Functions

Contents

Chapter 1

Functions

1.1 TurboCalc by Michael Friedrich

TurboCalc - copyright Michael Friedrich.

Full Table of Contents of this file

Main Table of Contents of all files

Full Index of all files

Besides operators and normal values, formulas consist mainly of functions which are described on the following pages, grouped by category.

First, here is an overview of all common "values" which are needed by functions as input parameters (and also as return value).

Numbers: These are numbers in the common format (+/-123[.123][E+-1]). You will find a detailed description in chapter three "input". (Position and Length are used interchangeably below with Number to make descriptions clear.)

Booleans: These are TRUE or FALSE. Instead of this full notation you can also use normal numbers. (Hence 0 represents FALSE, any other value means TRUE).

Text: Strings may consist of any available character. If used as parameter, a string has to be placed in quotation marks. If the result of a formula is text, it will be shown without quotation marks in the sheet.

Note: If the quotation mark itself is to be used in a text, you must use two quotation marks following each other for each quotation mark ("a""b" is the text a"b).

Date: This is a continuous integer value (0 means 1.1.1900, 1 means 2.1.1900...), which should be entered as a correct date and not as a number. Therefore the function DATEVALUE ("text") or simply VALUE ("text") must be used in contrast to the input in a cell. See below (e.g. VALUE ("20 Jan 94")).

Time: This is a continuous integer value (in seconds) (0 means midnight, 1 means 00:00:01.) which should be entered as a correct time and not as a number. Therefore the function TIMEVALUE ("text") or simply VALUE ("text") must be used, in contrast to the input in a cell. See below (e.g. VALUE ("12:03:07")).

Cell-Reference: Read the contents of another cell and used as a value. References consist of one, two or three letters followed by a number (without blanks!), e.g. A1,XE23 or XXA107. The letters indicate the column and the number the row. If cell C7 is filled with e.g. 123, the formula "=C7" results in 123.

Absolute references can also be entered (that means they will not be changed during copying, moving,...), by putting a Dollar sign in front of the "fixed" direction (or both). E.g. C7 or $C7 or C$7.

If you want to obtain values from other sheets, you may use the function @ (or AT), further details can be found in chapter "Functions-Cell Functions"

Range: These are parameters for the range- and database functions. Ranges consist of two references separated by a colon (e.g. A1:C5 or $C3:$E5) and determine the rectangle between these two cells. (A range may also consist of one reference only, in this case the 1 by 1 dimensioned range of the reference is meant).

Names: For all types mentioned above, you can use names (that means variables) as well. For details, see chapter eight "Names".

Note: The "appendix" has a list of all functions in alphabetical order (with respective parameters). Additionally all functions can be found in the "index" to allow an easy and fast access to this reference.

1.2 Operators

Operators

Pri. Op. Description

5 ^ Power

4 * Multiplication

4 / Division

4 MOD calculates the modulo (the remainder of a division, e.g. 4MOD3=1)

3 + Addition (also text, times and dates, see the corresponding function descriptions)

3 - Subtraction (also text,... see "+")

2 = compares both values (or text, dates or times) and returns either TRUE OR FALSE

<>

>=

<=

1 AND Logically ANDs two values. Further details can be found under "Boolean Functions".

1 OR Logically ORs two values. Further details can be found under "Boolean Functions".

1 XOR Logically XORs two values. Further details can be found under "Boolean Functions".

Here, the priority (Pri.) indicates the order in which the calculation is executed if no brackets are set. In this case, the operations with the highest priority will be executed first, the others successively. If several operators have the same priority the calculation is executed from the left to the right. This corresponds to common mathematical conventions.

Examples:

2+2*3^2+2 will be calculated as (2+(2*(3^2)))+2.

2-2-2 will be calculated as (2-2)-2 (in case of same priority from the left to the right) and results in -2, but 2-(2-2)=2 would also be possible).

1.3 Mathematical Functions

Mathematical Functions

These are the usual mathematical functions. Just about any pocket calculator or programming language provides these.

Number: Unless specified otherwise, the input parameter "number" is a real number.

ABS(Number)

ARCCOS(Number)

ARCSIN(Number)

ARCTAN(Number)

COS(Number)

COSHYP(Number)

DEGTORAD(Angle)

EXP(Number)

FAC(Number)

INTEGER(Number)

LG(Number)

LN(Number)

LOG(Number)

LOG10(Number)

MOD(Number1;Number2)

PI()

RADTODEG(Angle)

RND()

ROUND(Number;Places)

SIGN(Number)

SIN(Angle)

SINHYP(Number)

SQR(Number)

SQRT(Number)

TAN(Angle)

TANHYP(Number)

1.4 ABS(Number)

ABS(Number)

The result of the ABS function is an absolute value (i.e. the amount) of a number, that means the number without sign.

Example:

ABS(2) is 2

ABS(-2) is 2

Related Functions:

SIGN

1.5 ARCCOS(Number)

ARCCOS(Number)

The function ARCCOS returns the arccosine of a number. This is the angle which the cosine (COS) results in the argument number. The resulting angle is given in the arc measure with a value range from 0 to PI. (For the conversion into degree measure: see RADTODEG)

Number is the cosine of the desired angle and must lie between -1 and 1 (otherwise the error #VALUE will occur).

Example:

ARCCOS(-0.5) is 2.094

RADTODEG(ARCCOS(-0.5)) is 120 (degree)

Related Functions:

COS , PI , RADTODEG

1.6 ARCSIN(Number)

ARCSIN(Number)

The function ARCSIN returns the arcsine of a number. This is the angle of which the sine (SIN) results in the arguments number. The resulting angle is given in the arc measure with a value range from -PI/2 to PI/2. (For the conversion into degree measure: see RADTODEG)

Number is the sine of the desired angle and must lay between -1 and 1 (otherwise the error #VALUE will occur).

Example:

ARCSIN(-0.5) is 0.524

RADTODEG(ARCSIN(-0,5)) is 30 (degree)

Related Functions:

SIN , PI , RADTODEG

1.7 ARCTAN(Number)

ARCTAN(Number)

The ARCTAN function returns the arctangent of a number. This is the angle of which the tangent (TAN) results in the argument number. The resulting angle is given in the arc measure with a value range from -PI/2 to PI/2. (For the conversion into degree measure: see RADTODEG)

Example:

ARCTAN(1) is 0.785

RADTODEG(ARCTAN(-1)) is 45 (degree)

Related Functions:

TAN , PI , RADTODEG

1.8 COS(Number)

COS(Number)

The function COS returns the cosine of the angle (in arc measure). For the conversion of angles into the arc measure see DEGTORAD().

Example:

COS(1.047) is 0.5

COS(DEGTORAD(60)) is 0.5

Related Functions:

ARCCOS , PI , DEGTORAD

1.9 COSHYP(Number)

COSHYP(Number)

The function COSHYP returns the hyperbolic cosine of the angle (that means = 1/2*(e^z+e^-z))

Example:

COSHYP(2) is 3.762

Related Functions:

EXP , SINHYP , TANHYP

1.10 DEGTORAD(Angle)

DEGTORAD(Angle)

This function transforms an angle given in degrees into arc measure.

Hence normal angle specifications can be used as parameters for the functions Sine, Cosine and Tangent.

Example:

DEGTORAD(30) is 0.523 (=PI/6)

SIN(DEGTORAD(20)) is 0.5

Related Functions:

RADTODEG , SIN , COS , TAN

1.11 EXP(Number)

EXP(Number)

The function EXP returns the value e (=2.718281828) to the power of number.

Example:

EXP(1) is 2.71828...

EXP(LN(5)) is 5

Related Functions:

LN , LOG , LOG10 , LG

1.12 FAC(Number)

FAC(Number)

The function FAC returns the factorial of a number, that means 1*2*3*...*number

Number should be integer and positive. (Decimal numbers will be truncated).

Note: Instead of FAC (3) you can simply write 3!.

Example:

FAC(5) is 120

FAC(5.2) is 120

5! is 120, too

1.13 INTEGER(Number)

INTEGER(Number)

(or shortly INT)

The function INTEGER rounds down the parameter number to the next integer number.

Example:

INTEGER(5.4) is 5

INTEGER(5.7) is 5

INT(-5.2) is -6

Related Functions:

MOD , ROUND

1.14 LG(Number)

LG(Number)

Synonym to LOG10 (see below), which is well known to mathematicians.

1.15 LN(Number)

LN(Number)

The function LN gives the natural logarithm of a number (base is e=2.718281828...).

Number is a positive real number (that means any number >=0). Otherwise the error #VALUE appears.

LN is the inverse function of EXP.

Example:

LN(100) = 4.605

LN(EXP(3) = 3

Related Functions:

EXP , LOG , LOG10

1.16 LOG(Number)

LOG(Number)

Synonym of LG (see above), which is well known to mathematicians.

1.17 LOG10(Number)

LOG10(Number)

The function LOG10 returns the logarithm of a number to base 10.

Number is a positive real number (that means any number >=0). Otherwise the error #VALUE appears.

Example:

LN(100) =2

LN(10ˆ5) =5

Related Functions:

EXP , LN , LG , LOG , LOG10

1.18 MOD(Number1;Number2)

MOD(Number1;Number2)

The function MOD determines the remainder of the division of number1 by number2.

Number1 and number2 may represent any real number (number2 must not be 0.)

Example:

MOD (10;3) is 1

MOD (10.5;3) is 1.5

MOD (2.25;0.5) is 0,.25

Related Functions:

The arithmetic division: "/"

1.19 PI()

PI()

The function PI provides the number 3.1415926...

Example:

PI()/2 = 1.5707...

SIN(PI()/2) = 1

It is used for the calculation of the circumference and area of circles/spheres. If the radius of a circle is given in cell A1, the following expressions return:

2*PI()*A1 the circumference and

PI()*(A1ˆ2) the area of the circle.

1.20 RADTODEG(Angle)

RADTODEG(Angle)

This function transforms an angle given in arc measure into degrees.

Hence all angle-specifications of the functions ARCSIN, ARCCOS and ARCTAN can be converted into common values.

Example:

RADTODEG(0.5236) = 30

RADTODEG(ARCSIN(0.5)) = 30

Related Functions:

DEGTORAD , ARCSIN , ARCCOS , ARCTAN

1.21 RND()

RND()

The function RND returns a random number between 0 and 1.

If a random number between 1 and n is needed, it can be produced as follows:

INTEGER(RND()*N+1)

Example:

INTEGER(RND()*6+1) produces a random number between 1 and 6 and simulates a die.

1.22 ROUND(Number;Places)

ROUND(Number;Places)

The function ROUND rounds a number according to the required number of decimal places.

Number is the real number which should be rounded.

Places determines to how many decimal places the argument should be rounded.

Places>0: determines the places after the decimal point.

Places<0: rounds to the respective places to the left of the decimal point (e.g. 10, 100, 1000...)

Positions=0: rounds to the next integer number

Example:

ROUND(3.13;1) = 3.1

ROUND(3.15;1) = 3.2

ROUND(314.5;-2) = 300

1.23 SIGN(Number)

SIGN(Number)

The function SIGN produces the results -1, 0, 1 depending on the leading sign of the argument:

-1 if number<0, 0 if number=0 and 1 if number>0.

Example:

SIGN(20) = 1

SIGN(-20) =-1

SIGN(0) = 0

SIGN(A1)*ABS(A1) just returns A1 (if A1 is filled with a numeric value)

Related Functions:

ABS

1.24 SIN(Angle)

SIN(Angle)

The function SIN returns the sine of angle (in arc measure). For the conversion of angles into the arc measure, see DEGTORAD().

Example:

SIN(1.047) is 0.8659

SIN(DEGTORAD(60) is 0.8659

Related Functions:

ARCSIN , PI , DEGTORAD

1.25 SINHYP(Number)

SINHYP(Number)

The function SINHYP returns the hyperbolic sine of number (that means = 1/2*(e^z-e^-z))

Example:

SINHYP(2) is 3.627

Related Functions:

EXP , COSHYP , TANHYP

1.26 SQR(Number)

SQR(Number)

The function SQR returns the square of number (i.e. number*number).

Example:

SQR(5) = 25

5^2=25

Related Functions:

SQRT

1.27 SQRT(Number)

SQRT(Number)

The function SQRT returns the positive square root of a number.

Number is a positive real value (that means any number >=0). Otherwise the error #VALUE appears.

Example:

SQRT(25) = 5

SQRT(SQR(5)) = 5

SQRT(ABS(-25)) = 5

Related Functions:

SQR , ABS

1.28 TAN(Angle)

TAN(Angle)

The function TAN returns the tangent of the parameter angle in arc measure (i.e. SIN/COS). For the conversion of angles into the arc measure, see DEGTORAD().

Example:

TAN(0.7854) is 1

TAN(DEGTORAD(45) is 1

Related Functions:

ARCTAN , COS , PI , DEGTORAD , SIN

1.29 TANHYP(Number)

TANHYP(Number)

The function TANHYP returns the hyperbolic tangent of number (that means SINHYP(Number)/COSHYP(Number))

Example:

TANHYP(2) is 0.964

Related Functions:

EXP , SINHYP , COSHYP

1.30 Boolean Functions

Boolean Functions

These functions return a boolean value (TRUE or FALSE, that means 1 or 0) or process such booleans.

Therefore, these functions are very suitable for conditional calculations and decisions (Above all, see IF).

AND(Value1; Value2;...)

FALSE()

IF(Condition;Value1;Value2)

ISDATE(Value)

ISEMPTY(Reference)

ISERROR(Reference)

ISSTRING(Value)

ISTIME(Value)

ISVALUE(Value)

NOT(Value)

OR(Value1; Value2;...)

TRUE()

XOR(Value1;Value2;...)

1.31 AND(Value1; Value2;...)

AND(Value1; Value2;...)

The function AND returns the result of the logical AND of all values as boolean, i.e. TRUE will be returned if all values are TRUE and non-zero. If only one value is zero or FALSE, FALSE will be returned.

Values: You can compare as many values as you wish separated by a semicolon. Accepted values are booleans or numbers (A number will be interpreted as TRUE, if it is not 0.).

Note: Instead of this notation, AND can also be used as an operator in in-line notation (like "+"): A1 AND A3. Above all, it increases the legibility of an expression if only two values should be ANDed. (Furthermore, AND can be used in this notation to AND two (binary) numbers: Then, 7 AND 4 is 4).

Example:

AND(TRUE;FALSE) is FALSE

AND(TRUE;1;2) is TRUE

AND(FALSE;TRUE;1) is FALSE

IF(AND(0;1);3;4) is 4

this also corresponds to:

TRUE AND FALSE

FALSE AND 1 AND 2

FALSE AND TRUE AND 1

IF(0 AND 1;3;4)

3 AND 1 is 1

7 AND 3 is 3

Related Functions:

OR , NOT , XOR

1.32 FALSE()

FALSE()

The function FALSE returns the value FALSE (=0). It can either be used as FALSE or FALSE().

Example:

IF(FALSE);1;2) is 2

1.33 IF(Condition;Value1;Value2)

IF(Condition;Value1;Value2)

The function IF returns, depending on the condition, either value1 (if condition complied, TRUE) or value2.

Condition may be any Boolean expression.

Value1, value2 may be any expression (text, numbers...). Further IF(..;..;..) contructions as parameters are also possible, i.e. IF constructs may be nested.

Note: This function is very practical, as it allows the easy construction of case distinctions.

Example:

IF(TRUE;"Hello", "you") is "hello"

IF(A1>0;"profit", "loss") determines, depending on A1, if a profit or loss was made.

IF(A1<10;25;25+(A1-10)*0.23) calculates the telephone costs of A1 units (in this case for Germany: Base fee: 25 DM, unit 23 PF and 10 units free of charge).

Related Functions:

CHOOSE

1.34 ISDATE(Value)

ISDATE(Value)

The function ISDATE checks if the given value is a date or not. If so, TRUE will be returned, otherwise FALSE.

Example:

ISDATE(12) is FALSE

1.35 ISEMPTY(Reference)

ISEMPTY(Reference)

The function ISEMPTY checks if the reference cell is empty. In this case, TRUE will be returned, otherwise FALSE.

Reference: Is a reference to a cell, e.g. A1

Example:

ISEMPTY(A1) is FALSE, if A1 is filled with a numeric value, text or any other admitted input.

1.36 ISERROR(Reference)

ISERROR(Reference)

The function ISERROR checks if the cell the reference points to contains an error message. In this case, TRUE will be returned, otherwise FALSE.

Example:

ISERROR(A1) with A1 containing "12" is FALSE

1.37 ISSTRING(Value)

ISSTRING(Value)

The function ISSTRING checks if the given value is text. If so, TRUE will be returned, otherwise FALSE.

Example:

ISSTRING("12") is TRUE

1.38 ISTIME(Value)

ISTIME(Value)

The function ISTIME checks if the given value is a time. If so, TRUE will be returned, otherwise FALSE.

Example:

ISTIME(12) is FALSE

ISTIME(A1) is TRUE, if A1 is filled with time data.

1.39 ISVALUE(Value)

ISVALUE(Value)

The function ISVALUE checks if the given value is a number. If so, TRUE will be returned, otherwise FALSE.

Example:

ISVALUE("12") is FALSE

1.40 NOT(Value)

NOT(Value)

The function NOT negates the input boolean value, that means if the value is TRUE, FALSE will be returned and vice versa.

Value: Can be a boolean or a number. 0 represents FALSE, all other numbers are interpreted as TRUE.

Example:

NOT(TRUE) is FALSE

NOT(TRUE AND FALSE) is TRUE

NOT(FALSE) is TRUE

NOT(3) is FALSE

Related Functions:

AND , OR , XOR

1.41 OR(Value1; Value2;...)

OR(Value1; Value2;...)

The function OR returns the result of all values ORed as boolean, that means TRUE will be returned if at least one value is TRUE or non-zero. If all values are zero or FALSE, FALSE will be returned.

Values: You can compare as many values as you wish, separated by a semicolon. Accepted values are booleans or numbers. (A number will be interpreted as TRUE, if it is not 0.)

Note: Instead of this notation, OR can also be used as an operator in in-line notation (like "+"): A1 OR A3. Above all, it increases the legibility of an expression if only two values are to be ORed. (Furthermore, OR can be used in this notation to OR two (binary) numbers: Then, 3 OR 4 is 7).

Example:

OR(TRUE;FALSE) is TRUE

OR(FALSE;0;0) is FALSE

OR(FALSE;0;1) is TRUE

IF(OR(0;1);3;4) is 4

this also corresponds to:

TRUE OR FALSE

FALSE OR 0 OR 0

FALSE OR 0 OR 1

IF(0 OR 1;3;4)

2 OR 1 is 3 and 3 OR 2 is 3

Related Functions:

AND , NOT , XOR

1.42 TRUE()

TRUE()

The function TRUE returns the value TRUE (=1). It can either be written as TRUE or TRUE().

Example:

IF(TRUE ;1;2) is 1

1.43 XOR(Value1;Value2;...)

XOR(Value1;Value2;...)

The function XOR returns the result of the exclusive-or of all values as boolean, that means FALSE will be returned if an even number of values is TRUE or non-zero. Otherwise, TRUE will be returned.

value1	value2	value1 XOR value2
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

Values: You can compare as many values as you wish, separated by a semicolon. Accepted values are booleans or numbers. (A number will be interpreted as TRUE, if it is not 0.)

Note: Instead of this notation, XOR can also be used as an operator in in-line notation (like "+"): A1 XOR A3. Above all, it increases the legibility of an expression if only two values are to be XORed. (Furthermore, XOR can be used in this notation to XOR two (binary) numbers: Then, 3 XOR 4 is 7).

Example:

XOR(TRUE;FALSE) is TRUE

XOR(FALSE;0;0) is FALSE

XOR(FALSE;0;1) is TRUE

IF(XOR(0;1);3;4) is 3

this also corresponds to:

TRUE XOR FALSE

FALSE XOR 0 XOR 0

FALSE XOR 0 XOR 1

IF(0 OR 3;3;4)

2 XOR 1 is 3

3 XOR 2 is 1

Related Functions:

OR ; AND , NOT

1.44 Text Functions

Text Functions

Plus

CHAR(ASCII-Code)

CLEAN(Text)

CODE(Text)

COMPRESS(String[;List])

INSTRING(String;Pattern[;Pos])

HEX(number text)

LEFT(Text;Length)

LENGTH(Text)

LOWER(Text)

MID(Text;Position;Length)

PART(Text;Position;Length)

REPEAT(Text;Number)

REVERSE(Text)

RIGHT(Text;Number)

SHIFTL(Text)

SHIFTR(Text)

TEXT(Data[;Format])

TRIM(Text)

UPPER(Text)

UPPER2(Text)

VALUE(Text)

1.45 Plus

Plus

The operator "+" can be used to concatenate text (to append one after the other). The syntax is the same as with normal numbers.

Warning: A minus operation, as well as an operation between text and a number is undefined! A #TYPE-error will appear in these cases.

Example:

"Hello" + " " + "world" is "hello world"

"Ab" + "c" is "Abc"

1.46 CHAR(ASCII-Code)

CHAR(ASCII-Code)

The function CHAR returns the character (i.e. the text of length 1), which is represented by the given ASCII-code.

This function is the inverse of CODE.

The argument ASCII-code must range between 1 and 255!

Example:

CHAR(65) is "A"

CHAR(CODE("T")) is "T"

Related Functions:

CODE

1.47 CLEAN(Text)

CLEAN(Text)

The function CLEAN removes all unprintable characters from text, i.e. all control codes. This is useful if you have imported text which contained such foreign control codes (indicated by a rectangular box on the screen), for it avoids manual editing of the argument.

Example:

CLEAN(CHAR(7)+"text") produces "text"

Related Functions:

CHAR , TRIM

1.48 CODE(Text)

CODE(Text)

This function code returns the ASCII-Code (American Standard Code for Information Interchange, see Amiga manual) of the first letter of the text.

Example:

CODE("A") is 65

CODE("ABC") is 65

CODE("a") is 97 (because the "a" is now given in lower case!)

Related Functions:

CHAR

1.49 COMPRESS(String[;List])

COMPRESS(String[;List])

Removes all characters from List out of String and returns the reult. (If List is omitted, all blanks will be removed).

String: Text to be compressed.

List: Text with all characters to be removed. If this parameter is omitted, "" (blanks) will be assumed.

Example:

COMPRESS (" a b c") is "abc"

COMPRESS ("++a-b++c","+-") is "abc"

1.50 INSTRING(String;Pattern[;Pos])

INSTRING(String;Pattern[;Pos])

Returns the first position (starting at Pos), from which Pattern in String occurs. If the text has not been found, 0 will be returned.

String: Text which is to be searched through.

Pattern: Text which is to be searched for in String.

Pos: The position from which on the search has to be started (1=first position - if this is omitted, the search will start at the beginning.

Example:

INSTRING("toad road";"oa") is 2

INSTRING("toad road";"oa";3) is 7

INSTRING("toad road";"abc") is 0 (that means not found)

1.51 HEX(number text)

HEX(number text)

This function changes the text number text from the hexadecimal system to a `normal´ number in the decimal system.

Number text should consist of the numerals `0´-`9´ and the characters `A´-`F´ (the use of small letters is possible). Every other character terminates the conversion.

Example:

HEX("C8") results in 200

HEX("C8X1") results in 200, the conversion will be terminated at `X´.

1.52 LEFT(Text;Length)

LEFT(Text;Length)

LEFT returns at most Length of the left-most letters of Text. Length must be greater than 0 (otherwise the error message #VALUE appears). If length is greater than the total length of the text, the whole text will be returned.

If length is omitted (LEFT(text)), only the first letter of the text will be returned.

Example:

LEFT("Michael Friedrich";4) is "Mich"

LEFT("Hello") is "H"

Related Functions:

MID , PART , RIGHT

1.53 LENGTH(Text)

LENGTH(Text)

The function LENGTH returns the number of characters in a string (i.e. its length) as a number.

Example:

LENGTH("Michael Friedrich") results in 17

LENGTH("") is 0

1.54 LOWER(Text)

LOWER(Text)

The function LOWER converts all letters of the argument into lower-case letters.

Example:

LOWER("This is Text") becomes "this is text"

Related Functions:

UPPER , UPPER2

1.55 MID(Text;Position;Length)

MID(Text;Position;Length)

The function MID produces a substring from the given argument text, starting at Position and at most Length characters long.

Position must always be greater than 0 (otherwise the error message #VALUE appears) and determines the starting position from which the text shall be taken.

Length must be greater than 0 (otherwise the error message #VALUE is returned as well) and determines how many letters shall be returned. If number2 is greater than the length of the rest of the text, this rest will be returned.

Example:

MID("Michael Friedrich";2;3) is "ich"

MID("Michael Friedrich";8;20) is "Friedrich"

MID("Michael Friedrich";20;20) is ""

Related Functions:

LEFT , RIGHT , PART

1.56 PART(Text;Position;Length)

PART(Text;Position;Length)

This function is equivalent to MID. It was added to the instruction set for compatibility reasons only. Description, see MID.

1.57 REPEAT(Text;Number)

REPEAT(Text;Number)

The function REPEAT returns a string, which consists of the number of repetitions of the text-parameter.

Number must be higher than 0, otherwise #VALUE will be returned.

Example:

REPEAT("-";5) is "-----"

REPEAT("text";3) is "texttext"

1.58 REVERSE(Text)

REVERSE(Text)

Reverses the Text. i.e. Produces it backwards.

Text: any text

Example:

REVERSE("Hello") is "olleH"

1.59 RIGHT(Text;Number)

RIGHT(Text;Number)

RIGHT returns the Number right-most characters from Text. Number must be higher than 0 (otherwise the error message #VALUE appears). If number is higher than the total length of the text, the whole text will be returned.

If number is omitted (RIGHT(text)), only the last letter of the text will be returned.

Example:

RIGHT("Michael Friedrich";3) is "ich"

RIGHT("Hello") is "o"

Related Functions:

LEFT ; MID , PART ,

1.60 SHIFTL(Text)

SHIFTL(Text)

Shifts (rotates) the text one position to the left; i.e. the second character becomes the first one, the first character at the left will be added at the end.

Text: any text

Example:

SHIFTL("Hello") is "elloH"

1.61 SHIFTR(Text)

SHIFTR(Text)

Shifts (rotates) the text one position to the right, the last character will be removed and inserted at the beginning.

Text: any text

Example:

SHIFTR("Hello") is "oHell"

1.62 TEXT(Data[;Format])

TEXT(Data[;Format])

Converts all types of data (number, boolean, date, time and text) into text.

Data: Can be any expression.

Format: Determines the format, how the argument data is to be formatted as text. 0 or omitting the parameter corresponds to the standard format. Further format numbers can be found in the table of the macro instruction NUMERICFORMAT.

This function is the counterpart of VALUE.

Example:

TEXT(123:3) is "123.00"

TEXT(12)+"Dollar" is "12 Dollar"

VALUE(TEXT(123.45;1)) is 123 (because TEXT(123.45;1) produces "123").

1.63 TRIM(Text)

TRIM(Text)

TRIM deletes all "unnecessary" blanks in the text. It reduces all blank-sequences to one single space for separation between words..

Example:

TRIM("January February March") becomes "January February March"

Related Functions:

CLEAN

1.64 UPPER(Text)

UPPER(Text)

UPPER converts all letters of the argument into upper-case (capital) letters.

Example:

UPPER("This is text") becomes "THIS IS TEXT"

Related Functions:

UPPER2 , LOWER

1.65 UPPER2(Text)

UPPER2(Text)

(Initial caps) UPPER2 converts only the first letter of each word into capital letters while converting the rest into small letters.

Example:

UPPER2("This is a text") becomes "This Is A Text"

Related Functions:

UPPER , LOWER

1.66 VALUE(Text)

VALUE(Text)

The function VALUE converts the given text into a number, a date, a time or an error value. This is advisable, if date- or time-values are to be used in formulas, as normal specification is not possible there. Moreover this function can be used to check certain cell contents without risking an error message (if the cells contain numbers, these numbers will simply be passed on, in case of normal texts, 0 will be returned).

This function is the counterpart of TEXT.

Value: can also be a number, boolean, date, time as parameter. In this case this will be returned correspondingly. (VALUE(42)=42, VALUE(b1) with b1=13:02 is 13:03)

Thus cell contents which can contain numbers as well as texts can very easily be tested and you will not receive a #TYPE error message. (e.g. IFGOTO(VALUE(b1)=0;c5) - GOTOs when b1 is 0 or contains text which cannot be interpreted as a number)

Example:

VALUE("1000") is 1000.

VALUE("20 Jan 93") produces 20.01.1993

If cell A1 contains a date, the days, beginning from 1.1.1992, can be calculated with the formula A1-VALUE("1.1.1992")

1.67 Date and Time Functions

Date and Time Functions

Most of the functions mentioned below deal with date and time values (that means, they must receive a date or time as their argument or they return one).

Date: The date is internally saved as an integer number. The 1.1.1900 is allocated to the number 0. Thus, the date 1.7.93 has the value 34149.

(With the help of the function <Format-Numeric Format> you can force TurboCalc to display date values as a number ("0") or with the date format "DD-MM-YYYY" - standard decides for itself).

This possibility can be used for advanced applications ("date - date" is just the number of days between these two days, see below.)

Time: The time is saved as the number of seconds since midnight!

Therefore, if a date (or time) is required as parameter of a function, a normal value can be entered as well. In this case, please note the interpretation of the number!

Plus, Minus

DATE(Year;Month;Day)

DATEVALUE(Text)

DAY(Date)

MONTH(Date)

NOW()

TIMEVALUE(Text)

TODAY()

WEEKDAY(Date)

YEAR(Date)

1.68 Plus, Minus

Plus, Minus

The operators "+" and "-" are also defined for date and time, according to the following rules:

date + number =date (increases date by number of days)

number + date =date (increases date by number of days)

date - number =date (diminishes date by number of days)

date - date =number (number of days between the two dates)

time + number =time (increases time by number of seconds)

number + time =time (increases time by number of seconds)

time - number =time (diminishes time by number of seconds)

time - time =number (number of seconds between the given times)

Concerning the type number the value of "1" represent one single second, values with decimals are ignored)

Note: In all other operations (e.g. number - date) the date value will be transformed into a number. Thus, TurboCalc goes on calculating with two numbers).

Example:

VALUE("5-1-1993")-VALUE("1-1-1993") produces 4

VALUE("18:00")-VALUE("12:00") produces 21600 (=6*60*60*)

Related Functions:

VALUE , DATEVALUE , TIMEVALUE , DATE

1.69 DATE(Year;Month;Day)

DATE(Year;Month;Day)

The function DATE converts the three parameters into a date.

Year, month and day must represent an existing day from 1.1.1900 until 31.12.2400 - otherwise the error #VALUE will be returned.

Example:

DATE(1993;4;1) produces the 1st of April 1993

Related Functions:

DATEVALUE , DAY , MONTH , YEAR , TODAY , NOW , TIMEVALUE

1.70 DATEVALUE(Text)

DATEVALUE(Text)

The function DATEVALUE converts a text into a date. (You can also use VALUE, which allows a conversion of all formats, see there.)

Text can be a any date format in text-form, e.g.:

"30-9-93"

"30 Sep 93"

"Sep 93"

"30 Sep"

(for a complete description of all possibilities refer to chapter three, "The Input")

Note: This function is very useful if date-values are to be used in a formula as the dates mentioned above cannot be entered directly.

Example:

DATEVALUE("1.4.93") produces the 1st of April 93

Related Functions:

VALUE , DATE , TODAY , TIMEVALUE

1.71 DAY(Date)

DAY(Date)

The function DAY produces the day-of-month of a date as a number.

Example:

DAY(VALUE("23-3-93")) is 23

DAY(2000) = 24

Related Functions:

MONTH , YEAR , WEEKDAY , TODAY , NOW

1.72 MONTH(Date)

MONTH(Date)

The function MONTH produces the month of a date as a number.

Example:

MONTH(VALUE("23-3-93") is 3

MONTH(2000) = 6

Related Functions:

YEAR , DAY , WEEKDAY , TODAY , NOW

1.73 NOW()

NOW()

The function NOW returns the current system time.

Note: Please check that the time is set correctly before using this function. If an internal (buffered) clock is installed, this should be guaranteed. Otherwise you should adjust date and time with the CLI Instruction DATE or with Preferences.

Example:

NOW() produces the current system time.

Related Functions:

TIMEVALUE , TODAY

1.74 TIMEVALUE(Text)

TIMEVALUE(Text)

The function TIMEVALUE converts text into a date. (You can also use VALUE, which allows a conversion of all formats, see there.)

Text can be any time format in textform:

13:03 or

13:03:05 (with seconds)

Note: This function is very useful if time-values are to be used in a formula as the times mentioned above cannot be entered directly.

Example:

TIMEVALUE("13:45") produces 13 h 45 and 0 seconds

Related Functions:

VALUE , DATE , TODAY , DATEVALUE

1.75 TODAY()

TODAY()

The function TODAY returns the date of the current day.

Note: Please check that the date is set correctly before using this function. If an internal (buffered) clock is installed, this should be guaranteed. Otherwise you should adjust date and time with the CLI Instruction DATE or with Preferences.

Example:

TODAY()+2 produces the date of the day after tomorrow.

Related Functions:

DATE , DATEVALUE , NOW

1.76 WEEKDAY(Date)

WEEKDAY(Date)

The function WEEKDAY calculates the weekday for the given date and returns it as a number from 1 to 7. (1=Sunday, 7 = Saturday)

Example:

WEEKDAY(DATE("1 June 92")) is 3 (Tuesday)

CHOOSE(WEEKDAY(TODAY);"Su";"Mo";"Tu";"We";"Th";"Fr";"Sa") produces a two-letter abbreviation for the current weekday.

Related Functions:

YEAR , MONTH , DAY , TODAY , NOW

1.77 YEAR(Date)

YEAR(Date)

The function YEAR produces the year-number of a date as a number.

Example:

YEAR(VALUE("23-3-93") is 1993

YEAR(2000) = 1905

Related Functions:

MONTH , DAY , WEEKDAY , TODAY , NOW

1.78 Sheet Functions

Sheet Functions

All sheet functions require a list of ranges as parameters (i.e. one or more ranges, separated by semicolon).

A range is either a cell (A1...) or a rectangular cell block, which is specified by cell1 + ":" + cell2 (e.g. A1:C3).

Naturally, the cells/ranges can be absolute (with dollar sign) or relative or be replaced by names, for further details, see "Names".

All of the following functions can process any number of ranges (separated by a semicolon).

Furthermore, TurboCalc allows a direct input of numbers where parameters are required, which is very useful in connection with the functions MIN and MAX.

AVERAGE(Range)

COUNT(Range)

COUNT2(Range)

MAX(Range)

MIN(Range)

PRODUCT(Range)

STDEV(Range)

SUM(Range)

VAR(Range)

For all examples the following "example sheet" is used:

A B C

1: 2 4

2: 3 5

3: text

4:

5:

1.79 AVERAGE(Range)

AVERAGE(Range)

The function AVERAGE calculates the average of all numbers in the given range. Text and empty cells are ignored. If the range contains no numbers, 0 will be returned.

Example:

AVERAGE(A1:A5) = 2.5*

AVERAGE(A1;A2:B2;A3:B3) = 3.5

AVERAGE(A1;A4) = 0

Related Functions:

COUNT , COUNT2 , MAX , SUM , PRODUCT

1.80 COUNT(Range)

COUNT(Range)

The function COUNT returns the number of numeric data within the given range. Text and empty cells are ignored.

Example:

COUNT(A1:A5) = 2

COUNT(A1;A2:B2;A3:B3) = 4

Related Functions:

COUNT2 , MAX , MIN , SUM , PRODUCT

1.81 COUNT2(Range)

COUNT2(Range)

The function COUNT2 returns the number of cells which are not empty within the given range. Empty cells are ignored but text, dates and times are counted.

Example:

COUNT(A1:A5) = 3

COUNT(A1;A2:B2;A3:B3) = 4

Related Functions:

COUNT , MAX , MIN , SUM , PRODUCT

1.82 MAX(Range)

MAX(Range)

The function MAX returns the highest number within the given range. Text and empty cells are ignored. If the range contains no numbers, 0 will be returned.

Example:

MAX(A1:A5) = 3

MAX(A1;A2:B2;A3:B3) = 5

MAX(A1:F20;100) determines the maximum value of the block A1:F20. If it is less than 100, 100 is returned.

Related Functions:

COUNT , COUNT2 , MAX SUM , PRODUCT

1.83 MIN(Range)

MIN(Range)

The function MIN returns the smallest number within the given range. Texts and empty cells are ignored. If the range contains no numbers, 0 will be returned.

Example:

MIN(A1:A5) = 1

MIN(A1;A2:B2;A3:B3) = 1

MIN(A1;A4) = 0

MIN(A1:F20;100) determines the minimum value of the block A1:F20. If it is greater than 100, 100 is returned.

1.84 PRODUCT(Range)

PRODUCT(Range)

The function PRODUCT returns the product of all numbers within the given range. Text and empty cells are ignored. If the range contains no numbers, 0 will be returned.

If one of the numbers is 0, the multiplication results in 0 as well.

Example:

PRODUCT(A1:A5) = 6

PRODUCT(A1;A2:B2;A3:B3) = 120

Related Functions:

COUNT , COUNT2 , MAX , MIN , SUM

1.85 STDEV(Range)

STDEV(Range)

The function STDEV returns the standard deviation of all numbers within the given range. If the range contains no numbers, 0 will be returned.

Example:

SUM(A1:A5) = 5

SUM(A1;A2:B2;A3:B3) = 14

Related Functions:

COUNT , COUNT2 , MAX , MIN , PRODUCT

1.86 SUM(Range)

SUM(Range)

The function SUM returns the sum of all numbers within the given range. If the range contains no numbers, 0 will be returned.

Example:

SUM(A1:A5) = 5

SUM(A1;A2:B2;A3:B3) = 14

Related Functions:

COUNT , COUNT2 , MAX , MIN , PRODUCT

1.87 VAR(Range)

VAR(Range)

The function VAR returns the variance of all numbers within the given range. If the range contains no numbers, 0 will be returned.

Example:

SUM(A1:A5) = 5

SUM(A1;A2:B2;A3:B3) = 14

Related Functions:

COUNT , COUNT2 , MAX , MIN , PRODUCT

1.88 Database Functions

Database Functions

These functions are very similar to the sheet functions described above, but here a database-range instead of a "normal" range is required. Therefore, all following functions have the same syntax:

Dfunction (Database Range; Column; Criteria Range)

Important: If you are familiar with TurboCalc 2.0 you will notice that the names of the database functions have changed: Now, they are only prefixed by "D" (in version 2 the prefix was "DB"). Old formulas will be automatically converted.

Database Range: This is a range as described in chapter seven "Database" (i.e. first row: title, remaining rows: data). You are not limited to the current database, but the name "DATABASE" always refers to the current one.

Column: This parameter specifies the column (of the database range), which should be affected by the function (not by the criteria range). It can be entered as a number (0=first column, 1=second column...) or as column-title text (in quotation marks!).

Criteria Range: This range determines, which data records should be taken for calculation. Only those data records (of the selected column) are used, which match the criteria. For details (also about creation of a criteria range), see chapter seven "Database", paragraph "search criteria".

Note: These functions are not limited to the current database range only, but may also be used for any (correctly installed database) range (e.g. to count or add up columns with a value>2000 or a date between 1.1.1990 and 1.1.1991...).

Note: Because these function need quite some time for calculation (especially for big ranges, for they have to compare all datasets of the range), it is advisable to switch off the automatical recalculation beforehand.

DAVERAGE(Database;Column;Criteria)

DCOUNT(Database; Column, Criteria)

DCOUNT2(Database; Column, Criteria)

DMAX(Database;Column;Criteria)

DMIN(Database;Column;Criteria)

DPRODUCT(Database;Column;Criteria)

DSTDEV(Database;Column;Criteria)

DSUM(Database;Column;Criteria)

DVAR(Database;Column;Criteria)

1.89 DAVERAGE(Database;Column;Criteria)

DAVERAGE(Database;Column;Criteria)

The function DAVERAGE returns the average of all numbers of the desired column of all rows of the database range which match the criteria - for details about the parameters, see introduction to the database functions.

If the criteria are not met, or if there are no numbers in the rows of the selected column, 0 will be returned.

Example:

DAVERAGE(DATABASE;0;CRITERIA) returns the average of the first column of the database range, which match the criteria.

DAVERAGE(DATABASE;"number";CRITERIA) does the same, if the first row has the title "number".

Related Functions:

Databasefunctions

1.90 DCOUNT(Database; Column, Criteria)

DCOUNT(Database; Column, Criteria)

The function DCOUNT returns the amount of numerical data within the database range, which match the criteria - for details about the parameters, see introduction to the database functions.

Example:

DCOUNT(DATABASE;0;CRITERIA) returns the count of numbers in the first column of the database range, which match the criteria.

DCOUNT(DATABASE;"number";CRITERIA) does the same, if the first column has the title "number".

Related Functions:

Databasefunctions

1.91 DCOUNT2(Database; Column, Criteria)

DCOUNT2(Database; Column, Criteria)

The function DCOUNT2 returns the number of non-empty cells within the database range which match the criteria - for details about the parameter, see introduction to the database functions.

Empty cells are ignored but text, dates and times are counted.

This function is very useful to determine the number of data records which correspond to the criteria.

Example:

DBCOUNT2(DATABASE;0;CRITERIA) produces the number of non-empty cells of the first column of the database range, which match the criteria.

DCOUNT2(DATABASE;"number";CRITERIA) does the same, if the first column has the title "number".

Related Functions:

Databasefunctions

1.92 DMAX(Database;Column;Criteria)

DMAX(Database;Column;Criteria)

The function DMAX returns the highest number of the desired column for all rows of the database range which match the criteria - for details about the parameters, see introduction to the database functions.

If the criteria do not match any rows, or if there are no numbers in the rows of the selected column, 0 will be returned.

Example:

DMAX(DATABASE;0;CRITERIA) returns the highest number of the first column of the database range, which match the criteria.

DMAX(DATABASE;"number";CRITERIA) does the same, if the first row has the title "number".

Related Functions:

Databasefunctions

1.93 DMIN(Database;Column;Criteria)

DMIN(Database;Column;Criteria)

The function DMIN returns the smallest number of the desired column of all rows of the database range which match the criteria - for details about the parameters, see introduction to the database functions.

If the criteria do not match any rows, or if there are no numbers in the rows of the selected column, 0 will be returned.

Example:

DMIN(DATABASE;0;CRITERIA) returns the smallest number of the first column of the database range, which match the criteria.

DMIN(DATABASE;"number";CRITERIA) does the same, if the first row has the title "number".

Related Functions:

Databasefunctions

1.94 DPRODUCT(Database;Column;Criteria)

DPRODUCT(Database;Column;Criteria)

The function DPRODUCT returns the product of all numbers in the desired column of all rows of the database range which match the criteria - for details about the parameters, see introduction to the database functions.

If the criteria do not match any rows, or if there are no numbers in the rows of the selected column, 0 will be returned.

Example:

DPRODUCT(DATABASE;0;CRITERIA) multiplies all numbers of the first column of the database range, which match the criteria.

DPRODUCT(DATABASE;"number";CRITERIA) does the same, if the first row has the title "number".

Related Functions:

Databasefunctions

1.95 DSTDEV(Database;Column;Criteria)

DSTDEV(Database;Column;Criteria)

The function DSTDEV returns the standard deviation of all numbers in the desired column of all rows of the database range which match the criteria - for details about the parameters, see introduction to the database functions.

If the criteria do not match any rows, or if there are no numbers in the rows of the selected column, 0 will be returned.

Related Functions:

Databasefunctions

1.96 DSUM(Database;Column;Criteria)

DSUM(Database;Column;Criteria)

The function DSUM returns the sum of all numbers in the desired column of all rows of the database range which match the criteria - for details about the parameters, see introduction to the database functions.

If the criteria do not match any rows, or if there are no numbers in the rows of the selected column, 0 will be returned.

Example:

DSUM(DATABASE;0;CRITERIA) adds up all numbers of the first column of the database range, which match the criteria.

DSUM(DATABASE;"number";CRITERIA) does the same, if the first row has the title "number".

Related Functions:

Databasefunctions

1.97 DVAR(Database;Column;Criteria)

DVAR(Database;Column;Criteria)

The function DBVARreturns the variance of all numbers in the desired column of all rows of the database range which match the criteria - for details about the parameters, see introduction to the database functions.

If the criteria do not match any rows, or if there are no numbers in the rows of the selected column, 0 will be returned.

Related Functions:

Databasefunctions

1.98 Cell Functions

Cell Functions

#Reference

@Sheet; reference

CELL(Row;Column)

CELLABS(Row;Column)

CHOOSE(Index; Value1;Value2;Value3...)

COLUMNNUMBER(Range)

INDIRECT(Text)

HLOOKUP(Value;Range;Offset;[Exact])

LOOKUP(Value;Range;[Exact])

RANGEABS(Row;Column;Height;Width)

RANGEHEIGHT(Range)

RANGEWIDTH(Range)

RANGEX([Range])

RANGEY([Range])

ROWNUMBER(Range)

SHEETNAME()

VLOOKUP(Value;Range;Offset;[Exact])

1.99 #Reference

#Reference

(This function is defined for macro instructions only, otherwise an error message or an useless value will be returned).

References within macro instructions generally refer to the sheet out of which the macro was called and not to the separate macro sheet (exception: CALL, IFGOTO see there).

If you want to address a cell within the macro sheet, it can be done with #reference. Here, reference represents the desired cell or the selected range.

(#reference is identical to the expression "@macrosheet;reference",if "macrosheet" is the name of the sheet in which the currently running macro is originally located.)

Example:

The following macro extract implements a safety requestor:

A10 =REQUEST("Yes or No?")

A12 here, the main program can be found

=RETURN()

A11 =IFGOTO(NOT(#A10);A20)

.....

A20 =RETURN()

The first instruction REQUEST asks, if you want to proceed (briefly here: "yes or no?"). The user may click on >Ok< or >Abort<. When clicking on >Ok<, cell A10 is filled with TRUE, otherwise FALSE.

This value is then tested in cell A11 (If you missed the # here and you started the macro from another sheet, the value of the cell of this sheet (the other one) would have been used, which was not your intention!).

If it is FALSE, then NOT(#A10) is TRUE and a jump to cell A20 is executed. In this case, the main routine is omitted.

Related Functions:

@Sheet;reference

1.100 @Sheet; reference

@Sheet; reference

(or AT(sheet;reference)

The AT-function (that means the "commercial at" "@") provides the values of cells from other sheets. Therefore, this function can be used to link several sheets.

Sheet: This is the name of a sheet (as it appears in the window-title). If the file is loaded, the name without specification of the path and the ending, will be sufficient, if the original file name has the tag ".TCD". The name must be terminated by a semicolon ";". The name can be put in quotation marks (and must be, if it contains a semicolon ";"!). If the sheet is not loaded, the specification of the path must also be indicated to allow TurboCalc to find the file (also see below).

Hence, the loaded file "DhO:data/Test.TCD" can be addressed with Test, Test.TCD, data/Test.TCD or dho:data/Test.TCD.

Reference: Is a reference which complies with the TurboCalc format (e.g. A1 or A1). It is of no importance if the reference is relative or absolute (with or without dollar signs).

Note: You can use @ or AT interchangeably. The two parameters (separated by a semicolon!) can also be put in brackets.

From version 3.0 on, you can also access closed sheets with the AT function: (But only to read cells, e.g. @Test;a7 - for range references with this instruction the other sheet will still have to be opened.)

If the desired sheet is not open, a read of the indicated cell out of the file will be attempted. The read result will be buffered internally so that the disk will only be accessed once. The search sequence will be as follows:

i) If the project name contains a colon":", the search will only take place in this path (for it is an "absolute" path specification).

ii) Otherwise the search will take place in succession in the following paths:

1. Path of the sheet in which the function had been input. (If a path had been set, that means if a ":" or "/" exists in the sheet name.)

2. Path from which a sheet had been started by the WorkBench (double click on icon or shift-click on icon and program).

3. Path in which TurboCalc exists.

4. Active path when loading TurboCalc (Shell only).

If the sheet could not be found, ".TCD" will be added and the search will be repeated from 1. through 4..

Therefore: The specification of .TCD is not necessary. But it is recommended to specify the path completely (if necessary also .TCD). The better the specifications the faster TurboCalc can find the sheet. But: This will also complicate transferring data. Therefore, depending on the application, a "relative" specification (concerning the current sheet) will be helpful.

Note: The read value (or text) is internally buffered with sheet name and cell, so that it will only have to be read once, but: here a distinction is made between different names (with/without ".TCD", different path) - even if all names will probably point to the same data! (names are NOT case-sensitive). Therefore: If possible, always use the same spelling (e.g. simply by copying the formula). Otherwise it can happen that the same data will be read and buffered several times. (Certainly this will not cause errors, but increases the necessary memory space.)

Note: If the sheet could not be opened, an error message will still appear (if the cell is empty, 0 will be returned.)

Warning: Cells protected by a password cannot be accessed here (they must always be opened). They will be interpreted as "not found".

Important Note: In the older version of TC2 (until TC2.18) the memory routine contained a small error, which prevented the reading of values from sheets saved in this way (as soon as formulas were saved). Simple fix: Loading the sheet(s) and saving with the current version.

Examples:

If the sheet "DhO:data/Test.TCD" is open, and the value 123 can be found in cell C5, all following expressions will return this value:

Testdata/TestDhO:data/TestAT(Test;C5)

If the exact file name is e.g. "accounting January;February.TCD" quotation marks are necessary:

accounting January;Februaryaccounting January;February.TCDRelated Functions:

#Reference

1.101 CELL(Row;Column)

CELL(Row;Column)

Returns the contents of the cell whose relative position is given by parameters Row and Column.

Column and Row can be any integer number from 1 onwards. 0 addresses the current row/column. Negative numbers determine how far the cell should be located to the left/top of the sheet. Positive numbers correspond to the directions right/bottom.

During the execution of macro instructions (i.e. the parameters of these instructions), this function returns the cell relatively to the cursor position of the current window. Here, CELL(0;0) addresses the cell in which the cursor is located. (Exception: the parameter of the macro-instructions CALL and IFGOTO, see there!).

Note: Please note, that the row is followed by the column in this notation. It complies with the mathematical standard for a matrix - but in a co-ordinate system this would mean Y,X!

Example:

CELL(-1;-1) returns the value above nd to the left of the current cell

Related Functions:

CELLABS , INDIRECT

1.102 CELLABS(Row;Column)

CELLABS(Row;Column)

The function CELLABS returns the contents of the cell whose position is given absolutely with the parameters Row and Column.

Column and Row can be any positive numbers from 1 onwards (all smaller values produce the error message #VALUE).

Note: Please note, that the row is followed by the column in this notation. It complies with the mathematical standard for a matrix - but in a co-ordinate system this would mean Y,X!

Note: This function is suitable if you want to visit several cells with a loop construct - please, refer to the example of IFGOTO.

Note: If you want to specify a block, the function CELLABS can be used: e.g. CELLABS(1;3):CELLABS(7;5) - or simply use RANGEABS, see there.

Example:

CELLABS(1:1) returns the value which is located in cell A1

CELLABS(3;2) returns the value of cell B3

Related Functions:

CELL , INDIRECT , RANGEABS

1.103 CHOOSE(Index; Value1;Value2;Value3...)

CHOOSE(Index; Value1;Value2;Value3...)

The function CHOOSE returns the value1,value2..., depending on the parameter index.

Index is a positive number from 1 onwards. It determines, which of the following values is to be returned. If "index" is less than 1 or greater than the total number of values, the error message #VALUE will be returned.

Value1,2... can be any expression (texts, numbers...). Also a further CHOOSE (..;..;..) as parameter is possible.

Example:

CHOOSE(1;"hello";"you";123) is "hello"

CHOOSE(2;"hello";"you";123) is "you"

CHOOSE(3;"hello";"you";123) is 123

CHOOSE(4;"hello";"you";123) is #VALUE

CHOOSE(WEEKDAY(TODAY);"Su";"Mo";"Tu";"We";"Th";"Fr";"Sa") produces a two-letter abbreviation of the current weekday (WEEKDAY returns a number between 1 and 7!)

Related Functions:

IF

1.104 COLUMNNUMBER(Range)

COLUMNNUMBER(Range)

This function returns the number of the column in the selected range (left, top corner), or of the current cell, if no range is specified. The cell A1 corresponds to column number 1.

This function is very useful in connection with macros, or with the function CELLABS. You will find an example in chapter eight "Names - Setting Names".

Example:

COLUMNNUMBER(F3) is 6

COLUMNNUMBER(F3:H7) is 6

COLUMNNUMBER() returns the column number of the current cell.

Related Functions:

RANGEWIDTH ; RANGEHEIGHT ; ROWNUMBER ; CELLABS ; RANGEABS

1.105 INDIRECT(Text)

INDIRECT(Text)

The function INDIRECT extracts the cell reference from the parameter text.

Text: Is text (in quotation marks!) which contains a valid cell reference (or name). (The functions CELL, CELLABS and INDIRECT are possible within the text).

Example:

INDIRECT("A1") returns the cell A1

INDIRECT(A1) is F5, if the text "F5" is located in cell A1.

Related Functions:

CELL , CELLABS , RANGEABS

1.106 HLOOKUP(Value;Range;Offset;[Exact])

HLOOKUP(Value;Range;Offset;[Exact])

Horizontal search for a value within a certain range: This HLOOKUP function searches horizontally in the first row of a certain range given in the parameter list for the first cell with a content greater than or equal to the parameter value. If such a cell is found, the HLOOKUP function will return the contents of the cell which is Offset cells above (if Offset is negative) or below (if Offset is positive) the matching cell.

This can be used to read out a certain value which depends on the contents of another cell.

Value: Can be any admitted value (including Text, Date, Time or Boolean values). Texts require a complete equality with the search criterion value (not case-sensitive). Otherwise the operation is stopped at the first entry greater than or equal to the specified value.

Range: Is the range, which should be searched. (The sheet of the range, that means the first row, should normally be sorted.)

Offset: Determines how many rows above/below the matching value the cell is to be read (e.g. 1= one row below, -1=one row above).

Exact: (Can be omitted) If this parameter is given and not equal to 0 (e.g. TRUE or 1), the search is bound to be "exact", i.e. TurboCalc does not look for the first entry greater than or equal to value, but for exact equality (in this case, the parameter range need not be sorted)

Note: HLOOKUP and VLOOKUP allow the omission of the parameter Offset. If you do so, you have to skip the parameter Exact as well. Furthermore, Offset is assumed to be one.

Example:

The following expressions have all the same effect:

HLOOKUP(10,A1:F5;1)

HLOOKUP(10,A1:A5;1)

HLOOKUP(10;A1:F5;1;0)

Second Example:

A B C

1: 5 15 20 minimum quantity

2: 22 15 10 price

This small example contains different prices in row 1 for a product depending on the ordered quantity (given in row 2). If you have the current order in cell A3, you can compute the total amount of your invoice with:

(result for 3: 45, for 6: 60, for 10: 120)

=A3*LOOKUP(A3;A2:C2;1).

TurboCalc takes the quantity of cell A3, compares it with the entries in A2 to C2 and selects the appropriate price.

Further examples can be found in the file "LOOKUP.TCD" on your TurboCalc-diskette.

Related Functions:

VLOOKUP , LOOKUP

1.107 LOOKUP(Value;Range;[Exact])

LOOKUP(Value;Range;[Exact])

This function is an abbreviation for either HLOOKUP or VLOOKUP (depending on range) with an Offset of 1. For further details, see one of these functions.

Normally, a call to this function corresponds to HLOOKUP(Value;Range;1;Exact).

If the parameter range is only one column wide, VLOOKUP(Value;Range;1;Exact) is executed.

Related Functions:

HLOOKUP , VLOOKUP

1.108 RANGEABS(Row;Column;Height;Width)

RANGEABS(Row;Column;Height;Width)

The function RANGEABS defines a block with the dimensions Width and Height starting at the position Row, Column (All values have to be greater than 0 or a #VALUE-error will appear).

Note: Please note that the row is followed by the column in this notation. It complies with the mathematical standard for a matrix - but in a co-ordinate system this would mean Y,X!

Note: This function is suitable for all functions or commands where a range is needed as a parameter. It allows easy specification of user-defined ranges in any macro-instruction.

Note: You may use CELLABS to define a range as well, e.g. CELLABS(1;3):CELLABS(7;5)

Example:

RANGEABS(3;4;1;2) is the block (D3:D4).

SELECT(RANGEABS(3;4;1;2)) selects this block.

Related Functions:

CELL , CELLABS , INDIRECT

1.109 RANGEHEIGHT(Range)

RANGEHEIGHT(Range)

This function returns the height of the parameter Range. This might be very useful for some macros.

If Range is omitted, this function will return the height of the current range (the one that is marked in the current sheet). Thus, you can determine the dimensions of the current block from within a macro (together with RANGEX, RANGEY and RANGEWIDTH)

Example:

RANGEHEIGHT(A1:C5) is 5

RANGEHEIGHT(F7) is 1

Related Functions:

RANGEWIDTH ; ROWNUMBER ; COLUMNNUMBER

1.110 RANGEWIDTH(Range)

RANGEWIDTH(Range)

This function returns the width of the parameter Range. This might be very useful for some macros.

If Range is omitted, this function will return the width of the current range (the one that is marked in the current sheet). Thus, you can determine the dimensions of the current block from within a macro (together with RANGEX, RANGEY and RANGE-HEIGHT)

Example:

RANGEWIDTH(A1:C5) is 3

RANGEWIDTH(F7) is 1

Related Functions:

RANGEHEIGHT ; ROWNUMBER ; COLUMNNUMBER

1.111 RANGEX([Range])

RANGEX([Range])

This function returns the x-position of the parameter Range. This might be very useful for some macros.

If Range is omitted, this function will return the x-position of the current range (the one that is marked in the current sheet). So you can determine the dimensions of the current block from within a macro (together with RANGEY, RANGEWIDTH and RANGEHEIGHT)

Remark: This function is similar to the function COLUMNNUMBER - but with the following important difference: COLUMNNUMBER always returns the x-position of the cursor, but this need not to be identical with the upper left corner of the block!

Example:

RANGEX(A1:C5) returns 1

RANGEX(F7) is 5

RANGEX() returns the x-position of the current range.

Related Functions:

RANGEY , RANGEWIDTH , RANGEHEIGHT , ROWNUMBER , COLUMNNUMBER

1.112 RANGEY([Range])

RANGEY([Range])

This function returns the y-position of the parameter Range. This might be very useful for some macros.

If Range is omitted, this function will return the y-position of the current range (the one that is marked in the current sheet). So you can determine the dimensions of the current block from within a macro (together with RANGEX, RANGEWIDTH and RANGEHEIGHT)

Remark: This function is similar to the function ROWNUMBER - but with the following important difference: ROWNUMBER always returns the y-position of the cursor, but this need not to be identical with the upper left corner of the block!

Example:

RANGEY(A1:C5) returns 1

RANGEY(F7) is 7

RANGEY() returns the y-position of the current range.

Related Functions:

RANGEX , RANGEWIDTH , RANGEHEIGHT , ROWNUMBER , COLUMNNUMBER

1.113 ROWNUMBER(Range)

ROWNUMBER(Range)

This function returns the number of the row in the selected range (left, top corner), or of the current cell, if no range is specified. The cell A1 corresponds to row number 1.

This function is very useful in connection with macros, or with the function CELLABS.

Example:

ROWNUMBER(F3) is 3

ROWNUMBER(F3:H7) is 3

ROWNUMBER() returns the row number of the current cell.

Related Functions:

RANGEWIDTH ; RANGEHEIGHT ; ROWNUMBER ; CELLABS ; RANGEABS

1.114 SHEETNAME()

SHEETNAME()

Returns the name of the sheet, in which this formula is located, or -while in the macro mode- the current sheet name. This instruction is very useful in connection with the macro-instruction SELECTSHEET.

Example:

SHEETNAME()

In the standard sheet "Sheet1", the text "Sheet1" is returned.

1.115 VLOOKUP(Value;Range;Offset;[Exact])

VLOOKUP(Value;Range;Offset;[Exact])

Vertical search for value within the range: This LOOKUP function searches vertically in the first column of a certain range given in the parameter list for the first cell with a content greater than or equal to the parameter value. If such a cell is found, the VLOOKUP function will return the contents of the cell which is Offset cells to the right of the matching cell (if Offset is negative left of this cell).

If the search operation fails, the error message #VALUE will be returned.

This can be used to find a certain value which depends on the contents of another cell.

Value: Can be any legal value (including Text, Date, Time or Boolean values). Text requires equality with the search criterion value (not case-sensitive). Otherwise the operation is stopped, when the first entry greater than or equal to value is found.

Range: Is the range, which should be searched. (It is recommended, that the search range is already sorted at this time.

Offset: Determines, how many columns to the right/left of the found value the cell shall be read (e.g. 1= one column right, -1=one column left).

Exact: (Can be omitted) If this parameter is given and not equal to 0 (e.g. TRUE or 1), the search is bound to be "exact", i.e. TurboCalc does not look for the first entry greater than or equal to value but for equality (in this case, the parameter range need not be sorted)

Note: HLOOKUP and VLOOKUP allow the omission of the parameter Offset. If you do so, you have to skip the parameter Exact as well. Furthermore, Offset is assumed to be 1.

Example:

VLOOKUP(10,A1:F5;1)

Related Functions:

HLOOKUP , LOOKUP

1.116 Financial Functions

Financial Functions

TurboCalc has many built-in financial functions. Most of the following functions use these parameters:

Value: A fixed amount, the principal, which is a basis for calculation.

At the start of an investment, the present value (PV) is increased (or decreased) to the future value (FV).

Payment: The amount which is paid monthly...(see period).

Rate: Interest rate (e.g. 7.5% or 0.075, but not 7.5).

Term: Indicates the period of time in which interest is paid (most of the time:Years). Here, 1 corresponds to 1 year; 1/12 to a month, 1/360 to a (bank-)day.

Frequency: Determines how often during the "year" the interest or installment payment should take place. (1 means yearly, 2 half-yearly, 4 quarterly, 12 monthly...) This parameter is always the last within a functional notation and can also be omitted - in this case 1 (yearly) is assumed.

Important: If you are familiar with TurboCalc 2.0 you will notice that the names of these functions have changed: Old formulas will be automatically converted .

CFV(PV;Rate;Term[;Frequency])

CTERM(Present Value;Future value;Rate[;Frequency])

FV(Payment;Rate;Term[;Frequency])

LRATE(Present Value;Future value;Term[;Frequency])

PV(Future value;Rate;Term[;Frequency])

PMT(Future value;Rate;Term[;Frequency])

TERM(Future value;Payment;Rate;[;Frequency])

1.117 CFV(PV;Rate;Term[;Frequency])

CFV(PV;Rate;Term[;Frequency])

This function calculates the future value of a non-recurring investment using the complex interest method. You have to enter the present value, term and interest rate.

The common question which can be answered using this function is: "What amount is reached, if I invest a certain sum for a fixed term at my bank today?"

Example 1:

After the birth of a child a fixed amount of $2500 is invested in an account with a yearly interest rate of 3.5%. Calculate the resulting amount until the 18th birthday of the child!

CFV(2500;0.035;18)

Result: The principal has grown to $4643.73 in 18 years.

Example 2:

An amount of $10000 is to be invested as fixed deposit for a term of 18 months. The bank offers an interest rate of 7.5% as half-year-interest. Calculate the future amount after 18 months!

CFV(1000;7.5%;1.5;2)

Result: After 18 months the principal has grown to an amount of $11167.71.

1.118 CTERM(Present Value;Future value;Rate[;Frequency])

CTERM(Present Value;Future value;Rate[;Frequency])

The function "CTERM" calculates, how long an amount, with a known interest rate, must be invested to reach the desired future value.

Example:

Your son wants to buy a new computer. He has saved $4000. Unfortunately the desired equipment costs $4500. How long does he have to leave the money on his deposit account (interest rate 3%), to reach the necessary amount?

CTERM(4000;4500;3)

Result: Your son has to wait about 4 years until the future value is reached.

1.119 FV(Payment;Rate;Term[;Frequency])

FV(Payment;Rate;Term[;Frequency])

With this function you can determine the future value of an investment, when paying in a fixed amount at regularly intervals over a fixed term. (This is called an annuity of sinking fund.) A fixed interest rate for the whole term is assumed.

Example:

An apprentice saves a monthly amount of $200 during his three years of apprenticeship. The bank pays 6% interest monthly on his payments. 3 years later he wants to buy a used car with his savings. How much money will he have for his car after 3 years?

FV(200;6%;3;12)

Result: After his apprenticeship he can buy a car for $7867.22.

1.120 LRATE(Present Value;Future value;Term[;Frequency])

LRATE(Present Value;Future value;Term[;Frequency])

This formula calculates the interest rate of a non-recurring investment. Present and future values must be known as well as the term and frequency of interest payments.

Example:

Two years ago you invested an amount of $6000 at your local bank. At the end of the term the amount has increased to $6933.80. Because you have mislaid your contract you try to find out the correct interest rate by yourself.

LRATE(6000;69938.80;2)

Result: Your invested capital has received an interest rate of 7.5%.

1.121 PV(Future value;Rate;Term[;Frequency])

PV(Future value;Rate;Term[;Frequency])

This formula allows to calculate a necessary starting principal for a known or desired future amount. The interest rate per year, the term and frequency of interest payments must be known in advance.

You need this function e.g. if you are planning to have large expenditures in the near future. It would not be wise to keep the complete amount ready. Instead of this, only invest the amount, which results in the future amount inclusive the interest.

Example:

You want to buy a new car in 3 years time. Somehow, you know that the car will cost $30000. What amount do you have to invest to reach a sum of $30000 with an interest rate of 7.5%, within 3 years?

PV(30000;7.5%;3)

Result: Today you have to invest $24,148.82 to receive the necessary amount within 3 years.

1.122 PMT(Future value;Rate;Term[;Frequency])

PMT(Future value;Rate;Term[;Frequency])

You will need this function to find out the amount of the regularly payments to reach an already known or desired future value. It is assumed that the interest rate remains constant during the term.

Example:

A family wants to buy a new living-room furniture for about $8000 in five years. The bank offers an interest rate of 6.5% on the regularly payments. What does the family have to pay to the account to reach $8000 within five years?

PMT(8000;6.5%;5;12)

Result: The family has to pay a monthly installment of $113.20 to the deposit account.

1.123 TERM(Future value;Payment;Rate;[;Frequency])

TERM(Future value;Payment;Rate;[;Frequency])

With this function you can find out, how long you will have to pay a regular amount to get the desired future value. Interest rate, payment amount and the frequency of the payment of the payments must be known in advance.

Example:

A dream of your life is a frightfully expensive sports car for $80000. Because you do not have the money at your disposal right now, you decide to save a fixed amount of $500 monthly, until the principal has reached the required amount including the annual interest of 6,5%. How long will you have to pay?

TERM(80000;500;6.5%;12)

Result: You will have to be patient for another 10 years, before you can afford the car.

1.124 Miscellaneous Functions

Miscellaneous Functions

CELLINFO(Nr[;Cell])

DEMOVERSION()

FILEEXISTS(File)

INFO(Nr)

LASTERROR()

OBJID(text)

OBJECTINFO(Name;Nr)

REVISION()

SETxxx(Condition;Value1;Value2[;Reference])

TCFUNCTION(TCLib;Offset;Num1;Num2;Text)

VERSION()

1.125 CELLINFO(Nr[;Cell])

CELLINFO(Nr[;Cell])

This functions gives specific information about a certain cell.

Nr indicates the type of information required.

Cell determines the cell, in which the information exists. Can be omitted. Then the cell, in which the formula exists, will be used.

Nr can be one of the following values:

0 content of the cell (*always* as text)

1 formula (as text)

2 Data type (O: empty cell, 1: floating point number, 2: integer, 3: date, 4: time, 5: boolean, 6: text, 7: cell reference, 8: error) (7 will normally not be possible)

3 number format

4 text color

5 background color

6 pattern (0-15)

7 text style (1: underlined, 2: bold, 4: italic - and their sums)

8 horizontal alignment (0: default, 1. left, 2: mid, 3: right)

9 vertical alignment (0: default, 1: top, 3: mid, 2: bottom)

10 frame (2 bit (0-3, out-max) each page-> 0-255!)

11 protection flags (1: protected, 2: hide formula - sum!)

12 column width (here the column in which the indicated cell lies will be used)

13 column height (here the row in which the indicated cell lies will be used)

For examples see example sheet of the same name.

1.126 DEMOVERSION()

DEMOVERSION()

This function returns TRUE if the current TurboCalc-Version is a demo version, otherwise FALSE (so if you use this function in your commercial version, you should read FALSE).

This function can be used to check the environment when writing macros.

1.127 FILEEXISTS(File)

FILEEXISTS(File)

This function checks if the indicated file exists and returns a corresponding TRUE (if existing) or FALSE result.

File is a text string holding the file name.

Note: Although this is a function, it should only be used in connection with macros (see example). Otherwise at least "automatic recalculation" should be deactivated. If this is used as normal function in the sheet, it will be checked during every recalculation whether the file exists and this will retard the recalculation process.

Warning: No requesters with "Please insert ... diskette" will appear (if a drive does not exist, e.g. at "XXX:tes"), but in this case FALSE will immediately be returned.

Example:

=IFGOTO(NOT(FILEEXISTS("test.tcd"));A20)

=LOAD("test.tcd")

This fragment checks whether the file "test.tcd" exists. If this is the case, it will be loaded, otherwise it will branch to A20 for error handling.

1.128 INFO(Nr)

INFO(Nr)

This function gives you some interesting values:

Nr is one of the following numbers:

0 number of opened sheet

1 number of opened sheet windows

2 number of opened and visible sheet windows (the hidden windows are not counted)

3 free chip memory

4 free fast core

5 total free memory

1.129 LASTERROR()

LASTERROR()

Returns the error number of the last error. Especially useful for error handling routines. For further details see macro instruction ONERROR().

1.130 OBJID(text)

OBJID(text)

This function calculates the appropriate object type-ID for the instruction OBJECT from text (length 1 to 4) and will allow a better selection of the object type.

Text: Text from which the object type-ID will be calculated, only the first four characters are significant but case-dependent.

(For insiders: The function sums up the ASCII-values one after another, multiplied with 256ˆ3, 256ˆ2 or 256).

Example:

OBJID ("TEXT") - see also OBJECT-Macro.

1.131 OBJECTINFO(Name;Nr)

OBJECTINFO(Name;Nr)

This function gives information about certain parameters of an object.

Name: The name of the object. If the object does not exist, the error message "VALUE" will appear.

Nr: Determines the type of information, one of the following numbers:

0 Name (Not very helpful because the name must be indicated, only exists for completeness).

1 macro-text

2 macro active (TRUE/FALSE)

3 background color (0=off, 1=color 0 ...)

4 frame (0=off, 1=normal, 2=3D, 3=3D-2)

5 activated (TRUE/FALSE)

6 x-position

7 y-position

8 width

9 height

1.132 REVISION()

REVISION()

This function returns the revision of the current TurboCalc-Version (the number after the version number and the decimal point).

This function can be used to check the environment when writing macros.

1.133 SETxxx(Condition;Value1;Value2[;Reference])

SETxxx(Condition;Value1;Value2[;Reference])

This functions allows you to set/change the formatting of the current or other cells depending on the given condition. So it is very simple to change the sheet automatically, corresponding to the input (without macro programming).

xxx can be replaced by the following options:

SETALIGNMENT changes the current alignment (left, right)

SETCOLOR changes the color of the font

SETFORMAT changes the numeric format

SETSTYLE changes the font style

Condition: is a boolean value. If it is TRUE (or not 0) the respective formatting is set to Value1, otherwise to Value2. is a boolean value. If it is TRUE (or not 0) the respective formatting is set to Value1, otherwise to Value2.

Value1, value2 are from the following allowable values to which the respective style feature is set according to the condition:

* Alignment: 0=standard, 1=left, 2=right, 3=centered

* Color: 0=standard color, 1=color1...(also refer to macro-instruction COLORS)

* Format: 0=standard, 1...specifies the respective numeric-format, refer to macro-instruction NUMERICFORMAT

* Style: 0=normal, 1=underlined, 2=bold, 4=italic (several styles together are also possible, refer to macro-instruction FONT).

Reference: Specifies the cells for which the formatting should be determined. If this parameter is missing, the current cell will be affected.

All functions return the value 0, if the operation was finished successfully. Otherwise (if the target cell is empty!) an appropriate error message will appear.

Note: Because of the minor importance of the return parameter, you can ignore it as follows: Simply add this formula to the former cell contents (because 0 is returned, nothing is changed). Also refer to the first example below.

Example:

A1+SETCOLOR(A1>=0;5;6) shows the value of cell A1 in the current cell. If the value is positive it will be shown in color 5, otherwise in color 6 (this can very useful to achieve a special emphasis for e.g. negative income).

SETCOLOR(A1>0;5;6;A1) see above, but the color of cell A1 will be changed directly and 0 is returned to the current cell.

A1+SETALIGNMENT(A1>=0;2;1) prints the contents of A1 right aligned if it is positive, otherwise left aligned.

SUM(A1:A10)+SETSTYLE(SUM(A1:A10)>2000;3;4) calculates the result of cell A1 to A10 and prints it bold and underlined if it is greater than 2000, otherwise the style is set to italic.

1.134 TCFUNCTION(TCLib;Offset;Num1;Num2;Text)

TCFUNCTION(TCLib;Offset;Num1;Num2;Text)

Here you can integrate and call external functions. This is a simple method of expanding TurboCalc functionality. TurboCalc libraries will be used for this purpose. Further details in the corresponding paragraph of the chapter "Projects".

TCLib: This is the name of the TurboCalc-Library (as text).

Offset: Indicates the function number (thus several functions can exist in one library).

Num1, Num2, Text: These are the possible parameters of the function (Num1 and Num2 are numerical values, text is a text).

The appropriate library will define precise functional details, parameter settings and return value. If you want to create your own library, you will find details in the above-mentioned section about TCLibs.

Note: If you want to call external macro instructions, use TCMACRO.

1.135 VERSION()

VERSION()

This function returns the version of the current TurboCalc-Version (the number before the decimal point and the revision number).

This function can be used to check the environment when writing macros.

1.136 Table of Contents

Table of Contents

1.137 Index

WEEKDAY(Date)

X

XOR(Value1;Value2;...)

Y

YEAR(Date)