Basics about MAR Cal

- MAR Cal allows input of mathematical expressions in plain text
- MAR Cal uses the decimal point for numbers
- Finalise your input with the return key
- Input is automatically stored and can be used again in another session
- A mouse-click brings back stored lines into the input fields

The following rules apply to arithmetic operations:

- Use **round brackets**:

`(5 + 10 + 20) - (10 - 15)`	results in: 40
`(5 + 10) * 2`	results in: 30
`(2.5 * (5 - 21 / 7)) / 4`	results in: 1.25
`(5 - 2 ^ 2) + ( - 2 ^ 2 + 5 ) + 8.5`	results in: 10.5
`(5 - 2 ^ 2) - ( - 2 ^ 2 + 5 ) + 8.5`	results in: 8.5
`lg ( lg ( 3 + 2 * sqr ( sin(x) ) ) )`	results in: -.1905 für x = 30 Grad

- Addition	`4 + 3`	results in: 7
- Subtraction	`4 - 3`	results in: 1
- Multiplication	`5 * 1.5`	results in: 7.5
- Division	`12 / 5`	results in: 2.4
- Integer division	`12 \ 5`	results in: 2
- Factorial	`4 !`	results in: 24

- Exponentiation: to raise a number to a power, use the < ^ > as operator character:

`x ^ 2.3`	is x raised to the power of 2.3
`x ^ (2-a)`	is x raised to the power of (2-a)

Note that the ^ key shows a visible result only in combination with a second keystroke.

- Square root: use **SQR(x)**

`SQR(4)` is the square root of 4 and the result is 2.

Instead of using the standard function < **SQR** >, you could alternatively use either of the following inputs:

```
4^0.5
4^(1/2)
```

Number symbols (name of variables or constants)

Number symbols can consist of characters and numbers. The length of names is not restricted. Symbols must start with an alphabetic character A-Z. No special characters, except the underscore ("_"), are accepted:

`AbcX * 5`	is AbcX multiplied by 5	<AbcX is a **number symbol**
`A * 5`	is A multiplied by 5	<A> is a **number symbol**
`A_123 * 5`	is A_123 multiplied by 5	<A_123> is a **number symbol**

`123A * 5`	123 multiplied by the number symbol <A> multiplied by 5. For A=2 the result would be 1230.

Applying the scientific format

You can use the scientific format, for example:

`1.2E3 + 5.5`	delivers as result 1205.5
`w1 = (2.5e-3)*2`	delivers as result w1 = .005
`w2 = 3.0e5 / (1.5E6)`	delivers as result w2 = .2
`Abc = (3! + Lg(100) + 2 * sin(30)) ^2 + 1.5E-2)`	delivers as result: Abc=81.015

Standard functions and predefined names

You cannot use predefined names, the character E or the names of standard functions as names of number symbols (variables or constants).

`PI` = 3.14159265...
`Exp(x)` is e raised to the power of x, with e = 2.71828183...
`< ! >` is the factorial character. Example: 3! = 3*2*1 = 6

`ln(x)` is the logarithm of x to the base e
`lg(x)` is the logarithm of x to the base 10
`lb(x)` is the logarithm of x to the base 2

`sin(x)` is the sine of x
`cos(x)` is the cosine of x
`tan(x)` is the tangent of x
`cot(x)` is the inverse tangent (cotangent) of x

When working with trigonometrical functions, the angle **input** can be either in **degrees** or in **radians**. The default setting is degrees. You can change from degrees to radians with the **Menu <Settings>**:

`arcsin(x)` is the arcsine of x
`arccos(x)` is the arccosine of x
`arctan(x)` is the arctangent of x
`arccot(x)` is the arccotangent of x

`sinh(x)` is the hyperbolic sine of x
`cosh(x)` is the hyperbolic cosine of x
`tanh(x)` is the hyperbolic tangent of x
`coth(x)` is the hyperbolic cotangent of x

`arsinh(x)` is the inverse hypberbolic sine of x
`arcosh(x)` is the inverse hypberbolic cosine of x (defined for x >= 1)
`artanh(x)` is the inverse hypberbolic tangent of x (defined for |x| < 1)
`arcoth(x)` is the inverse hypberbolic cotangent of x (defined for |x| > 1)

Use apostrophe < ' > for Comment

You can add a comment to your formulae and symbols. To initiate a comment, use the single **apostrophe < ' >.** You can place any number of blank characters before or after the apostrophe:

```
a = 2.4 + 0.65 + 1.9          ' a is length of room 1
b = 4.5 + 1.0 + 0.5 + 1.1     ' b is width of room 1
```

The whole input line including the comment is stored in the formula storage field and the result

- a = 4.95 and
- b = 7.1

is stored in the constant storage field.

Automatic Storage

MAR Cal automatically saves input in the formula storage field. In the constant storage field, the result of a calculation is stored. They can be returned to the input field with a mouse click. In the above example, the calculated constants a and b are in the constant storage field and you can use them for further calculations. For example:

```
A = a * b   ' A is area of room 1
```

The new input is saved in the formula storage field. In the constant storage field, the new result is stored:

- A = 35.145

Equation or formula

Generally, you can enter your input in the form of an equation or directly as a formula. For example, if you wish to compute **`< 4+5 >`**, you have the following 2 options:

- `y = 4 + 5` the input is an **equation**
- `4 + 5` the input is a **formula**

The difference between the two lies in the storage method.

When using an **equation**, the equation itself is stored in the formula storage field and the result of the computation is stored in the constant storage field.

On the other hand, by entering a **formula**, the formula is stored in the formula storage field but the result will not be saved.

Button < Define variable >

You can declare a number symbol as a variable and generate pairs of values for this variable. Let's look at the function, which should be in the formula input field:

```
x^2 + x - 1
```

Assume that we wish to obtain the solution for the range from 1 to 5 for x, with a step width of 1. To get pairs of values, instead of a single solution, just click on the button <Define variable>. A pop-up window comes up, in which we enter the name of the variable, its start value, end value and the step width. For our example, the input is:

Variable	x
Start value	1
End value	5
Step	1

As we have already entered the formula, a click on the **push-button <Run!>** in the same window starts the computation. MAR Cal outputs the pairs of values calculated in a separate window.

Button < Run! >

Starts the calculation. Alternatively, you can finalise your input with the return key.

Button < Clear input >

Clears the input line.

Button < Group >

Data in the input storage field as well as the constant storage field can be grouped. To form groups, enter numbers in the comment part of the input. Blank characters after the initial apostrophe or the grouping number are allowed, provided that you enter the number before any additional input. Otherwise, MasterAllRound interprets the number as a comment. Clicking on the <Group> button sorts the group. The same number can be used several times.

Button < Line up >

Moves the highlighted line in the input storage field one line up.

Button < Line down >

Moves the highlighted line in the input storage field one line down.

Button < Delete line >

Deletes the highlighted line of the input storage field.

Button < Undo >

Brings the deleted line back into the input storage.

Button < Edit input storage >

Opens a window to edit the input storage field. In this window you can edit, i. e. modify, delete or add input.

Lines marked for grouping are automatically grouped, stored and written back into the storage field when leaving the window with the button **< Save >**. Using the **< Quit >** button exits the window without saving the modifications. You can choose the kind of grouping via the menu **< Settings >**. Lines not marked for grouping are kept in the original order.

Button < Edit constant storage >

Opens a window to edit the constant storage field. In this window you can edit, i. e. modify, delete or add input.

Lines marked for grouping are automatically grouped, stored and written back into the storage field when leaving the window with the button **< Save >**. Using the **< Quit >** button exits the window without saving the modifications. You can choose the kind of grouping via the menu **< Settings >**. Lines not marked for grouping are kept in the original order.

Menu < ShowList >

When having generated pairs of values you can obtain the maxima and minima of a function via the **menu <ShowList>**. To list them, you have two options:

- Menu <Max-min values of points>
- Menu <Max-min values of curve>

For example, for

```
        sin(x)
```

with x as variable and a step width of 14 degrees running from -111 to +111,

Max-min values of points would list the following values:

```
        -97         -.9925
        85          .99619
```

Max-min values of curve would list the following values:

```
        -90.00      -1
        89.9999     1
```

Menu < Settings >

< Reset window > resets the look back to the default.

< Group ascending, top >: First the groups are ordered ascending followed by the rest of the lines.

< Group descending, top >: First the groups are ordered descending followed by the rest of the lines.

< Group ascending, bottom >: The groups are ordered ascending after the non-grouped input lines.

< Group descending, bottom >: The groups are ordered descending after the non-grouped input lines.

Menu < @Sub-equation >

To work with sub-equations means **split a complex term into several parts**. Working with sub-equations makes sense when an equation is relatively complex built.

Let's assume that you wish to analyse the following equation, which is a special solution of the crack growth equation:

```
Nf = ( ai ^ (1-m/2) - aj ^ (1-m/2)) / (( C * ( Y * dS * PI ^ .5) ^ m ) * ( m/2 - 1 ))
```

You could lay out this term more clearly by using three sub-equations on the right side of the equation, namely:

```
P1 = ai ^ ( 1-m/2 )
P2 = aj ^ ( 1-m/2 )
P3 = ( Y * dS * PI ^ .5 ) ^ m
```

To enter the sub-equations, click on the **menu <@Sub-equation>**. In the pop-up window enter the input, which you finalise with the return key. Having entered all sub-equations, close the window via the buttom < Close > Now key in the following equation, which you also finalise with the return key:

```
Nf = ( P1-P2 ) / (( C*P3 )*( m/2-1 ) )    '@ ### Crack growth equation ###
```

Note the use of the
< @ >
sign. To notify MAR Cal of using sub-equations, a < @ > - sign must appear somewhere after the comment sign < ' >. You also need to declare the constants ai, aj, m, C, Y and dS before you start the calculation.

Menu < SmartCalculator >

Brings up the Smart calculator, the smart version of the universal calculator. Here, you can define number symbols to use subsequently in the matrix interface.

MAR Cal folders and files

MAR Cal always creates a folder called < **MARUser** >. In this folder all entered formulae, equations and number symbols (constants) are automatically saved. You can delete these files. In this case, a new <MARuser> folder is created automatically at the next program start - however it is then empty.