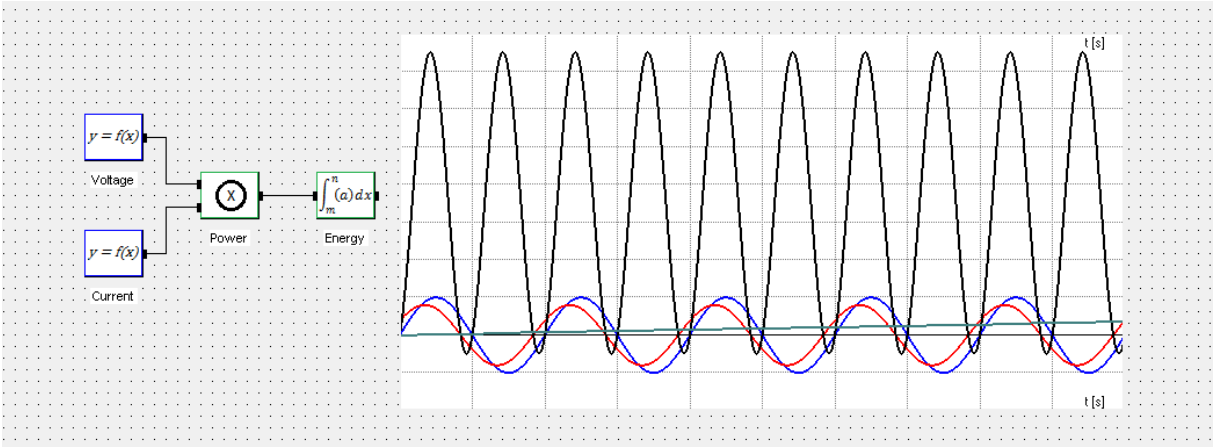
User Manual



Brace*Calc* Calculation Studio 4

Contents

1 Overview

1.1 General

BraceCalc is an independent software. It provides graphic presentations of mathematical calculations
BraceCalc runs with Microsoft Windows 7, 8, 10.

Main parts:

- Windows program with ribbon menu.
- Schematic object view.
- Curve object view with zoom and cursors.
- Function definition with ***BraceCalc*** script.
- Mathematical methods.
- Digital signal analyse
- Analog and digital filter calculations.

1.2 Support

Customer support and technical support for ***BraceCalc***:

Eicher Engineering
Software Development
Frigadenstrasse 23
CH-8739 Rieden SG
Switzerland

www.bracecalc.ch
brace@eichereng.ch

1.3 Installation, Uninstallation

BraceCalc is delivering as a Microsoft Installer Format file MSI and it can be installed on every Windows PC.
The uninstallation happens in the system control <Programs uninstallation>

1.4 Manufacture Information

1.4.1 Trademarks

Windows 7, Windows 8, Windows 10 are registered trademarks of the Microsoft Corporation.

1.4.2 Copyright

It's not allowed to copy or change parts of this software. Please read the licensing agreement.

1.5 Licensing Agreement

The use of ***BraceCalc*** is governed by the following terms and conditions:

Acceptance of License Agreement
You should carefully read the following terms and conditions before using ***BraceCalc*** (the 'Software'). Unless you have a different license agreement signed by Eicher Engineering, your use of this software indicates your agreement to these terms and conditions.
If you do not accept all of these terms and conditions, you must cease using the Software immediately.

Copyright
Customer acknowledges that the Software, License Key and accompanying user documentation
('Documentation') are copyrighted works owned by Eicher Engineering and
that Customer has no rights in the foregoing except as expressly granted herein.

Free trial
Eicher Engineering hereby grants you a 30 day license to use the Software free of charge without a valid License Key. The 30 day license is without any Software limitations.
Use of the Software more than 30 days without a valid License Key is a violation of Swiss and international copyright laws

License Key

A unique key that will allow you to use the Software without any limited functions may be purchased from the Eicher Engineering website,
which is currently http://www.bracecalc.ch. License Keys may be purchased for single or multiple users, and are priced according to the price list on the ***BraceCalc*** website.
Upon purchase of a License Key, Eicher Engineering hereby grants you a nonexclusive, non-transferable license to use the Software as follows:

Personal License Key:

A single-user License Key may either be used by a single user who uses the Software personally on one or more computers, or installed on a single computer used by multiple people, but not both.

Multi-User License Key:

A multiple-user License Key may be purchased based upon either (a) the number of users who have access to the Software on any number of computers,
or (b) based upon the number of computers on which the Software will be installed for use by any number of users. You may not use the Software in excess of either (a) the number of purchased users or (b) the number of purchased computers, whichever is applicable. A Multi-User License Key may not be used by your subsidiary companies, customers, or any other third parties.

You shall be responsible for maintaining the License Key in a safe location and are specifically prohibited from distributing the License Key, intentionally or unintentionally, to any third party. Upon loss or distribution of the License Key, Customer shall be required to pay a reinstatement fee at Eicher Engineering's discretion.

Distribution of the Software

Provided that you do not copy or distribute the License Key, and you include a copy of this License Agreement, you may (a) make copies of the Software; (b) give exact, unmodified copies of the Software to anyone; and (c) distribute the Software in its unmodified form via electronic means. You are specifically prohibited from charging any fees for any such copies or distributions.

Term and Termination

You may continue to use the Software for as long as you comply with the terms and conditions of this License Agreement. Eicher Engineering may terminate this License Agreement immediately upon notice to you in the event that Eicher Engineering has reason to believe you have breached this License Agreement. Upon termination, you shall immediately cease all use of the Software, License Key and Documentation and shall not be entitled to a refund of any fees paid.

Governing Law

The validity and interpretation of this Agreement shall be governed by the laws of Switzerland and the Kanton of St.Gallen. Customer agrees that the federal and state courts located in Switzerland and Kanton St.Gallen, shall be the appropriate site of venue for actions relating to this Agreement, and hereby consents to the exclusive jurisdiction and venue of such courts.

Disclaimer of Warranty

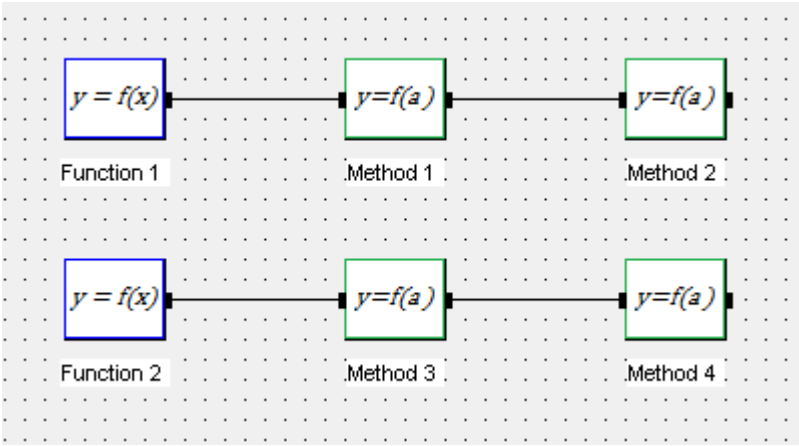
THE SOFTWARE AND DOCUMENTATION ARE PROVIDED 'AS IS.' TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, EICHER ENGINIEERING DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. ANY LIABILITY OF EICHER ENGINEERING WILL BE LIMITED EXCLUSIVELY TO REFUND OF THE PURCHASE PRICE.

2 Introduction

2.1 General

BraceCalc allows you to calculate, display, analyze and print mathematical aspects.

Mathematical calculation can be separated in functions and methods.



A mathematical term in ***BraceCalc*** is a function definition with one or none variable.

$y = f(x)$ z.B. $x - x^2 + 4$, $\sin(x)$, 5, etc.

This function definition calculates ***BraceCalc*** and save the results as a number pair buffer {x, y}.

A method can execute to such a buffer. The result is a new buffer.

$$\text{Function1 } \{x_n, y_n\}\ n=0\ldots k \xrightarrow{\text{Method}} \text{Function2 } \{x_n, y_n\}\ n=0\ldots k$$

The mathematical terms and functions will be defined with the simple ***BraceCalc*** script language.

2.2 BraceCalc Objects

2.2.1 General

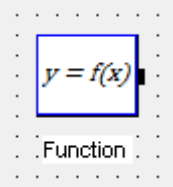
The main parts of a ***BraceCalc*** file and its calculations are the ***BraceCalc*** objects.
There are four ***BraceCalc*** object types:

- Source function
- Method
- Digital filter method
- Analog filter
- Auxiliary object

Depending on the type, an object can stand alone or within a calculation chain.

2.2.2 Function

Function objects are source objects and belong always at the first place of a calculation chain. Therefore they don't own any inputs. They can be used as single objects.

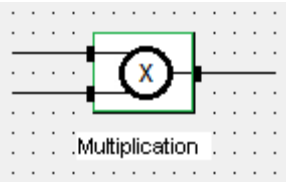


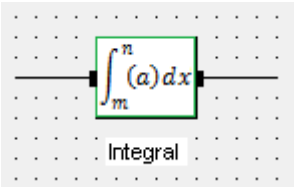
Examplel

```
Term = sin(x * 2 * [Pi]);
```

2.2.3 Method

Method objects own always one input and one output.
Therefore these objects aren't at the first place of a calculation chain.
They also can't be a single object.





The method object type is given from the definition. Therefore it doesn't need a script definition coercively like the multiplication method for example.

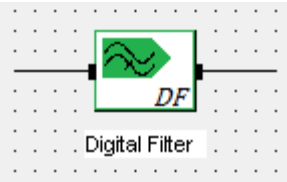
There are optional parameters in every object.

Example

```
begin = 1;
end   = 7;
```

2.2.4 Digital Filter Method

Digital filter method shows the Bode diagram additionally to the normal method.
This kind of method object can be defined as single object.



Example

```
sampling = 1000;
az.0 = 6.923E-4;
az.1 = 2 * 6.923E-4;
az.2 = 6.923E-4;
bz.0 = 1;
bz.1 = -1.937;
bz.2 = 0.94;
```

2.2.5 Analog Filter

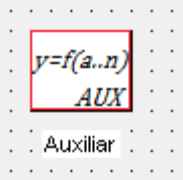
This object type can only be defined as single object.
Also the Bode diagram can be shown.

Example

```
threshold = 1000;
form = lowpass;
as.0 = 1.2872;
bs.0 = 0.4142;
```

2.2.6 Auxiliary Objects

This single objects can further calculate other and not connected objects.
For example, a whole filter curve can be calculated from some single filter objects.



Example

```
Term = bode{1} * bode{2};
```

2.3 BraceCalc Script

Definitions and object properties are defined with the simple ***BraceCalc*** script language. The word ´language´ is a little bit overacted. It is a simple assignment from values to predefined key words.

There is a text field for the script in every object detail window.

General provision:
[***BraceCalc*** Property] = [Value, Expression, Constant, Name]

Line comments are possible with ´//´. These won't consider during calculation.

BraceCalc script works from top to down. Constants, function ranges, function parts etc. have to defined in the right order.

Examples:

```
Term   = 2 * x;         begin = 0; end = 1;
Term   = 2 * (-x) + 4; begin = 1; end = 2;
Repeat = 5;

R    = 100;
C    = 1E-6;
FA   = 2000; // entire steps
FG   = 1 / 2 / [PI] / ([R] * [C]) / [FA];
K    = 1 / tan([PI] * [FG]);
az.0 = 2 / (1 + [K]);
bz.1 = (1 - [K]) / (1 + [K]);
```

2.4 Basis of Calculation

2.4.1 General

BraceCalc calculates all values in the double (64 bit floating) format.
Every object have got its own values buffers.
The buffer length depends directly of the step definition in the calculation menu.

Beware step values above 100000. Such large calculation can bring your PC to the limit.

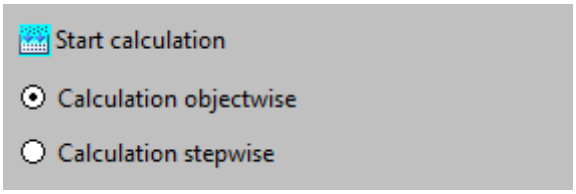
The range of the calculation is defined also in the calculation menu.

There are two kinds of calculations

- Object wise calculation
- Step wise calculation

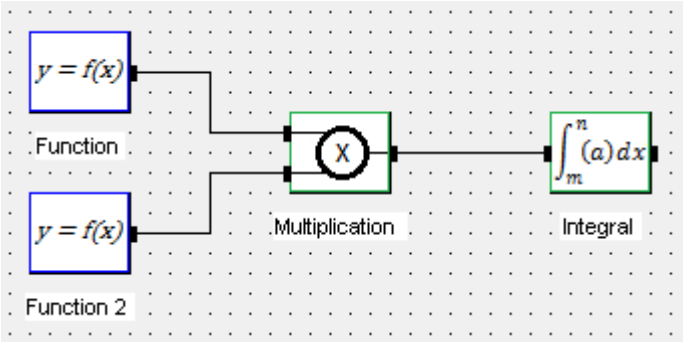
These kinds result from the object type and the calculation chain.

The calculation kind and the calculation itself are defined in the calculation menu.



2.4.2 Object Calculation

Every object will calculate for itself in order of the signal path.

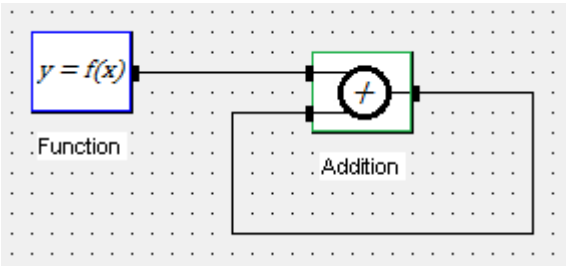


Calculation order:

a) Function
b) Function 2
c) Multiplication from the output buffers of function and function 2.
d) Integral from the output buffer of the multiplication.

2.4.3 Step Calculation

A feedback needs coercively a step wise calculation.



Calculation order:

a) Value 1 of function in the output buffer.
b) Value 1 of a) makes value 1 of the addition input buffer A.
Addition input Buffer B is the value 1 of addition output buffer (still 0).
c) Addition output buffer is input value A1 plus input value B0.
d) Value 2 of function in the output buffer.
e) Value 2 of d) makes value 2 of the addition input buffer A.
Addition input Buffer B is now the result of the addition output buffer.
f) Addition output buffer is input value A2 plus input value B1.
g) Value 3 of function in the output buffer.
h) Value 3 of g) makes value 3 of the addition input buffer A.
Addition input buffer B is now the result of the addition output buffer.
i) …
j) …
k) …

2.5 Scopes

2.5.1 General

The draw of the calculated buffers as a graphical curve is one of the main parts of ***BraceCalc***.

Every calculated buffer could be draw and print as curve.

We use four different curve types.

- Function curves from the output buffers.
- Spectrum curves from the output buffers.
- Bode amplitude curves from filter object buffers.
- Bode phase curves from filter object buffers.

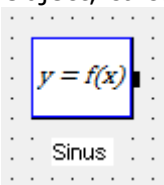
The range and style are defined in the calculated menu.

2.5.2 Function Curves

The general draw is the direct image of the value buffer. This means, the resolution and the range depend directly from the value in the calculation menu.

Example sine curve

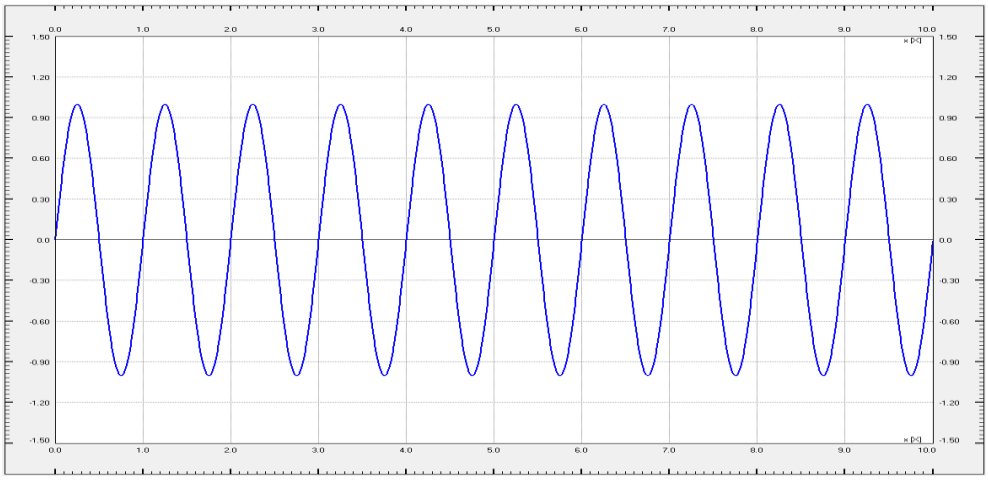
Object, calculation steps, range



Script

```
Term = sin([2Pi] * x);
```

Curve



2.5.3 Spectra

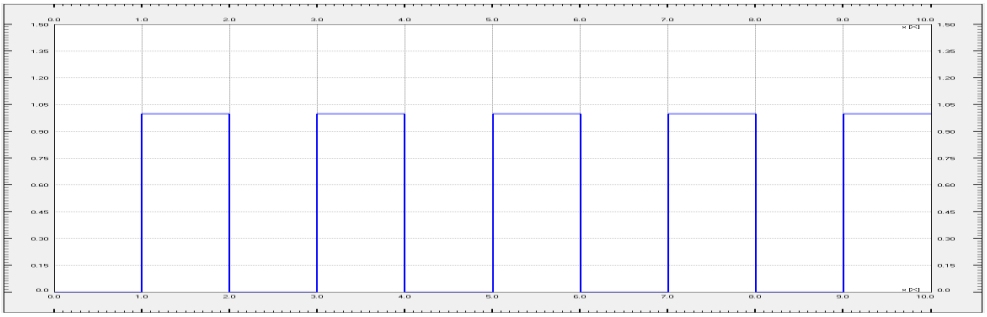
Every function curve can be draw as a spectrum.
The script of the function object needs the FFT parameter which is set with the maximum frequency of the spectrum.

Example rectangle

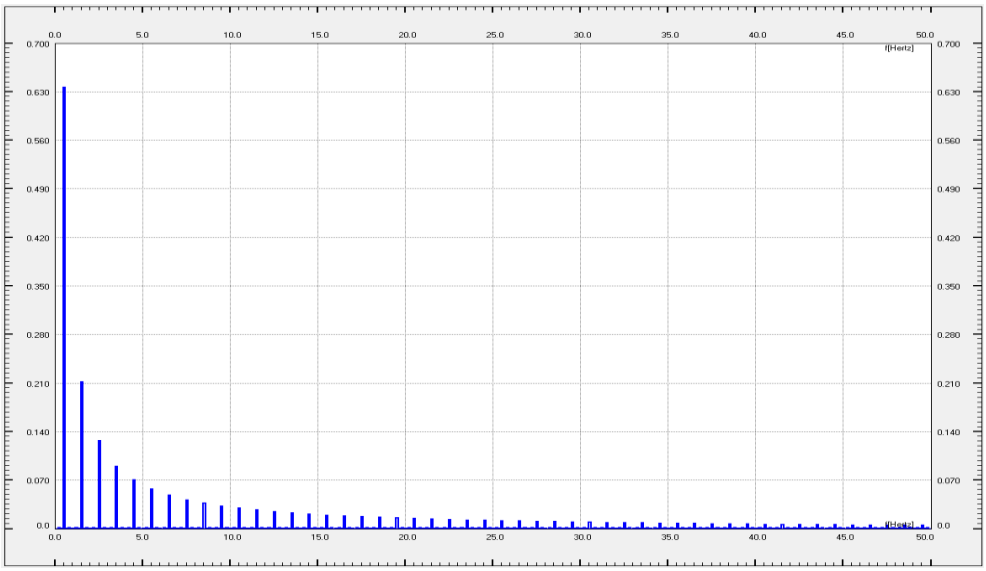
Script

```
Term = 0; begin = 0; end = 1;
Term = 1; begin = 1; end = 2;
Term = 0; begin = 2; end = 3;
Term = 1; begin = 3; end = 4;
Term = 0; begin = 4; end = 5;
Term = 1; begin = 5; end = 6;
Term = 0; begin = 6; end = 7;
Term = 1; begin = 7; end = 8;
Term = 0; begin = 8; end = 9;
Term = 1; begin = 9; end = 10;
FFT = 100;
```

Curve



Spectrum



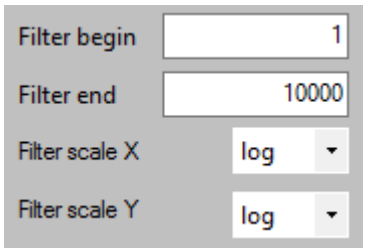
Attention:
The calculation time can be very long with too large maximum frequency.

2.5.4 Bode Diagram

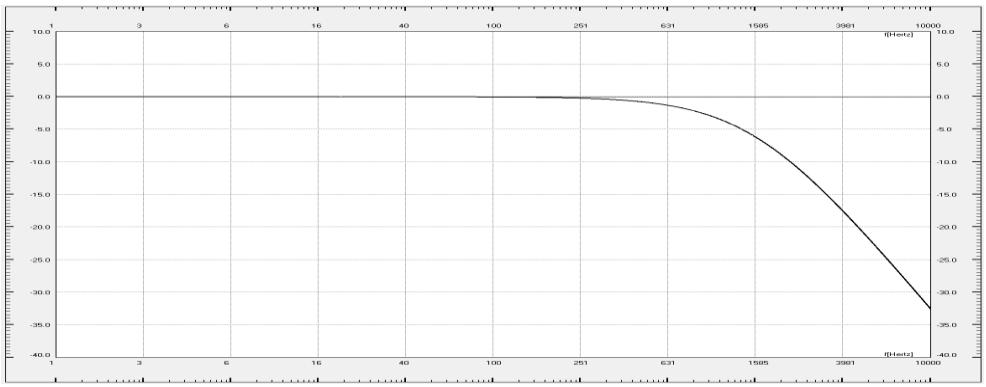
The calculation buffer of the digital and analog filter objects could be draw as Bode diagrams (amplitude and phase).
Range and scale are defined in the calculation menu.

Example analog low pass filter second order

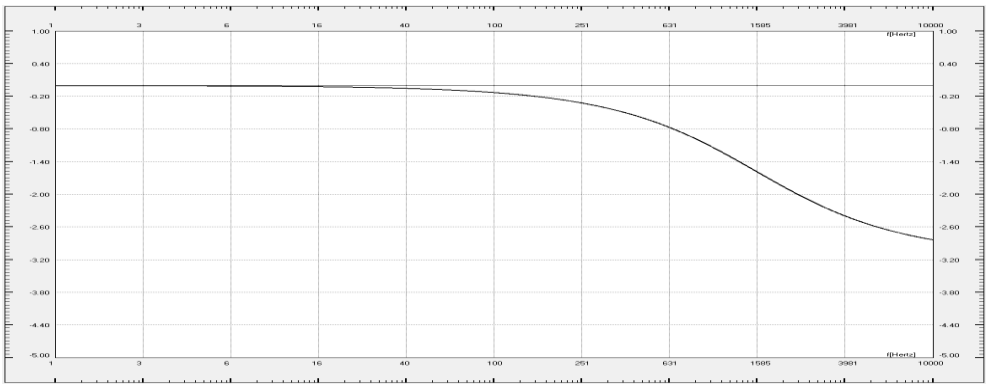
Object, filter range and scale



Bode diagram amplitude

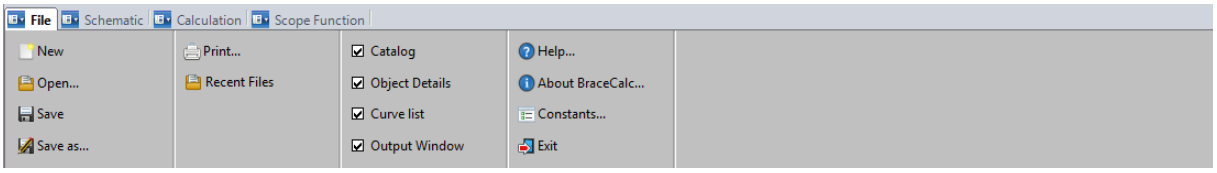


Bode diagram phase



3 References

3.1 File



Every ***BraceCalc*** files own the file ending BCX. The format based on a XML format.
It contains ***BraceCalc*** types and elements.

Attention:
Please don't change the *.bcx files directly into an editor.
The file can be damage and corrupt.

3.1.1 New (CTRL-N)

Creates a new ***BraceCalc*** file.
It gets these default properties:

- Name Untitled.bcx
- One object Mathematical source
- Function Sine
- Range 0…10, linear
- Steps 20000

3.1.2 Open… (CTRL-O)

Opens an existing ***BraceCalc*** file.
A still open file has to close first.

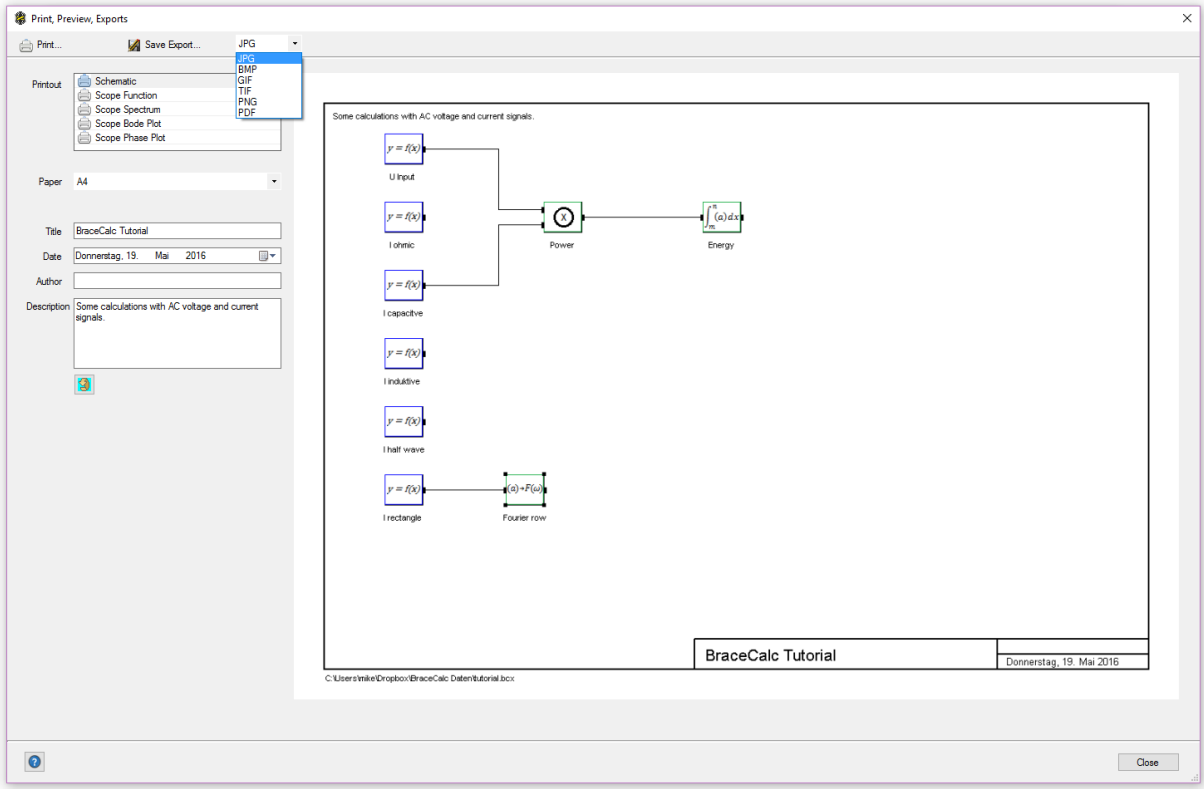
3.1.3 Save (CTRL-S)

Saves the current open ***BraceCalc*** file.
A read only file requests a new file name.

3.1.4 Save as…

The current open ***BraceCalc*** file will be save with a new file name.
For that, the <Save as…> Windows dialog appears.

3.1.5 Print... (CTRL-P)



The print dialog window contains the direct print preview, some user specific data.

All print types are similar with the scope and schematic views in the ***BraceCalc*** main window.

- Schematic
- Function curves
- Spectrum
- Bode diagram amplitude
- Bode diagram phase

Additional the print preview can also save as a graphical file into the follow formats.

- JPG
- BMP
- GIF
- TIF
- PNG
- PDF

3.1.6 Last files

A pop up window opens with the list of the last open ***BraceCalc*** files.

3.1.7 Docking Window Catalog

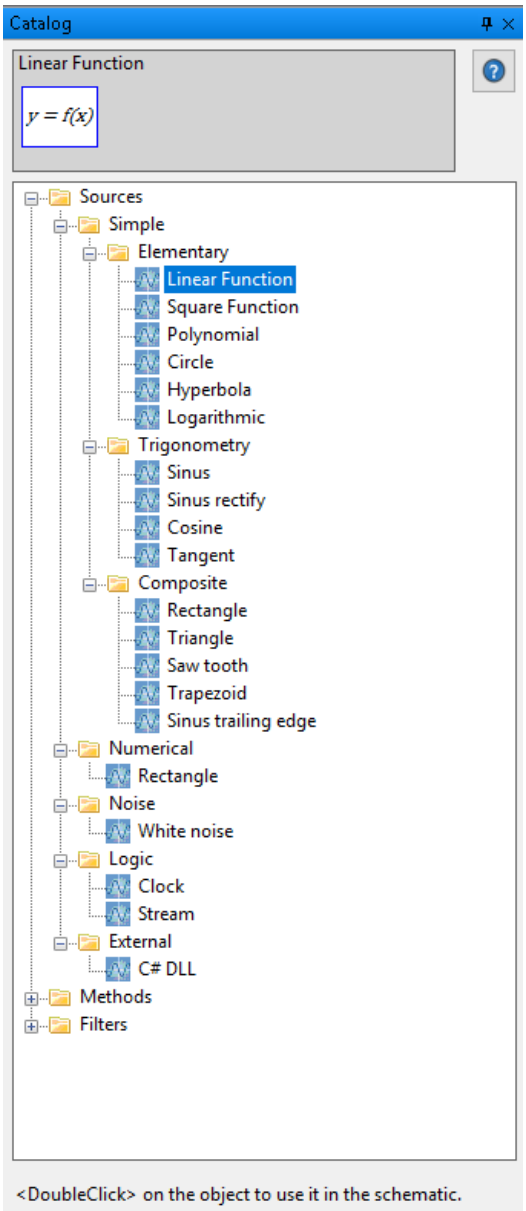
The catalog window provides a large selection of different predefined ***BraceCalc*** objects. You can include these objects directly with a double click on it.

The catalog contains three parts.

- Sources
- Methods
- Filters

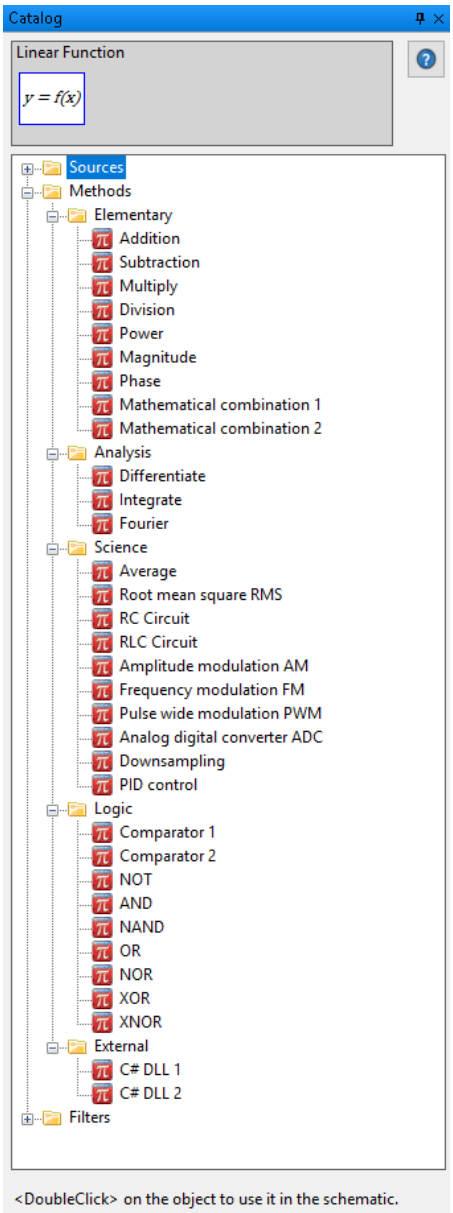
3.1.7.1 Sources

All source objects contain one output and no input.
They could use as single objects.



3.1.7.2 Methods

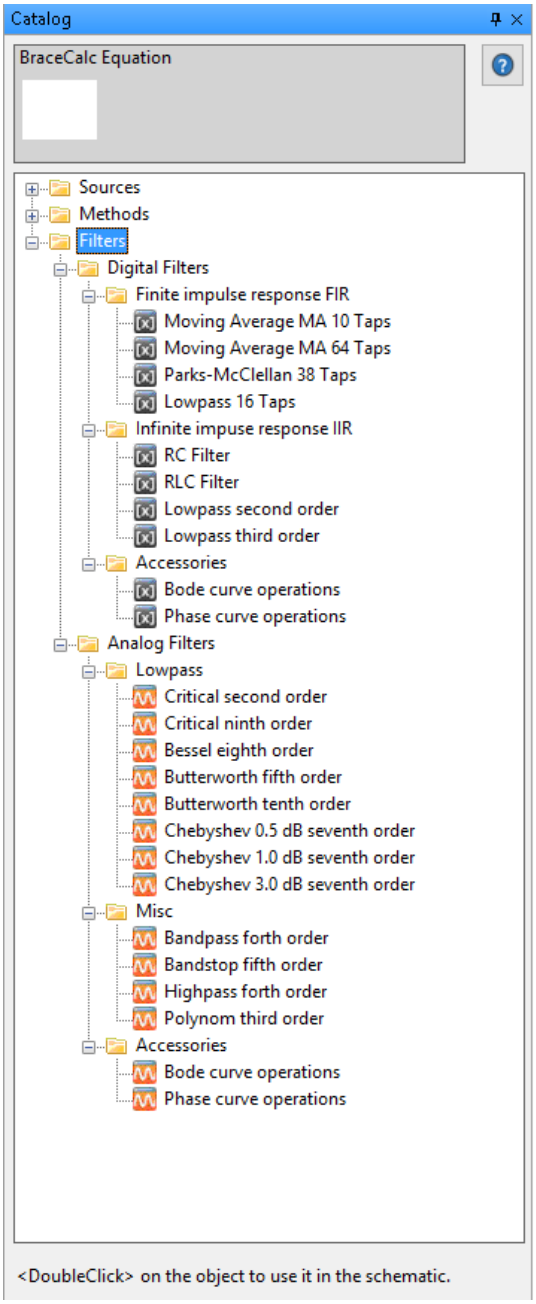
All method objects contain one output and one or two inputs.
They could not be use as single objects.



These methods are unique. For example, an addition method is an addition. You cannot use the addition as a multiplication with a special script parameter.

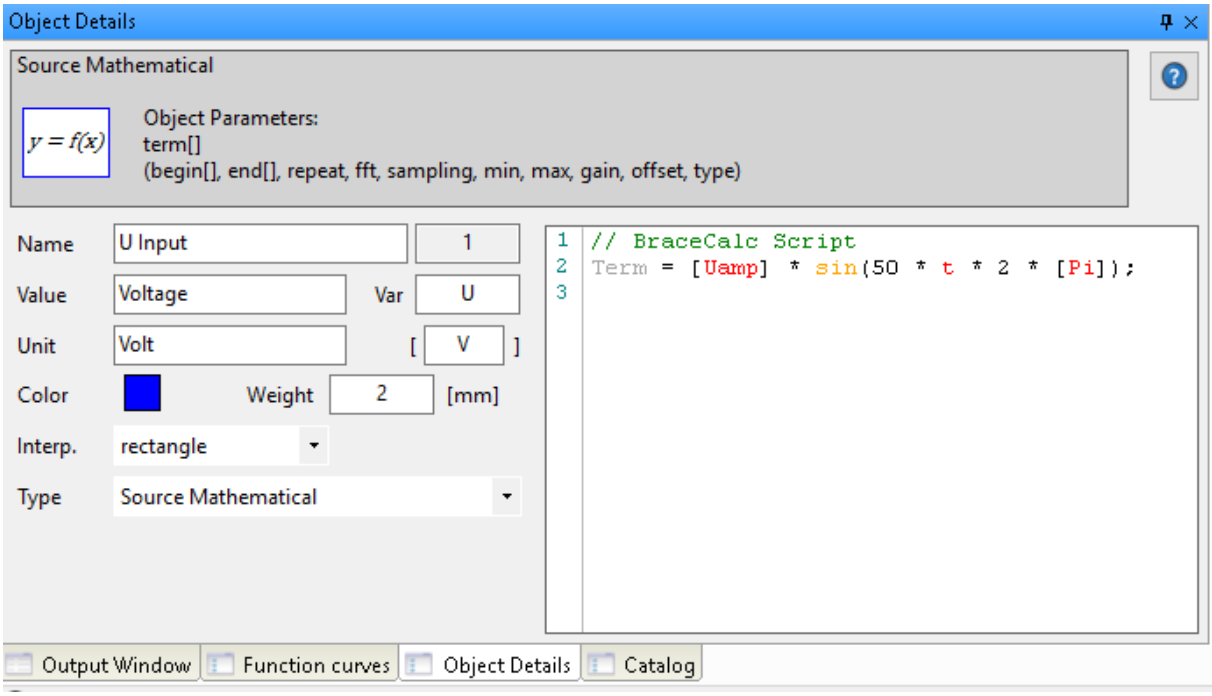
3.1.7.3 Filters

All filter objects are digital or analog. Just the digital filters could work as a method within a signal path.
Analog filter objects could just be single objects with bode diagrams.



3.1.8 Docking Window Object Details

The object properties are define in the object details.



- Overview
  An overview for the chosen object type appears at the top of the window. All mandatory and optional parameter are listed.

- Name
  Object name. This name appears in the schematic below the object symbol and in the curve list window.

- Value optional
  Physical value of the object output signal.

- Unit optional
  Physical unit of the object output signal.

- Color
  Curve color of the output signal.

- Strength
  Line strength of the output signal curve in [mm].

- Interpolation
  There are two kinds how the curve drawing connect two calculated points.
  As connection with a line or with a perpendicular.

The different ***BraceCalc*** objects and the parameters will be described in the next chapters.

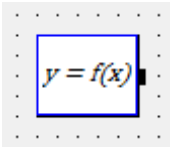
3.1.8.1 Script Parameter

Name	Wert	Bezeichnung	Bemerkung
term[]	ℝ {} + - * / ^ sqr, sqrt, sin, cos, tan, atan sinh, cosh, cotan, exp, ln, log abs, sign, trunc	Unit operations	
begin[]	ℝ {}	Begin of a function part.	
end[]	ℝ {}	End of function part.	
repeat	ℕ {}	Repeats of the function part.	
noise	white	White random noise.	
gap	ℝ {}	Difference between two ´num´values.	
num	ℝ {} ℤ {} hex	Target value of a numeric series.	Hexadecimal values with ´0x´prefix.
format	u8, s8 u16, s16 u32, s32 u64, s64	Format of the num values with hexadecimal form.	Default s16
gain	ℝ {}	Gain of a curve.	
offset	ℝ {}	Offset of a curve.	
power	ℝ {}	Power value	
fft	ℕ {}	Maximum frequency of the FFT calculation.	This value starts a FFT
max	ℝ {}	Maximum value	Higher values will be ignored.
min	ℝ {}	Minimum value	Lower values will be ignored.
type	u8, s8 u16, s16 u32, s32 u64, s64 f32, double	End type of the calculation.	The calculation is always with doubles. The typecast is at the end of the calculation.
bits	ℕ {}	Analog Digital Converter Bit depth	
waves	ℕ {}	Number of Fourier components in the output.	Only valid with object wise calculation.
freq	ℕ {}	Carrier frequency for AM and FM	
hub	ℕ {}	Frequency hub [Hz] for FM.	
sampling	ℕ {}	Object specific sampling.	Only valid if smaller than the sampling in the calculation menu.
Form	polynom lowpass bandpass highpass bandstop	Analog filter types.	

threshold	ℕ {}	Threshold frequency [Hz] for analog filters.	
band	ℕ {}	Filter band [Hz] for analog band filters	
as[]	ℝ {}	A coefficient for analog filters.	
bs[]	ℝ {}	B coefficient for analog filters.	
az[]	ℝ {}	A coefficient for digital filters.	
bz[]	ℝ {}	B coefficient for digital filters.	
prop	ℝ {}	Proportional factor PID control	
inte	ℝ {}	Integral factor PID control	
diff	ℝ {}	Differential factor PID control	
pidmax	ℝ {}	Maximum value. PID control	
res	ℝ {}	Resistor [Ohm]	
ind	ℝ {}	Inductance [H]	
cap	ℝ {}	Capacity [F]	
taps	ℕ {}	Mean value and root mean square value buffer.	
low	ℝ {}	Low output value of a logic object.	
high	ℝ {}	High output value of a logic object	
lohi	ℝ {}	Low to High threshold of a logic object	
hilo	ℝ {}	High to Low threshold of a logic object	
stream[]	ℤ {} hex	Byte array of a stream.	Big Endian
display	ℝ {}	Display offset. Won't be considered for calculation.	Helpful to arange logic signals stacked.

3.1.8.2 Source Mathematical

Single part or multi part function as a mathematical term.



$$f_{out}(x) = f_{in}(x)$$

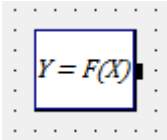
Input	-	
Output	1	
Parameter	Term[]	mathematical term
Optional	begin[]	Begin of the function
	end[]	End of the function
	repeat	Number of repeats of the begin/end range.
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Term = 2 * x;          begin = 0; end = 1;
Term = 2 * (-x) + 4; begin = 1; end = 2;
repeat = 5;


Term = sin(x * 2 * [Pi]);
```

3.1.8.3 Source Numerical

Given numerical values.



$$f_{out}(x) = F_{in}(X)$$

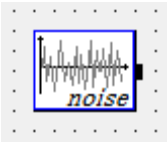
Input	-	
Output	1	
Parameter	Num[]	Single points
	NumXY[][]	Singel points
	Begin	Begin of the first point
	Gap	Gap between two points
Optional	format	Point format in hexadecimal form.
	fft	FFT calculation with maximum frequency of the output.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
begin = 1;
gap = 0.01;
num = 1753;
num = 1357;
num = 1583;
num = 2000;
num = 2115;
num = 1593;
numxy = 1.1, 1593;
numxy = 1.2, 1593;
```

3.1.8.4 Source Noise

The output is a noise signal.



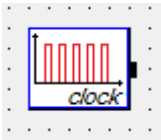
Input	-	
Output	1	
Parameter	Noise	Noise type
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
noise = white;
Max = 1;
```

3.1.8.5 Source Clock

The output is logical clock signal.



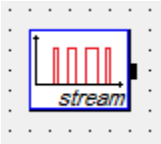
Input	-	
Output	1	
Parameter	sampling	Clock frequency depends on the calculation steps.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	low	Low value of the clock
	high	High value of the clock
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
sampling = 100;
```

3.1.8.6 Source Stream

The output is a binary byte stream signal.



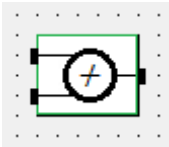
Input	-	
Output	1	
Parameter	stream[]	Bytes in Hex.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Stream frequency depends on the calculation steps.
	low	Low value of the stream
	high	High value of the stream
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Beispiel

```
stream = 0x1234;
stream = 0x00;
stream = 0x5678;
stream = 0x00;
stream = 0x9ABC;
sampling = 5;
```

3.1.8.7 Method Addition

Method for the addition of two inputs.



$$f_{out}(x) = f_{in}(x) + g_{in}(x)$$

Input	2
Output	1

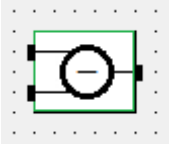
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.8 Method Subtraction

Method for the substraction of two inputs.



$$f_{out}(x) = f_{in}(x) - g_{in}(x)$$

Input	2
Output	1

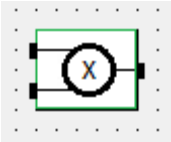
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.9 Method Multiplication

Method for the multiplication of two inputs.



$$f_{out}(x) = f_{in}(x) * g_{in}(x)$$

Input	2
Output	1

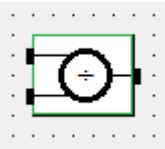
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.10 Method Division

Method for the division of two inputs.



$$f_{out}(x) = \frac{f_{in}(x)}{g_{in}(x)}$$

Input	2
Output	1

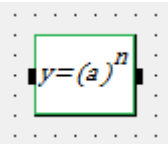
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.11 Methode Potenz

Method for the power calculation of one input.



$$f_{out}(x) = f_{in}(x)^n$$

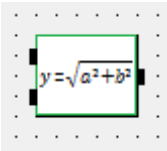
Input	1	
Output	1	
Parameter	Power	Power number n
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Power = 12.5;
```

3.1.8.12 Method Magnitude

Method for the magnitude calculation of two inputs.



$$f_{out}(x) = \sqrt{f_{in}(x)^2 + g_{in}(x)^2}$$

Input	2
Output	1

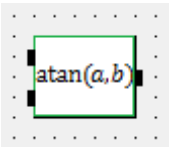
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.13 Method Phase

Method for the phase calculation of two inputs.



$$f_{out}(x) = atan2\big(f_{in}(x), g_{in}(x)\big)$$

Input	2
Output	1

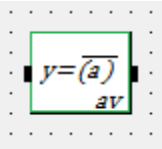
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.14 Method Mean Value

Method for a mean value calculation with buffer from one input.



$$f_{out}(t) = \frac{1}{n}\sum_{i=0}^{n-1} f_{in}(t-i)$$

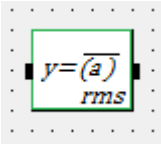
Input	1	
Output	1	
Parameter	Taps	Length of the buffer n per calculation.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Taps = 16;
```

3.1.8.15 Method RMS

Method for root mean square calculation with buffer from one input.



$$f_{out}(t) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} f_{in}(t-i)^2}$$

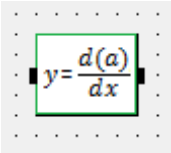
Input	1	
Output	1	
Parameter	Taps	Length of the buffer n per calculation.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Taps = 16;
```

3.1.8.16 Method Derivation

Method for a derivation calculation of one input.



$$f_{out}(x) = \frac{f_{in}(x)}{dx}$$

Input	1
Output	1

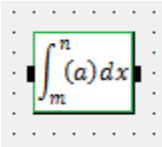
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.17 Method Integral

Method for a integral calculation of one input.



$$f_{out}(x) = \int_m^n f_{in}(x)\,dx$$

Input	1
Output	1

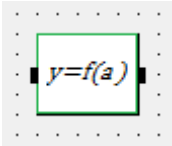
Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

3.1.8.18 Method Mathematical 1 Input

Method for an output calculation from one input and a mathematical term.



$$f_{out}(x) = f(f_{in}(x))$$

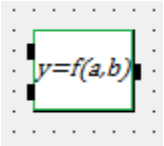
Input	1	
Output	1	
Parameter	Term[]	Mathematical term
	InputA	Input signal
Optional	begin[]	Begin of the function
	end[]	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Term = inputA * 2; begin = 1; end = 2;
Term = inputA / 2; begin = 3; end = 4;
```

3.1.8.19 Method Mathematical 2 Inputs

Method for an output calculation from two inputs and a mathematical term.



$$f_{out}(x) = f(f_{in}(x), g_{in}(x))$$

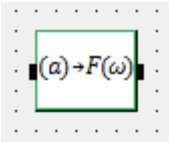
Input	2	
Output	1	
Parameter	Term[]	Mathematical term
	InputA	Input signal A
	InputB	Input signal B
Optional	begin[]	Begin of the function
	end[]	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Term = inputA * inputB; begin = 1; end = 3;
Term = inputA / inputB; begin = 5; end = 9;
```

3.1.8.20 Method Fourier

Method for an output calculation which is composited from a number of waves from the calculated fourier serie of the input signal.



$$f_{out}(x) = \frac{a_0}{2} + \sum_{1}^{n} a_n \cos(nx) + b_n \sin(nx)$$

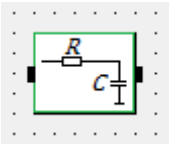
Input	1	
Output	1	
Parameter	Waves	Number n of the fourier serie components.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Down sampling of the calculation points.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Waves = 20;
```

3.1.8.21 Method RC

Method for a resistor-capacity calculation from an input.



$$u_{out}(t) = RC(u_{in}(t))$$

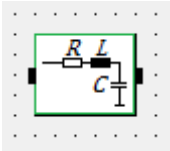
Input	1	
Output	1	
Parameter	Res	Resistor [Ohm]
	Cap	Capacity [F]
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Res = 1000;  // Ohm
Cap = 50E-6; // Farad
```

3.1.8.22 Method RLC

Method for a resistor-capacity-inductivity calculation from an input.



$$u_{out}(t) = RLC(u_{in}(t))$$

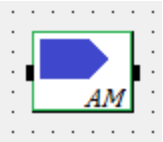
Input	1	
Output	1	
Parameter	Res	Resistor [Ohm]
	Cap	Capacity [F]
	Ind	Inductivity [H]
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Res = 10;     // Ohm
Cap = 0.001;  // Farad
Ind = 0.2;    // Henry
```

3.1.8.23 Method AM

Method for an amplitude-modulated calculation from an input.



$$u_{out}(t) = AM(u_{in}(t))$$

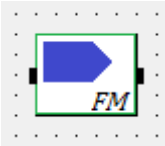
Input	1	
Output	1	
Parameter	Freq	Carrier frequency [Hz]
	Min	Minimum amplitude input (-> 0% AM Signal)
	Max	Maximum amplitude input (-> 100% AM Signal)
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Freq = 100;
Min = -1;
Max = 1;
```

3.1.8.24 Method FM

Method for a frequency-modulated calculation from an input.



$$u_{out}(t) = \sin\left(2\pi f t + \frac{h * u_{in}(t)}{|\max - min|}\right)$$

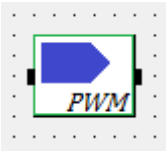
Input	1	
Output	1	
Parameter	Freq	Carrier frequency f [Hz]
	Min	Minimum modulation
	Max	Maximum modulation
	Hub	Factor h of the angle modulation
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Freq = 10;
Min = 0;
Max = 0.1;
Hub = 2;
```

3.1.8.25 Method PWM

Method for a pulse-width-modulated calculation from an input.



$$u_{out}(t) = PWM(u_{in}(t))$$

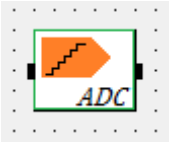
Input	1	
Output	1	
Parameter	Sampling	Number of ticks at 100% duty cycle
	Min	Minimum amplitude input (-> 0% duty cycle)
	Max	Maximum amplitude input (-> 100% duty cycle)
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Sampling = 100;
Min = -0.1;
Max = 2.1;
```

3.1.8.26 Method ADC

Method for an analog-digital-converter calculation from an input.



$$u_{out}(t) = ADC(u_{in}(t))$$

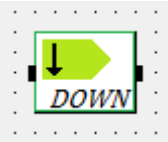
Input	1	
Output	1	
Parameter	Sampling	Sampling frequency.
	Max	Maximum input value (-> maximum ADC value)
	Bits	Bit depth of the output signal.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Sampling = 5000;
Bits = 12;
Max = 3;
```

3.1.8.27 Method Down Sampling

Method for a down sampling of the input.



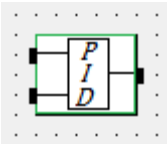
Input	1	
Output	1	
Parameter	Sampling	Number of ticks per second.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Sampling = 1000;
```

3.1.8.28 Method PID Control

Method of a PID control calculation with two inputs (ACTUAL, TARGET).



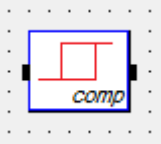
Input	2	
Output	1	
Parameter	Sampling	Sampling frequency of the control
	Prop	Proportional parameter
	Inte	Integral parameter
	Diff	Differential parameter
	PidMax	Maximum control value
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
sampling = 1000;
prop = 0.01;
inte = 0.01;
diff = 0;
max = 100000;
```

3.1.8.29 Method Comparator 1 Input

Method of a comparator with one input signal.



Input	1
Output	1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling frequency depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
low = 0;
high = 10;
lohi = 8;
hilo = 2;
```

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.30 Method Comparator 2 Inputs

Method fort the compare of the input signals.

Input	2
Output	1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

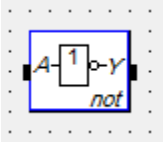
Example

Default values:

```
low = 0;
high = 5;
```

3.1.8.31 Method NOT

Method for a logical NOT operation.



Input	1
Output	1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

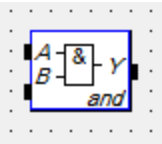
Example

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.32 Method AND

Method for a logical AND operation.



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Input 2
Output 1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

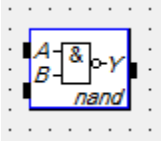
Example

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.33 Method NAND

Method for a logical NAND operation



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Input 2
Output 1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

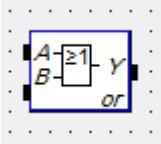
Example

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.34 Method OR

Method for a logical OR operation.



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Input 2
Output 1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

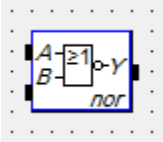
Example

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.35 Method NOR

Method for a logical NOR operation.



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Input 2
Output 1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

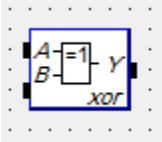
Example

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.36 Method XOR

Method for a logical XOR operation.



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Input 2
Output 1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

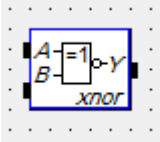
Example

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.37 Method XNOR

Method for a logical XOR operation.



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Input 2
Output 1

Parameter

Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	sampling	Sampling depends on the calculation steps.
	low	Low value of the output
	high	High value of the output
	lohi	Low to High threshold
	hilo	High to Low threshold
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

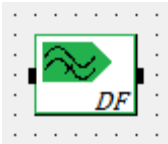
Example

Default values:

```
low = 0;
high = 5;
lohi = 2.4
hilo = 0.8;
```

3.1.8.38 Method Digital Filter

Method of a digital filter calculation.



$$A(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} a_k z^{-k}}{1 + \sum_{k=0}^{N} b_k z^{-k}}$$

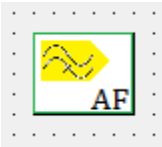
Input	1	
Output	1	
Parameter	Sampling	Sampling frequency of the filter
	az[]	A components
	bz[]	B components
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the output.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
sampling = 2000;
gain = 1 / 10;
az.0 = 1;
az.1 = 1;
az.2 = 1;
az.3 = 1;
az.4 = 1;
az.5 = 1;
az.6 = 1;
az.7 = 1;
az.8 = 1;
az.9 = 1;
```

3.1.8.39 Object Analog Filter

Calculates the Bode diagrams of an analog filter transfer function.



It is essential

$$P = j\frac{\omega}{\omega_g}\ bzw. P = j\frac{f}{f_g}$$

$$Band = \Delta\omega$$

General transfer function

$$A(P) = \frac{A_0 + A_1 P + A_2 P^2 + \cdots + A_n P^n}{B_0 + B_1 P + A_2 P^2 + \cdots + B_n P^n}$$

Analog low pass filter

$$A(P) = \frac{A_0}{\prod_i 1 + a_i P + b_i P^2}$$

Analog high pass filter

$$A(P) = \frac{A_\infty}{\prod_i 1 + \frac{a_i}{P} + \frac{b_i}{P^2}}$$

Analog band pass filter

$$A(P) = \frac{A_0}{\prod_i 1 + a_i \frac{P + \frac{1}{P}}{\Delta\omega} + b_i \left(\frac{P + \frac{1}{P}}{\Delta\omega}\right)^2}$$

Analog band block filter

$$A(P) = \frac{A_\infty}{\prod_i 1 + a_i \frac{\Delta\omega}{P + \frac{1}{P}} + b_i \left(\frac{\Delta\omega}{P + \frac{1}{P}}\right)^2}$$

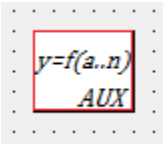
Input	-	
Output	-	
Parameter	Form	Analog filter type (Polynom, Lowpass, Highpass, Bandpass, Bandstop)
	Threshold	Working frequency
	as[]	A components
	bs[]	B components
Optional	gain	Gain factor of the outputs

Example

```
threshold = 1000;
form = lowpass;
as.0 = 1.2872;
bs.0 = 0.4142;
```

3.1.8.40 Object Auxiliary

This auxiliary object calculates any operation with output signals from other objects.



$$f_{out}(x) = f(f_{out\ 1}(x), f_{out\ 2}(x), \dots, f_{out\ n}(x))$$

$$f_{out}(\omega) = f(f_{out\ 1}(\omega), f_{out\ 2}(\omega), \dots, f_{out\ n}(\omega))$$

$$f_{out}(\varphi) = f(f_{out\ 1}(\varphi), f_{out\ 2}(\varphi), \dots, f_{out\ n}(\varphi))$$

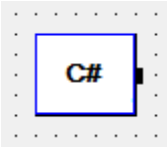
Input	-	
Output	-	
Parameter	Term[]	Mathematical term
	Output{n}	Output signal of an object n
	Bode{n}	Bode amplitude curve of a filter n
	Phase{n}	Bode phase curve of a filter n
Optional	begin[]	Begin of the function
	end[]	End of the function
	min	Minimum of the considered values.
	max	Maximum of the considered values.
	gain	Gain factor of the output.
	offset	Offset value of the output.
	type	Type of the output.

Example

```
Term = bode{1} * bode{2}; // filter ids in the braces
```

3.1.8.41 C# Source

Output of external programmed C# DLL.



Input	-	
Output	1	
Parameter	dll	Path and name of the external C# DLL.
	namespace	Namespace name of the static C# class and its functions.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the ouput
	gain	Gain factor to the output
	offset	Offset value to the output
	type	Type of the output values

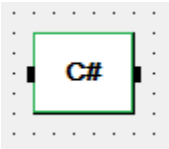
Example

```
dll = c:\ownDll\bin\Debug\ownDll.dll;
namespace = OwnDll;
```

```
namespace OwnDll
{
  //
  // "BraceCalcSource" class is mandatory. Don't change the name.
  //
  static public class BraceCalcSource
  {
    //
    // "Init" method is mandatory. Don't change it.
    //
    static public void Init(uint steps, double begin, double end)
    {
      //
      // Fill in some initialization code
      //
    }
    //
    // "Step" method is mandatory. Don't change it.
    //
    static public double Step(double position, int stepNr)
    {
      //
      // Fill in some calculation code. Stepwise.
      //
    }
    //
    // "End" method is mandatory. Don't change the definition.
    //
    static public void End(double[] values)
    {
      //
      // Fill in some finish code.
      // !!! Will be excecuted just by BraceCalc objectwise calculation !!!
      //
    }
  }
}
```

3.1.8.42 C# Method 1 Input

Method to calculate an output signal with an input signal from an external C# DLL.



Input	1	
Output	1	
Parameter	dll	Path and name of the external C# DLL.
	namespace	Namespace name of the static C# class and its functions.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the ouput
	gain	Gain factor to the output
	offset	Offset value to the output
	type	Type of the output values

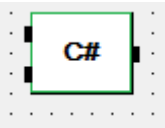
Example

```
dll = c:\ownDll\bin\Debug\ownDll.dll;
namespace = OwnDll;
```

```
namespace OwnDll
{
  //
  // "BraceCalcSource" class is mandatory. Don't change the name.
  //
  static public class BraceCalcSource
  {
    //
    // "Init" method is mandatory. Don't change it.
    //
    static public void Init(uint steps, double begin, double end)
    {
      //
      // Fill in some initialization code
      //
    }
    //
    // "Step" method is mandatory. Don't change it.
    //
    static public double Step(double position, int stepNr, double inputA)
    {
      //
      // Fill in some calculation code. Stepwise.
      //
    }
    //
    // "End" method is mandatory. Don't change the definition.
    //
    static public void End(double[] values)
    {
      //
      // Fill in some finish code.
      // !!! Will be excecuted just by BraceCalc objectwise calculation !!!
      //
    }
  }
}
```

3.1.8.43 C# Method 2 Inputs

Method to calculate an output signal with two input signals from an external C# DLL.



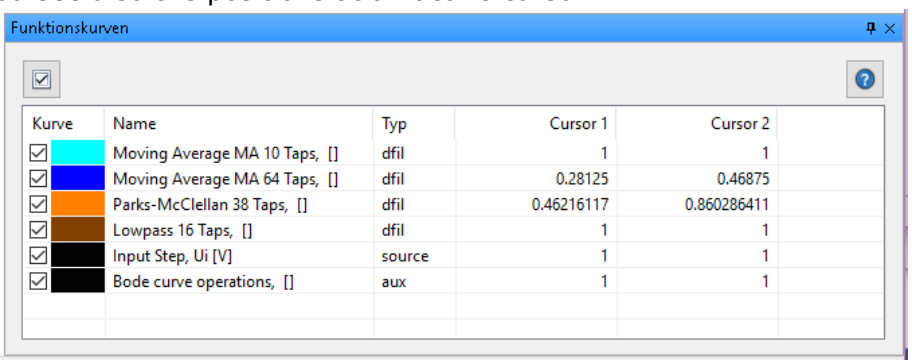
Input	1	
Output	1	
Parameter	dll	Path and name of the external C# DLL.
	namespace	Namespace name of the static C# class and its functions.
Optional	begin	Begin of the function
	end	End of the function
	fft	FFT calculation with maximum frequency of the ouput
	gain	Gain factor to the output
	offset	Offset value to the output
	type	Type of the output values

Example

```
dll = c:\ownDll\bin\Debug\ownDll.dll;
namespace = OwnDll;

namespace OwnDll
{
  //
  // "BraceCalcSource" class is mandatory. Don't change the name.
  //
  static public class BraceCalcSource
  {
    //
    // "Init" method is mandatory. Don't change it.
    //
    static public void Init(uint steps, double begin, double end)
    {
      //
      // Fill in some initialization code
      //
    }
    //
    // "Step" method is mandatory. Don't change it.
    //
    static public double Step(double position, int stepNr, double inputA, double inputB)
    {
      //
      // Fill in some calculation code. Stepwise.
      //
    }
    //
    // "End" method is mandatory. Don't change the definition.
    //
    static public void End(double[] values)
    {
      //
      // Fill in some finish code.
      // !!! Will be excecuted just by BraceCalc objectwise calculation !!!
      //
    }
  }
}
```

3.1.9 Docking Window Curve List

All curves are listed in the curve list.
You see also the positions at an active cursor.

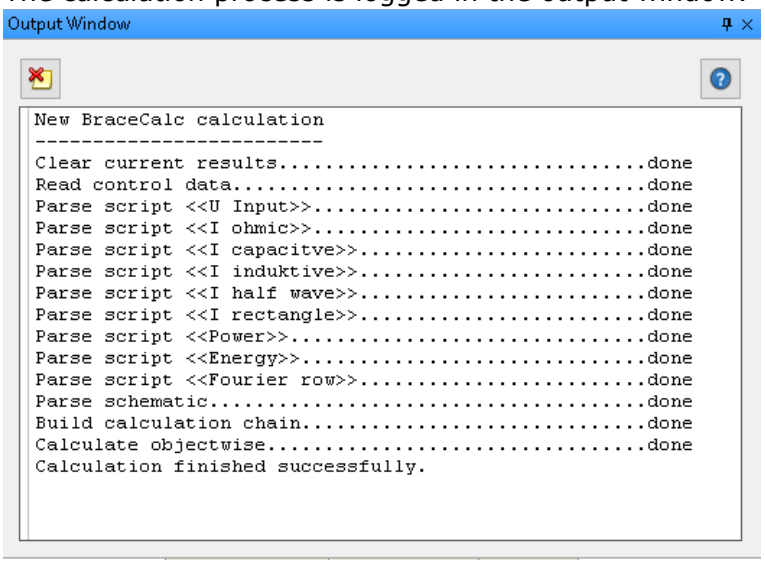


Kurve	Name	Typ	Cursor 1	Cursor 2
☑	Moving Average MA 10 Taps, []	dfil	1	1
☑	Moving Average MA 64 Taps, []	dfil	0.28125	0.46875
☑	Parks-McClellan 38 Taps, []	dfil	0.46216117	0.860286411
☑	Lowpass 16 Taps, []	dfil	1	1
☑	Input Step, Ui [V]	source	1	1
☑	Bode curve operations, []	aux	1	1

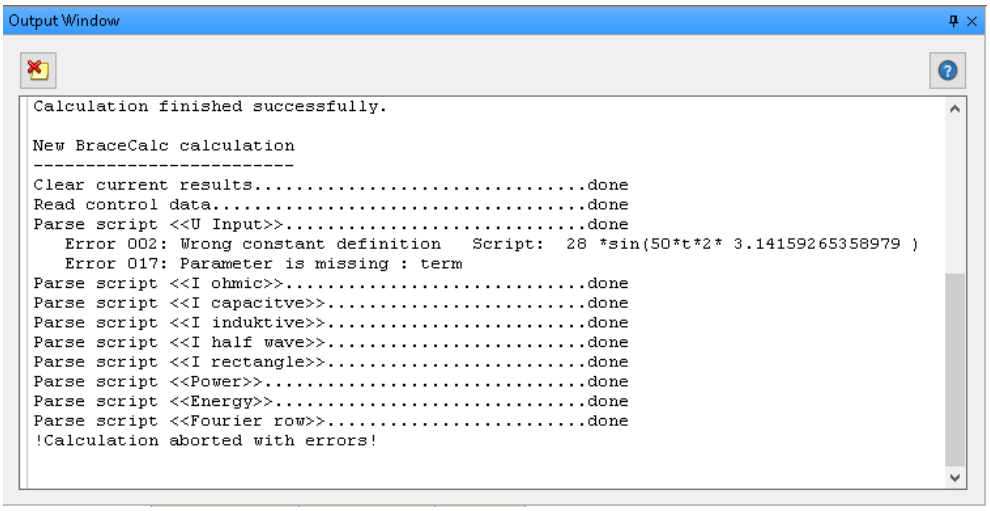
You can check or uncheck the curve for display in the scope window.

3.1.10 Docking Window Output

The calculation process is logged in the output window.



You see the errors if a definition is incorrect.



3.1.11 Error List

Fehler	Bezeichnung	Bemerkung
Error001	No script definition.	A mandatory parameter definition is missing in the object.
Error002	Wrong constant definition	The constant definition is wrong. Check the order.
Error006	Wrong parameter definition	The parameter definition is wrong for the object.
Error007	Wrong parameter value	The range value is wrong for the parameter.
Error008	Wrong term definition.	The mathematical term is wrong.
Error010	Parameter not found.	A mandatory parameter is missing for the object.
Error011	Wrong coefficient definition.	Filter coefficient is wrong.
Error012	Wrong object singal path.	The signal path is not working with these objects.
Error013	Source object has an input.	You try to connect an input to a source object.
Error014	Method object has an invalid input.	The input signal is wrong at the object.
Error015	Method object has an invalid input.	Two input signals are wrong at the object.
Error016	No object defined.	It needs one source object at least for a calculation.
Error017	No parameter defined.	A mandatory parameter for the object is missing.
Error018	Too many signal connections.	There are up to 19 object steps in one calculation chain allowed.
Error019	Calculation broken.	There is too less memory for the calculation..
Error020	Wrong calculation parameter	Some parameter in the calculation menu are wrong.
Error021	DLL not found or missing	The given DLL path and name couldn't be found.

3.1.12 Help…

This user manual will open for the help.

3.1.13 About BraceCalc…

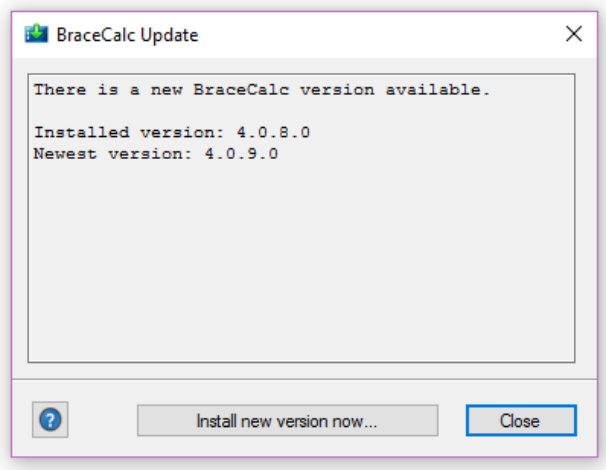
3.1.13.1 Language

You can choose between English and German program language.

Windows components like the ´File open dialog´ appears with the Windows language.

3.1.13.2 Proof…

BraceCalc seek for the actual version.
A newer version can be installed with <Install new version now…>.



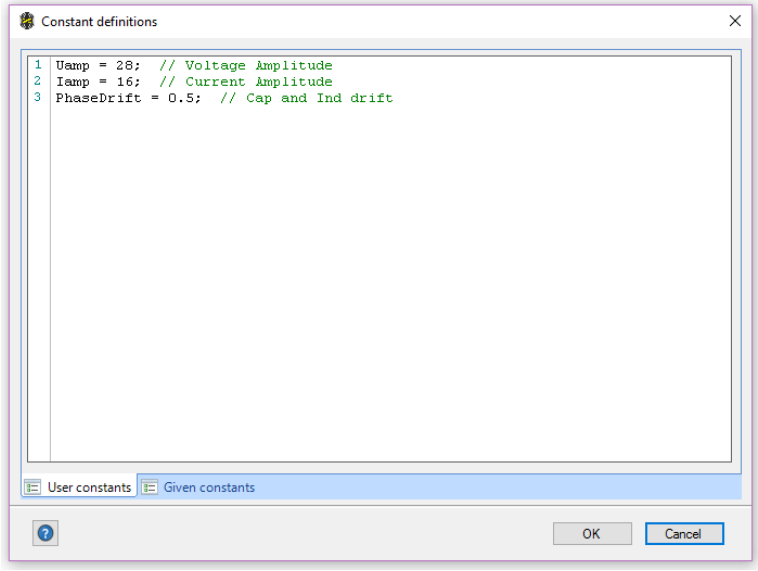
3.1.13.3 Activate

You can use ***BraceCalc*** for 30 days without a valid key.

Enter the valid key in the field and push <Activate> for activate ***BraceCalc***.

3.1.14 Constants

You can define constants for the current file.

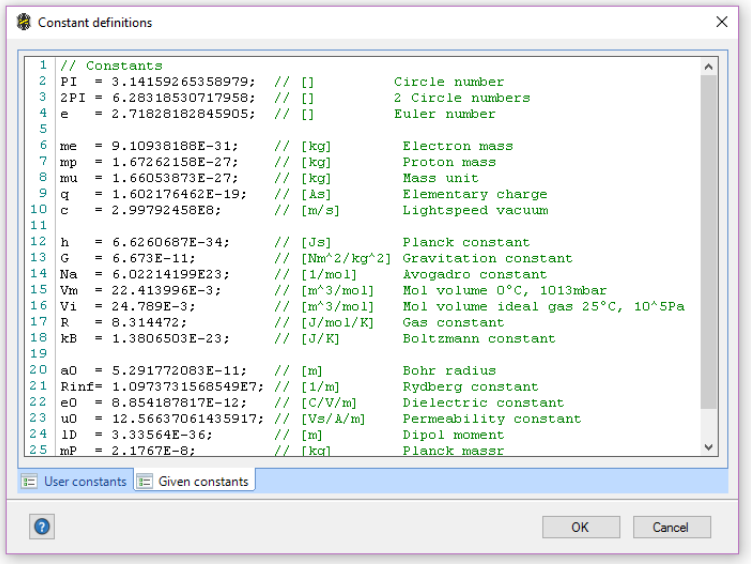


You can use these constants in every object script. The name has to written with brackets [].

Example:

```
Term = [Iamp] * x;
```

Additional to the user constants there is a choice of predefined constants.



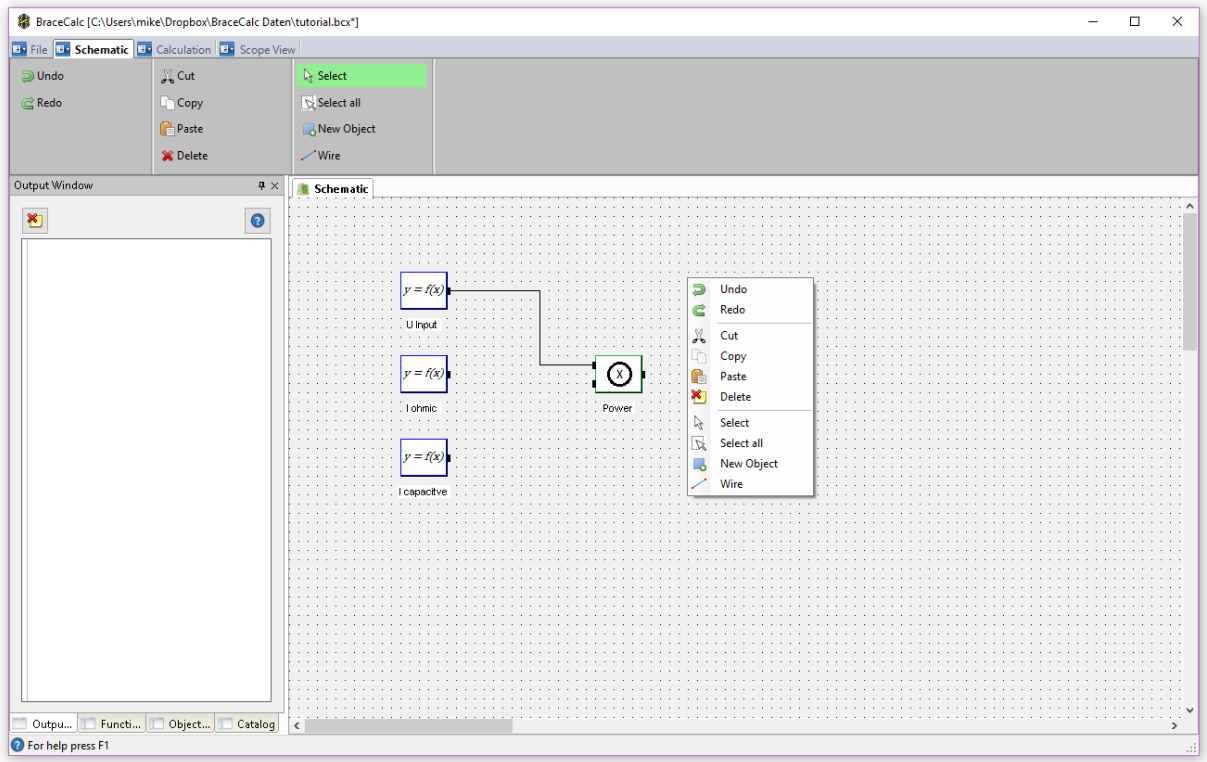
3.1.15 Exit

BraceCalc is closing. You have to save unsaved data.

3.2 Schematic

3.2.1 General

The ***BraceCalc*** schematic is the base for a calculation. It includes all objects. They can be work as single objects or they can be connected with others.



3.2.2 Undo (CTRL-Z)

A done action can be undo.

3.2.3 Redo (CTRL-Y)

An undo action can be redo.

3.2.4 Cut (CTRL-X)

Cutting out marked objects to paste them later.

3.2.5 Copy (CTRL-C)

Copy marked objects to paste them later.

3.2.6 Paste (CTRL-V)

Paste objects after a cut or copy.

3.2.7 Delete

Delete marked objects. They won't copy in the clipboard.

3.2.8 Select

You can mark objects and connections with the mouse.

3.2.9 Select all (CTRL-A)

You can mark all objects and connection in the schematic..

3.2.10 New Object

You can insert a new default source ***BraceCalc*** object in the schematic.
Use the object details window for further changes.

3.2.11 Connection

The connection draw tool is active.
You can draw connections and intersections with the mouse.
Use the left mouse button for a connection and the right mouse button for an intersection.

3.2.12 Key Control

With the keys

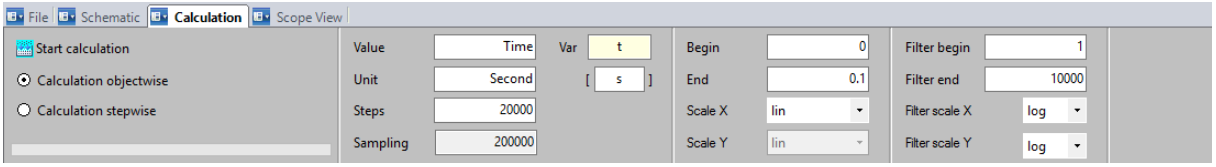
<up>
<down>
<left>
<right>

You can move marked objects.

3.3 Calculation

3.3.1 General

The ***BraceCalc*** calculation is also a main part of ***BraceCalc***.
All necessary parameters and definitions are insert in this menu.



3.3.2 Start Calculation (F5)

A new calculation starts with <Start calculation> for all object and signals.

Calculation order:

- Analyse of the calculation parameters
- Analyse of the schematic
- Analyse of every object and its script
- Calculation of every signal
- Display of all curves in the scopes.

3.3.3 Calculation Object by Object

BraceCalc calculates every object by object. The calculation order is given through the schematic connections and the object types.
Every input signal of an object has to be calculated before the object itself can be calculated.

For example, a fourier series can only be calculated object by object.

See chapter 2

3.3.4 Calculation Step by Step

Every single step is calculated through the complete signal chain.
Only source objects are calculated first complete.

For example, a feedback control calculation is just possible with a step by step calculation.

See chapter 2

3.3.5 Value and Var

The field <Value> is optional.

Very important is the field <Var>
This shortcut name is our function variable of the X-Axe. It is necessary in every object script with mathematical terms.

<Var> can be a single letter or a word. Special sign and space are not allowed.

We use ´t´ for time functions.

3.3.6 Unit

Optional
The unit of the X-Axe. We use ´second´ and ´[s]´ for a time unit.

3.3.7 Steps

The complete number of calculation steps is written here in this field.
Steps above 200000 can break the calculation because of too less memory.

3.3.8 Sampling

Read only.
The sampling is calculated from the steps and the range (begin to end).

3.3.9 Begin

The start point of the calculation. This has to be lower than the end point.

3.3.10 End

The end point of the calculation. This has to be higher than the begin point.

3.3.11 X Scale

You can choose between linear and logarithmic scale.
A logarithmic scale needs a begin point higher than zero.

3.3.12 Y Scale

Read only. It is just a linear scale possible.

3.3.13 Filter Begin

The filter calculation needs a separate begin and end range.
This is the filter begin point. It has to be lower than the filter end point.

3.3.14 Filter End

The filter end point of the calculation. It has to be higher than the filter begin point.

3.3.15 X Filter Scale

You can choose between linear and logarithmic scale.
A logarithmic scale needs a begin point higher than zero.

3.3.16 Y Filter Scale

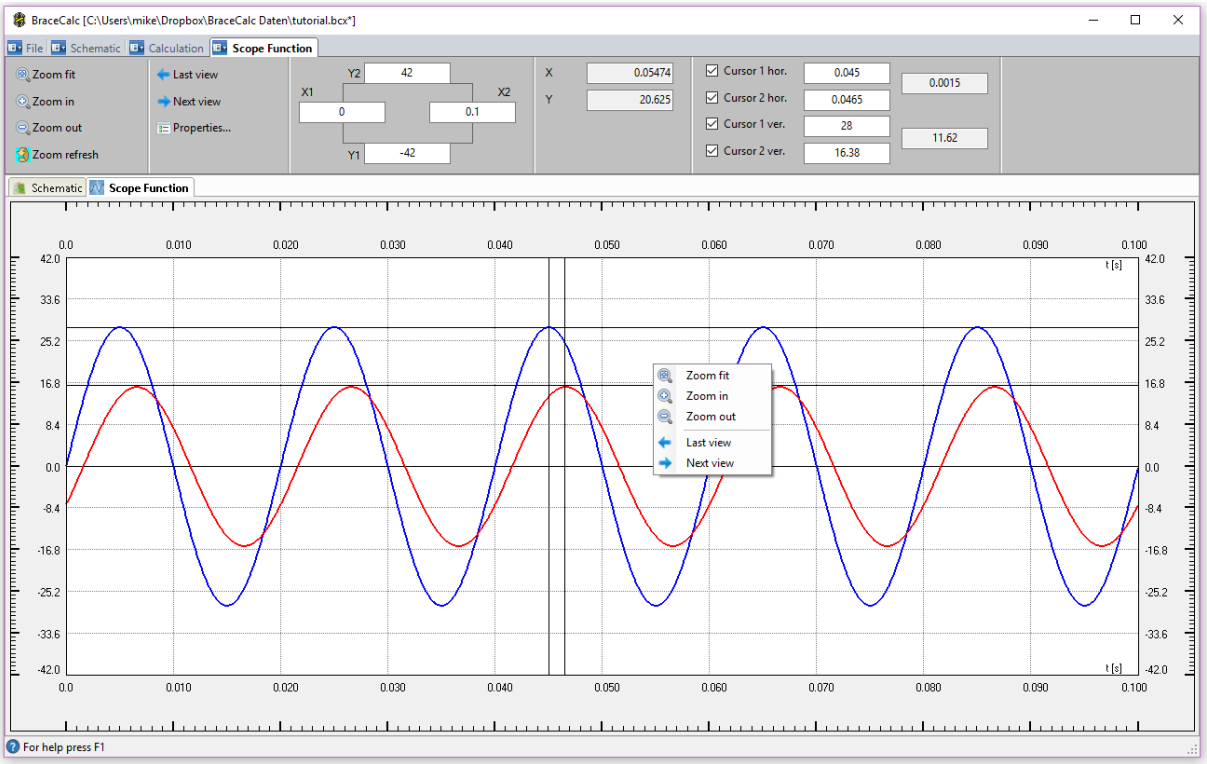
Here is a logarithmic Y calculation possible. The display and the graduation is calculated in **Dezibel [dB]**.

3.4 Scopes

3.4.1 General

There are four different scopes in ***BraceCalc***

- Function scope
- Spectrum scope
- Bode diagram amplitude
- Bode diagram phase



Every scope has its own properties.

Remark:
The actual display is keeping for the print.

3.4.2 Zoom Fit

The x-axe display range is setting to the specification of begin point and end point in the calculation parameters.
The y-axe range is the result of the maximum and minimum calculation points.

An indefinite y-axe can be the result of an infinite calculation.
You can set a maximum or minimum in the object script to avoid this.

3.4.3 Zoom In

The view is scaling up.

3.4.4 Zoom Out

The view is scaling down.

3.4.5 Zoom refresh

The display is refreshing with the actual values in the menu.
New values for the range or the cursors can refreshing the display also with <Enter>.

3.4.6 Last View

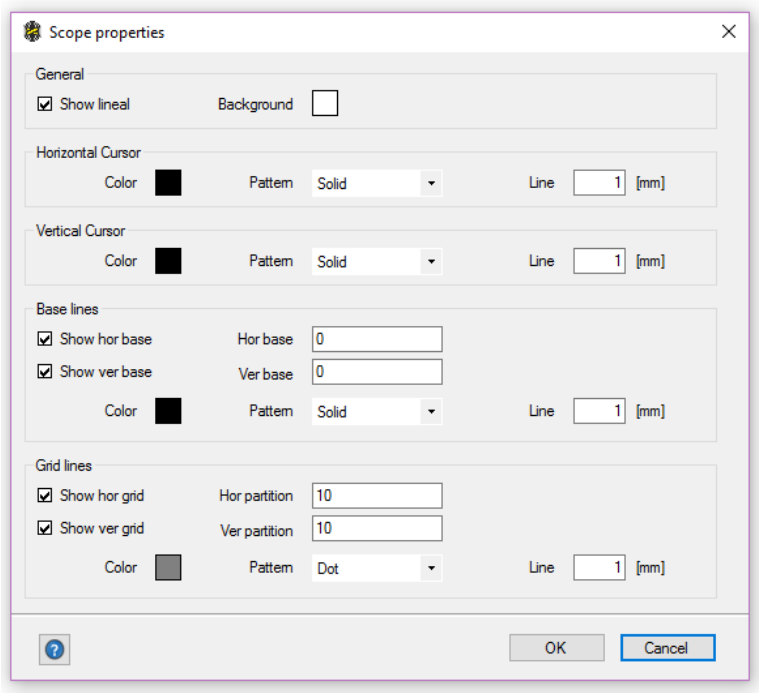
The last view is displayed.

3.4.7 Next View

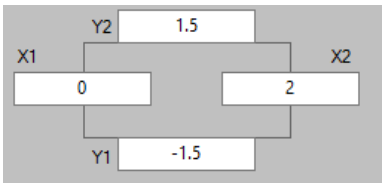
The last view can be repeated.

3.4.8 Properties...

The scope properties can be adjusted with this dialog.



3.4.9 View Range Field



The view range field can be adapted individually.
You can insert the ranges manually.

X1 has to be lower than X2 and Y1 has to be higher than Y2.

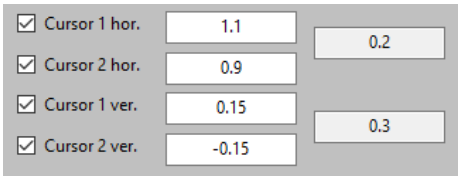
The view is refreshing wiht <Zoom refresh> or <Enter>.

3.4.10 Scope Mouse Position

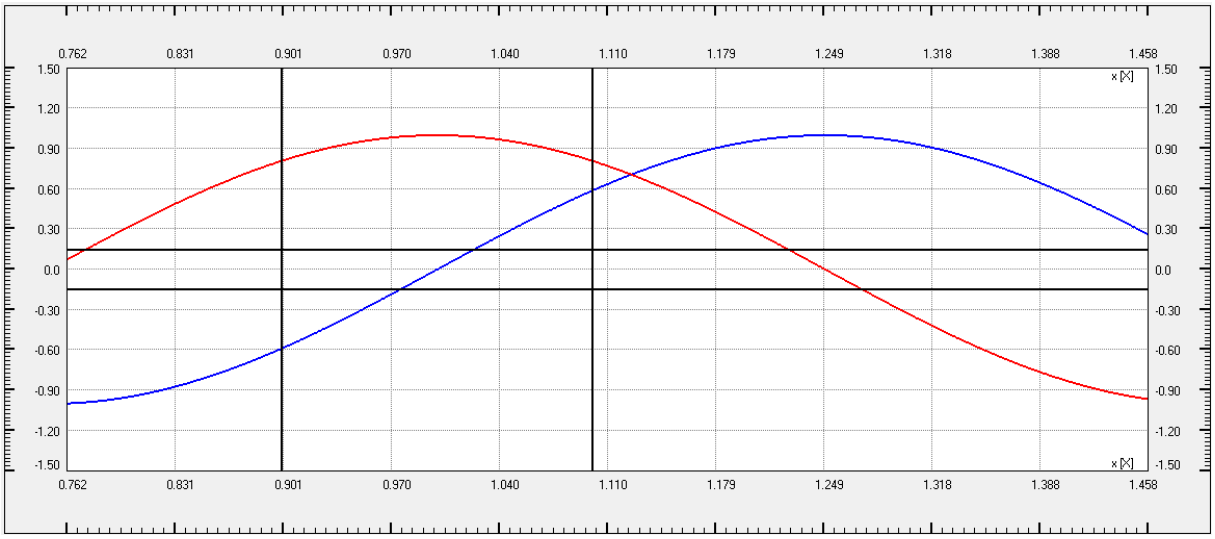
You see the actual coordinates of the mouse cursor.

3.4.11 Cursor

You can activate two horizontal and two vertical cursors for every scope.



You see the difference between two cursors if both are activated.



Key navigation

Cursor 1 horizontal
<left>
<right>

Cursor 2 horizontal
<SHIFT><left>
<SHIFT><right>

Cursor 1 vertical
<up>
<down>

Cursor 2 vertical
<SHIFT><up>
<SHIFTt><down>

3.5 C# DLL creation

3.5.1 General

It is possible to program your own BraceCalc sources and methods with external C# DLLs.

- Source with output
- Method with one input signal and an output
- Method with two input signals and an output

Following aspects has to consider:

- Unique user defined namespace name
- Static class with user defined name
- Static functions with given definitions

3.5.1.1 Unique namespace

The namespace has to be unique and is a BraceCalc script parameter.

```
namespace MyBraceCalc
{
}
```

3.5.1.2 Static class

The class name is also user defined. It isn't used as BraceCalc parameter.

```
static public class MyBraceCalcClass
{
}
```

3.5.1.3 Static init function

The init function is called at the begin of the BraceCalc calculation. It can be empty.

```
static public void Init(uint steps, double begin, double end)
{
}
```

3.5.1.4 Static step function

This step function is called at every step of the BraceCalc calculation. It mustn't be empty.

There are two different kinds of calculation:

- Objectwise calculation. Every object is calculated self-contained.
- Stepwise calculation. Every step goes through the object chain.

These kinds are to consider in the programming.

Step function of a source:

```
static public double Step(double position, int stepNr)
{
}
```

Step function of a method with one input:

```
static public double Step(double position, int stepNr, double inputA)
{
}
```

Step function of method with two inputs:

```
static public double Step(double position, int stepNr, double inputA, double inputB)
{
}
```

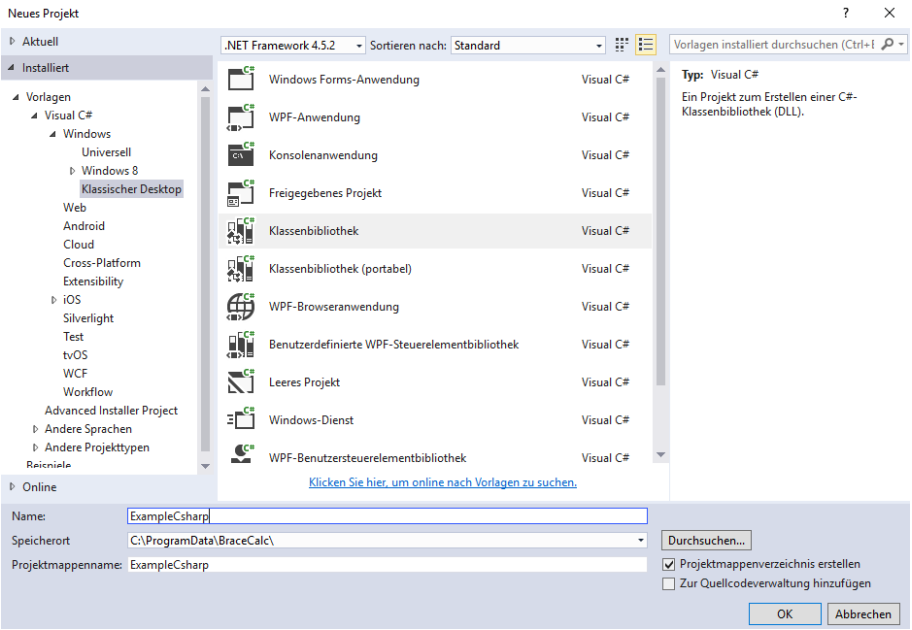
3.5.1.5 Static end function

The end function is called at the end of a BraceCalc function. The complete calculated output signal array is the parameter. It can calculated additional.

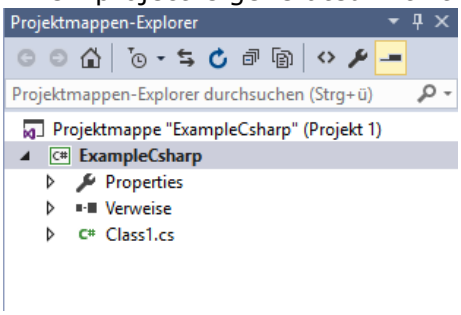
```
static public void End(double[] values)
{
}
```

3.5.2 Visual Studio 2015 Project

Start Visual Studio 2015 and create a new C# class library for the classic desktop.



A new project is generated with this content:



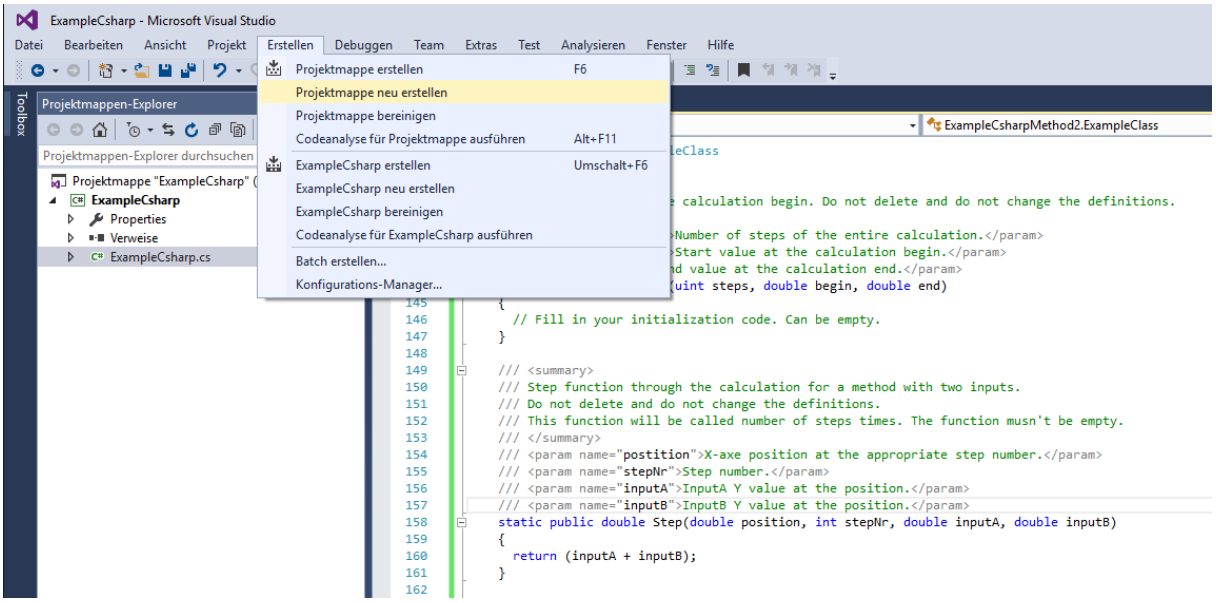
Delete the given code in Class1.cs and copy the following template code:

```
//===========================================================================
//
//                Eicher Engineering
//                BraceCalc the Calculation Program
//
//===========================================================================
//
// Comment:
//    BraceCalc external C# DLL Template
//
//===========================================================================
//
// Created 08.10.2016
//
//===========================================================================
//
//
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExampleCsharpSource
{
  static public class ExampleClass
  {
    /// <summary>
    /// Init function at the calculation begin. Do not delete and do not change the definitions.
    /// </summary>
    /// <param name="steps">Number of steps of the entire calculation.</param>
    /// <param name="begin">Start value at the calculation begin.</param>
    /// <param name="end">End value at the calculation end.</param>
    static public void Init(uint steps, double begin, double end)
    {
      // Fill in your initialization code. Can be empty.
    }

    /// <summary>
    /// Step function through the calculation for a source.
    /// Do not delete and do not change the definitions.
    /// This function will be called number of steps times. The function musn't be empty.
    /// </summary>
    /// <param name="postition">X-axe position at the appropriate step number.</param>
    /// <param name="stepNr">Step number.</param>
    static public double Step(double position, int stepNr)
    {
      // Fill in your calculation code. Simple sinus.
      return Math.Sin(position);
    }

    /// <summary>
    /// Step function through the calculation for a method with one input.
    /// Do not delete and do not change the definitions.
    /// This function will be called number of steps times. The function musn't be empty.
    /// </summary>
    /// <param name="postition">X-axe position at the appropriate step number.</param>
    /// <param name="stepNr">Step number.</param>
    /// <param name="inputA">InputA Y value at the position.</param>
    static public double Step(double position, int stepNr, double inputA)
    {
      // Fill in your calculation code.
      return inputA * 10;
    }

    /// <summary>
    /// Step function through the calculation for a method with two inputs.
    /// Do not delete and do not change the definitions.
    /// This function will be called number of steps times. The function musn't be empty.
    /// </summary>
    /// <param name="postition">X-axe position at the appropriate step number.</param>
    /// <param name="stepNr">Step number.</param>
```

```
    /// <param name="inputA">InputA Y value at the position.</param>
    /// <param name="inputB">InputB Y value at the position.</param>
    static public double Step(double position, int stepNr, double inputA, double inputB)
    {
      return (inputA + inputB);
    }


    /// <summary>
    /// End function at the calculation end. Do not delete and do not change the definitions.
    /// Attention: this function is called just with objectwise BraceCalc calculation.
    /// </summary>
    /// <param name="values">Array with all calculated values from the step function.</param>
    static public void End(double[] values)
    {
      // Fill in your end code. Can be empty.
    }
  }
}

namespace ExampleCsharpMethod1
{
  static public class ExampleClass
  {
    /// <summary>
    /// Init function at the calculation begin. Do not delete and do not change the definitions.
    /// </summary>
    /// <param name="steps">Number of steps of the entire calculation.</param>
    /// <param name="begin">Start value at the calculation begin.</param>
    /// <param name="end">End value at the calculation end.</param>
    static public void Init(uint steps, double begin, double end)
    {
      // Fill in your initialization code. Can be empty.
    }

    /// <summary>
    /// Step function through the calculation for a method with one input.
    /// Do not delete and do not change the definitions.
    /// This function will be called number of steps times. The function musn't be empty.
    /// </summary>
    /// <param name="postition">X-axe position at the appropriate step number.</param>
    /// <param name="stepNr">Step number.</param>
    /// <param name="inputA">InputA Y value at the position.</param>
    static public double Step(double position, int stepNr, double inputA)
    {
      // Fill in your calculation code.
      return inputA * 10;
    }

    /// <summary>
    /// End function at the calculation end. Do not delete and do not change the definitions.
    /// Attention: this function is called just with objectwise BraceCalc calculation.
    /// </summary>
    /// <param name="values">Array with all calculated values from the step function.</param>
    static public void End(double[] values)
    {
      // Fill in your end code. Can be empty.
    }
  }
}

namespace ExampleCsharpMethod2
{
  static public class ExampleClass
  {
    /// <summary>
    /// Init function at the calculation begin. Do not delete and do not change the definitions.
    /// </summary>
    /// <param name="steps">Number of steps of the entire calculation.</param>
    /// <param name="begin">Start value at the calculation begin.</param>
    /// <param name="end">End value at the calculation end.</param>
    static public void Init(uint steps, double begin, double end)
    {
      // Fill in your initialization code. Can be empty.
```

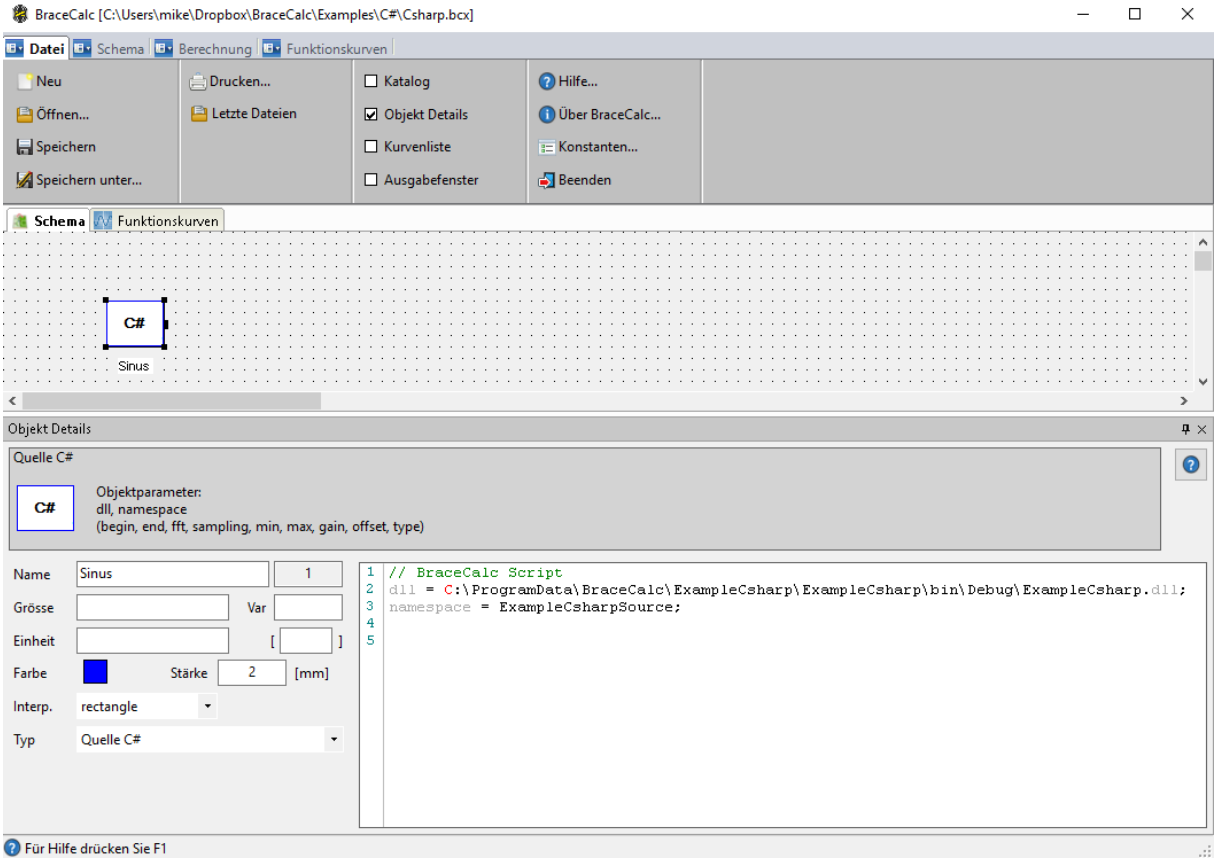
```
    }

    /// <summary>
    /// Step function through the calculation for a method with two inputs.
    /// Do not delete and do not change the definitions.
    /// This function will be called number of steps times. The function musn't be empty.
    /// </summary>
    /// <param name="postition">X-axe position at the appropriate step number.</param>
    /// <param name="stepNr">Step number.</param>
    /// <param name="inputA">InputA Y value at the position.</param>
    /// <param name="inputB">InputB Y value at the position.</param>
    static public double Step(double position, int stepNr, double inputA, double inputB)
    {
      return (inputA + inputB);
    }


    /// <summary>
    /// End function at the calculation end. Do not delete and do not change the definitions.
    /// Attention: this function is called just with objectwise BraceCalc calculation.
    /// </summary>
    /// <param name="values">Array with all calculated values from the step function.</param>
    static public void End(double[] values)
    {
      // Fill in your end code. Can be empty.
    }
  }
}
```

Now build the DLL

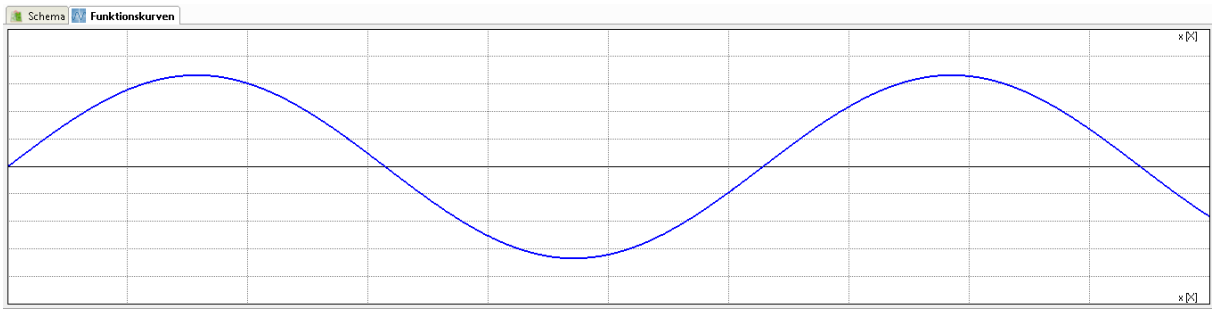


3.5.3 Integrate DLL in C# objects

Insert a C# object into your BraceCalc schematic. Use the parameters “dll” and “namespace” as parameters. Write the DLL path and name to the dll parameter. Write the name of the namespace to the namespace parameter.



Calculate the BraceCalc file.



The result of the example DLL gives you a simple sinus curve.

Version	Date	Description
4.0.10.0	24.05.2016	Release
4.0.11.0	08.06.2016	Added logical objects.
4.0.12.0	08.10.2016	Added C# DLL objects
4.0.13.0	15.10.2016	Added numerical xy values.