Analysis Option
Perception
LICENSE AGREEMENT AND WARRANTY
For information about LICENSE AGREEMENT AND WARRANTY refer to www.hbm.com/terms.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong> Analysis Option</td>
<td></td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>11</td>
</tr>
<tr>
<td>1.1.1 How to install Analysis</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Formula database</td>
<td>13</td>
</tr>
<tr>
<td>1.3 Definitions</td>
<td>16</td>
</tr>
<tr>
<td>1.3.1 Constants</td>
<td>16</td>
</tr>
<tr>
<td>1.3.2 Variables</td>
<td>16</td>
</tr>
<tr>
<td>1.3.3 Functions</td>
<td>17</td>
</tr>
<tr>
<td>1.4 Modifying the layout</td>
<td>18</td>
</tr>
<tr>
<td>1.4.1 Add, delete and clear</td>
<td>18</td>
</tr>
<tr>
<td>1.5 Creating formulas</td>
<td>19</td>
</tr>
<tr>
<td>1.5.1 Function creator</td>
<td>20</td>
</tr>
<tr>
<td>1.5.2 Entering comment</td>
<td>25</td>
</tr>
<tr>
<td>1.5.3 Entering formulas</td>
<td>26</td>
</tr>
<tr>
<td>1.5.4 Problems in formulas</td>
<td>27</td>
</tr>
<tr>
<td>1.6 Formula menu</td>
<td>30</td>
</tr>
<tr>
<td>1.6.1 Load Formulas</td>
<td>30</td>
</tr>
<tr>
<td>1.6.2 Save Formulas</td>
<td>31</td>
</tr>
<tr>
<td>1.6.3 Print Formulas</td>
<td>31</td>
</tr>
<tr>
<td>1.6.4 Move Sheet</td>
<td>31</td>
</tr>
<tr>
<td><strong>2</strong> Formula Database Functions</td>
<td></td>
</tr>
<tr>
<td>2.1 General</td>
<td>33</td>
</tr>
<tr>
<td>2.2 Overview</td>
<td>34</td>
</tr>
<tr>
<td>2.2.1 Function Overview</td>
<td>34</td>
</tr>
<tr>
<td>2.2.2 Function Overview - HIC</td>
<td>40</td>
</tr>
<tr>
<td>2.2.3 Function Overview - Cycle</td>
<td>41</td>
</tr>
<tr>
<td><strong>3</strong> Arithmetic Operations</td>
<td></td>
</tr>
<tr>
<td>3.1 + (Addition)</td>
<td>43</td>
</tr>
<tr>
<td>3.2 - (Subtraction)</td>
<td>45</td>
</tr>
<tr>
<td>3.3 * (Multiplication)</td>
<td>47</td>
</tr>
<tr>
<td>3.4 / (Division)</td>
<td>49</td>
</tr>
<tr>
<td>3.5 – (Unary minus)</td>
<td>51</td>
</tr>
<tr>
<td>3.6 @Modulo</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>Reference Guide</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>4.1</td>
<td>@Abs</td>
</tr>
<tr>
<td>4.2</td>
<td>@ACosine</td>
</tr>
<tr>
<td>4.3</td>
<td>@And</td>
</tr>
<tr>
<td>4.4</td>
<td>@Area</td>
</tr>
<tr>
<td>4.5</td>
<td>@ASine</td>
</tr>
<tr>
<td>4.6</td>
<td>@ATan</td>
</tr>
<tr>
<td>4.7</td>
<td>@ATan2</td>
</tr>
<tr>
<td>4.8</td>
<td>@BlockFFT</td>
</tr>
<tr>
<td>4.9</td>
<td>@Clip</td>
</tr>
<tr>
<td>4.10</td>
<td>@Comparator</td>
</tr>
<tr>
<td>4.11</td>
<td>@Cos</td>
</tr>
<tr>
<td>4.12</td>
<td>@CurveFitting</td>
</tr>
<tr>
<td>4.13</td>
<td>@Cut</td>
</tr>
<tr>
<td>4.14</td>
<td>@Cycles</td>
</tr>
<tr>
<td>4.15</td>
<td>@Diff</td>
</tr>
<tr>
<td>4.16</td>
<td>@DQ0Transformation</td>
</tr>
<tr>
<td>4.17</td>
<td>@Energy</td>
</tr>
<tr>
<td>4.18</td>
<td>@EqualTo</td>
</tr>
<tr>
<td>4.19</td>
<td>@Exp</td>
</tr>
<tr>
<td>4.20</td>
<td>@ExpWave</td>
</tr>
<tr>
<td>4.21</td>
<td>@FallTime</td>
</tr>
<tr>
<td>4.22</td>
<td>@FilterButterworthLP</td>
</tr>
<tr>
<td>4.23</td>
<td>@FilterButterworthHP</td>
</tr>
<tr>
<td>4.24</td>
<td>@FilterButterworthBP</td>
</tr>
<tr>
<td>4.25</td>
<td>@FilterButterworthBS</td>
</tr>
<tr>
<td>4.26</td>
<td>@FilterBesselLP</td>
</tr>
<tr>
<td>4.27</td>
<td>@FilterBesselHP</td>
</tr>
<tr>
<td>4.28</td>
<td>@FilterBesselBP</td>
</tr>
<tr>
<td>4.29</td>
<td>@FilterBesselBS</td>
</tr>
<tr>
<td>4.30</td>
<td>@FilterChebyshevLP</td>
</tr>
<tr>
<td>4.31</td>
<td>@FilterChebyshevHP</td>
</tr>
<tr>
<td>4.32</td>
<td>@FilterChebyshevBP</td>
</tr>
<tr>
<td>4.33</td>
<td>@FilterChebyshevBS</td>
</tr>
<tr>
<td>4.34</td>
<td>@Frequency</td>
</tr>
<tr>
<td>4.35</td>
<td>@FormatDate</td>
</tr>
<tr>
<td>4.36</td>
<td>@FormatTime</td>
</tr>
<tr>
<td>Line</td>
<td>Function</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>4.37</td>
<td>@GreaterEqualThan</td>
</tr>
<tr>
<td>4.38</td>
<td>@GreaterThan</td>
</tr>
<tr>
<td>4.39</td>
<td>@Histogram</td>
</tr>
<tr>
<td>4.40</td>
<td>@IIF</td>
</tr>
<tr>
<td>4.41</td>
<td>@Integrate</td>
</tr>
<tr>
<td>4.42</td>
<td>@IntLookUp</td>
</tr>
<tr>
<td>4.43</td>
<td>@IntLookUp12</td>
</tr>
<tr>
<td>4.44</td>
<td>@isNaN</td>
</tr>
<tr>
<td>4.45</td>
<td>@Join</td>
</tr>
<tr>
<td>4.46</td>
<td>@Length</td>
</tr>
<tr>
<td>4.47</td>
<td>@LessEqualThan</td>
</tr>
<tr>
<td>4.48</td>
<td>@LessThan</td>
</tr>
<tr>
<td>4.49</td>
<td>@Ln</td>
</tr>
<tr>
<td>4.50</td>
<td>@Log</td>
</tr>
<tr>
<td>4.51</td>
<td>@Max</td>
</tr>
<tr>
<td>4.52</td>
<td>@MaxNum</td>
</tr>
<tr>
<td>4.53</td>
<td>@MaxPos</td>
</tr>
<tr>
<td>4.54</td>
<td>@Mean</td>
</tr>
<tr>
<td>4.55</td>
<td>@MedianFilter</td>
</tr>
<tr>
<td>4.56</td>
<td>@Min</td>
</tr>
<tr>
<td>4.57</td>
<td>@MinNum</td>
</tr>
<tr>
<td>4.58</td>
<td>@MinPos</td>
</tr>
<tr>
<td>4.59</td>
<td>@NextHillPos</td>
</tr>
<tr>
<td>4.60</td>
<td>@NextLvlCross</td>
</tr>
<tr>
<td>4.61</td>
<td>@NextValleyPos</td>
</tr>
<tr>
<td>4.62</td>
<td>@Noise</td>
</tr>
<tr>
<td>4.63</td>
<td>@Not</td>
</tr>
<tr>
<td>4.64</td>
<td>@Or</td>
</tr>
<tr>
<td>4.65</td>
<td>@Period</td>
</tr>
<tr>
<td>4.66</td>
<td>@Pow</td>
</tr>
<tr>
<td>4.67</td>
<td>@PrevHillPos</td>
</tr>
<tr>
<td>4.68</td>
<td>@PrevLvlCross</td>
</tr>
<tr>
<td>4.69</td>
<td>@PrevValleyPos</td>
</tr>
<tr>
<td>4.70</td>
<td>@Pulse</td>
</tr>
<tr>
<td>4.71</td>
<td>@PulseWidth</td>
</tr>
<tr>
<td>4.72</td>
<td>@Ramp</td>
</tr>
<tr>
<td>4.73</td>
<td>@ReadAsciiFile</td>
</tr>
<tr>
<td>Section</td>
<td>Function Name</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>4.74</td>
<td>@ReadLogFile</td>
</tr>
<tr>
<td>4.75</td>
<td>@Reduce</td>
</tr>
<tr>
<td>4.76</td>
<td>@RefCheck</td>
</tr>
<tr>
<td>4.77</td>
<td>@RelativeTime2Local</td>
</tr>
<tr>
<td>4.78</td>
<td>@RelativeTime2UTC</td>
</tr>
<tr>
<td>4.79</td>
<td>@RemoveGlitch</td>
</tr>
<tr>
<td>4.80</td>
<td>@Res2</td>
</tr>
<tr>
<td>4.81</td>
<td>@RiseTime</td>
</tr>
<tr>
<td>4.82</td>
<td>@RMS</td>
</tr>
<tr>
<td>4.83</td>
<td>@SAEJ211Filter</td>
</tr>
<tr>
<td>4.84</td>
<td>@Sin</td>
</tr>
<tr>
<td>4.85</td>
<td>@SineWave</td>
</tr>
<tr>
<td>4.86</td>
<td>@Smooth</td>
</tr>
<tr>
<td>4.87</td>
<td>@SpaceVectorInverseTransformation</td>
</tr>
<tr>
<td>4.88</td>
<td>@SpaceVectorTransformation</td>
</tr>
<tr>
<td>4.89</td>
<td>@Sqrt</td>
</tr>
<tr>
<td>4.90</td>
<td>@SquareWave</td>
</tr>
<tr>
<td>4.91</td>
<td>@StdDev</td>
</tr>
<tr>
<td>4.92</td>
<td>@STLX_SignalStart</td>
</tr>
<tr>
<td>4.93</td>
<td>@STLX_SignalEnd</td>
</tr>
<tr>
<td>4.94</td>
<td>@Sweep</td>
</tr>
<tr>
<td>4.95</td>
<td>@SweptSineWave</td>
</tr>
<tr>
<td>4.96</td>
<td>@Tan</td>
</tr>
<tr>
<td>4.97</td>
<td>@TimeMaxAbove</td>
</tr>
<tr>
<td>4.98</td>
<td>@TimeMaxAboveBegin</td>
</tr>
<tr>
<td>4.99</td>
<td>@TimeMaxBelow</td>
</tr>
<tr>
<td>4.100</td>
<td>@TimeMaxBelowBegin</td>
</tr>
<tr>
<td>4.101</td>
<td>@TimeMinAbove</td>
</tr>
<tr>
<td>4.102</td>
<td>@TimeMinAboveBegin</td>
</tr>
<tr>
<td>4.103</td>
<td>@TimeMinBelow</td>
</tr>
<tr>
<td>4.104</td>
<td>@TimeMinBelowBegin</td>
</tr>
<tr>
<td>4.105</td>
<td>@TimeTotalAbove</td>
</tr>
<tr>
<td>4.106</td>
<td>@TimeTotalAboveBegin</td>
</tr>
<tr>
<td>4.107</td>
<td>@TimeTotalBelow</td>
</tr>
<tr>
<td>4.108</td>
<td>@TimeTotalBelowBegin</td>
</tr>
<tr>
<td>4.109</td>
<td>@TriggerTime</td>
</tr>
<tr>
<td>4.110</td>
<td>@TriggerTimeToText</td>
</tr>
</tbody>
</table>
6.10 @CycleFundamentalRMS 301
6.11 @CycleInterval 303
6.12 @CycleLevel 305
6.13 @CycleMax 307
6.14 @CycleMean 308
6.15 @CycleMin 310
6.16 @CyclePeriod 311
6.17 @CyclePhase 312
6.18 @CycleRPM 314
6.19 @CycleRMS 315
6.20 @CycleStdDev 317
6.21 @CycleTHD 319
7 Pulse Measurement and Analysis 321
7.1 General 321
8 IIR Filters 325
8.1 Introduction 325
8.1.1 Bessel 326
8.1.1.1 Advantages 326
8.1.1.2 Disadvantages 327
8.1.2 Butterworth 327
8.1.2.1 Advantages 327
8.1.2.2 Disadvantages 327
8.1.3 Chebyshev (Type I) 327
8.1.3.1 Advantages: 327
8.1.3.2 Disadvantages: 327
8.1.4 Magnitude Spectrum 328
8.1.5 Impulse Response 331
8.1.6 Step Response 332
8.1.7 Phaseless Filtering 333
8.1.8 Importance of sampling rate and cutoff frequency 334
1 Analysis Option

1.1 Introduction
The Perception Analysis option allows you to perform calculations on measured data. A variety of built-in functions get you on the right track for analysis, ranging from basic statistics to advanced mathematics.

The analysis option comprises two main parts:

- A variety of functions
- The visible formula database

Using the formula database you can create your own set of additional functions without programming or sequencing. Just type in the required calculation and the result will be displayed. The results of new data is updated automatically, not only in the formula database but also directly on the displays and within the reports. Once defined the formulas can be saved for later use.

The formula database allows for an unlimited number of formulas, each with a name and units. A formula can be created using arithmetic operations on waveforms and scalars and combined with one of the built-in functions, cursor information, or the result of another formula. Auto-complete and in-line help guide you through the various options.

For ease of use we refer to the analysis option including the functions and the formula database as “the formula database”.

1.1.1 How to install the Analysis option
The Perception software requires a HASP key. HASP (Hardware Against Software Piracy) is a hardware-based (hardware key) software copy protection system that prevents unauthorized use of software applications. Each HASP key contains a unique ID number used for personalization of the application according to the features and options purchased. The key is also used for storing licensing parameters, applications and customer-specific data. If you have purchased the Analysis option as a separate item, you will receive a personalized "key file". Use this file to unlock the additional features.

You can find the serial number of your key in Help » About Perception

To update the key information:

1. Choose Help » Update Key...
2 In the Open dialog locate the Key File (*.pKey) and click **Open**.
3 If everything is OK you will see the following message:

![Perception software copy protection dialog]

**Figure 1.1:** Software copy protection dialog

4 **Click OK.**

After the installation you can go to **Help ▶ About Perception ▶ More...** to see all installed options.

You will need to restart the program before the changes take effect. The Analysis option is now available.
1.2 Formula database sheet
The Formula database sheet is used to create and edit formulas.

**Figure 1.2:** Formula database sheet

- **A** Wizard
- **B** Tools
- **C** Operators
- **D** Default help
- **E** Formula database row numbers with row pointer (function creator icon)
- **F** Formula name column
- **G** Formula column
- **H** Formula units
- **I** Additional help
A **Wizard** The function creator allows the user to enter and verify a function in a “guided” way. The function creator operates on the currently selected row. The currently selected row is denoted by the function creator icon in the row number column.

![Function creator icon](image)

**Figure 1.3:** Function creator icon

1 Calls on the function creator (see Figure 1.7 "Function Creator dialog" on page 21)

B **Tools** Tools are provided to add, delete and clear lines with formulas. The tools operate on the currently selected row. The currently selected row is denoted by the function creator icon in the row number column.

![Formula sheet tools](image)

**Figure 1.4:** Formula sheet tools

1 Insert row above current row
2 Insert row below current row
3 Delete row
4 Clear row

C **Operators** Use the operator buttons to insert basic operators.
Figure 1.5: Formula sheet operators

1. Add
2. Subtract
3. Multiply
4. Divide
5. Opening parenthesis
6. Closing parenthesis

**D** Default help This area displays basic help information on the selected function.

**E** Row numbers For easy reference each row has a row number. The currently active row has a function creator icon in front of the number.

**F-H** Formulas Each formula has a name, body and units. This area is also called the “formula editor”.

**I** Additional help For more detailed help information the additional help area is optionally available.

You may want to show or hide the additional help.

**To show or hide the additional help:**

- Click on the grip that is located on top of the formulas area.
1.3 Definitions
The formula database uses operators and functions. Operators require variables, functions require parameters.

1.3.1 Constants
Constants are numerical or string values which are entered straight into the formula. Numbers can be entered using floating point or integer notation. Internally all numerical values are floating point numbers. Strings are entered with quotation marks, like “text”.

1.3.2 Variables
Variables can be any variable (waveform, numeric or string) from the data source list. A variable can be selected using the data sources. Previously defined formulas are also available as variables.

Figure 1.6: Data Sources navigator with formula results

A Formula results

In the formula body a variable is referenced by the complete path of the variable in the data sources. For example in the diagram above the formula result T2 is referenced as Formula.T2
1.3.3 Functions
A function takes a number of parameters (which can be variables and / or constants) and uses them to produce a result. The type of the parameters and the type of the results depends on the function.

To enter a function:

1. Click once in the formula body of an empty row to select it or click twice in the formula body of an empty row to enter the edit mode.
2. Type the @ sign.
   - A list of available functions comes up.

3. To select a function from the list do the following:
   - Use the scroll bar to scroll through the list of available functions.
   - Click with the mouse on the function.
   - Use the arrow keys to select a function and press Enter.

The complete description of the selected function comes up in the Help area.

Note that as long as the formula has no name it is considered to be a comment line.
1.4 Modifying the layout

You can modify the layout of the rows by adding, deleting and clearing rows. You can use the tools directly or a right mouse click to call up a context menu with the same commands.

1.4.1 Add, delete and clear rows

You can add, delete and clear rows. You can do this on one row or on multiple rows.

To select multiple rows:

- To select consecutive rows:
  - Click the first row and drag to the last row.
  - Click the first row, press and hold down SHIFT, and then click the last row.
  - To select nonconsecutive rows, press and hold down CTRL, and then click each row.

To add one or more rows:

1. Select one or more rows.
   - The row pointer will move to the last selected row
2. Click on the appropriate tool:
   - Insert row(s) above to insert row(s) above the currently selected row.
   - Insert row(s) below to insert row(s) below the currently selected row.
   - Empty row(s) will be inserted.

To remove one or more rows:

Select one or more rows and do the following:

- To remove a row and the contents click the Delete selected row(s) tool.
- To remove the contents of the row but keep the row click the Clear selected row(s) tool.
- In the confirmation dialog that comes up click OK.

To clear a field:

You can clear a single field as follows:
1.4.2 Moving around

You can move through the fields by using the Tab and the Arrow keys.

- Select the field you want to clear and press Del
- Select the field you want to clear and press Enter: the field is open for editing and the complete text is selected. Press Del.
1.5 Creating formulas
Creating formulas and comment is easy. Using the function creator or various auto-completion techniques you are guided through the definition of formulas without too much typing. This system also reduces the possibility of errors.

The formula editor has a database alike layout with rows that represent records. Each row has three editable fields to enter comment and formulas.

1.5.1 Function creator
The function creator is a one-dialog feature that allows the user to enter and verify a function in a “guided” way.

To call on the function creator:

1. Select the row where you want to create or edit a function. There are two ways to open the function creator:
2A. Click on the function creator icon in the number column of the row.

2B. Click on the function creator icon in the Wizard panel (see Figure 1.2 "Formula database sheet" on page 13).
Figure 1.7: Function Creator dialog

A Default help
B Function name
C Function selection/search control
D Function units
E Function parameters
F Function result
G Show/hide preview
H OK button
I Cancel button

A Default help This area displays basic help information on the selected function.
B Function name The name of the function as it is shown in the data sources navigator.
C Function selection/search control Type in a part of the function’s name you are looking for. The functions containing the search query are listed.

![Automatically completed list of functions](image)

Figure 1.8: Function selection - Automatically completed list

The needed function can be selected with the mouse or the up- and down-arrow keys and confirmed with enter.

If the drop-down arrow of the search control is pressed, the available functions are shown per group.

![Drop-down arrow](image)

1 Drop-down arrow

D Function units The units of the function.

If the function has created its own unit (for example by performing an “Abs” on a waveform with the unit “Hz”), this is shown in the entry field as a default.

![Default function unit](image)

Figure 1.9: Default function unit
**Function parameters** If a function is selected, the parameters for this method are added here.

![Function parameters area](image)

**Figure 1.10:** Function parameters area

1. Required function parameter
   A parameter for the function which is required. This is indicated by the red color and the text "required".

2. Function parameter value
   Set the needed value for this parameter. Valid data sources containing the entered text will also be shown in the drop down list.

3. Data source button
   Opens the data source selection dialog, in which the valid data sources are shown. Calls on the default data source selector, which only shows the data sources which are allowed.

4. Invalid parameter indication
   If the value of the parameter is invalid, this is shown by a warning icon behind the parameter.

5. Optional function parameter
   A parameter for the function which is optional. This is indicated by the blue color and the text “optional”.
   Some functions allow you to add optional parameters. If you enter a value in the last optional parameter, a new optional parameter is added, and the previous parameter becomes required (see Figure 1.11).
Figure 1.11: Function parameters area with new optional parameter

F Function result  Shows the result of the function created by the function creator.

G Show/hide preview Expands or collapses the dialog to show or hide the preview.
There are three kinds of previews:
1 “Invalid” preview:

2 “Waveform” preview:
3 "Value" preview:

![Formula2]

3,142 Volt

**H** Click the **OK** button to apply the current function and to close the dialog.

**Note** The **OK** button is only enabled when the function is set up correctly. All required parameters and the function name must be filled in.

1 Click the **Cancel** button to discard the changes and to close the dialog.

### 1.5.2 Entering comment

You can enter comment in the formula editor to clarify the formulas you make. Comment is displayed in a green color. A comment row has no name.

**To enter comment:**

1 Click once in the formula body of an empty row to select it or click twice in the formula body of an empty row to enter the edit mode.

2 Start typing your text. Note that after typing the first character a list comes up:

![List of options]

3 Ignore this list. It will disappear when you entered your first word. Continue typing until done.

4 Press **Enter** when done or click anywhere else in the formula editor.
1.5.3 Entering formulas

Formulas are rules how to create new waveforms, numeric values or strings from existing data sources, variables and constants. A formula is a mathematical expression which can contain data sources, constants and functions. You can use brackets to change the precedence of the operators.

Formulas have a name, a body and optionally units. Formulas are displayed in blue (known keywords) and black.

To enter a formula:

Do the following:

1. Enter a name for the formula:
   a. Click once in the name field of an empty row to select it or click twice in the name field of an empty row to enter the edit mode.
   b. Enter a descriptive name.

2. Navigate to the body field of the formula as follows:
   a. Press the TAB.
   b. Press the right-arrow key.

3. Enter the body of the formula by using one or more of the following techniques:
   a. Type the @ sign to get a list of available functions and pick your function.
   b. Type any character: a drop down list comes up with a list of possible data sources roots. Select a root and type a dot: a new list drops down with branches from the selected root. Select a branch and type a dot, etc. until you have selected the variable or data source you want.
   c. Type in the full path of the variable / data source directly.
   d. Use operators and brackets to create more complex formulas.
   e. When available, help text is displayed in the Help area(s).

4. When done navigate to the Unit field and enter the units if applicable.

HINT/TIP

You can also drag data sources directly into a formula in the formula database. This allows you to quickly insert constants and variables into a function without the need to know the complete path of that variable. E.g simply drag a cursor X-position into your formula without typing the complete path like: Display.Display1.Cursor1.XPosition.
1.5.4 Problems in formulas

A lot of detailed input needs to be corrected when creating formulas in order for the formulas to work well. Typing errors or using incompatible types of input can cause problems in formulas. As the system of formulas grows, it becomes more difficult to pinpoint the cause of problems. To help find these problems in the formulas, the formula database automatically finds and marks most of the problems.

Problems found in the formulas are divided into two categories:

- **Errors**: These are problems that need to be solved actively.
- **Warnings**: These are problems in the formulas that could possibly be fixed at a later time automatically. For example, if input signals from a recording were missing, the problem would be solved simply by loading the recording into the system later.

Errors found in the functions are divided into two categories. The difference between the two types of errors is the timepoint when the error can be detected.

- **Syntax errors**: These errors can be detected when new formulas are added to the formula database. A common example of a syntax error is when incorrect separators are used between the parameters of a formula.

  **Note**  
  Typing errors can be avoided by using the function wizard.

- **Runtime errors**: These errors are detected when the result of a formula is used. This happens when the result of the formula is used on a display, on a meter, in a report, or by another formula in use. Runtime errors are typically more difficult to detect than syntax errors are, since more knowledge of the measurement, data-sources and formulas used is required. It can be difficult to determine if a formula or system of formulas contain an error, especially in a complex measurement environment.

Consider the following examples:

- **Syntax error**: Incorrect separator

![Figure 1.12: Syntax error (Incorrect separator)](image)

```
Num | Name      | Formula
--- | --------- |---------
1   | SyntaxErr | @SineWave(42;1k;7100)
```

A  Error message
SineWave normally uses three (3) arguments that are separated by a semi-colon. To illustrate this issue, the second semi-colon is mistyped as “??” in this example. The formula is now marked with an error icon. Additional information about the error can be retrieved by clicking on the icon. This will show the following detailed information for this example:

![Error Icon](image)

**Figure 1.13:** Formular Editor - Error message

- **Runtime error:** Incompatible sample rates

Runtime errors are only detected when a formula is used. Therefore, two erroneous formulas could be added, but only one of the erroneous formulas might be marked with an error. In the following example, two functions are created to add two waveforms. The error here is that the two waveforms are sampled at different speeds. Thus, the samples are not aligned and cannot be added. “Add1” (A) is used because it is shown on screen, whereas “Add2” (B) is defined but it has not used anywhere in Perception yet.

![Formulas](image)

**Figure 1.14:** Runtime error message

Even though both formulas are exactly the same, only the formula in use is marked with an error.

This problem is detected when the formula is used for the first time. It is very likely that the formula sheet is not visible at that time. If there is an error, no output is generated by the formula. This could be confusing. Therefore, Perception displays a notification in this case to provide as much feedback as possible.
This should enable you to find most common problems in the formulas.

**Note**  
*Not all problems in the formulas can be detected automatically. Perception can and will not show if a wrong signal was used as input for a formula when the incorrect signal is of the right type.*
1.6 **Formula menu**

The Formula menu lists commands related to Formula file handling. For layout and content management use the tools that are provided in the task pane on the left hand side of the formula editor.

The Formula menu is a dynamic menu and only available when the Formula sheet is on top, that is visible.

In the menu there are possibilities to save the formulas in a separate file. In general the formula database settings:

- comprises all formulas/functions as specified in the formulas sheet,
- can be stored in a separate file with the file extension `.pFormulas`
- are stored automatically when a workbench is saved and as part of a recording,
- are loaded automatically as part of a complete workbench,
- can be extracted / loaded out of a workbench or recording as separate settings,
- can be saved into a workbench or recording as separate settings.

![Formula menu](image)

**Figure 1.15:** Formula menu

### 1.6.1 Load Formulas

You can load formulas from a variety of sources.

**To load formulas:**

To load formulas from an external source proceed as follows:

1. Do one of the following:
   - In the Formula menu click **Load Formulas**....
   - When available in the toolbar click the **Load Formulas**... button 📧.
2 In the Load Formulas dialog that comes up select your file type if required:
   - Formula File (*.pFormulas)
   - Formulas out of a Virtual Workbench (*.pVWB)
   - Formulas out of an Experiment (*.PNRF)
3 Select the file you want to use.
4 Click Open.

1.6.2 Save Formulas
Much in the same way as you can load formulas you can also save formulas. You can also save into an existing virtual workbench or experiment. By doing so you will replace the formulas within that file. No other data will be altered.

To save formulas:

To save formulas into an external file proceed as follows:

1 Do one of the following:
   - In the Formulas menu click Save Formulas As....
   - When available in the toolbar click the Save Formulas As... buttons
2 In the Save Formulas As dialog that comes up select your file type if required:
   - Formula File (*.pFormulas)
   - Formulas out of a Virtual Workbench (*.pVWB)
   - Formulas out of an Experiment (*.PNRF)
3 Select the file you want to save into/replace or type a name for a new file.
4 Click Save.

1.6.3 Print Formulas
You can make a copy on the printer of the formulas.

To print a copy of the formulas:

1 Do one of the following:
   - In the Formulas menu click Print Formulas....
   - When available in the toolbar click the Print Formulas... button.
2 In the Print dialog that comes up enter your preferences.
3 Click Print.

1.6.4 Move Sheet
When the multiple workbook option is installed you can move the Formula sheet to another workbook.
To move the formula sheet to another workbook:

1. In the Formulas menu point to Move Sheet ‘Formula’ to ►
2. In the sub menu that comes up select a workbook.
2 Formula Database Functions

2.1 General
This document describes in full detail all functions available in the Perception formula database.

For a description of the formula database itself refer to the appropriate section in the User Manual.

Details on pulse characteristics, measurement and analysis can be found in chapter: “Pulse Measurement and Analysis “ on page 321.

Formulas in the formula database are defined as:

\[
\text{output} = \text{formula}
\]

The output is available in the Data Sources navigator as a variable. You can find it as an item in the Formula branch.

The output can be used as parameter in another function, regardless of the physical order in the database.

You can enter formulas just as you would record on paper. Standard mathematical rules are applicable.

Exception
No expressions are allowed as parameter(s) of an "@"-function.

Example:

Correct
\[
\begin{align*}
\text{AngleRad} &= \text{System.Constants.Pi} \ast \\
& \quad \text{Formula.Angle} / 180 \\
\text{CosAngle} &= \angle\cos(\text{Formula.AngleRad})
\end{align*}
\]

Incorrect
\[
\text{CosAngle} = \angle\cos(\text{System.Constants.Pi} \ast \\
& \quad \text{Formula.Angle} / 180)
\]

Note
All the functions work on static, single-time base, single-sweep data. The functions also work on multi-time base and/or multi-sweep data. However, the results may be unpredictable due to the nature of the function used. Some functions also work on dynamic (real-time) data. When applicable this is noted in the description of the function.
2.2 Overview
This section gives an alphabetical overview of all functions, including their name, a short description of the function and the page number.

2.2.1 Function Overview

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ (Add)</td>
<td>Add two expressions: &quot;+ (Addition)&quot; on page 43</td>
</tr>
<tr>
<td>- (Subtract)</td>
<td>Subtract two expressions: &quot;- (Subtraction)&quot; on page 45</td>
</tr>
<tr>
<td>* (Multiply)</td>
<td>Multiply two expressions: &quot;* (Multiplication)&quot; on page 47</td>
</tr>
<tr>
<td>/ (Divide)</td>
<td>Divide two expressions: &quot;/ (Division)&quot; on page 49</td>
</tr>
<tr>
<td>– (Invert)</td>
<td>Invert an expression: &quot;– (Unary minus)&quot; on page 51</td>
</tr>
<tr>
<td>@Modulo</td>
<td>Calculate the modulo: &quot;@Modulo&quot; on page 52</td>
</tr>
<tr>
<td>@Abs</td>
<td>Absolute value of a numerical value or a waveform: &quot;@Abs&quot; on page 53</td>
</tr>
<tr>
<td>@ACosine</td>
<td>Calculate the arccosine: &quot;@ACosine&quot; on page 54</td>
</tr>
<tr>
<td>@And</td>
<td>Logical AND: &quot;@And&quot; on page 55</td>
</tr>
<tr>
<td>@Area</td>
<td>Area under curve of a waveform: &quot;@Area&quot; on page 56</td>
</tr>
<tr>
<td>@ASine</td>
<td>Calculate the arcsine: &quot;@ASine&quot; on page 58</td>
</tr>
<tr>
<td>@ATan</td>
<td>Calculate the arctangent: &quot;@ATan&quot; on page 59</td>
</tr>
<tr>
<td>@ATan2</td>
<td>Calculate the arctangent of two parameters: &quot;@ATan2&quot; on page 60</td>
</tr>
<tr>
<td>@BlockFFT</td>
<td>Calculate the main frequency within a block of data: &quot;@BlockFFT&quot; on page 63</td>
</tr>
<tr>
<td>@Clip</td>
<td>Clip the amplitude of a waveform: &quot;@Clip&quot; on page 65</td>
</tr>
<tr>
<td>@Comparator</td>
<td>Compare values of two waveforms or one waveform and a numerical value: &quot;@Comparator&quot; on page 67</td>
</tr>
<tr>
<td>@Cos</td>
<td>Calculate the cosine: &quot;@Cos&quot; on page 69</td>
</tr>
<tr>
<td>@CurveFitting</td>
<td>Fit a linear or parabolic waveform to a given waveform: &quot;@CurveFitting&quot; on page 70</td>
</tr>
<tr>
<td>@Cut</td>
<td>Cut out a specific segment of a waveform: &quot;@Cut&quot; on page 72</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>@Cycles</td>
<td>Count the number of cycles in a waveform: &quot;@Cycles&quot; on page 74</td>
</tr>
<tr>
<td>@Diff</td>
<td>Differentiate a waveform: &quot;@Diff&quot; on page 76</td>
</tr>
<tr>
<td>@DQ0Transformation</td>
<td>Apply a DQ0Transformation on the input waveforms: &quot;@DQ0Transformation&quot; on page 78</td>
</tr>
<tr>
<td>@Energy</td>
<td>Calculate the energy under curve of a waveform: &quot;@Energy&quot; on page 82</td>
</tr>
<tr>
<td>@EqualTo</td>
<td>Equal-to evaluation: &quot;@EqualTo&quot; on page 84</td>
</tr>
<tr>
<td>@Exp</td>
<td>Exponential: calculate the power (base e) of the input: &quot;@Exp&quot; on page 85</td>
</tr>
<tr>
<td>@ExpWave</td>
<td>Generate an exponential waveform: &quot;@ExpWave&quot; on page 86</td>
</tr>
<tr>
<td>@FallTime</td>
<td>Determine the falltime of a pulse in a waveform: &quot;@FallTime&quot; on page 87</td>
</tr>
<tr>
<td>@FilterButterworthLP</td>
<td>Filters the input signal: &quot;@FilterButterworthLP&quot; on page 89</td>
</tr>
<tr>
<td>@FilterButterworthHP</td>
<td>Filters the input signal: &quot;@FilterButterworthHP&quot; on page 91</td>
</tr>
<tr>
<td>@FilterButterworthBP</td>
<td>Filters the input signal: &quot;@FilterButterworthBP&quot; on page 93</td>
</tr>
<tr>
<td>@FilterButterworthBS</td>
<td>Filters the input signal: &quot;@FilterButterworthBS&quot; on page 95</td>
</tr>
<tr>
<td>@FilterBesselLP</td>
<td>Filters the input signal: &quot;@FilterBesselLP&quot; on page 97</td>
</tr>
<tr>
<td>@FilterBesselHP</td>
<td>Filters the input signal: &quot;@FilterBesselHP&quot; on page 99</td>
</tr>
<tr>
<td>@FilterBesselBP</td>
<td>Filters the input signal: &quot;@FilterBesselBP&quot; on page 101</td>
</tr>
<tr>
<td>@FilterBesselBS</td>
<td>Filters the input signal: &quot;@FilterBesselBS&quot; on page 103</td>
</tr>
<tr>
<td>@FilterChebyshevLP</td>
<td>Filters the input signal: &quot;@FilterChebyshevLP&quot; on page 105</td>
</tr>
<tr>
<td>@FilterChebyshevHP</td>
<td>Filters the input signal: &quot;@FilterChebyshevHP&quot; on page 108</td>
</tr>
<tr>
<td>@FilterChebyshevBP</td>
<td>Filters the input signal: &quot;@FilterChebyshevBP&quot; on page 110</td>
</tr>
<tr>
<td>@FilterChebyshevBS</td>
<td>Filters the input signal: &quot;@FilterChebyshevBS&quot; on page 112</td>
</tr>
<tr>
<td>@FormatDate</td>
<td>Formats a date data source into a string: &quot;@FormatDate&quot; on page 117</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>@FormatTime</td>
<td>Formats a time data source into a string: &quot;@FormatTime&quot; on page 119</td>
</tr>
<tr>
<td>@Frequency</td>
<td>Determine the frequency of a waveform: &quot;@Frequency&quot; on page 115</td>
</tr>
<tr>
<td>@GreaterEqualThan</td>
<td>Greater-than-or-equal-to evaluation: &quot;@GreaterEqualThan&quot; on page 121</td>
</tr>
<tr>
<td>@GreaterThan</td>
<td>Greater-than evaluation: &quot;@GreaterThan&quot; on page 122</td>
</tr>
<tr>
<td>@Histogram</td>
<td>Calculate amplitude histogram: &quot;@Histogram&quot; on page 123</td>
</tr>
<tr>
<td>@IIF</td>
<td>Conditional result: &quot;@IIF&quot; on page 125</td>
</tr>
<tr>
<td>@Integrate</td>
<td>Integrate a waveform: &quot;@Integrate&quot; on page 127</td>
</tr>
<tr>
<td>@IntLookUp</td>
<td>Modify a waveform by using the waveform data as an index to an external conversion table: &quot;@IntLookUp&quot; on page 128</td>
</tr>
<tr>
<td>@IntLookUp12</td>
<td>Modify a waveform by using the waveform data as an index to a conversion table. Optimized for 12-bit data: &quot;@IntLookUp12&quot; on page 131</td>
</tr>
<tr>
<td>@IsNaN</td>
<td>Determine if the input parameter is not a number: &quot;@IsNaN&quot; on page 133</td>
</tr>
<tr>
<td>@Join</td>
<td>Concatenate two or more waveforms: &quot;@Join&quot; on page 134</td>
</tr>
<tr>
<td>@Length</td>
<td>Return the length (in samples) of a waveform: &quot;@Length&quot; on page 136</td>
</tr>
<tr>
<td>@LessEqualThan</td>
<td>Less-than-or-equal-to evaluation: &quot;@LessEqualThan&quot; on page 137</td>
</tr>
<tr>
<td>@LessThan</td>
<td>Less-than evaluation: &quot;@LessThan&quot; on page 138</td>
</tr>
<tr>
<td>@Ln</td>
<td>Calculate the natural logarithm: &quot;@Ln&quot; on page 139</td>
</tr>
<tr>
<td>@Log</td>
<td>Calculate the logarithm base 10: &quot;@Log&quot; on page 141</td>
</tr>
<tr>
<td>@Max</td>
<td>Determine the maximum value (amplitude) of a waveform: &quot;@Max&quot; on page 142</td>
</tr>
<tr>
<td>@MaxNum</td>
<td>Determine the maximum value of a range of numerical values: &quot;@MaxNum&quot; on page 144</td>
</tr>
<tr>
<td>@MaxPos</td>
<td>Return the position of the waveform maximum: &quot;@MaxPos&quot; on page 145</td>
</tr>
<tr>
<td>@Mean</td>
<td>Calculate the mean value of a waveform: &quot;@Mean&quot; on page 147</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>@MedianFilter</td>
<td>Filter a waveform using a median filter: &quot;@MedianFilter&quot; on page 149</td>
</tr>
<tr>
<td>@Min</td>
<td>Determine the minimum value (amplitude) of a waveform: &quot;@Min&quot; on page 152</td>
</tr>
<tr>
<td>@MinNum</td>
<td>Return the position of the waveform minimum: &quot;@MinNum&quot; on page 154</td>
</tr>
<tr>
<td>@MinPos</td>
<td>Return the position – in time – of the absolute minimum: &quot;@MinPos&quot; on page 155</td>
</tr>
<tr>
<td>@NextHillPos</td>
<td>Determine the position of the next local maximum in a waveform: &quot;@NextHillPos&quot; on page 157</td>
</tr>
<tr>
<td>@NextLvlCross</td>
<td>Determine the position of the next crossing of a waveform with a specified signal level: &quot;@NextLvlCross&quot; on page 159</td>
</tr>
<tr>
<td>@NextValleyPos</td>
<td>Determine the position of the next local minimum in a waveform: &quot;@NextValleyPos&quot; on page 161</td>
</tr>
<tr>
<td>@Noise</td>
<td>Generate a waveform containing noise: &quot;@Noise&quot; on page 163</td>
</tr>
<tr>
<td>@Not</td>
<td>Logical NOT: &quot;@Not&quot; on page 164</td>
</tr>
<tr>
<td>@Or</td>
<td>Logical OR: &quot;@Or&quot; on page 165</td>
</tr>
<tr>
<td>@Period</td>
<td>Determine the period of a waveform: &quot;@Period&quot; on page 166</td>
</tr>
<tr>
<td>@Pow</td>
<td>Exponentiation, base raised to the power exponent: &quot;@Pow&quot; on page 168</td>
</tr>
<tr>
<td>@PrevHillPos</td>
<td>Determine the position of the previous local maximum in a waveform: &quot;@PrevHillPos&quot; on page 169</td>
</tr>
<tr>
<td>@PrevLvlCross</td>
<td>Determine the position of the previous crossing of a waveform with a specified signal level: &quot;@PrevLvlCross&quot; on page 171</td>
</tr>
<tr>
<td>@PrevValleyPos</td>
<td>Determine the position of the previous local minimum in a waveform: &quot;@PrevValleyPos&quot; on page 173</td>
</tr>
<tr>
<td>@PulseWidth</td>
<td>Determine the width of a pulse in a waveform: &quot;@PulseWidth&quot; on page 177</td>
</tr>
<tr>
<td>@Ramp</td>
<td>Generate a linear ramp waveform: &quot;@Ramp&quot; on page 179</td>
</tr>
<tr>
<td>@ReadAsciiFile</td>
<td>Read waveform data from an ASCII (text) file: &quot;@ReadAsciiFile&quot; on page 181</td>
</tr>
<tr>
<td>@ReadLogFile</td>
<td>Read a sequence of numerical values from a Perception log file and creates a waveform: &quot;@ReadLogFile&quot; on page 183</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>@Reduce</td>
<td>Reduce the number of samples in a waveform by resampling: &quot;@Reduce&quot; on page 187</td>
</tr>
<tr>
<td>@RefCheck</td>
<td>Verify a waveform against one or two waveform envelopes: &quot;@RefCheck&quot; on page 188</td>
</tr>
<tr>
<td>@RelativeTime2Local</td>
<td>Returns a string representing the absolute local time corresponding to the elapsed time: &quot;@RelativeTime2Local&quot; on page 190</td>
</tr>
<tr>
<td>@RelativeTime2UTC</td>
<td>Returns a string representing the absolute UTC time corresponding to the elapsed time: &quot;@RelativeTime2UTC&quot; on page 192</td>
</tr>
<tr>
<td>@RemoveGlitch</td>
<td>Remove undesirable samples in a waveform: &quot;@RemoveGlitch&quot; on page 194</td>
</tr>
<tr>
<td>@Res2</td>
<td>Sample a waveform so that its length becomes a power of two: &quot;@Res2&quot; on page 195</td>
</tr>
<tr>
<td>@RiseTime</td>
<td>Determine the risetime of a pulse in a waveform: &quot;@RiseTime&quot; on page 196</td>
</tr>
<tr>
<td>@RMS</td>
<td>Calculate the Root Mean Square value of a waveform: &quot;@RMS&quot; on page 198</td>
</tr>
<tr>
<td>@SAEJ211Filter</td>
<td>Filter waveform based on SAE J211 recommendations: &quot;@SAEJ211Filter&quot; on page 199</td>
</tr>
<tr>
<td>@Sin</td>
<td>Calculate the sine: &quot;@Sin&quot; on page 200</td>
</tr>
<tr>
<td>@SineWave</td>
<td>Generate a sine wave: &quot;@SineWave&quot; on page 201</td>
</tr>
<tr>
<td>@Smooth</td>
<td>Smooth a waveform over a selected number of samples: &quot;@Smooth&quot; on page 202</td>
</tr>
<tr>
<td>@SpaceVectorInverseTransformation</td>
<td>Calculate a Space Vector Inverse waveform: &quot;@SpaceVectorInverseTransformation&quot; on page 204</td>
</tr>
<tr>
<td>@SpaceVectorTransformation</td>
<td>Calculate a Space Vector waveform: &quot;@SpaceVectorTransformation&quot; on page 208</td>
</tr>
<tr>
<td>@Sqrt</td>
<td>Calculate the square root: &quot;@Sqrt&quot; on page 211</td>
</tr>
<tr>
<td>@SquareWave</td>
<td>Generate a square wave: &quot;@SquareWave&quot; on page 212</td>
</tr>
<tr>
<td>@StdDev</td>
<td>Calculate the standard deviation of a waveform: &quot;@StdDev&quot; on page 213</td>
</tr>
<tr>
<td>@Sweep</td>
<td>Select a sweep in a multi-sweep recording: &quot;@Sweep&quot; on page 218</td>
</tr>
</tbody>
</table>
## Perception Analysis

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@SweptSineWave</td>
<td>Generates a waveform described by the sine function sweeping from a start frequency to an end frequency: &quot;@SweptSineWave&quot; on page 219</td>
</tr>
<tr>
<td>@Tan</td>
<td>Calculate the tangent: &quot;@Tan&quot; on page 221</td>
</tr>
<tr>
<td>@TimeMaxAbove</td>
<td>Return a number representing the time interval of the longest period that the signal is above a specified level: &quot;@TimeMaxAbove&quot; on page 222</td>
</tr>
<tr>
<td>@TimeMaxAboveBegin</td>
<td>Return a number representing the start time of the longest period that the signal is above a specified level: &quot;@TimeMaxAboveBegin&quot; on page 224</td>
</tr>
<tr>
<td>@TimeMaxBelow</td>
<td>Return a number representing the time interval of the longest period that the signal is below a specified level: &quot;@TimeMaxBelow&quot; on page 226</td>
</tr>
<tr>
<td>@TimeMaxBelowBegin</td>
<td>Return a number representing the start time of the longest period that the signal is below a specified level: &quot;@TimeMaxBelowBegin&quot; on page 228</td>
</tr>
<tr>
<td>@TimeMinAbove</td>
<td>Return a number representing the time interval of the shortest period that the signal is above a specified level: &quot;@TimeMinAbove&quot; on page 230</td>
</tr>
<tr>
<td>@TimeMinAboveBegin</td>
<td>Return a number representing the start time of the shortest period that the signal is above a specified level: &quot;@TimeMinAboveBegin&quot; on page 232</td>
</tr>
<tr>
<td>@TimeMinBelow</td>
<td>Return a number representing the time interval of the shortest period that the signal is below a specified level: &quot;@TimeMinBelow&quot; on page 234</td>
</tr>
<tr>
<td>@TimeMinBelowBegin</td>
<td>Return a number representing the start time of the shortest period that the signal is below a specified level: &quot;@TimeMinBelowBegin&quot; on page 236</td>
</tr>
<tr>
<td>@TimeTotalAbove</td>
<td>Return a number representing the total amount of time that the signal is above a specified level: &quot;@TimeTotalAbove&quot; on page 238</td>
</tr>
<tr>
<td>@TimeTotalAboveBegin</td>
<td>Return a number representing the start of the first period where the signal is above the specified level: &quot;@TimeTotalAboveBegin&quot; on page 240</td>
</tr>
<tr>
<td>@TimeTotalBelow</td>
<td>Return a number representing the total amount of time that the signal is below a specified level: &quot;@TimeTotalBelow&quot; on page 242</td>
</tr>
</tbody>
</table>
## Name Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>@TimeTotalBelowBegin</td>
<td>Return a number representing the start of the first period where the signal is below the specified level: &quot;@TimeTotalBelowBegin&quot;</td>
<td>244</td>
</tr>
<tr>
<td>@TriggerTime</td>
<td>Return the trigger position: &quot;@TriggerTime&quot; on page 246</td>
<td></td>
</tr>
<tr>
<td>@TriggerTimetoText</td>
<td>Return the trigger position in a time-date formatted string: &quot;@TriggerTimetoText&quot; on page 248</td>
<td></td>
</tr>
<tr>
<td>@TrueRMS</td>
<td>Calculate RMS value: &quot;@TrueRMS&quot; on page 252</td>
<td></td>
</tr>
<tr>
<td>@TrueRMSRef</td>
<td>Returns a waveform that represents the true RMS per number of cycles: &quot;@TrueRMSRef&quot; on page 253</td>
<td></td>
</tr>
<tr>
<td>@Value</td>
<td>Return the amplitude value of a waveform at a specified x-position: &quot;@Value&quot; on page 255</td>
<td></td>
</tr>
<tr>
<td>@XDelta</td>
<td>Return the sampling interval of a waveform: &quot;@XDelta&quot; on page 256</td>
<td></td>
</tr>
<tr>
<td>@XDeltaHigh</td>
<td>Return the maximum sampling interval in a multi-timebase recording: &quot;@XDeltaHigh&quot; on page 257</td>
<td></td>
</tr>
<tr>
<td>@XDeltaLow</td>
<td>Return the minimum sampling interval in a multi-timebase recording: &quot;@XDeltaLow&quot; on page 258</td>
<td></td>
</tr>
<tr>
<td>@XFirst</td>
<td>Return the x-coordinate (with respect to the trigger point) of the first sample in a waveform: &quot;@XFirst&quot; on page 259</td>
<td></td>
</tr>
<tr>
<td>@XLast</td>
<td>Return the x-coordinate (with respect to the trigger point) of the last sample in a waveform: &quot;@XLast&quot; on page 260</td>
<td></td>
</tr>
<tr>
<td>@XShift</td>
<td>Shift a waveform in time: &quot;@XShift&quot; on page 261</td>
<td></td>
</tr>
<tr>
<td>@XYArray</td>
<td>Create a waveform from a list of X/Y-value pairs: &quot;@XYArray&quot; on page 262</td>
<td></td>
</tr>
<tr>
<td>@YArray</td>
<td>Create a waveform from a list of Y-values: &quot;@YArray&quot; on page 264</td>
<td></td>
</tr>
</tbody>
</table>

### 2.2.2 Function Overview - HIC section

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Con3ms</td>
<td>Return the highest acceleration level: &quot;@Con3ms&quot; on page 266</td>
</tr>
<tr>
<td>@Con3ms_T</td>
<td>Return the start time of the highest acceleration level: &quot;@Con3ms_T&quot; on page 268</td>
</tr>
</tbody>
</table>
### 2.2.3 Function Overview - Cycle Math section

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@CycleArea</td>
<td>Calculate the area of every cycle detected on a reference signal: @CycleArea on page 286</td>
</tr>
<tr>
<td>@CycleCount</td>
<td>Count the number of cycles detected on a reference signal: @CycleCount on page 288</td>
</tr>
<tr>
<td>@CycleCrestFactor</td>
<td>Calculates the “Crest Factor” of every cycle detected on a reference signal: @CycleCrestFactor on page 289</td>
</tr>
<tr>
<td>@CycleDetect</td>
<td>Perform level crossing detection on the input waveform: @CycleDetect on page 291</td>
</tr>
<tr>
<td>@CycleEnergy</td>
<td>Calculate the energy of cycles detected on a reference signal: @CycleEnergy on page 293</td>
</tr>
<tr>
<td>@CycleFrequency</td>
<td>Calculate the frequency of cycles detected on the reference signal: @CycleFrequency on page 295</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>@CycleFundamental</td>
<td>Generate the Waveform of the Cycle Fundamental: &quot;@CycleFundamental&quot; on page 297</td>
</tr>
<tr>
<td>@CycleFundamental Phase</td>
<td>Calculate the phase difference between the Cycle Fundamentals: &quot;@CycleFundamentalPhase&quot; on page 299</td>
</tr>
<tr>
<td>@CycleInterval</td>
<td>Generate a square waveform with a specified interval between the high and low states: &quot;@CycleInterval&quot; on page 303</td>
</tr>
<tr>
<td>@CycleLevel</td>
<td>Return a waveform that uses the value of the input signal at the start of a cycle of the reference signal and repeat that value during half, one or multiple cycles of that reference signal: &quot;@CycleLevel&quot; on page 305</td>
</tr>
<tr>
<td>@CycleMax</td>
<td>Calculate the maximum value in cycles detected on a reference signal: &quot;@CycleMax&quot; on page 307</td>
</tr>
<tr>
<td>@CycleMean</td>
<td>Calculate the mean value in cycles detected on a reference signal: &quot;@CycleMean&quot; on page 308</td>
</tr>
<tr>
<td>@CycleMin</td>
<td>Calculate the minimum value in cycles detected on a reference signal: &quot;@CycleMin&quot; on page 310</td>
</tr>
<tr>
<td>@CyclePeriod</td>
<td>Calculate the period of cycles detected on a reference signal: &quot;@CyclePeriod&quot; on page 311</td>
</tr>
<tr>
<td>@CyclePhase</td>
<td>Calculates the phase difference $\theta$ between two waveforms: &quot;@CyclePhase&quot; on page 312</td>
</tr>
<tr>
<td>@CycleRPM</td>
<td>Calculates the revolutions per minute: &quot;@CycleRPM&quot; on page 314</td>
</tr>
<tr>
<td>@CycleTHD</td>
<td>Calculates the Total Harmonic Distortion: &quot;@CycleTHD&quot; on page 319</td>
</tr>
<tr>
<td>@CycleRMS</td>
<td>Calculate the root mean square value of cycles detected on a reference signal: &quot;@CycleRMS&quot; on page 315</td>
</tr>
<tr>
<td>@CycleStdDev</td>
<td>Calculate the standard deviation of every cycle detected on a reference signal: &quot;@CycleStdDev&quot; on page 317</td>
</tr>
</tbody>
</table>
3 Arithmetic Operations

3.1 + (Addition)

Function
Adds left and right expression.

Syntax
Expression1 + Expression2

Parameters
Expression1 Left expression
Expression2 Right expression

Output
The result is the sum of the left and right expression.

Description
An expression can be:

- Waveform variable
- Function call
- Numerical variable
- Constant value

Normal operator precedence (first multiplication and division, then addition and subtraction) applies. Parentheses can be used to change the operator precedence in more complicated expressions.

Example
Some examples of valid expressions using addition are:

- $2 + 3$
- $\text{var1} + 4$
- $5 + \text{var2}$
- $1000 + \text{@Noise}(1E6; 1000)$
- $(\text{Var1} + \text{Var2}) \times (\text{Var3} - \text{Var4})$

When adding numerical values, the output is a numerical value.
When adding a waveform and a numerical value, the output is a waveform. The number of points (length) of the output waveform is equal to the length of the input waveform.

When adding two waveforms, the output is a waveform formed by a point-by-point addition of the two input waveforms. The x-scaling of the output waveform is equal to the x-scaling of the left waveform. The length of the output waveform is equal to the shortest of the lengths of the two input waveforms. For a meaningful result, the x-scaling of both waveforms should be equal.

See Also
"* (Multiplication)" on page 47, "- (Subtraction)" on page 45, "/ (Division)" on page 49
3.2 - (Subtraction)

Function
Subtracts left and right expression.

Syntax
Expression1 – Expression2

Parameters
Expression1  Left expression
Expression2  Right expression

Output
The result is the difference of the left and right expression.

Description
An expression can be:
- Waveform variable
- Function call
- Numerical variable
- Constant value

Normal operator precedence (first multiplication and division, then addition and subtraction) applies. Parentheses can be used to change the operator precedence in more complicated expressions.

Example
Some examples of valid expressions using subtraction are:
- 3 - 2
- var1 - 4
- 5 - var2
- @SineWave(1E6; 1000; 1k) - 1
- (Var1 + Var2) * (Var3 - Var4)

When subtracting numerical values, the output is a numerical value.

When subtracting a waveform and a numerical value, the output is a waveform. The number of points (length) of the output waveform is equal to the length of the input waveform.
When subtracting two waveforms, the output is a waveform formed by a point-by-point subtraction of the two input waveforms. The x-scaling of the output waveform is equal to the x-scaling of the left waveform. The length of the output waveform is equal to the shortest of the lengths of the two input waveforms. For a meaningful result, the x-scaling of both waveforms should be equal.

See Also
"* (Multiplication)" on page 47, "+ (Addition)" on page 43, "/ (Division)" on page 49
3.3 * (Multiplication)

Function
Multiplies left and right expression.

Syntax
Expression1 * Expression2

Parameters
Expression1  Left expression
Expression2  Right expression

Output
The result is the product of the left and right expression.

Description
An expression can be:
- Waveform variable
- Function call
- Numerical variable
- Constant value

Normal operator precedence (first multiplication and division, then addition and subtraction) applies. Parentheses can be used to change the operator precedence in more complicated expressions.

Example
Some examples of valid expressions using multiplication are:
- 2 * 3
- var1 * 4
- 5 * var2
- 10 * @SineWave(1E6; 1000; 1k)
- (Var1 - AvgVar1) * (Var2 - AvgVar2)

When multiplying numerical values, the output is a numerical value.

When multiplying a waveform and a numerical value, the output is the scaled waveform. The number of points (length) of the output waveform is equal to the length of the input waveform.
When multiplying two waveforms, the output is a waveform formed by a point-by-point multiplication of the two input waveforms. The x-scaling of the output waveform is equal to the x-scaling of the left waveform. The length of the output waveform is equal to the shortest of the lengths of the two input waveforms. For a meaningful result, the x-scaling of both waveforms should be equal.

See Also
"+ (Addition)" on page 43, "- (Subtraction)" on page 45, "/ (Division)" on page 49
3.4  / (Division)

Function
Divides left and right expression.

Syntax
Expression1 / Expression2

Parameters
Expression1   Left expression (dividend)
Expression2   Right expression (divisor)

Output
The result is the quotient of the left and right expression.

Description
An expression can be:

- Waveform variable
- Function call
- Numerical variable
- Constant value

Normal operator precedence (first multiplication and division, then addition and subtraction) applies. Parentheses can be used to change the operator precedence in more complicated expressions.

Example
Some examples of valid expressions using division are:

- 3 / 2
- var1 / 4
- 5 / var2
- @SineWave(1E6; 1000; 1k) / 10
- (Var1 + Var2) / (Var3 - Var4)

When dividing numerical values, the output is a numerical value.

When dividing a waveform and a numerical value, the output is a waveform. The number of points (length) of the output waveform is equal to the length of the input waveform.
When dividing two waveforms, the output is a waveform formed by a point-by-point division of the two input waveforms. The x-scaling of the output waveform is equal to the x-scaling of the left waveform. The length of the output waveform is equal to the shortest of the lengths of the two input waveforms. For a meaningful result, the x-scaling of both waveforms should be equal.

Returns an undefined (unknown) value when the divisor is a numerical value equal to zero or when the denominator is a waveform containing a sample with the value zero.

See Also
"* (Multiplication)" on page 47, "+ (Addition)" on page 43, "- (Subtraction)" on page 45
3.5 – (Unary minus)

Function
Inverts the sign of an expression.

Syntax
– Expression

Parameters
Expression   Expression to be inverted

Output
The result is the expression multiplied by -1 (minus one).

Description
The expression can be any expression containing:
- Waveform variables
- Function calls
- Numerical variables
- Constant values

Normal operator precedence (first multiplication and division, then addition and subtraction) applies. Parentheses can be used to change the operator precedence in more complicated expressions.

Example
Some examples of valid expressions using the unary minus are:
- - 2
- - var1
- - @SineWave(1E6; 1000; 1k) - 1
- - ((Var1 + Var2) * (Var3 - Var4))

When inverting a numerical value, the output is a numerical value. When inverting a waveform, the output is the negated waveform. The number of points (length) of the output waveform is equal to the length of the input waveform.

See Also
"* (Multiplication)" on page 47, "+ (Addition)" on page 43, "- (Subtraction)" on page 45, "/ (Division)" on page 49
3.6 @Modulo

Function
Returns a waveform or numerical value with the modulo result.

Syntax
@Modulo(Par1, Par2)

Parameters
Par1 Input waveform or numerical value
Par2 Numerical value

Output
Waveform or numerical value containing the modulo operation of the input.

Description
The modulo operation divides the input with the supplied argument Par2. The remainder of that division is returned.

Example
The below would divide the input signal Formula.MyAngle by 360 and return the remainder. In this case it is ensured the angle signal never exceeds the 360 value.

```
Angle = @Modulo(Formula.MyAngle; 360 )
```
4 Reference Guide

4.1 @Abs

**Function**
Calculates the absolute value of the parameter.

**Syntax**
@Abs(Par)

**Parameters**

Par  
Input waveform or numerical value.

**Output**
Absolute value of the waveform or numerical value.

**Description**
Calculates the absolute value of the input waveform or numerical value. Positive values remain unchanged, negative values change their sign. This function can be used to rectify signals or to force positive values for results.

**Example**
The following example creates a sine wave and rectifies this signal:

```
Signal = @SineWave(1E6; 1000; 1k)
Rectif  = @Abs(Formula.Signal)
```
4.2 @ACosine

Function
Returns a waveform or numerical value representing the arccosine of a waveform or a number. The result is in radians.

Syntax
@ACosine(Input)

Parameters
Input
Input waveform or numerical value.

Output
Waveform or numerical value containing the arccosine of the input.

Description
The arccosine function returns the angle for which the cosine equals the argument. The angle returned is in radians. The arccosine is the inverse trigonometric function of the cosine function. That is the function:

\[ y = \arccos(x) \]

is defined so that:

\[ \sin(y) = x \]

The input domain of arccosine is:

\[-1 \leq x \leq 1\]

for real result values in the range:

\[0 \leq y \leq \pi\]

Example
The function below would calculate the angle in radians for which the cosine would amount to 0.

\[
\text{CosValue} = \@ACos(1)
\]

See Also
"@Cos" on page 69
4.3 @And

Function
Performs a logical AND evaluation on the input parameters.

Syntax
@And(Param1; …; ParamN)

Parameters
Param1 Number: first parameter used for the AND evaluation.
ParamN Last parameter used for the AND evaluation. With N >= 2.

Output
The output is 1 or a 0.

Description
The @And function performs a logical AND evaluation on the input parameters. Depending on the evaluation the result will be 1 or 0. A numerical value not equal to 0 corresponds to a logical “True” and a numerical 0 corresponds to a logical “False”.

The truth table of the AND function is:

<table>
<thead>
<tr>
<th>Param1</th>
<th>Param2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

Example
The following is a list of examples and their return value.

AndExampl1 = @And(1; 1; 1) => 1 (=true)
AndExampl2 = @And(1; 4; 10) => 1 (=true)
AndExampl3 = @And(1; 4; 0)  => 0 (=false)
AndExampl3 = @And(0; 0; 0)  => 0 (=false)

See Also
"@Not" on page 164 and "@Or" on page 165
4.4 @Area

Function
Calculates the area under curve of a waveform.

Syntax
@Area(Waveform)
@Area(Waveform; Begin)
@Area(Waveform; Begin; End)

Parameters
Waveform Input waveform for which the area under curve is to be calculated.
Begin Number: segment begin.
End Number: segment end.

Output
The output is a numerical value.

Description
The area under curve is calculated using the following formula:

\[
\text{Area} = \left[ \sum_{n=n_1}^{n_2} y(n) \right] - \frac{y(n_1) + y(n_2)}{2} \cdot \Delta x
\]

\[n_1 = \text{first sample with } x \geq \text{Begin}\]
\[n_2 = \text{last sample with } x \leq \text{End}\]
\[\Delta x = x - \text{difference between two samples}\]

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not as samples.

The numerical integration of the curve is performed assuming linear interpolation of the curve between the samples.
Example
The following example creates a sine wave of 50 Hz and calculates the area under curve of the first half period of the signal:

\[
\text{Signal} = \text{@SineWave(50k; 1000; 50)} \\
\text{Area} = \text{@Area(\text{Formula.Signal}; 0; 10m)}
\]

See Also
"@Energy" on page 82 and "@Mean" on page 147
4.5  @ASine

Function
Returns a waveform or numerical value representing the arcsine of a waveform or a number. The result is in radians.

Syntax
@ASine(Input)

Parameters
Input  Input waveform or numerical value.

Output
Waveform or numerical value containing the arcsine of the input.

Description
The arcsine function returns the angle for which the sine equals the argument. The angle returned is in radians. The arcsine is the inverse trigonometric function of the sine function. That is the function:

\[ y = \arcsine(x) \]

is defined so that:

\[ \sin(y) = x \]

The input domain of arcsine is:

\[-1 \leq x \leq 1\]

for real result values in the range:

\[-\pi/2 \leq y \leq \pi/2\]

Example
The function below would calculate the angle in radians for which the sine would amount to 1.

\[ \text{CosValue} = @ASine(1) \]

See Also
"@Sin" on page 200
4.6 @ATan

Function
Calculates the arctangent of the input parameter.

Syntax
@ATan(Par)

Parameters
Par Input waveform or numerical value.

Output
Waveform or numerical value containing the arctangent of the input.

Description
The arctangent function returns the angle for which the tangent equals the argument. The angle is returned in radians. The arctangent is the inverse trigonometric function of the tangent.

Example
The following example calculates Pi by multiplying ATan(1) by four:

\[ \pi = 4 \times \text{ATan}(1) \]

See Also
"@Cos" on page 69, "@Sin" on page 200 and "@Tan" on page 221
4.7 @ATan2

Function
Calculates the arctangent of the two input parameters.

Syntax
@ATan2(Waveform1; Par2)

Parameters
Waveform1 First input waveform.
Par2 Second input waveform or numerical value.

Output
A waveform containing the arctangent of the two input parameters (either two
input waveforms or an input waveform and a numerical value).
The values of the output waveform are in radians.

Description
The Atan2 function is the arctangent function with two arguments. The purpose
of using two arguments instead of one (Atan function) is to gather information
on the signs of the inputs in order to return the appropriate quadrant of the
computed angle, which is not possible for the single-argument arctangent
function.

For any arguments x and y not both equal to zero, atan2(y, x) is the angle in
radians between the positive x-axis of a plane and the point given by the
coordinates (x, y) on it. The angle is positive for counter-clockwise angles
(upper half-plane, y > 0), and negative for clockwise angles (lower half-plane,
y < 0).
Figure 4.1: Atan2 round a circle

Example
The following example calculates the angle of a point \((y,x)\) traveling a circle. The ‘\(y\)’ value of the point follows a sinus function (starting at 0). The ‘\(x\)’ value of the point follows a cosinus (starting at 1).

\[
\begin{align*}
\text{sig}_\text{sin} &= \text{@SineWave}(1k;1k+1;5) \\
\text{sig}_\text{cos} &= \text{@SineWave}(1k;1k+1;5;90) \\
\text{atan2} &= \text{@Atan2}(\text{Formula.sig}_\text{sin}; \text{Formula.sig}_\text{cos})
\end{align*}
\]
Result:

Figure 4.2: Result of ATan2 Example

See Also
"@ATan" on page 59
4.8  @BlockFFT

**Function**
Returns a waveform representing the maximum frequency detected per block of the input waveform.

**Syntax**
@BlockFFT(Waveform; Size; Space)
@BlockFFT(Waveform; Size; Space; Begin)
@BlockFFT(Waveform; Size; Space; Begin; End)

**Parameters**
- **Waveform**: Input waveform
- **Size**: Number: block size in milliseconds.
- **Space**: Number: spacing between start of two successive blocks in milliseconds.
- **Begin**: Number: start position of BlockFFT function
- **End**: Number: end position of BlockFFT function.

**Output**
Waveform containing frequency versus time.

**Description**
This function calculates the maximum frequency per block using an FFT algorithm. The parameter Size determines the size (length) of the block in milliseconds. The Space parameter determines the spacing between the start of two successive blocks in milliseconds.

Refer to the following diagram as an example for the relation between Space and Block.
Figure 4.3: Example - Relation between spacing and block size

The output of the function is a waveform containing a single maximum frequency per block as function of time. The sample spacing of this output waveform is equal to the Space parameter.

The segment limits (Begin and End) are used to select a range of the waveform in which BlockFFT’s are calculated. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Examples
The following example calculates the BlockFFT of a composed waveform:

\[
\begin{align*}
\text{Sine1} & = 2 \times \text{@SineWave}(1000k; 30k; 10k) \\
\text{Sine2} & = \text{@SineWave}(1000k; 30k; 15k) \\
\text{Sine3} & = 1.5 \times \text{@SineWave}(1000k; 30k; 5k) \\
\text{Signal} & = \text{@Join}(\text{Formula.Sine1}; \text{Formula.Sine2}; \text{Formula.Sine3}) \\
\text{Result} & = \text{@BlockFFT}(\text{Formula.Signal}; 4; 2)
\end{align*}
\]
4.9  @Clip

Function
Clips a waveform between the lower and upper bound specified.

Syntax
@Clip(Waveform; LowerBound; UpperBound)

Parameters
Waveform  Waveform whose amplitude range is to be clipped.
LowerBound Number: lower bound value used for clipping.
UpperBound Number: upper bound value used for clipping.

Output
Waveform with all sample values clipped between the lower and upper bound.

Description
For each sample in the waveform a comparison is made against the lower and upper bound of the clipping range.
If the sample value is between these two values, it is not modified.
If the sample value is larger than the upper bound, it is set to the upper bound.
If the sample value is lower than the lower bound, it is set to the lower bound.

Example
The example clips the 1.2 V sine wave to the limits -1 and 1 to simulate input overflow.

\[
\text{Signal} = 1.2 \times @\text{SineWave}(20k; 1000; 50) \\
\text{InpSignal} = @\text{Clip}(\text{Formula.Signal}; -1; 1)
\]

The following example determines the cumulative time an acceleration signal is above 150 g. The technique used is to first clip the signal between 150 g and 0.001 g more than 150 g. From the clipped signal 150 is subtracted and the clipped signal is scaled up by a factor of 1000. This technique results in 0 when the acceleration is below 150 g and in 1 when it is above 150 g. The area under this curve is the cumulative time above 150 g.

\[
\text{Accel} = 150 + @\text{SineWave}(20k; 100; 50) \\
\text{Temp} = 1000 \times (@\text{Clip}(\text{Formula.Accel}; 150; 150.001) - 150)
\]
CumTime = @Area(Formula.Temp)

See Also
"@Cut" on page 72
4.10  @Comparator

Function
Compared the sample values of two waveforms or of one waveform and a
numerical value.

Syntax
@Comparator(Waveform1; Par2)

Parameters
Waveform1  First input waveform.
Par         Second input waveform or numerical value.

Output
A waveform that is the result of the comparison between the sample values of
the first waveform and the second waveform or the numerical value.
When INP1 < INP2 then the output value is -1.
When INP1 is equal to INP2 then the output value is 0.
When INP1 > INP2 then the output value is 1.

Description
The Comparator function compares the two input waveforms with each other
and returns -1, 0 or 1 dependent on the relationship between the input values.

The second parameter can also be a numerical value. In that case this value is
used for the comparison.

Example
The following example calculates a waveform that indicates the relationship
between the two input signals.

    sig_sin = @SineWave(1k;1k+1;5)
    sig_cos = @SineWave(1k;1k+1;5;90)
    comp     = @Comparator(Formula.sig_sin; Formula.sig_cos)
Figure 4.4: Result of Comparator example
4.11  @Cos

Function
Calculates the cosine of the input parameter.

Syntax
@Cos(Par)

Parameters
Par  Input waveform or numerical value.

Output
Waveform or numerical value containing the cosine of the input.

Description
The trigonometric function cosine is calculated assuming the input parameter is the angle in radians. When a waveform parameter is used, the cosine is calculated for each individual sample.

Example
The following example calculates the cosine of the variable "Angle" specified in degrees:

\[
\begin{align*}
\text{Angle} & = 33 \\
\text{AngleRad} & = \text{System.Constants.Pi} * \text{Formula.Angle} / 180 \\
\text{CosAngle} & = @\text{Cos}(@\text{Formula.AngleRad})
\end{align*}
\]

See Also
"@ATan" on page 59, "@Sin" on page 200 and "@Tan" on page 221
4.12 @CurveFitting

Function
Returns a waveform that has the best fit to a series of data points from the input signal, using linear interpolation or a parabolic regression.

Syntax
@CurveFitting(Waveform)
@CurveFitting(Waveform; Order)
@CurveFitting(Waveform; Order; BeginIntv)
@CurveFitting(Waveform; Order; BeginIntv; EndIntv)
@CurveFitting(Waveform; Order; BeginIntv; EndIntv; Begin)
@CurveFitting(Waveform; Order; BeginIntv; EndIntv; Begin; End)

Parameters
Waveform  Input waveform
Order      Number: regression order. 1 = linear, 2 = parabolic. Default is linear regression.
BeginIntv Number: begin of the interval used for interpolation. Default the start of the input waveform is used.
EndIntv   Number: end of the interval used for interpolation. Default the end of input waveform is used.
Begin     Number: begin time from where the output curve starts. Default the start time of the input waveform is used.
End       Number: end time to where the output curve ends. Default the end time of the input waveform is used.

Output
A waveform being the linear or parabolic regressing of the input waveform or a part of it.

Description
This function seeks a curve that has the best fit to a series of data points from the input signal. Depending on the parameters all data points or just a limited interval of data points from the input signal are used.

If the Order parameter is set to 1 a linear regression will be done: the output signal will be a straight line.

If the Order parameter is set to 2 a parabolic regression will be done: the output signal will be a parabola.
The least squares algorithm is used to do the calculations.

Example
The following examples create a linear and a parabolic fit:

\[
\text{Signal} = \text{@SineWave(8000; 8001; 5)}
\]

\[
\text{LineFit} = \text{@CurveFitting(\text{Formula.Signal}; 1; 90m; 110m)}
\]

\[
\text{ParabolicFit} = \text{@CurveFitting(\text{Formula.Signal}; 2; 125m; 175m)}
\]
4.13  @Cut

Function
Cuts out a specific part of a waveform.

Syntax
@Cut(Waveform; Begin; End)

Parameters
Waveform  Input waveform from which a segment is to be selected.
Begin     Number: segment begin.
End       Number: segment end.

Output
Waveform segment.

Description
A specific part of a waveform can be selected for further processing. The function calculates the number of samples for the output based on the Begin and End values. The first sample is the sample with an x-coordinate nearest to Begin. The last sample is the sample with an x-coordinate nearest to End.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not as samples. If the segment limits are located outside the x-range of the waveform, these values are limited to that range.

Example
The following example cuts out the segment of a signal between 100 ms and 200 ms:

\[
\begin{align*}
\text{Signal} &= @\text{SineWave}(10000; 10000; 100) \\
\text{Segment} &= @\text{Cut}(\text{Formula.Signal}; 100m; 200m)
\end{align*}
\]

If a specific range of samples is required, the information functions can be used to calculate the proper range to select.

The following example selects the first 1024 points of the previous signal, and uses -1E20 as a very small value:

\[
\begin{align*}
\text{XEnd} &= @\text{XBegin}(\text{Formula.Signal}) \\
& \quad + 1023 \times @\text{XDelta}(\text{Formula.Signal})
\end{align*}
\]
First1024 = @Cut(Formula.Signal; -1E20; Formula.XEnd)

See Also
"@Join" on page 134, "@Length" on page 136, "@XFirst" on page 259, "@XDelta" on page 256 and "@XLast" on page 260
4.14  @Cycles

Function
Calculates the number of cycles in a waveform or waveform segment.

Syntax
@Cycles(Waveform)
@Cycles(Waveform; Begin; End)
@Cycles(Waveform; Begin)

Parameters
Waveform  Input waveform from which the cycles are calculated.
Begin  Number: segment begin
End  Number: segment end

Output
The number of cycles.

Description
The @Cycles function calculates the number of times a periodically repeated sequence of samples (=Cycle) occurs in the waveform. The 50% level between the maximum and minimum amplitude values is used to count the level crossings to get the number of cycles.

The segment limits (Begin and End) are used to select a range of samples in which cycles are calculated. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example: time) and not in samples.

Example
The following example calculates the number of cycles of a complete signal, and between the two cursors within a display named "display".

\[
\begin{align*}
\text{Signal} &= @\text{SineWave}(10k; 10001; 50) \\
\text{Cycles} &= @\text{Cycles}(\text{Formula.Signal}) \\
\text{Start} &= \text{Display.Display.Cursor1.XPosition}
\end{align*}
\]
End = Display.Display.Cursor2.XPosition
Cycles_BC = @Cycles(Formula.Signal;
                 Formula.Start; Formula.End)

See Also
"@Frequency" on page 115 and "@Period" on page 166
4.15 @Diff

Function
Differentiates a waveform.

Syntax
@Diff(Waveform)

Parameters
Waveform Waveform to be differentiated.

Output
Differentiated waveform.

Description
The derivative is a measurement of how a function changes when the values of its inputs change. Loosely speaking, a derivative can be thought of as how much a quantity is changing at some given point. The process of finding a derivative is called differentiation. Otherwise stated differentiation is used to determine the slope of a signal instead of the value. The slope is calculated by calculating the difference between adjacent samples and dividing this difference by the sampling interval:

\[ \text{Diff}(n) = \frac{y(n) - y(n-1)}{\Delta x} \quad \text{for } n = 2, ..., N \]

\[ \text{Diff}(1) = 0 \]

\[ \Delta x = x - \text{difference between two samples} \]
\[ N = \text{number of samples} \]

Differentiation enhances the high frequency components like high frequency noise and the digitization errors caused by rounding effects. To get a better estimate for the slope, smoothing of the waveform (either before or after differentiation) is recommended.

Example
The following example creates a noisy sine wave and differentiates this signal. The resulting waveform is smoothed to get a more accurate estimate of the slope of the sine wave.
Signal = @SineWave(10k; 1000; 50)
Differ = @Diff(Formula.Signal)
Slope = @Smooth(Formula.Differ; 7)

See Also
"@Integrate" on page 127 and "@Smooth" on page 202
4.16  **@DQ0Transformation**

**Function**
Applies the DQ0 Transformation on the input signals.

**Syntax**

@DQ0Transformation(Signal1; Signal2; Signal3; ReferenceAngle; Component; Transformation; OrientationRule)

**Parameters**

- **Signal1**  
  Input waveform of phase1 signal.

- **Signal2**  
  Input waveform of phase2 signal.

- **Signal3**  
  Input waveform of phase3 signal.

- **Reference Angle**  
  Waveform or Value of Reference Angle [rad]

- **Component**  
  Required output Component (0="direct", 1="quadrature", 2="zero")

- **Transformation Optional**  
  Invariant Transformation Rule (0="amplitude", 1="power") default is "amplitude"

- **Orientation Rule Optional**  
  Orientation Rule (0="wiki",1="matlab") default is "wiki"

**Output**
The waveform which results for the required component, transformation and required orientation rule transformations.

**Description**
The direct-quadrature-zero (DQ0) transformation (also known as Park’s transformation) is a mathematical transformation that is used to simplify the analysis of three-phase power systems.

The DQ0 Transformation is defined by the following matrix and can be applied for any three-phase quantities (e.g. voltage, currents, etc.)
For the “wiki” orientation rule, the following formula is applied for the amplitude-invariant transformation:

\[
\begin{bmatrix}
I_{Ad} \\
I_{aq} \\
I_{A0}
\end{bmatrix} = \frac{2}{3} \begin{bmatrix}
\cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\
-\sin(\theta) & -\sin(\theta - \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3}) \\
1/2 & 1/2 & 1/2
\end{bmatrix}
\begin{bmatrix}
I_u \\
I_v \\
I_w
\end{bmatrix}
\]

And for the power-invariant transformation, the following formula is applied:

\[
\begin{bmatrix}
I_{pd} \\
I_{pq} \\
I_{p0}
\end{bmatrix} = \frac{2}{\sqrt{3}} \begin{bmatrix}
\cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\
-\sin(\theta) & -\sin(\theta - \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3}) \\
1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2}
\end{bmatrix}
\begin{bmatrix}
I_u \\
I_v \\
I_w
\end{bmatrix}
\]

For the “Matlab” orientation rule, the following formula is applied for the amplitude-invariant transformation:

\[
\begin{bmatrix}
I_{Ad} \\
I_{aq} \\
I_{A0}
\end{bmatrix} = \frac{2}{3} \begin{bmatrix}
\sin(\theta) & \sin(\theta - \frac{2\pi}{3}) & \sin(\theta + \frac{2\pi}{3}) \\
\cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\
1/2 & 1/2 & 1/2
\end{bmatrix}
\begin{bmatrix}
I_u \\
I_v \\
I_w
\end{bmatrix}
\]

And for the power-invariant transformation, the following formula is applied:

\[
\begin{bmatrix}
I_{pd} \\
I_{pq} \\
I_{p0}
\end{bmatrix} = \frac{2}{\sqrt{3}} \begin{bmatrix}
\sin(\theta) & \sin(\theta - \frac{2\pi}{3}) & \sin(\theta + \frac{2\pi}{3}) \\
\cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\
1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2}
\end{bmatrix}
\begin{bmatrix}
I_u \\
I_v \\
I_w
\end{bmatrix}
\]
Where:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_u )</td>
<td>Value of the phase1 signal</td>
</tr>
<tr>
<td>( I_v )</td>
<td>Value of the phase2 signal</td>
</tr>
<tr>
<td>( I_w )</td>
<td>Value of the phase3 signal</td>
</tr>
<tr>
<td>( \theta )</td>
<td>Value of the reference angle</td>
</tr>
<tr>
<td>( I_{Ad} )</td>
<td>Value of the amplitude-invariant direct component</td>
</tr>
<tr>
<td>( I_{Aq} )</td>
<td>Value of the amplitude-invariant quadrature component</td>
</tr>
<tr>
<td>( I_{A0} )</td>
<td>Value of the amplitude-invariant zero component</td>
</tr>
<tr>
<td>( I_{Pd} )</td>
<td>Value of the power-invariant direct component</td>
</tr>
<tr>
<td>( I_{Pq} )</td>
<td>Value of the power-invariant quadrature component</td>
</tr>
<tr>
<td>( I_{P0} )</td>
<td>Value of the power-invariant zero component</td>
</tr>
</tbody>
</table>

Notes:
- The Transformation parameter is optional and has the default value 0="amplitude".
- The Orientation Rule parameter is optional and has the default value 0="wiki".
- The input waveforms (phase1, phase2 & phase3) all required the same sampling frequency. The same applies to the Reference Angle when it is a Waveform.
- This function is only available in the Formula database sheet (see "Formula database sheet" on page 13) if the eDrive option (1-PERC-OP-EDR) is part of your license.

Example

Given that the three-phase input signals are available as Formula.Phase1, Formula.Phase2, Formula.Phase3 and the Reference Waveform Formula.RefVal, the following DQ0 amplitude-invariant transformation waveforms can be defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQ0ATd</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;Formula.refAng;0)</td>
</tr>
<tr>
<td>DQ0ATq</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;Formula.refAng;1)</td>
</tr>
<tr>
<td>DQ0ATz</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;Formula.refAng;2)</td>
</tr>
</tbody>
</table>
The Reference Waveform may also be specified as a constant value:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQ0ATd_2</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3; System.Constants.Pi; 0)</td>
</tr>
</tbody>
</table>

Or the Reference Waveform has a value of 0 (zero):

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQ0ATd_3</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;0;0)</td>
</tr>
</tbody>
</table>

Likewise the power-invariant variations are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQ0PTd</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;Formula.refAng;0;1)</td>
</tr>
<tr>
<td>DQ0PTq</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;Formula.refAng;1;1)</td>
</tr>
<tr>
<td>DQ0PTz</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;Formula.refAng;2;1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQ0PTd_2</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;System.Constants.Pi; 0;1)</td>
</tr>
<tr>
<td>DQ0PTd_3</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3;0;0;1)</td>
</tr>
</tbody>
</table>

All of the above transformations use the default "wiki" Orientation Rule. For example, to use the "matlab" Orientation Rule for the power-invariant transformation, the following would be used:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQ0PTdm</td>
<td>@DQ0Transformation(Formula.phase1;Formula.phase2;Formula.phase3; Formula.refAng;0;1;1)</td>
</tr>
</tbody>
</table>

See Also
"@SpaceVectorTransformation" on page 208
@Energy

Function
Calculates the energy under curve of a waveform.

Syntax
@Energy(Waveform)
@Energy(Waveform; Begin)
@Energy(Waveform; Begin; End)

Parameters
Waveform Input waveform for which the energy under curve is to be calculated.
Begin Number: segment begin.
End Number: segment end.

Output
The output is a numerical value.

Description
The energy under curve is calculated using the following formula:

\[
\text{Energy} = \left[ \sum_{n=n_1}^{n_2} y^2(n) \right] - \frac{y^2(n_1) + y^2(n_2)}{2} \cdot \Delta x
\]

\[n_1 = \text{first sample with } x \geq \text{Begin}\]
\[n_2 = \text{last sample with } x \leq \text{End}\]
\[\Delta x = x - \text{difference between two samples}\]

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used. The numerical integration is performed assuming linear interpolation of the squared curve between the samples.

Example
The following example creates a sine wave of 50 Hz and calculates the energy in one period of the signal:
\[
\text{Signal} = \text{@SineWave}(20k; 1000; 50) \\
\text{E1} = \text{@Energy}(\text{Signal}; 0; 20m)
\]

See Also
"@Area" on page 56 and "@RMS" on page 198
@EqualTo

Function
This function performs an equal-to (=) evaluation on the two numerical input parameters.

Syntax
@EqualTo(Param1; Param2)

Parameters
- Param1: Number: first parameter used for evaluation
- Param2: Number: second parameter used for evaluation

Output
The output is 1 or 0

Description
The EqualTo function performs an 'equal-to' evaluation on the input parameters. If Param 1 = Param 2 then the return value will be 1 (true) else the return value will be 0 (false).

The EqualTo function is typically used in combination with the IIF function.

Example
The following example compares input parameters and provides a result that depends on the outcome:

\[
\begin{align*}
\text{EqualToExampl1} &= @\text{EqualTo}(5; 5) \Rightarrow 1 \text{ (true)} \\
\text{EqualToExampl2} &= @\text{EqualTo}(12; 10) \Rightarrow 0 \text{ (false)} \\
\text{IIFExample} &= @\text{IIF}(@\text{EqualToExampl2}; \text{"TRUE"}; \text{"FALSE"})
\end{align*}
\]

See Also
"@IIF" on page 125, "@GreaterEqualThan" on page 121, "@GreaterThan" on page 122, "@LessEqualThan" on page 137 and "@LessThan" on page 138
4.19  @Exp

Function
Exponential function, the mathematical operation, written \( e^n \), involving two parameters: the base "e" (2.7...) and the exponent "n".

Syntax
@Exp(Par)

Parameters
Par  
Input waveform or numerical value.

Output
Waveform or numerical value containing the exponentiation (base e) of the input.

Description
The exponential function calculates the power (base e) of the input. When a waveform parameter is used, the exponential function is calculated for each individual sample. This function is the inverse function of @Ln.

Example
The exponential function can also be achieved with the function @Pow. The following formulas are equivalent, assuming a formula database variable named Input:

\[
\text{Result} = \@Exp(\text{Formula.Input}) \\
\text{Result} = \@Pow(\text{System.Constants.e}; \text{Formula.Input})
\]

The following example shows an alternative for the system variable System.Constants.e:

\[
\text{Euler} = \@Exp(1)
\]

See Also
"@ExpWave" on page 86, "@Ln" on page 139 and "@Pow" on page 168
4.20  @ExpWave

Function
Generates a waveform containing an Exponential function.

Syntax
@ExpWave(Rate; Count; Alpha; Beta)
@ExpWave(Rate; Count; Alpha; Beta; X0)

Parameters
- Rate: Number: sampling frequency
- Count: Number of samples
- Alpha: Number: amplitude multiplier
- Beta: Number: exponent multiplier
- X0: Number: exponent multiplier modifier

Output
The output is a waveform containing an exponentially growing signal.

Description
This function generates a waveform using the formula:

\[ f(x) = \alpha \cdot e^{(\beta \cdot (x - x_0))} \]

The sampling frequency, number of samples, rate of growth and multiplication factor can be specified. The length of a generated waveform is limited to 1 GigaSamples (1 000 000 000).

The natural logarithm ln(x) is the inverse function of the exponential function. The possibility to generate an exponential wave function can be used to synthesize a variety of waveforms. The simulated data can be used as input for other analysis functions.

Example
The following example creates an exponential waveform.

\[
\text{Signal} = \text{@ExpWave}(1; 100; 2; 100m)
\]

See Also
"@Ln" on page 139, "@Pulse" on page 175, "@SineWave" on page 201 and "@SquareWave" on page 212
4.21  @FallTime

Function
Determines the fall time of a pulse in a waveform.

Syntax
@FallTime(Waveform)
@FallTime(Waveform; Begin)
@FallTime(Waveform; Begin; End)

Parameters
- Waveform: Input waveform containing the trailing edge of a pulse.
- Begin: Number: segment begin.
- End: Number: segment end.

Output
The output is a numerical value. This function works “while recording” when the Begin and End parameters are set. The result will be calculated when the data between Begin and End is available.

Description
The fall time is computed by taking the time difference between the distal point (90% magnitude transition) and the proximal point (10% magnitude transition) on the first trailing edge of a pulse in the waveform (or waveform segment).

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Example
The following example creates a pulse and calculates the 90%-10% fall time of the trailing edge. The result is 40 ms:

```
Sig1 = @Ramp(1k; 100; 0; 0)
Sig2 = @Ramp(1k; 51; 0; 10)
Sig3 = @Ramp(1k; 100; 10; 10)
Sig4 = @Ramp(1k; 51; 10; 0)
Signal = @Join(Formula.Sig1; Formula.Sig2; Formula.Sig3; Formula.Sig4; Formula.Sig1)
```
Falltime = @FallTime(Formula.Signal)

See Also
"@PulseWidth" on page 177 and "@RiseTime" on page 196
4.22  @FilterButterworthLP

Function
Filters the input signal with a direct form IIR lowpass Butterworth filter.

Syntax
@FilterButterworthLP(Signal; Order; Fc)
@FilterButterworthLP(Signal; Order; Fc; Phaseless)

Parameters
- **Signal**: Input waveform
- **Order**: Number: Filter order
- **Fc**: Number: The cut off Frequency in Hz
- **Phaseless**: Number: Filter method
  - 0  Filter does not work phase less (default)
  - 1  Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR lowpass Butterworth filtering.

The cut off frequency is the frequency at which the magnitude of the response is -3 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

Example

\[ \text{Signal} = @\text{FilterButterworthLP(Formula.Signal; 2; 200)} \]
If the Formula Signal is sampled at 20 kHz than the frequency and phase response looks like:

Figure 4.5: Magnitude Spectrum - FilterButterworthLP

Figure 4.6: Phase Response - FilterButterworthLP

See Also
"@FilterButterworthBS" on page 95, "@FilterButterworthBP" on page 93 and "@FilterButterworthHP" on page 91
4.23  @FilterButterworthHP

**Function**
Filters the input signal with a direct form IIR highpass Butterworth filter.

**Syntax**
@FilterButterworthHP(Signal; Order; Fc)
@FilterButterworthHP(Signal; Order; Fc; Phaseless)

**Parameters**
- **Signal**: Input waveform
- **Order**: Number: Filter order
- **Fc**: Number: The cut off Frequency in Hz
- **Phaseless**: Number: Filter method
  - 0  Filter does not work phase less (default)
  - 1  Filter works phase less

**Output**
The output is the filtered waveform.

**Description**
This function performs a direct form IIR highpass Butterworth filtering.

The cut off frequency is the frequency at which the magnitude of the response is -3 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

**Example**

```plaintext
Signal = @FilterButterworthHP(Formula.Signal; 2; 200)
```
If the *Formula.Signal* is sampled at 20 kHz than the frequency and phase response looks like:

**Figure 4.7:** Magnitude Spectrum - FilterButterworthHP

**Figure 4.8:** Phase Response - FilterButterworthHP

**See Also**
"@FilterButterworthBS" on page 95, "@FilterButterworthBP" on page 93 and "@FilterButterworthLP" on page 89
4.24  @FilterButterworthBP

Function
Filters the input signal with a direct form IIR bandpass Butterworth filter.

Syntax
@FilterButterworthBP(Signal; Order; LowFc; UpFc)
@FilterButterworthBP(Signal; Order; LowFc; UpFc; Phaseless)

Parameters
Signal  Input waveform
Order  Number: Filter order
LowFc  Number: The lower cut off Frequency in Hz
UpFc  Number: The upper cut off Frequency in Hz
Phaseless  Number: Filter method
            0  Filter does not work phase less (default)
            1  Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR bandpass Butterworth filtering.

The cut off frequencies are the frequencies at which the magnitude of the response is -3 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

Example
    Signal  = @FilterButterworthBP(Formula.Signal; 2; 200; 1000)
If the `Formula.Signal` is sampled at 20 kHz than the frequency and phase response looks like:

![Magnitude Spectrum](image1)

**Figure 4.9: Magnitude Spectrum - FilterButterworthBP**

![Phase Response](image2)

**Figure 4.10: Phase Response - FilterButterworthBP**

See Also
"`@FilterButterworthBS` on page 95, `@FilterButterworthHP` on page 91 and `@FilterButterworthLP` on page 89"
4.25  @FilterButterworthBS

Function
Filters the input signal with a direct form IIR bandstop Butterworth filter.

Syntax
@FilterButterworthBS(Signal; Order; LowFc; UpFc)
@FilterButterworthBS(Signal; Order; LowFc; UpFc; Phaseless)

Parameters
- **Signal**: Input waveform
- **Order**: Number: Filter order
- **LowFc**: Number: The lower cut off Frequency in Hz
- **UpFc**: Number: The upper cut off Frequency in Hz
- **Phaseless**: Number: Filter method
  - 0  Filter does not work phase less (default)
  - 1  Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR bandstop Butterworth filtering.

The cut off frequencies are the frequencies at which the magnitude of the response is -3 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

Example

```
Signal = @FilterButterworthBS(Formula.Signal; 2; 200; 1000)
```
If the \textit{Formula.Signal} is sampled at 20 kHz than the frequency and phase response looks like:

![Magnitude Spectrum](image)

**Figure 4.11**: Magnitude Spectrum - FilterButterworthBS

![Phase Response](image)

**Figure 4.12**: Phase Response - FilterButterworthBS

\textbf{See Also}

"@FilterButterworthBP" on page 93, "@FilterButterworthHP" on page 91 and "@FilterButterworthLP" on page 89
4.26  @FilterBesselLP

Function
Filters the input signal with a direct form IIR lowpass Bessel filter.

Syntax
@FilterBesselLP(Signal; Order; Fc)
@FilterBesselLP(Signal; Order; Fc; Phaseless)

Parameters
- **Signal**: Input waveform
- **Order**: Number: Filter order
- **Fc**: Number: The cut off Frequency in Hz
- **Phaseless**: Number: Filter method
  - 0  Filter does not work phase less (default)
  - 1  Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR lowpass Bessel filtering.

The cut off frequency is the frequency at which the magnitude of the response is -3 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

Example

```plaintext
Signal = @FilterBesselLP(Formula.Signal; 2; 200)
```
If the Formula.Signal is sampled at 20 kHz than the frequency and phase response looks like:

**Figure 4.13:** Magnitude Spectrum - FilterBesselLP

**Figure 4.14:** Phase Response - FilterBesselLP

See Also
"@FilterBesselBP" on page 101, "@FilterBesselBS" on page 103 and "@FilterBesselHP" on page 99
4.27 \texttt{@FilterBesselHP}

\textbf{Function}
Filters the input signal with a direct form IIR highpass Bessel filter.

\textbf{Syntax}
\texttt{@FilterBesselHP(Signal; Order; Fc)}
\texttt{@FilterBesselHP(Signal; Order; Fc; Phaseless)}

\textbf{Parameters}
- \textit{Signal} \hspace{1cm} Input waveform
- \textit{Order} \hspace{1cm} Number: Filter order
- \textit{Fc} \hspace{1cm} Number: The cut off Frequency in Hz
- \textit{Phaseless} \hspace{1cm} Number: Filter method
  - 0 \hspace{0.5cm} Filter does not work phase less (default)
  - 1 \hspace{0.5cm} Filter works phase less

\textbf{Output}
The output is the filtered waveform.

\textbf{Description}
This function performs a direct form IIR highpass Bessel filtering.

The cut off frequency is the frequency at which the magnitude of the response is -3 dB.

More detailed information can be found in the \textit{IIR Filters} chapter on page 325.

\textbf{Example}
\texttt{Signal = @FilterBesselHP(Formula.Signal; 2; 200)}
If the Formula.Signal is sampled at 20 kHz than the frequency and phase response looks like:

![Magnitude Spectrum](image1)

**Figure 4.15: Magnitude Spectrum - FilterBesselHP**

![Phase Response](image2)

**Figure 4.16: Phase Response - FilterBesselHP**

See Also

"@FilterBesselBP" on page 101, "@FilterBesselBS" on page 103 and "@FilterBesselLP" on page 97
4.28  @FilterBesselBP

Function
Filters the input signal with a direct form IIR bandpass Bessel filter.

Syntax
@FilterBesselBP(Signal; Order; LowFc; UpFc)
@FilterBesselBP(Signal; Order; LowFc; UpFc; Phaseless)

Parameters
Signal  Input waveform
Order   Number: Filter order
LowFc   Number: The lower cut off Frequency in Hz
UpFc    Number: The upper cut off Frequency in Hz
Phaseless  Number: Filter method
            0  Filter does not work phase less (default)
            1  Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR bandpass Bessel filtering.

The cut off frequencies are the frequencies at which the magnitude of the response is -3 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

Example
    Signal = @FilterBesselBP(Formula.Signal; 2; 200; 1000)
If the *Formula.Signal* is sampled at 20 kHz than the frequency and phase response looks like:

**Figure 4.17: Magnitude Spectrum - FilterBesselBP**

**Figure 4.18: Phase Response - FilterBesselBP**

See Also

"@FilterBesselBS" on page 103, "@FilterBesselHP" on page 99 and "@FilterBesselLP" on page 97
4.29  @FilterBesselBS

Function
Filters the input signal with a direct form IIR bandstop Bessel filter.

Syntax
@FilterBesselBS(Signal; Order; LowFc; UpFc)
@FilterBesselBS(Signal; Order; LowFc; UpFc; Phaseless)

Parameters
Signal  Input waveform
Order   Number: Filter order
LowFc   Number: The lower cut off Frequency in Hz
UpFc    Number: The upper cut off Frequency in Hz
Phaseless Number: Filter method
          0  Filter does not work phase less (default)
          1  Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR bandstop Bessel filtering.

The cut off frequencies are the frequencies at which the magnitude of the response is -3 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

Example
Signal = @FilterBesselBS(Formula.Signal; 2; 200; 1000)
If the *Formula.Signal* is sampled at 20 kHz than the frequency and phase response looks like:

![Magnitude Spectrum - FilterBesselBS](image1.png)

**Figure 4.19:** Magnitude Spectrum - FilterBesselBS

![Phase Response - FilterBesselBS](image2.png)

**Figure 4.20:** Phase Response - FilterBesselBS

**See Also**

[@FilterBesselBP](#) on page 101, [@FilterBesselHP](#) on page 99 and [@FilterBesselLP](#) on page 97
4.30 @FilterChebyshevLP

Function
Filters the input signal with a direct form IIR lowpass Chebyshev filter.

Syntax
@FilterChebyshevLP(Signal; Order; Fc)
@FilterChebyshevLP(Signal; Order; Fc; Ripple)
@FilterChebyshevLP(Signal; Order; Fc; Ripple; Phaseless)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>Input waveform</td>
</tr>
<tr>
<td>Order</td>
<td>Number: Filter order</td>
</tr>
<tr>
<td>Fc</td>
<td>Number: The cut off Frequency in Hz</td>
</tr>
<tr>
<td>Ripple</td>
<td>Number: The amplitude of the stop band ripple in decibels, the default value is 1 dB.</td>
</tr>
<tr>
<td>Phaseless</td>
<td>Number: Filter method</td>
</tr>
<tr>
<td></td>
<td>0 Filter does not work phase less (default)</td>
</tr>
<tr>
<td></td>
<td>1 Filter works phase less</td>
</tr>
</tbody>
</table>

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR lowpass Chebyshev filtering.

The cut off frequency is not the -3 dB point as defined at the Butterworth and Bessel filters but it is the highest frequency at which the magnitude of the response is equal to the specified ripple below the maximum magnitude. If for example the ripple is defined as 2 dB and Fc is 200 Hz than the magnitude at 200 Hz will be -2 dB, see Figure 4.21.
Figure 4.21: Magnitude Spectrum - FilterChebyshevLP (Ripple)

More detailed information can be found in the IIR Filters chapter on page 325.

Example

```plaintext
Signal = @FilterChebyshevLP
        (Formula.Signal; 2; 200; 2)
```
If the Formula.Signal is sampled at 20 kHz than the frequency and phase response looks like:

**Figure 4.22:** Magnitude Spectrum - FilterChebyshevLP

**Figure 4.23:** Phase Response - FilterChebyshevLP

See Also

"@FilterChebyshevBP" on page 110, "@FilterChebyshevBS" on page 112 and "@FilterChebyshevHP" on page 108
4.31  @FilterChebyshevHP

**Function**
Filters the input signal with a direct form IIR highpass Chebyshev filter.

**Syntax**
@FilterChebyshevHP(Signal; Order; Fc)
@FilterChebyshevHP(Signal; Order; Fc; Ripple)
@FilterChebyshevHP(Signal; Order; Fc; Ripple; Phaseless)

**Parameters**
- **Signal**: Input waveform
- **Order**: Number: Filter order
- **Fc**: Number: The cut off Frequency in Hz
- **Ripple**: Number: The amplitude of the stop band ripple in decibels, the default value is 1 dB.
- **Phaseless**: Number: Filter method
  - 0: Filter does not work phase less (default)
  - 1: Filter works phase less

**Output**
The output is the filtered waveform.

**Description**
This function performs a direct form IIR highpass Chebyshev filtering.

The cut off frequency is not the -3 dB point as defined at the Butterworth and Bessel filters but it is the lowest frequency at which the magnitude of the response is equal to the specified ripple below the maximum magnitude. If for example the ripple is defined as 2 dB and Fc is 200 Hz than the magnitude at 200 Hz will be -2dB. More detailed information can be found in the IIR Filters chapter on page 325.

**Example**
```plaintext
Signal = @FilterChebyshevHP(Formula.Signal; 2; 200)
```
If the Formula.Signal is sampled at 20 kHz than the frequency and phase response looks like:

**Figure 4.24:** Magnitude Spectrum - FilterChebyshev

**Figure 4.25:** Magnitude Spectrum - FilterChebyshev

**See Also**

"@FilterChebyshevBP" on page 110, "@FilterChebyshevBS" on page 112 and "@FilterChebyshevLP" on page 105
4.32  @FilterChebyshevBP

Function
Filters the input signal with a direct form IIR bandpass Chebyshev filter.

Syntax
@FilterChebyshevBP(Signal; Order; LowFc; UpFc)
@FilterChebyshevBP(Signal; Order; LowFc; UpFc; Ripple)
@FilterChebyshevBP(Signal; Order; LowFc; UpFc; Ripple; Phaseless)

Parameters
- **Signal**: Input waveform
- **Order**: Number: Filter order
- **LowFc**: Number: The lower cut off Frequency in Hz
- **UpFc**: Number: The upper cut off Frequency in Hz
- **Ripple**: Number: The amplitude of the stop band ripple in decibels, the default value is 1 dB.
- **Phaseless**: Number: Filter method
  - 0: Filter does not work phase less (default)
  - 1: Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR bandpass Chebyshev filtering. The cut off frequencies are not the -3d B points as defined at the Butterworth and Bessel filters but they are the lowest or highest frequencies at which the magnitude of the response is equal to the specified ripple below the maximum magnitude. If for example the ripple is defined as 2 dB and LowFc is 200 Hz than the magnitude at 200 Hz will be -2 dB.

More detailed information can be found in the IIR Filters chapter on page 325.

Example
```
Signal = @FilterChebyshevBP(Formula.Signal; 2; 200; 1000; 2)
```
If the Formula.Signal is sampled at 20 kHz than the frequency and phase response looks like:

![Magnitude Spectrum](image1.png)

**Figure 4.26: Magnitude Spectrum - FilterChebyshevBP**

![Phase Response](image2.png)

**Figure 4.27: Phase Response - FilterChebyshevBP**

See Also

"@FilterChebyshevBS" on page 112, "@FilterChebyshevHP" on page 108 and "@FilterChebyshevLP" on page 105
4.33  @FilterChebyshevBS

Function
Filters the input signal with a direct form IIR bandstop Chebyshev filter.

Syntax
@FilterChebyshevBS(Signal; Order; LowFc; UpFc)
@FilterChebyshevBS(Signal; Order; LowFc; UpFc; Ripple)
@FilterChebyshevBS(Signal; Order; LowFc; UpFc; Ripple; Phaseless)

Parameters
Signal   Input waveform
Order    Number: Filter order
LowFc    Number: The lower cut off Frequency in Hz
UpFc     Number: The upper cut off Frequency in Hz
Ripple   Number: The amplitude of the stop band ripple in decibels, the default value is 1 dB.
Phaseless Number: Filter method
   0  Filter does not work phase less (default)
   1  Filter works phase less

Output
The output is the filtered waveform.

Description
This function performs a direct form IIR bandstop Chebyshev filtering. The cut off frequencies are not the -3 dB points as defined at the Butterworth and Bessel filters but they are the lowest or highest frequencies at which the magnitude of the response is equal to the specified ripple below the maximum magnitude. If for example the ripple is defined as 2 dB and LowFc is 200 Hz than the magnitude at 200 Hz will be -2 dB.

If for example the ripple is defined as 2 dB and Fc is 200 Hz than the magnitude at 200 Hz will be -2 dB.

More detailed information can be found in the IIR Filters chapter on page 325.
Example

\[ \text{Signal} = @\text{FilterChebyshevBS}(\text{Formula}\_\text{Signal}; 2; 200; 1000) \]

If the \text{Formula}\_\text{Signal} is sampled at 20 kHz then the frequency and phase response looks like:

\begin{figure}
\centering
\includegraphics[width=\textwidth]{magnitude_spectrum.png}
\caption{Magnitude Spectrum - FilterChebyshevBS}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{phase_response.png}
\caption{Phase Response - FilterChebyshevBS}
\end{figure}
See Also
"@FilterChebyshevBP" on page 110, "@FilterChebyshevHP" on page 108 and "@FilterChebyshevLP" on page 105
4.34  @Frequency

Function
Determines the frequency of a waveform.

Syntax
@Frequency(Waveform)
@Frequency(Waveform; Begin)
@Frequency(Waveform; Begin; End)

Parameters
Waveform  Input waveform for which the frequency is to be determined.
Begin     Number: segment begin.
End       Number: segment end.

Output
The output is a numerical value. This function works also “while recording” when both the Begin and End parameters are set. The result will be calculated continuously from the data that is readily available.

Description
The frequency of the waveform or waveform segment is determined. The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.

The frequency is determined as follows:
- First the minimum and maximum values of the waveform are determined. The mean of these is considered to be the zero level and the number of zero crossings in the range Begin to End is calculated. The crossings should be in the same direction as the first one. The crossings are determined using a +/-5% hysteresis around the zero level in order to suppress the effects of noise.
- From the time difference between the first and last zero crossing in the same direction and the number of zero crossings in between, the period of the waveform is determined. Frequency is the reciprocal of the period.
Example

The following example determines the frequency of a 50 Hz signal:

\[
\begin{align*}
\text{Signal} & = \text{SineWave}(20k; 1000; 50) \\
\text{Freq} & = \text{Frequency}(\text{Formula.Signal})
\end{align*}
\]

See Also

"@Period" on page 166
4.35  @FormatDate

Function
Returns a date data source as a string, this date string can be formatted.

Syntax
@FormatDate(Date; Format)

Parameters
Date        Input date data source.
Format      Date format string.

Output
A string representing the input date in a user specified format.

Description
This function is used to format a date data source. The input data source can be the UTCDate of the RecordingInformation of an opened recording.

The format string can use the following formatting codes:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>display month as a numerical (1 .. 12)</td>
</tr>
<tr>
<td>MM</td>
<td>display month as a numerical (01 .. 12)</td>
</tr>
<tr>
<td>MMM</td>
<td>display three-letter month (Apr)</td>
</tr>
<tr>
<td>MMMM</td>
<td>display complete name of month (April)</td>
</tr>
<tr>
<td>d</td>
<td>display day of the MONTH (1 .. 31)</td>
</tr>
<tr>
<td>dd</td>
<td>display day of the MONTH (01 .. 31)</td>
</tr>
<tr>
<td>ddd</td>
<td>display three-letter day of the WEEK (Tue)</td>
</tr>
<tr>
<td>dddd</td>
<td>display complete name of day of the WEEK (Tuesday)</td>
</tr>
<tr>
<td>yy</td>
<td>display two-digit year (15)</td>
</tr>
<tr>
<td>yyyy</td>
<td>display four-digit year (2015)</td>
</tr>
</tbody>
</table>

An example of some formulas:

Name          | Formula                                           |
--------------|---------------------------------------------------|
FormattedDate_1 = @FormatDate(Active.Recording Information.UTCDate;“ddd dMM-yyy”)
FormattedDate_2 = @FormatDate(Active.Recording Information.UTCDate;“dd-MM-yyyy”)
The formulas above give the following results:

<table>
<thead>
<tr>
<th>Data source</th>
<th>Output</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormattedDate_1</td>
<td>Tue 20JAN-2015</td>
<td>&quot;ddd ddMMyyyy&quot;</td>
</tr>
<tr>
<td>FormattedDate_2</td>
<td>200115</td>
<td>&quot;dd-MM-yy&quot;</td>
</tr>
<tr>
<td>FormattedDate_3</td>
<td>January-20-2015</td>
<td>&quot;MMMM-d-yyyy&quot;</td>
</tr>
</tbody>
</table>

See Also
"@FormatTime" on page 119
4.36  @FormatTime

**Function**
Returns a time data source as a string, this time string can be formatted.

**Syntax**
@FormatTime(Time; Format)

**Parameters**
- **Time**: Input time data source.
- **Format**: Time format string.

**Output**
A string representing the input time in a user specified format.

**Description**
This function is used to format a time data source. The input data source can be the UTCTime of the RecordingInformation of an opened recording. The format string can use the following formatting codes:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HH:mm</td>
<td>13:47</td>
</tr>
<tr>
<td>HH:mm:ss</td>
<td>13:47:08</td>
</tr>
<tr>
<td>hh:mm:ss</td>
<td>01:47:08</td>
</tr>
<tr>
<td>h:mm:ss</td>
<td>1:47:08</td>
</tr>
<tr>
<td>t</td>
<td>1:47 PM</td>
</tr>
<tr>
<td>T</td>
<td>1:47:08 PM</td>
</tr>
</tbody>
</table>

An example of some formulas:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormattedTime_1</td>
<td>@FormatTime(Active.RecordingInformation.UTCTime;&quot;HH:mm:ss&quot;)</td>
</tr>
<tr>
<td>FormattedTime_2</td>
<td>@FormatTime(Active.RecordingInformation.UTCTime;&quot;T&quot;)</td>
</tr>
</tbody>
</table>

The formula above give the following result:

<table>
<thead>
<tr>
<th>Data source</th>
<th>Output</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormattedDate_1</td>
<td>10:08:37</td>
<td>&quot;HH:mm:ss&quot;</td>
</tr>
<tr>
<td>FormattedDate_2</td>
<td>10:08:37 AM</td>
<td>&quot;T&quot;</td>
</tr>
</tbody>
</table>
See Also
"@FormatDate" on page 117
4.37 @GreaterEqualThan

Function
This function performs a greater-than-or-equal-to (≥) evaluation on the two numerical input parameters.

Syntax
@GreaterEqualThan(Param1; Param2)

Parameters
Param 1 Number: first parameter used for evaluation.
Param 2 Number: second parameter used for evaluation.

Output
The output is 1 or 0.

Description
The GreaterEqualThan function performs a "greater-than-or-equal-to" evaluation on the input parameters.

If Param 1 ≥ Param 2 then the return value will be 1 (= true) else the return value will be 0 (=false).

The GreaterEqualThan function is typically used in combination with the IIF function.

Example
The following example compares input parameters and provides a result that depends on the outcome:

GETExam1 = @GreaterEqualThan(5; 5) => 0 (false)
GETExam2 = @GreaterEqualThan(12; 10) => 1 (true)
IIFExample = @IIF(Formula.GETExam2; "TRUE"; "FALSE")

See Also
"@EqualTo" on page 84, "@GreaterThan" on page 122, "@IIF" on page 125, "@LessEqualThan" on page 137 and "@LessThan" on page 138
4.38  @GreaterThan

Function
This function performs a greater-than (>) evaluation on the two numerical input parameters.

Syntax
@GreaterThan(Param1; Param2)

Parameters

- **Param1**: Number: first parameter used for evaluation.
- **Param2**: Number: second parameter used for evaluation.

Output
The output is 1 or 0.

Description
The GreaterThan function performs a 'greater-than' evaluation on the input parameters.

If Param 1 > Param 2 then the return value will be 1 (true) else the return value will be 0 (false).

The GreaterThan function is typically used in combination with the IIF function.

Example
The following example compares input parameters and provides a result that depends on the outcome:

```
GTExam1 = @GreaterThan(5; 100) => 0 (false)
GTExam2 = @GreaterThan(12; 10) => 1 (true)
IIFExample = @IIF(Formula.GTExam2; "TRUE"; "FALSE")
```

See Also
"@EqualTo" on page 84, "@GreaterEqualThan" on page 121, "@IIF" on page 125, "@LessEqualThan" on page 137 and "@LessThan" on page 138
4.39  @Histogram

Function
Calculates amplitude histogram.

Syntax
@Histogram(Waveform; YLow; YHigh; Number)

Parameters
- **Waveform**: Input waveform from which the histogram is to be calculated.
- **YLow**: Number: lowest amplitude to include in the histogram.
- **YHigh**: Number: highest amplitude to include in the histogram.
- **Number**: Number of divisions.

Output
Waveform containing the amplitude histogram.

Description
The histogram function determines the number of occurrences of amplitude values (values of a waveform) in each amplitude division. The amplitude divisions are specified by dividing the range between YLow and YHigh into Number divisions. The width of each division is (YHigh - YLow)/Number. Each value of the waveform is sorted to one of these divisions. The histogram shows how many values were sorted in each division. The maximum allowed number of divisions is 4096. Typically values between 100 and 1000 are used.

The parameter YLow and YHigh determine the range of amplitudes included in the histogram. Any value outside this range is discarded. To ensure that out-of-range values are included in a division of the histogram, use the @Clip function to limit the amplitude values before calculating the histogram.

Example
The following example calculates an amplitude histogram between -5.5 V and +5.5 V using 11 divisions. This means each division has a width of 1 V.

\[
\text{Signal} = 5 \times \text{@SineWave}(5000; 1000; 10) \\
\text{Histo} = \text{@Histogram(Formula.Signal; -5.5; 5.5; 11)}
\]

The following example calculates a histogram of 100 divisions between the minimum and maximum of the waveform, assuming a formula database variable Signal:
MinSignal = @Min(Formula.Signal)
MaxSignal = @Max(Formula.Signal)
Histo = @Histogram(Formula.Signal;
    Formula.MinSignal;
    Formula.MaxSignal; 100)

See Also
"@Clip" on page 65, "@Max" on page 142 and "@Min" on page 152
4.40  @IIF

Function
This function performs an Immediate If (IIF) and returns one value if a condition evaluates to TRUE and another value if it evaluates to FALSE.

Syntax
@IIF(Param; IfTrue; IfFalse)

Parameters
- **Param**: Waveform/Number/String to be evaluated.
- **IfTrue**: Waveform/Number/String output if the evaluation of Param returns true.
- **IfFalse**: Waveform/Number/String output if the evaluation of Param returns false.

Output
The output can be a waveform, number or string.

Description
The IIF function performs an evaluation on the first input parameter; the sort of evaluation depends on the type of this parameter and is described in the table below. The evaluation will be done and the result is either true or false. If the result is true then the second parameter will be returned as output of the IIF function else the third parameter will be returned.

The following table describes the evaluation of the first parameter:

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>Not 0</td>
<td>Equal to 0</td>
</tr>
<tr>
<td>String</td>
<td>Not empty</td>
<td>Empty</td>
</tr>
<tr>
<td>Waveform</td>
<td>Not Null</td>
<td>Null</td>
</tr>
</tbody>
</table>

Example
The following example uses a generated sine wave. If this signal is shown in the display with name Display than the IIF function will return a string “Above 0.5” if the waveform value at the active cursor position is above 0.5 else the string “Below 0.5” will be returned.

The GreaterThan function is used to compare the waveform value at the active cursor position against the 0.5 value. The result is 1 (true) if the value is greater than 0.5 else the result is 0 (false). This result is the first parameter of the IIF function.
Signal = @SineWave(8000; 8001; 5)
GreaterHalf = @GreaterThan
    (Display.Display.ActiveCursor.YValue; 0.5)
IIFExample = @IIF(Formula.GreaterHalf; "Above 0.5"; "Below 0.5")

See Also
"@EqualTo" on page 84, "@GreaterEqualThan" on page 121, "@GreaterThan" on page 122, "@LessEqualThan" on page 137 and "@LessThan" on page 138
4.41  @Integrate

Function
Integrates a waveform.

Syntax
@Integrate(Waveform)

Parameters
Waveform       Waveform to be integrated.

Output
Integrated waveform.

Description
Integration calculates the running sum of all sample values:

\[
\begin{align*}
  i(1) &= y(1) \\
  i(n) &= i(n-1) + y(n) \cdot \Delta x \quad \text{for } n = 2, \ldots, N
\end{align*}
\]

The integral calculates the area under curve of the waveform from the starting point up to the end point.

Example
The following example integrates the instantaneous power to obtain energy as a function of time:

\[
\begin{align*}
  \text{Signal} &= \@SineWave(10k; 1000; 50) \\
  \text{Power} &= (\text{Formula.Signal} \ast \text{Formula.Signal}) / 600 \\
  \text{Energy} &= \@Integrate(\text{Formula.Power})
\end{align*}
\]

See Also
"@Diff" on page 76 and "@Energy" on page 82
4.42 @IntLookUp

Function
Uses the data from a waveform as an index (pointer) into a Conversion Look Up Table (CLUT). The CLUT is stored in a separate file.

Syntax
@IntLookUp(Waveform; CLUT)
@IntLookUp(Waveform; CLUT; IndexOffset)

Parameters
Waveform Input waveform
CLUT String: the full path giving the name and the unique location of the CLUT-file.
IndexOffset Number: offset to the index pointer.

Output
Converted waveform.

Description
This function uses the data from a waveform as an index to a Conversion Look Up Table. The data can be 16-bit integer.

The Conversion Look Up Table (CLUT) is stored in an ASCII (text) file.

Each line in this file contains a floating-point number. The line number is used as the index for the CLUT.

The first line is associated by default with index 0. If the function however uses the optional 'IndexOffset' parameter then the first line is associated with – IndexOffset. This allows for working with negative values.

When using 16-bit integer data, the CLUT needs to contain $2^{16} = 65536$ points. When the data is signed (positive and negative) use an offset of 32768. Now the first point in the CLUT is associated with –32768.

Example
Assume the following look-up table:
<table>
<thead>
<tr>
<th>Line Number</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.3</td>
</tr>
<tr>
<td>2</td>
<td>21.4</td>
</tr>
<tr>
<td>3</td>
<td>31.5</td>
</tr>
<tr>
<td>4</td>
<td>41.4</td>
</tr>
<tr>
<td>5</td>
<td>51.2</td>
</tr>
<tr>
<td>6</td>
<td>61.3</td>
</tr>
<tr>
<td>7</td>
<td>71.5</td>
</tr>
<tr>
<td>8</td>
<td>81.6</td>
</tr>
<tr>
<td>9</td>
<td>91.2</td>
</tr>
<tr>
<td>10</td>
<td>101.4</td>
</tr>
<tr>
<td>11</td>
<td>111.2</td>
</tr>
</tbody>
</table>

Assume the original waveform to be:

6, 2, 7, 8

The first value 6 is associated with line 7 (index is zero-based). The corresponding value from the CLUT table is 71.5

The converted output waveform looks like:

71.5, 31.5, 81.6, 91.2

If the optional "IndexOffset" parameter is used and set to 2:

The first value 6 is associated with line 5 (index 6 – 2, zero-based). The corresponding value from CLUT table is 51.2

The converted output waveform looks like:

51.2, 11.3, 61.3, 71.5

If the input waveform contains an index greater than the maximum index stored in the CLUT then the last value of the CLUT is used. If the input waveform contains an index smaller than the minimum index stored in the CLUT then the first value of the CLUT is used.

To use the external lookup table "Lookup.asc" located in the directory "C:\CalibData":
Signal = @Ramp(1000; 2001; 0; 4096; 11)
Calib = @IntLookUp
  (Formula.Signal; "C:\CalibData\Lookup.asc")

See Also
"@IntLookUp12" on page 131
4.43 @IntLookUp12

Function
Uses the data from a waveform as an index (pointer) into a Conversion Look Up Table (CLUT).

Syntax
@IntLookUp12(Waveform; Clut)

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform</td>
<td>Waveform to be converted.</td>
</tr>
<tr>
<td>Clut</td>
<td>Conversion Look Up Table</td>
</tr>
</tbody>
</table>

Output
Converted waveform.

Description
This function uses the data from a waveform as an index (pointer) to a Conversion Look Up Table.

Note
This function is optimized for 12-bit integer data.

The Conversion Look Up Table has 4096 values.

Each value of the original waveform is used as a pointer into the CLUT. The corresponding value in the CLUT is used to produce the output.

The CLUT can be created:

- using a high level programming language and the Perception CSI option
- with a (combination of) formula database function(s)

The following diagram shows an example of the use of this function.
The top trace shows the original recording.

The middle trace shows the contents of the CLUT.

The bottom trace shows the converted data.

To create this diagram the following formulas are used:

\[
\begin{align*}
\text{CLUT} &= @\text{SineWave}(200k; 4096; 48.8) \\
\text{Org} &= @\text{Ramp}(2M; 40960; 4095; 10) \\
\text{Calib} &= @\text{IntLookUp12}(\text{Formula.Org}; \text{Formula.CLUT})
\end{align*}
\]

First a sine wave is generated of 4096 points with a frequency of 48.8 Hz. With a sample rate of 200 kHz one cycle fits in the waveform.

The result named “Calib” is produced by taking the generated waveform Formula.Org and using this waveform as an index for the CLUT.

**Note**

The original data should be integer data (2 bytes) in range -32767 to + 32768 with 12-bit resolution. Before the data is used, the integer value is divided by 16 to obtain the correct index. Therefore also 14- or 16-bit data can be used, but this will result in a loss of the least significant bits and may produce unpredictable or unexpected results.

**See Also**

"@IntLookUp" on page 128
4.44  @IsNaN

Function
This function determines if the input parameter is Not a Number (NaN).

Syntax
@IsNaN(Param)

Parameters
Param          Number: parameter to be validated.

Output
The Output is 1 or 0.

Description
When the input parameter is Not a Number (NaN) then the return value will be 1 (true) else the return value will be 0 (false).

Example
The following example will return the value 1 (true):

    isValid = @IsNaN(10.0/0.0)

See Also
"@GreaterThan" on page 122, "@LessThan" on page 138, "@EqualTo" on page 84, "@GreaterEqualThan" on page 121 and "@LessEqualThan" on page 137
4.45  @Join

**Function**
Joins two or more waveforms by concatenation.

**Syntax**
@Join(Waveform1; ..., WaveformN)

**Parameters**
- *Waveform1*  First waveform.
- *WaveformN*  Last waveform, with N >= 2

**Output**
Concatenated waveform.

**Description**
This function concatenates two or more waveforms to form a new waveform. The waveforms are joined by concatenation of the end of a waveform with the begin of the next waveform. The horizontal axis is determined by the first waveform. The x-coordinate of the first sample of the output waveform is equal to the x-coordinate of the first sample of the first waveform. The samples from the other waveform(s) are simply concatenated to the samples of the first waveform. The length of the output waveform is the sum of the lengths of the waveforms specified as parameters.

When the input waveforms have different time bases, the output waveform will also contain multiple time bases.

**Example**
This function is especially useful for generating simulated signals as shown in the following example:

\[
\begin{align*}
\text{Seg1} &= @\text{Ramp}(1k; 100; 0; 0) \\
\text{Seg2} &= @\text{Ramp}(1k; 50; 0; 1) \\
\text{Seg3} &= @\text{Ramp}(1k; 100; 1; 1) \\
\text{Signal} &= @\text{Join} (\text{Formula.Seg1}; \text{Formula.Seg2}; \text{Formula.Seg3})
\end{align*}
\]

The Join function takes the first valid data point of a waveform and concatenates that to the previous waveform. This means that the original end and start time of the two concatenated waveforms may differ. However, in the output waveform they will be one (1) sample interval apart.
The following example illustrates this:

\[
\text{Seg}1 = \text{@Ramp}(1k; 100; 0; 0)
\]
\[
\text{Seg}2 = \text{@Ramp}(1k; 100; 0; 0)
\]

Shift second segment by 100 horizontal units using @XShift:

\[
\text{Seg}3 = \text{@XShift(Formula.Seg2; 100)}
\]
\[
\text{Signal} = \text{@Join(Formula.Seg1; Formula.Seg3)}
\]

See Also
"@Cut" on page 72
4.46  @Length

Function
Returns the number of samples in a waveform.

Syntax
@Length(Waveform)

Parameters
Waveform  Input waveform

Output
The output is a numerical value.

Description
This function returns the number of samples that is available within the specified input waveform. This value can be used as input parameter for other functions.

Example
The following example creates a sine wave consisting of 1000 samples. The variable NSamples receives the length of this signal (1000).

\[
\text{Signal} = 25 \times \@SineWave(20k; 1000; 50) \\
\text{NSamples} = \@Length(\text{Formula.Signal})
\]

See Also
"@XDelta" on page 256, "@XFirst" on page 259 and "@XLast" on page 260
4.47 @LessEqualThan

Function
This function performs a less-than-or-equal-to (≤) evaluation on the two numerical input parameters.

Syntax
@LessEqualThan (Param1; Param2)

Parameters
- **Param1** Number: first parameter used for evaluation.
- **Param2** Number: second parameter used for evaluation.

Output
The output is 1 or 0.

Description
The LessEqualThan function performs a 'less-than-or-equal-to' evaluation on the input parameters.

If Param 1 ≤ Param 2 then the return value will be 1 (true) else the return value will be 0 (false).

The LessEqualThan function is typically used in combination with the IIF function.

Example
The following example compares input parameters and provides a result that depends on the outcome:

\[
\begin{align*}
\text{LETExam1} &= @\text{LessEqualThan}(5; 5) \Rightarrow 1 \text{ (true)} \\
\text{LETExam2} &= @\text{LessEqualThan}(12; 10) \Rightarrow 0 \text{ (false)} \\
\text{IIFExample} &= @\text{IIF(Formula.LETExam2; "true"; "false")}
\end{align*}
\]

See Also
"@EqualTo" on page 84, "@GreaterEqualThan" on page 121, "@GreaterThan" on page 122, "@IIF" on page 125 and "@LessThan" on page 138
4.48 @LessThan

Function
This function performs a less-than (<) evaluation on the two numerical input parameters.

Syntax
@LessThan(Param1; Param2)

Parameters

Param1       Number: first parameter used for evaluation.
Param2       Number: second parameter used for evaluation.

Output
The output is 1 or 0.

Description
The LessThan function performs a 'less-than' evaluation on the input parameters. If Param 1 < Param 2 then the return value will be 1 (true) else the return value will be 0 (false).

The LessThan function is typically used in combination with the IIF function.

Example
The following example compares input parameters and provides a result that depends on the outcome:

LessThanExample1 = @LessThan(5; 100) => 1 (true)
LessThanExample2 = @LessThan(12; 10) => 0 (false)
IIFExample   = @IIF(Formula.LessThanExampl2; "TRUE"; "FALSE")

See Also
"@EqualTo" on page 84, "@GreaterEqualThan" on page 121, "@IIF" on page 125 and "@LessEqualThan" on page 137
4.49 @Ln

Function
Natural logarithm of a given number, (or logarithm to base “e”), written as ln(x) or log_e(x): the power to which you need to raise the base (e) in order to get the number.

Syntax
@Ln(Par)

Parameters
Par Input waveform or numerical value.

Output
Waveform or numerical value containing the natural logarithm of the input.

Description
The logarithm base e is calculated (e = 2.718...). This logarithm is called the natural logarithm. When a waveform parameter is used, the log function is calculated for each individual sample.

If a number is smaller than 1E-30, the value is set to ln(1E-30) = -69.08.

Formally, ln(a) may be defined as the area under the graph (integral) of 1/x from 1 to a, that is:

\[ \ln(a) = \int_{1}^{a} \frac{1}{x} \, dx. \]

This function is the inverse function of @Exp.

Example
The following example shows that the natural logarithm of the system variable System.Constants.e equals 1:

One = @Ln(System.Constants.e)

The following example shows how the natural logarithm can be used on a waveform:
Wave     = @Ramp(10k; 1k; 1E-5; 10)
WaveLn   = @Ln(Formula.Wave)

See Also
"@Exp" on page 85 and "@Log" on page 141
4.50  @Log

Function
Logarithm of a given number to base 10, written as $\log_{10}(x)$: the power to which you need to raise the base (10) in order to get the number.

Syntax
@Log(Par)

Parameters
Par  Input waveform or numerical value.

Output
Waveform or numerical value containing the logarithm of the input.

Description
The logarithm base 10 is calculated. When a waveform parameter is used, the log function is calculated for each individual sample.

If a number is smaller than 1E-30, the value is set to $\log(1E^{-30}) = -30$.

Example
If $10^x = 100$ then $x = \log_{10}(100)$

$$x = \text{@Log}(100)$$

The result will be $x = 2$

See Also
"@Ln" on page 139 and "@Pow" on page 168
Function
Determines the maximum value of a waveform.

Syntax
@Max(Waveform)
@Max(Waveform; Begin)
@Max(Waveform; Begin; End)

Parameters
Waveform Input waveform for which the maximum is to be determined.
Begin Number: segment begin.
End Number: segment end.

Output
The output is a numerical value. This function works also “while recording” when both the Begin and End parameters are set. The result will be calculated continuously from the data that is readily available.

Description
The maximum of the waveform or waveform segment is determined. The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not as samples.

Example
The following example creates a 50 Hz sine wave. The maximum value of the first 20 ms (one period of 50 Hz) of the signal is calculated to determine the (peak) amplitude:

\[
\text{Signal} = 25 \times \text{@SineWave(20k; 1000; 50)}
\]
\[
\text{Ampl} = \text{@Max(Formula.Signal; 0; 20m)}
\]

If the position of the maximum is to be determined, use the @MaxPos function instead. The maximum value at the position found can then be obtained using the @Value function:
TMax = @MaxPos(Formula.Signal; 0; 20m)
Ampl = @Value(Formula.Signal; Formula.TMax)

See Also
"@MaxPos" on page 145, "@Min" on page 152 and "@MinPos" on page 155
"@Value" on page 255
Function
Determines the maximum value of two or more numerical values.

Syntax
@MaxNum(Par1; ...; ParN)

Parameters
Par1 First numerical value.
ParN Last numerical value, N >= 2.

Output
Maximum of all numerical values.

Description
This function returns the largest numerical value of all parameters specified. Each parameter must be a numerical value containing a valid value. If one of the parameters has the incorrect type or contains no value, the function returns no value.

Example
The following example calculates the minimum of a waveform but clips the value if it is below zero. The waveform is assumed to be a real-world signal:

Signal = Active.Group1.Recorder_A.Ch_A1
MinValue = @Min(Formula.Signal)
ClippedMin = @MaxNum(Formula.MinValue; 0)

The following example returns 10:

Ten = @MaxNum(-10; 3; 2.4; 10.0; -9.9)

See Also
"@MinNum" on page 154
4.53 @MaxPos

Function
Determines the position of the maximum of a waveform.

Syntax
@MaxPos(Waveform)
@MaxPos(Waveform; Begin)
@MaxPos(Waveform; Begin; End)

Parameters
Waveform Input waveform for which the position of the maximum is to be determined.
Begin Number: segment begin.
End Number: segment end.

Output
The output is the x-position of the maximum. This function works also “while recording” when both the Begin and End parameters are set. The result will be calculated continuously from the data that is readily available.

Description
The x-position of the maximum of the waveform or waveform segment is determined. The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.

Note The returned value is the position on the x-axis of the waveform in terms of x-position, not in terms of sample number. The units of this numerical value are the x-units of the waveform.

Example
The following example determines the time at which the pressure reaches its maximum value. The actual maximum pressure is determined using the @Value function for this time.

The pressure is assumed to be a real-world signal.
Pressure = Active.Group1.Recorder_A.Ch_A1
TimeOfMax = @MaxPos(Formula.Pressure)
MaxPress = @Value(Formula.Pressure;
                  Formula.TimeOfMax)

See Also
"@Max" on page 142, "@Min" on page 152 and "@MinPos" on page 155
4.54  @Mean

Function
Calculates the arithmetic mean value of a waveform.

Syntax
@Mean(Waveform)
@Mean(Waveform; Begin)
@Mean(Waveform; Begin; End)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform</td>
<td>Input waveform for which the mean value is to be calculated.</td>
</tr>
<tr>
<td>Begin</td>
<td>Number: segment begin.</td>
</tr>
<tr>
<td>End</td>
<td>Number: segment end.</td>
</tr>
</tbody>
</table>

Output
The output is a numerical value. This function works also “while recording” when both the Begin and End parameters are set. The result will be calculated continuously from the data that is readily available.

Description
The mean value is calculated using the following formula:

\[
\text{Mean} = \frac{1}{N} \sum_{n=n_1}^{n_2} y(n) \text{ with } N = (n_2 - n_1 + 1)
\]

The segment limits Begin (n1) and End (n2) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.

Example
The following example creates a signal (constant value 100) with hum and noise added. The mean value of the first 20 ms (one period of 50 Hz) of the signal is calculated:
Signal = 100 + 0.5 * @SineWave(20k; 1000; 50) +
       @Noise(20k; 1000)
Avg = @Mean(Formula.Signal; 0; 20m)

The value of Avg is a much more accurate reading of the signal than each individual sample in the waveform because of the averaging performed.

See Also
"@RMS" on page 198 and "@StdDev" on page 213
4.55  @MedianFilter

Function
Filters a waveform using a median filter.

Syntax
@MedianFilter(Waveform)
@MedianFilter(Waveform; N)

Parameters
Waveform  Waveform to be filtered.
N          The window size of the median filter. The default is 3. Range: 3 … 1001.

Output
Median filtered waveform.

Description
The median filter iterates through the input waveform sample by sample and takes the median of the sample and its neighboring samples. A sample together with its neighbors is called the “window”. The number of samples included in the window is the window size (N).
The median is the middle value after the samples in the window have been sorted numerically.

The median filter can be expressed with the following formula:

\[
\text{Output}(n) = \text{Median}\left( \text{Sort}\left(\text{Input}(n-M) \ldots \text{Input}(n+M)\right) \right)
\]

With:
N = Window size
M = (N-1)/2
n = Sample number in range 1 to 'end-of-waveform'

The window size N controls the number of samples used before sorting and taking the median. If N is not an odd integer, it will be rounded to the nearest odd integer first. If the value is smaller than 3, it is set to 3. If the value is larger than 1001, it is set to 1001.
Median filtering is a smoothing technique, similar to a moving average filter (See "@Smooth" on page 202). Smoothing techniques are effective at removing noise in smooth regions of an input waveform, but adversely affect edges. To reduce the edge effects, additional samples ('M') are added before the first resp. after the last input sample.

The median filter filters out high frequency components i.e. noise. The best filtering is achieved if the window size is equal to some integer multiple of the period of the disturbances. Due to the sorting, in combination with taking the median, the characteristic waveform features of the waveform are preserved. At locations where the input signal goes from a rising signal to a falling signal and vice versa, some filtering effects might be visible however.

The length of the output waveform is the same as the length of the input waveform.

Example

Figure 4.31: Median filtering on a noisy sinusoidal waveform
Where:

- \( \text{sin} \) \( \Rightarrow \) @SineWave(10k;10k+1;10)
- \( \text{sinwithnoise} \) \( \Rightarrow \) Formula.sin + @SineWave(10k;10k+1;1000)
- \( \text{median} \) \( \Rightarrow \) @MedianFilter(Formula.sinwithnoise;51)

See Also

"@Smooth" on page 202
4.56 @Min

Function
Determines the minimum value of a waveform.

Syntax
@Min(Waveform)
@Min(Waveform; Begin)
@Min(Waveform; Begin; End)

Parameters

Waveform Input waveform for which the minimum is to be determined.
Begin Number: segment begin.
End Number: segment end.

Output
The output is a numerical value. This function works also "while recording" when both the Begin and End parameters are set. The result will be calculated continuously from the data that is readily available.

Description
The minimum of the waveform or waveform segment is determined. The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters need to be specified in the units of the horizontal axis (for example time) and not as samples.

Example
The following example creates a 50 Hz sine wave on top of a DC value. The maximum and minimum value of the first 20 ms (one period of 50 Hz) of the signal is calculated to determine the (peak-to-peak) amplitude of the 50 Hz component:

\[
\begin{align*}
\text{Signal} & = 100 + 25 \times @\text{SineWave}(20k; 1000; 50) \\
\text{MinAmpl} & = @\text{Min}(\text{Formula.Signal}; 0; 20m) \\
\text{MaxAmpl} & = @\text{Max}(\text{Formula.Signal}; 0; 20m) \\
\text{Vpp} & = \text{Formula.MaxAmpl} - \text{Formula.MinAmpl}
\end{align*}
\]
If the position of the minimum is to be determined, use the @MinPos function instead. The minimum value at the position found can then be obtained using the @Value function:

\[
T\text{Min} = \text{@MinPos(Signal; } 0; 20\text{m)}
\]

\[
\text{Minim} = \text{@Value(Formula.Signal; Formula.T\text{Min})}
\]

See Also

"@Max" on page 142, "@MaxPos" on page 145 and "@MinPos" on page 155
4.57  @MinNum

Function
Calculates the minimum value of two or more numerical values.

Syntax
@MinNum(Par1; ...; ParN)

Parameters
Par1  First numerical value.
ParN  Last numerical value. N >= 2.

Output
Minimum of all numerical values.

Description
This function returns the smallest numerical value of all parameters specified. Each parameter must be a numerical value containing a valid value. If one of the parameters has the incorrect type or contains no value, the function returns no value.

Example
The following example calculates the maximum of a waveform but clips the value if it is larger than 100. The waveform is assumed to be a real-world signal:

```
Signal       = Active.Group1.Recorder_A.Ch_A1
MaxValue     = @Max(Formula.Signal)
ClippedMax   = @MinNum(Formula.MaxValue; 100)
```

The following example returns -10:

```
MinusTen = @MinNum(45.0; 10; -3.3; 20; -10.0; 5)
```

See Also
"@MaxNum" on page 144
4.58  **@MinPos**

**Function**
Determines the **position** of the **minimum** of a waveform.

**Syntax**
@MinPos(Waveform)
@MinPos(Waveform; Begin)
@MinPos(Waveform; Begin; End)

**Parameters**

- **Waveform**  
  Input waveform for which the position of the minimum is to be determined.

- **Begin**  
  Number: segment begin.

- **End**  
  Number: segment end.

**Output**
The output is the x-position of the minimum.

**Description**
The x-position of the minimum of the waveform or waveform segment is determined. The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

**Note**
The **Begin** and **End** parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.

**Note**
The returned value is the position on the x-axis of the waveform in terms of x-position, not a sample number. The units of this numerical value are the x-units of the waveform.

**Example**
The following example determines the time at which the pressure reaches its minimum value. The actual minimum pressure is determined using the @Value function for this time. The variable Pressure is assumed to be a real-world signal.

```
Pressure = Active.Group1.Recorder_A.Ch_A1
```
\begin{align*}
\text{TimeOfMin} &= \@MinPos(\text{Formula.Pressure}) \\
\text{MinPress} &= \@Value(\text{Formula.Pressure}; \text{Formula.TimeOfMin})
\end{align*}

\textbf{See Also}

"@Max" on page 142, "@MaxPos" on page 145 and "@Min" on page 152
4.59  @NextHillPos

Function
This function searches for the position of the next local maximum in a waveform.

Syntax
@NextHillPos(Waveform; StartPos; Hysteresis)

Parameters
Waveform    Input waveform for which the position of the next local maximum is to be determined.
StartPos    Number: start position in x-units where searching should start (forwards).
Hysteresis  Number: noise suppression hysteresis value.

Output
The output is the x-position of the next local maximum.

Description
The x-position of the next local maximum in the waveform is determined.

Searching starts from the x-position specified by StartPos to the right. If StartPos is before the beginning of the waveform, the start of the waveform is used as start position for the search.

The hysteresis value is used to suppress the effects of noise on the signal. If for instance 100 mV peak-to-peak noise is present on a signal, specifying 200 mV as hysteresis value will prevent the algorithm from determining small noise peaks as local maximum.

Note
The returned value is the position on the x-axis of the waveform in terms of x-position, not a sample number. The units of this numerical value are the x-units of the waveform.

Example
The following example determines the position of a local maximum on a 1 V sine wave with 10 mV noise, starting from 500 ms searching forwards, using a hysteresis of 40 mV:
Signal = @SineWave(10k; 10000; 50) + 0.01 * @Noise(10k; 10000)
NHPos = @NextHillPos(Formula.Signal; 500m; 40m)

See Also
"@NextValleyPos" on page 161, "@PrevHillPos" on page 169 and
"@PrevValleyPos" on page 173
4.60 \texttt{@NextLvlCross}

Function
Determines the position of the next crossing of a waveform with a specified signal level.

Syntax
\texttt{@NextLvlCross(Waveform; StartPos; Level)}
\texttt{@NextLvlCross(Waveform; StartPos; Level; Slope)}

Parameters
- \textit{Waveform}\hspace{1em} Input waveform for which the position of the next level crossing is to be determined.
- \textit{StartPos}\hspace{1em} Number: start position in x-units where searching should start.
- \textit{Level}\hspace{1em} Number: amplitude level to search for.
- \textit{Slope}\hspace{1em} Number: direction of the slope (-1, 0, 1)

Output
The output is the x-position of the next level crossing.

Description
The x-position of the next level crossing of the waveform with the specified level in the specified direction is determined.
Searching starts at the x-position specified by StartPos and goes to the right in the positive time direction. If StartPos is before the beginning of the waveform, the start of the waveform is used as start position for the search.

The Slope parameter controls the type of level crossing that is searched for:

- Slope = 1: Positive level crossing (from below level to above level)
- Slope = -1: Negative level crossing (from above level to below level)
- Slope = 0: Any crossing (either positive or negative)

Note
\textit{The returned value is the position on the x-axis of the waveform in terms of x-position, not a sample number. The units of this numerical value are the x-units of the waveform.}

Example
The following example determines the start and end time of the first TTL pulse present in the TTLSignal that is assumed to be available as real-world signal in “Active.Group1.Recorder_A.Ch_A1”.

\begin{itemize}
\item Slope = 1: Positive level crossing (from below level to above level)
\item Slope = -1: Negative level crossing (from above level to below level)
\item Slope = 0: Any crossing (either positive or negative)
\end{itemize}
TTSignal = Active.Group1.Recorder_A.Ch_A1
TimeStart = @NextLvlCross(Formula.TTSignal; -1E20; 2.5; 1)
TimeEnd = @NextLvlCross(Formula.TTSignal; Formula.TimeStart; 2.5; -1)

See Also
"@PrevLvlCross" on page 171
4.61 @NextValleyPos

Function
This function searches for the position of the next local minimum in a waveform.

Syntax
@NextValleyPos(Waveform; StartPos; Hysteresis)

Parameters

Waveform Input waveform for which the position of the next local minimum is to be determined.
StartPos Number: start position in x-units where searching should start (forwards).
Hysteresis Number: noise suppression hysteresis value.

Output
The output is the x-position of the next local minimum.

Description
The x-position of the next local minimum in the waveform is determined.

Searching starts from the x-position specified by StartPos to the right. If StartPos is before the beginning of the waveform, the start of the waveform is used as start position for the search.

The hysteresis value is used to suppress the effects of noise on the signal. If for instance 100 mV peak-to-peak noise is present on a signal, specifying 200 mV as hysteresis value will prevent the algorithm from determining small negative noise peaks as local minimum.

Note
The returned value is the position on the x-axis of the waveform in terms of x-position, not a sample number. The units of this numerical value are the x-units of the waveform.

Example
The following example determines the position of a local minimum on a 1 V sine wave with 10 mV noise, starting from 500 ms searching forwards, using a hysteresis of 40 mV:
Signal = @SineWave(10k; 10000; 50) + 0.01 * @Noise(10k; 10000)

NVPos = @NextValleyPos(Formula.Signal; 500m; 40m)

See Also
"@NextHillPos" on page 157, "@PrevHillPos" on page 169 and
"@PrevValleyPos" on page 173
4.62 @Noise

Function
Generates a waveform containing noise.

Syntax
@Noise(FSampling; NSamples)

Parameters
FSampling  Number: sampling frequency.
NSamples   Number of samples.

Output
Waveform containing noise.

Description
This function creates a waveform containing random noise with an amplitude between -1 and 1. The sampling frequency and number of samples to generate must be specified.

The possibility to generate noise can be very useful to investigate the influence of noise on the analysis of a measured or generated waveform.

Example
The following example generates a 5 V amplitude sine wave hidden in 10 V peak-to-peak noise.

\[ \text{Signal} = 5 * \text{SineWave}(10k; 1000; 50) + 10 * \text{Noise}(10k; 1000) \]

See Also
"@Pulse" on page 175, "@Ramp" on page 179, "@SineWave" on page 201 and "@SquareWave" on page 212
4.63  $@\text{Not}$

**Function**
This function performs a logical NOT evaluation on the input parameters.

**Syntax**
$@\text{Not}(\text{Param1})$

**Parameters**

$\text{Param1}$  
Number used for the NOT evaluation.

**Output**
The output is a 1 or a 0.

**Description**
The $@\text{Not}$ function performs a logical NOT evaluation on the input parameter. Depending on the evaluation the result will be 1 or 0. A numerical value not equal to 0 corresponds to a logical “True” and a numerical 0 corresponds to a logical “False”.

The truth table of the NOT function is:

<table>
<thead>
<tr>
<th>Param1</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

**Example**
The following is a list of examples and their return value.

- $\text{NotExampl1} = @\text{Not}(1) => 0 (= \text{false})$
- $\text{NotExampl2} = @\text{Not}(4) => 0 (= \text{false})$
- $\text{NotExampl3} = @\text{Not}(0) => 1 (= \text{true})$

**See Also**
"$@\text{And}$" on page 55 and "$@\text{Or}$" on page 165
4.64  @Or

Function
This function performs a logical OR evaluation on the input parameters.

Syntax
@Or(Param1; ...; ParamN)

Parameters
Param1  Number: first parameter used for the OR evaluation.
ParamN  Number: last parameter used for the OR evaluation.
        N >= 2

Output
The output is a 1 or a 0.

Description
The @Or function performs a logical OR evaluation on the input parameters. Depending on the evaluation the result will be 1 or 0. A numerical value not equal to 0 corresponds to a logical “True” and a numerical 0 corresponds to a logical “False”.

The truth table of the OR function is:

<table>
<thead>
<tr>
<th>Param1</th>
<th>Param2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

Example
The following is a list of examples and their return value.

OrExampl1 = @Or(1; 1)  => 1 (= true)
OrExampl2 = @Or(1; 4; 10) => 1 (= true)
OrExampl3 = @Or(1; 4; 0; 6) => 1 (= true)
OrExampl4 = @Or(0; 0)  => 0 (= false)

See Also
"@And" on page 55 and "@Not" on page 164
4.65  @Period

Function
Determines the **period** of a waveform. The period is the reciprocal of the frequency.

Syntax
@Period(Waveform)
@Period(Waveform; Begin)
@Period(Waveform; Begin; End)

Parameters
- **Waveform**: Input waveform for which the period is to be determined.
- **Begin**: Number: segment begin.
- **End**: Number: segment end.

Output
The output is a numerical value.

Description
The period of the waveform or waveform segment is determined. The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

**Note**
*The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.*

The period is determined using base and top magnitude algorithms. The mean of these magnitudes is considered to be the zero level. The number of zero crossings in the range Begin to End is calculated. The crossings should be in the same direction as the first one. The crossings are determined with a hysteresis around the zero level to suppress the effects of noise. The hysteresis is +/- 5% of the difference between top and base.

From the time difference between the first and last zero crossing in the same direction and the number of zero crossings in between, the period of the waveform is determined.

Example
The following example determines the period of a 50 Hz signal:
Signal = @SineWave(10k; 10k; 50) + 0.01 * @Noise(10k; 10k)

Period = @Period(Formula.Signal)

Freq = 1 / Formula.Period

See Also
"@Frequency" on page 115
Function
Exponentiation, the mathematical operation, written \( a^n \), involving two parameters: the base “a” and the exponent “n”.

Syntax
@Pow(Base; Exponent)

Parameters
- **Base**: Base waveform or numerical value.
- **Exponent**: Exponent waveform or numerical value.

Output
Waveform or numerical value containing the exponentiation of base and exponent.

Description
The exponential function calculates the base raised to the power exponent.

If both the base and the exponent are numerical values, the result is a numerical value.

If either the base or the exponent is a waveform, the exponentiation function is calculated for each individual sample in combination with the numerical value. The result is a waveform.

**Note**
*It is not possible to use a waveform for both base and exponent.*

Example
The following results are equivalent:

\[
\begin{align*}
\text{Input} & = 2 \times @\text{SineWave}(10k; 10k; 10) \\
\text{Result1} & = @\text{Exp}(\text{Formula.Input}) \\
\text{Result2} & = @\text{Pow}(\text{System.Constants.e}; \text{Formula.Input})
\end{align*}
\]

See Also
"@Exp" on page 85
@PrevHillPos

**Function**

This function searches for the position of the previous local maximum in a waveform.

**Syntax**

@PrevHillPos(Waveform; StartPos; Hysteresis)

**Parameters**

- **Waveform** Input waveform for which the position of the previous local maximum is to be determined.
- **StartPos** Number: start position in x-units where searching should start (backwards).
- **Hysteresis** Number: noise suppression hysteresis value.

**Output**

The output is the x-position of the previous local maximum.

**Description**

The x-position of the previous local maximum in the waveform is determined.

Searching starts from the x-position specified by StartPos to the left. If StartPos is past the end of the waveform, the end of the waveform is used as start position for the search.

The hysteresis value is used to suppress the effects of noise on the signal. If for instance 100 mV peak-to-peak noise is present on a signal, specifying 200 mV as hysteresis value will prevent the algorithm from determining small noise peaks as local maximum.

**Note**

*The returned value is the position on the x-axis of the waveform in terms of x-position, not a sample number. The units of this numerical value are the x-units of the waveform.*

**Example**

The following example determines the position of a local maximum on a 1 V sine wave with 10 mV noise, starting from 500 ms searching backwards, using a hysteresis of 40 mV:
Signal = SineWave(10k; 10000; 50) + 0.01 * Noise(10k; 10000)

PHPos = PrevHillPos(Formula.Signal; 500m; 40m)

See Also
"@NextHillPos" on page 157, "@NextValleyPos" on page 161 and
"@PrevValleyPos" on page 173
Function
Determines the position of the previous crossing of a waveform with a specified signal level.

Syntax
@PrevLvlCross(Waveform; StartPos; Level)
@PrevLvlCross(Waveform; StartPos; Level; Slope)

Parameters
Waveform Input waveform for which the position of the previous level crossing is to be determined.
StartPos Number: start position in x-units where searching should start (backwards).
Level Number: amplitude level to search for.
Slope Number: direction of the slope (-1, 0, 1)

Output
The output is the x-position of the previous level crossing.

Description
The x-position of the previous level crossing of the waveform with the specified level in the specified direction is determined.
Searching starts at the x-position specified by StartPos and goes to the left in the negative time direction.

The Slope parameter controls the type of level crossing to search for:

- Slope = 1: Positive level crossing (from below level to above level)
- Slope = -1: Negative level crossing (from above level to below level)
- Slope = 0: Any crossing (either positive or negative)

Note
The returned value is the position on the x-axis of the waveform in terms of x-position, not a sample number. The units of this numerical value are the x-units of the waveform.

Example
The following example determines the start and end time of the last TTL pulse present in the TTLSignal. This signal is assumed to be a real-world signal.
TTLSignal = Active.Group1.Recorder_A.Ch_A1
TimeEnd = @PrevLvlCrossing(Formula.TTLSignal; 1E20; 2.5; 1)
TimeStart = @PrevLvlCrossing(Formula.TTLSignal; Formula.TimeEnd; 2.5; -1)

See Also
"@NextLvlCross" on page 159
Function
This function searches for the position of the previous local minimum in a waveform.

Syntax
@PrevValleyPos(Waveform; StartPos; Hysteresis)

Parameters
Waveform Input waveform for which the position of the previous local minimum is to be determined.
StartPos Number: start position in x-units where searching should start (backwards).
Hysteresis Number: noise suppression hysteresis value.

Output
The output is the x-position of the previous local minimum.

Description
The x-position of the previous local minimum in the waveform is determined.

Searching starts from the x-position specified by StartPos to the left. If StartPos is past the end of the waveform, the end of the waveform is used as start position for the search.

The hysteresis value is used to suppress the effects of noise on the signal. If for instance 100 mV peak-to-peak noise is present on a signal, specifying 200 mV as hysteresis value will prevent the algorithm from determining small negative noise peaks as local minimum.

Note
The returned value is the position on the x-axis of the waveform in terms of x-position, not a sample number. The units of this numerical value are the x-units of the waveform.

Example
The following example determines the position of a local minimum on a 1 V sine wave with 10 mV noise, starting from 500 ms searching backwards, using a hysteresis of 40 mV:
\[
\text{Signal} = @\text{SineWave}(10k; 10000; 50) + 0.01 * @\text{Noise}(10k; 10000)
\]
\[
\text{PVPo} = @\text{PrevValleyPos}(\text{Formula}.\text{Signal}; 500m; 40m)
\]

See Also
"@NextHillPos" on page 157, "@NextValleyPos" on page 161 and "@PrevHillPos" on page 169
4.70  @Pulse

Function
Generates a waveform containing a single pulse.

Syntax
@Pulse(FSampling; NSamples; PulseStart)
@Pulse(FSampling; NSamples; PulseStart; PulseWidth)

Parameters
FSampling     Number: sampling frequency
NSamples      Number of samples
PulseStart    Number: Sample where pulse starts
PulseWidth    Number: Pulse width in samples the default value is set to 1.

Output
Waveform containing a single pulse.

Description
This function creates a single pulse waveform. The sampling frequency, number of samples to generate the single pulse can be specified. There is no limit on the length of the waveform. The amplitude of the pulse wave is 1 V.

The pulse function can be used to determine the impulse and step response of a filter function.

You can use this function to synthesize a variety of waveforms. This data can be used as input for other analysis functions.

Example
SignalPulse = @Pulse(80k; 80k; 20k)
ImpulseResponse = @FilterButterworthLP
(Formula.SignalPulse; 2; 200)
SignalStep = @Pulse(80k; 80k; 20k; 10k)
StepResponse = @FilterButterworthLP
(Formula.SignalStep; 2; 200)

If you are only interested in a part of the signal than you can use the pulse function to make all values zero outside of a specified interval. This interval can be defined by the PulseStart and PulseWidth parameters of the Pulse function.
SignalInterval = Formula.SignalStep * Active.Group1.Recorder_A.Ch_A1

See Also
"@Noise" on page 163, "@SineWave" on page 201, "@SquareWave" on page 212 and "@Ramp" on page 179
4.71 @PulseWidth

Function
Determines the pulse width of a pulse in a waveform.

Syntax
@PulseWidth(Waveform)
@PulseWidth(Waveform; Begin)
@PulseWidth(Waveform; Begin; End)

Parameters
Waveform Input waveform containing the pulse.
Begin Number: segment begin.
End Number: segment end.

Output
The output is a numerical value.

This function works "while recording" when the Begin and End parameters are set. The result will be calculated when the data between Begin and End is available.

Description
The pulse width is determined by taking the time difference between the first and second 50% magnitude transitions of the pulse in the waveform (or waveform segment).

Magnitude percentage crossings are calculated using the base and top magnitudes. The base and top magnitudes are determined by searching two dominant populations in the waveform (or waveform segment) magnitude histogram and taking the mean values of these two populations.

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used.

When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.
Example
The following example creates a pulse and calculates the 50%-50% pulse width. The result is 150 ms:

\[
\begin{align*}
\text{Sig1} &= \text{Ramp}(1k; 100; 0; 0) \\
\text{Sig2} &= \text{Ramp}(1k; 51; 0; 10) \\
\text{Sig3} &= \text{Ramp}(1k; 99; 10; 10) \\
\text{Sig4} &= \text{Sig4} \\
\text{Signal} &= \text{Join}(\text{Sig1}; \text{Sig2}; \text{Sig3}; \text{Sig4}; \text{Sig1}) \\
\text{Width} &= \text{PulseWidth}(\text{Signal})
\end{align*}
\]

See Also
"@FallTime" on page 87 and "@RiseTime" on page 196
4.72  @Ramp

Function
Generates a waveform containing one or more linear ramps.

Syntax
@Ramp(FSampling; NSamples; YStart; YEnd)
@Ramp(FSampling; NSamples; YStart; YEnd; Count)

Parameters
FSampling  Number: sampling frequency.
NSamples   Number of samples.
YStart      Number: value of first sample.
YEnd        Number: value of last sample.
Count       Number of ramps.

Output
Waveform containing one or more linear ramps.

Description
This function creates a waveform containing one or more linear ramps starting at YStart and ending at YEnd. The sampling frequency and number of samples to generate can also be specified. The length of the waveform is limited to 1 GigaSamples.

When specifying the optional parameter Count, the specified number of ramps will be generated. Using the Count value results in a sawtooth waveform. When Count is not specified, the value 0 will be used. This will create a single ramp only, not a single sawtooth.

The possibility to generate a ramp function in combination with the @Join function and other waveform generation functions is used to synthesize a variety of waveforms. The simulated data can be used as input for other analysis functions.

Example
The following example generates a simulated pulse of 10 V amplitude with 4 V peak-to-peak noise added:

    Sig1 = @Ramp(1k; 100; 0; 0)
Sig2  = @Ramp(1k; 51; 0; 10)
Sig3  = @Ramp(1k; 100; 10; 10)
Sig4  = @Ramp(1k; 51; 10; 0)
Signal = @Join(Formula.Sig1; Formula.Sig2;
               Formula.Sig3; Formula.Sig4) + 2 * @Noise(1k;
               302)

See Also
"@Noise" on page 163, "@Pulse" on page 175, "@SineWave" on page 201
and "@SquareWave" on page 212
4.73  @ReadAsciiFile

**Function**
Imports waveform data from an ASCII (text) file.

**Syntax**
@ReadAsciiFile(Filename)

**Parameters**

*Filename*   String: the full path giving the name and the unique location of the ASCII file.

**Output**
The output is a waveform.

**Description**
This function imports waveform data from an ASCII file. The samples of the imported data are interpreted as being equidistant.

The ASCII file must contain a header and a data part. Two different types of headers are supported: a short version and a long version.

**Short Header:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of header rows</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Data delimiter (dot, comma, tab or semi-column)</td>
<td>;</td>
</tr>
<tr>
<td>3</td>
<td>Number of data pairs</td>
<td>n</td>
</tr>
<tr>
<td>4</td>
<td>Scale factor for x and y (x; y)</td>
<td>5.0E-8;2.44E+0</td>
</tr>
<tr>
<td>5</td>
<td>Units for x and y (x; y)</td>
<td>s;A</td>
</tr>
</tbody>
</table>

**Long Header:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of header rows</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>Data delimiter (dot, comma, tab or semi-column)</td>
<td>;</td>
</tr>
<tr>
<td>3</td>
<td>Number of data pairs</td>
<td>n</td>
</tr>
<tr>
<td>4</td>
<td>Date of data generation</td>
<td>17.03.00</td>
</tr>
<tr>
<td>5</td>
<td>Time of data generation</td>
<td>23:59</td>
</tr>
</tbody>
</table>
Long Header:

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Extra information about producer of the TDG 0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>data, this line can be blank. Only for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>compatibility reasons.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Comments (max. 80 characters)</td>
<td>First example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Test 1;</td>
</tr>
<tr>
<td>8</td>
<td>Scale factor for x and y (x; y)</td>
<td>5.0E-8;2.44E+0</td>
</tr>
<tr>
<td>9</td>
<td>Units for x and y (x; y)</td>
<td>s;A</td>
</tr>
<tr>
<td>10</td>
<td>Name of scaled units (x; y)</td>
<td>time;current</td>
</tr>
<tr>
<td>11</td>
<td>Resolution of y-data in bit</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>Use of dynamic range in %</td>
<td>80</td>
</tr>
</tbody>
</table>

Data:

The data comes after the header and starts at line 6 or line 13, depending on the size of the header.

Each data line contains an x, y pair. The separator between the x and y value is defined at line 2 of the header.

The samples of the imported data are interpreted as being equidistant.

The first data line contains the first (lowest) x-value.

Example

The following example shows how the file “ReferenceCurve.asc” is read.

```
Signal = @ReadAsciiFile("d:\data\ReferenceCurve.asc")
```
4.74  @ReadLogFile

Function
Reads a sequence of numerical values from a Perception log file and creates a waveform from those values.

Syntax
@ ReadLogFile (Filename; Fieldname)
@ ReadLogFile (Filename; Fieldname; X-Units)
@ ReadLogFile (Filename; Fieldname; X-Units; SampleInterval)
@ ReadLogFile (Filename; Fieldname; X-Units; SampleInterval; X-Start)

Parameters
Filename  String: the full path giving the name and the unique location of the Perception log file to be read.
Fieldname  String: the name of the numerical field in the Perception logging file to be read.
X-Units  String: The X-Units of the new created waveform data source; the default value is empty.
SampleInterval  Number: The sample interval of the new created waveform data source; the sample interval is expressed in X-Units; the default value is 1.
X-Start  Number: The start of the new created waveform data source expressed in X-Units; the default value is 0

Output
The output is a waveform.

Description
This function reads numerical logged values from a Perception log file and creates a waveform from these values. Perception has the ability to save numerical values into a log file; these loggings can be done manually or as part of an automation setup; for example after each recording. For more information on Perception logging, please refer to the Perception Data Acquisition Software manual.
The log file is an xml structured file. See Figure 4.32 for an example of such a file.

Figure 4.32: Log file example
The same log file can also be opened into Microsoft® Excel:

![Microsoft Excel log file]

**Figure 4.33: Microsoft® Excel log file**

As can be seen each logging is a row in the Microsoft® Excel sheet. Via the `ReadLogFile()` function it is now possible to select one of the numerical columns and read the column data into Perception as a waveform data source. The selection is done by the second parameter the so called *Fieldname*; for example: `Active.UserData.X`

It is also possible to use `UserData.X` or even `X`; but if the field name is abbreviated, take care that the abbreviation is unique and does not conflict with other field names.

The `ReadLogFile()` function always compares the abbreviated name with the ending part of the original field name, therefore `X` is correct for `Active.UserData.X`, however `UserData` is incorrect and will result in an empty waveform.
When the logging rows are not equidistant in time, therefore the X-Units of the new created waveform is not set to time but is left empty.

Use for example “Setpoints” if a logging was done for each set-point whatever this might be. If there are however time equidistant loggings then you can use the SampleInterval to set the time between the loggings and set the X-Units to s.

Example

\[
Y = @ReadLogFile("C:\Users\Public\Documents\SharedLogfiles\VibrationLogfile2.XML"; "Active.UserData.Y")
\]

\[
X = @ReadLogFile("C:\Users\Public\Documents\SharedLogfiles\VibrationLogfile2.XML"; "X")
\]

The new X and Y waveforms can be shown in the Perception Y-t display or in an X-Y display (see Figure 4.34).

Figure 4.34: X and Y waveforms in Perception

See Also

"@ReadAsciiFile" on page 181
@Reduce

Function
Reduces the number of samples in a waveform by resampling.

Syntax
@Reduce(Waveform; Factor)

Parameters
Waveform Waveform to be resampled.
Factor Number: resampling factor.

Output
Resampled waveform.

Description
A waveform is resampled to reduce the amount of data. The resampling factor is rounded to the nearest integer N and should be in range 2, 3, ..., (waveform length)/2. The output waveform receives sample 1, sample 1+N, sample 1+2N, etc. until no more samples are available. The length of the output waveform is N times smaller than as the length of the input waveform. The sampling interval of the output waveform is N times the sampling interval of the input waveform. The x-coordinate of the first sample is not changed.

Resampling can cause aliasing effects when high frequency components are present in the waveform. In this case, use the smoothing function @Smooth for lowpass filtering of the data before resampling.

Example
The following example resamples a sine wave by a factor of 10, effectively lowering the sampling rate by a factor of 10. The number of samples is reduced from 1000 to 100 samples.

\[
\begin{align*}
    \text{Sine1000} &= \text{@SineWave(10k; 1000; 50)} \\
    \text{Sine100}  &= \text{@Reduce(Formula.Sine1000; 10)}
\end{align*}
\]

See Also
"@Res2" on page 195 and "@Smooth" on page 202
Function
Compares a waveform or waveform segment against one or two reference waveforms.

Syntax
@RefCheck(Waveform; UpperEnvelope)
@RefCheck(Waveform; UpperEnvelope; LowerEnvelope)
@RefCheck(Waveform; UpperEnvelope; LowerEnvelope; Begin)
@RefCheck(Waveform; UpperEnvelope; LowerEnvelope; Begin; End)

Parameters
Waveform Input waveform to be compared.
UpperEnvelope Waveform containing the upper envelope.
LowerEnvelope Waveform containing the lower envelope.
Begin Number: begin position of segment for comparison.
End Number: end position of segment for comparison.

Output
Number indicating success (between the reference waveforms) or failure.

Description
This function is used to compare a waveform against one or two reference waveforms. The output is the x position of the first level crossing of the input waveform with the Upper or Lower waveform.

If the input waveform is between the upper and lower waveform then the return value will be a “Not-a-number” (Nan) value.

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used.

When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Example
The following example creates a sine wave and two ‘envelope’ sine waves. The noise is used to create a success or failure. A higher noise value will result in failure.
Signal = @SineWave(8000; 8001; 5)
Upper  = Formula.Signal + 0.1
Lower  = Formula.Signal - 0.1
Noise  = 0.1 * @Noise(8000; 8001)
Signal2 = Formula.Signal + Formula.Noise
Check  = @RefCheck(Formula.Signal2; Formula.Upper; Formula.Lower)
Result = @IIF(Formula.Check; "NOT OK"; "OK")
4.77  @RelativeTime2Local

Function
Returns a string representing the absolute local time corresponding to the time elapsed since the start of the signal.

Syntax
@RelativeTime2Local(Signal; Rel Time)
@RelativeTime2Local(Signal; Rel Time; Decimals)
@RelativeTime2Local(Signal; Rel Time; Decimals; DateFormat)

Parameters
- **Signal**: Input waveform
- **Rel Time**: Number: input time elapsed since start (e.g. cursor X value) in seconds.
- **Decimals**: Number: number of decimals to display the fractional part of the seconds. Default 3. Number has to be between 0 and 9.
- **DateFormat**: String: date format: “None”, “Short” or “Long”. Default “None” means no date information.

Output
A string representing the absolute local time corresponding to the time elapsed since the start of the signal.

The output can look like:

Figure 4.35: @RelativeTime2Local - output
Description
This function is used to determine the absolute local time of a data sample. This function is often used to determine the absolute time at the cursor position.

Example
This example shows the absolute local time of the X-location of cursor 1.

\[ T_{\text{Local}} = @\text{RelativeTime2Local} \]
\[ \quad \text{(Active.Group1.Recorder_A.Input_shaft_vib; Display.Display.Cursor1.XPosition; 5; "Short")} \]

See Also
"@RelativeTime2UTC" on page 192
4.78  @RelativeTime2UTC

Function
Returns a string representing the absolute UTC time corresponding to the time elapsed since the start of the signal.

Syntax
@RelativeTime2UTC(Signal; Rel Time)
@RelativeTime2UTC(Signal; Rel Time; Decimals)
@RelativeTime2UTC(Signal; Rel Time; Decimals; DateFormat)

Parameters
Signal  Input waveform
Rel Time  Number: input time elapsed since start (e.g. cursor X value) in seconds.
Decimals  Number: number of decimals to display the fractional part of the seconds. Default 3. Number has to be between 0 and 9.
DateFormat  String: date format: “None”, “Short” or “Long”. Default “None” means no date information.

Output
A string representing the absolute UTC time corresponding to the time elapsed since the start of the signal.

Figure 4.36: @RelativeTime2UTC - Output

Description
This function is used to determine the absolute UTC time of a data sample. This function is often used to determine the absolute time at the cursor position.
Example
This example shows the absolute local time of the X-location of cursor 1.

\[ T_{UTC} = \text{@RelativeTime2UTC} \]
\[
(\text{Active.Group1.Recorder_A.Input_shaft_vib};
\text{Display.Display.Cursor1.XPosition}; 5;
"Short")
\]

See Also
"@RelativeTime2Local" on page 190
Function
Returns a waveform with new data between a specified start and end time by using a straight line between those two points.

Syntax
@RemoveGlitch(Waveform; Begin; End)

Parameters
Waveform  Input waveform for which the glitch is to be removed.
Begin     Number: glitch start time
End       Number: glitch end time

Output
The output is a waveform with the same length as the original waveform.

Description
The resulting waveform creates a straight line between the Begin and End times. The rest of the original waveform remains unchanged.

Note
If the formula is used to remove a glitch while recording, then the Begin and End parameters must be within the bounds of the available data. If either the Begin or End time does not have a corresponding data value in the waveform, then the RemoveGlitch function cannot be calculated.

Example
The following example creates a 50 Hz sine wave with duration of 2 seconds. Then the function creates a new 2-second-long waveform, where the samples in the interval between 1 second and 1.5 seconds are replaced with a straight line

\[
\text{Signal} = @\text{SineWave}(1k; 2000; 50)
\]
\[
\text{Cropped} = @\text{RemoveGlitch}(\text{Formula.Signal}; 1000m; 1500m)
\]
4.80  @Res2

Function
Resamples a waveform so that its length becomes a power of two.

Syntax
@Res2(Waveform)

Parameters
Waveform Waveform to be resampled.

Output
Resampled waveform.

Description
This function resamples a waveform so that the new number of samples is equal to the next power of two greater than or equal to the original length. This function can be used effectively for calculating spectra of any desired segment of a waveform. The number of samples in such a segment will typically not be equal to a power of two, which is required by an FFT algorithm. By using the @Res2 function on this segment first, a new waveform is created containing the same signal but sampled at a higher sampling rate in such a way that the number of samples will be a power of two. This resampled waveform can be used for FFT calculations without the need to truncate or zero-pad the waveform.

Because the resampling factor will typically not be an integer, values of the waveform between two samples have to be generated. This is done using linear interpolation.

The sampling interval of the waveform is corrected for the resampling factor.

Example
The following example generates a sine wave of 2500 samples and resamples it to a waveform with 4096 samples. The value of Length will be 4096.

\[
\begin{align*}
\text{Signal} & \quad = \quad @\text{SineWave}(10000; \; 2500; \; 100) \\
\text{Resampled} & \quad = \quad @\text{Res2}(\text{Formula.Signal}) \\
\text{Length} & \quad = \quad @\text{Length}(\text{Formula.Resampled})
\end{align*}
\]

See Also
"@Reduce" on page 187
4.81  @RiseTime

**Function**
Determines the *rise time* of a pulse in a waveform.

**Syntax**
@RiseTime(*Waveform*)
@RiseTime(*Waveform*; *Begin*)
@RiseTime(*Waveform*; *Begin*; *End*)

**Parameters**

- *Waveform*  
  Input waveform containing the leading edge of a pulse.
- *Begin*  
  Number: segment begin.
- *End*  
  Number: segment end.

**Output**
The output is a numerical value.

This function works "while recording" when the Begin and End parameters are set. The result will be calculated when the data between Begin and End is available.

**Description**
The rise time is determined by taking the time difference between the proximal point (10% magnitude transition) and the distal point (90% magnitude transition) on the first leading edge of a pulse in the waveform (or waveform segment).

Magnitude percentage crossings are calculated using the base and top magnitudes. The base and top magnitudes are determined by searching two dominant populations in the waveform (or waveform segment) magnitude histogram and taking the mean values of these two populations.

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

**Note**
The *Begin* and *End* parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.
Example
The following example creates a pulse and calculates the 10%-90% rise time of the leading edge. The result is 40 ms:

$$\text{Sig1} = \text{@Ramp}(1k; 100; 0; 0)$$
$$\text{Sig2} = \text{@Ramp}(1k; 51; 0; 10)$$
$$\text{Sig3} = \text{@Ramp}(1k; 100; 10; 10)$$
$$\text{Sig4} = \text{@Ramp}(1k; 51; 10; 0)$$
$$\text{Signal} = \text{@Join}(\text{Formula.Sig1}; \text{Formula.Sig2}; \text{Formula.Sig3}; \text{Formula.Sig4}; \text{Formula.Sig1})$$
$$\text{Rise} = \text{@RiseTime}(\text{Formula.Signal})$$

See Also
"@FallTime" on page 87 and "@PulseWidth" on page 177
4.82  @RMS

Function
Calculates the Root Mean Square of a waveform.

Syntax
@RMS(Waveform)
@RMS(Waveform; Begin)
@RMS(Waveform; Begin; End)

Parameters
Waveform       Input waveform for which the RMS is to be calculated.
Begin       Number: segment begin.
End       Number: segment end.

Output
The output is a numerical value.

Description
The RMS is calculated using the following formula:

\[
\text{RMS} = \sqrt{\frac{1}{N} \sum_{n=n_1}^{n_2} y^2(n) \text{with } N = (n_2 - n_1 + 1)}
\]

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used. When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.

Note
The Begin and End parameters have to be specified in the units of the horizontal axis (for example time) and not as samples.

Example
The following example creates a 10 V amplitude sine wave of 50 Hz and calculates the RMS value using the complete waveform:

\[
\text{Signal} = 10.0 \ast \text{@SineWave}(20k; 1000; 50) \\
\text{RMSAmpl} = \text{@RMS} \left( \text{Formula.Signal} \right)
\]

See Also
"@Energy" on page 82, "@Mean" on page 147 and "@StdDev" on page 213
4.83  @SAEJ211Filter

Function
Filters the input signal in accordance to the SAE J211 standard.

Syntax
@SAEJ211Filter(Signal; CFC)

Parameters
- Signal: Input waveform
- CFC: Number: Channel Frequency Class

Output
The output is the filtered waveform.

Description
The function performs a digital lowpass filtering to eliminate high frequency noise from raw data. The filter is a 4-pole phase-less lowpass Butterworth filter. The filter type complies with the SAE J211 standard that is widely used in the automotive industry.

The relation between the cut-off frequency (Fc) and the Channel Frequency Class (CFC) is defined as follows:

\[ Fc = CFC \times 1.67. \]

The cut off frequency should be not less than one-quarter of the sampling frequency.

Example
The following example generates a sine wave with some high frequency noise. The signal is filtered with the SAE J211 filter function:

\[
\begin{align*}
\text{Signal} & = 5 \times \text{@SineWave}(80000; 1000; 400) \\
& \quad + \text{@Noise}(80000; 1000) \\
\text{Filtered} & = \text{@SAEJ211Filter}(\text{Formula.Signal}; 300)
\end{align*}
\]

The cutoff frequency is: \( Fc = 1.67 \times 300 = 500 \text{ Hz} \)
Function
Calculates the sine of the input parameter.

Syntax
@Sin(Par)

Parameters
Par Input waveform or numerical value.

Output
Waveform or numerical value containing the sine of the input.

Description
The trigonometric function sine is calculated assuming the input parameter is the angle in radians. If a waveform parameter is used, the sine is calculated for each individual sample.

Example
The following example calculates the sine of the variable “Angle” specified in degrees:

\[
\begin{align*}
\text{Angle} & = 36 \\
\text{AngleRad} & = \text{System.Constants.Pi} * \text{Formula.Angle} / 180 \\
\text{SinAngle} & = @Sin(\text{Formula.AngleRad})
\end{align*}
\]

See Also
"@ATan" on page 59, "@Cos" on page 69 and "@Tan" on page 221
@SineWave

Function
Generates a waveform described by the sine function.

Syntax
@SineWave(FSampling; NSamples; FSignal)

Parameters
- **FSampling**: Number: sampling frequency.
- **NSamples**: Number of samples.
- **FSignal**: Number: signal frequency.

Output
Waveform containing a sine wave.

Description
This function creates a sine wave. The sampling frequency and number of samples that generate a sine wave frequency can be specified. There is no limit on the length of the waveform. The amplitude of the sine wave is 1 V.

You can use this function to synthesize a variety of waveforms. This data can be used as input for other analysis functions.

Example
The following example generates a 100 ms waveform segment containing a simulated 50 Hz sine wave of 10 V amplitude, sampled at 10 kHz with 4 V peak-to-peak noise added:

\[
\text{Signal} = 10 \times \@SineWave(10k; 1000; 50) + 2 \times \@Noise(10k; 1000)
\]

Or you can add multiple sine waves for more complex waveforms:

\[
\text{Signal2} = 5 \times \@SineWave(10k; 1000; 50) + 7 \times \@SineWave(10k; 1000; 60) + 12 \times \@SineWave(10k; 1000; 90)
\]

See Also
"@Noise" on page 163, "@Pulse" on page 175, "@Ramp" on page 179 and "@SquareWave" on page 212
4.86  @Smooth

Function
Smoothing a waveform using a moving average filter.

Syntax
@Smooth(Waveform; N)

Parameters
Waveform  Waveform to be smoothed.
N  Number of samples of the smoothing factor.

Output
Smoothed waveform.

Description
The waveform is smoothed by taking a moving unweighted average over the specified number of samples according to the following formula:

\[
\text{Output}(n) = \frac{1}{N} \sum_{k = n - N'}^{} \text{Input}(k)
\]

With:
- N = Smoothing factor
- N' = (N-1)/2
- n = Sample number in range 1 to 'end-of-waveform'

The smoothing factor N controls the number of samples used for building the average. If N is not an odd integer, it will be rounded to the nearest odd integer first. If the value is smaller than 3, it is set to 3. If the value is larger than 1001, it is set to 1001. A higher value of N produces a more pronounced smoothing effect.

Smoothing effectively filters out high frequency components. The best filtering is achieved if the number of samples is equal to some integer multiple of the period of the disturbances. Due to the symmetric nature of the smoothing the position of characteristic features of the waveform is not changed.

The length of the output waveform is the same as the length of the input waveform. For smoothing near the edges, the function assumes that the waveform is extended with the same data values as the edge.
If a triangular weighted smoothing is required, this can be achieved by smoothing the waveform twice. To suppress only high frequency disturbances, it is often more effective to use the @Smooth function several times using a smaller number of samples than once using a large number of samples.

**Example**
The following example extracts the high frequency noise from a 50 Hz sine wave by subtracting the smoothed signal from the original signal:

\[
\text{Mains} = 220 \times \text{@SineWave}(200k; 4096; 50) \\
+ 10 \times \text{@Noise}(200k; 4096) \\
\text{Smo1} = \text{@Smooth} (\text{Formula.Mains}; 55) \\
\text{Noise} = \text{Formula.Mains} - \text{Formula.Smo1}
\]

**See Also**
"@Integrate" on page 127
4.87  @SpaceVectorInverseTransformation

**Function**
Generates a Waveform of the required Space Vector Inverse Transformation component for either amplitude-invariant or power-invariant transformations.

**Syntax**
@SpaceVectorInverseTransformation(Alpha; Beta; Zero(System); Component; Transformation)

**Parameters**
- **Alpha**: Input waveform of the alpha signal.
- **Beta**: Input waveform of the beta signal.
- **Zero(System)**: Input waveform or Value [rad] of the zero(system) signal.
- **Component**: Required output Component (0="signal1", 1="signal2", 2="signal3", 3="magnitude", 4="angle[rad]")
- **Transformation**: Invariant Inverse Transformation Rule (0="amplitude", 1="power")

**Output**
Waveform of the required component and transformation.

**Description**
The Space Vector Transformation (also known as the Clarke transformation) is a mathematical transformation that is used to simplify the analysis of three-phase power systems. After applying the Space Vector Transformation to the three-phases circuits to obtain the alpha, beta & zero(system) component, the Space Vector Inverse Transformation can be applied to these resulting signals to re-obtain the original input signals of the thee-phases.

The following formula is applied for amplitude-invariant inverse transformation:

\[
\begin{bmatrix}
  I_1 \\
  I_2 \\
  I_3
\end{bmatrix} = \frac{3}{2} \begin{bmatrix}
  2/3 & 0 & 2/3 \\
  -1/3 & \sqrt{3}/3 & 2/3 \\
  -1/3 & -\sqrt{3}/3 & 2/3
\end{bmatrix} \begin{bmatrix}
  I_{\alpha} \\
  I_{\beta} \\
  I_{\theta}
\end{bmatrix}
\]
Waveforms for the magnitude and direction are defined as:

\[ I_{Am} = \sqrt{I_{A\alpha}^2 + I_{A\beta}^2} \]
\[ I_{A\phi} = \text{atan2}(I_{A\beta}, I_{A\alpha}) \]

And for the power-invariant transformation, the following formula is applied:

\[
\begin{bmatrix}
I_1 \\
I_2 \\
I_3
\end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix}
\frac{2}{\sqrt{3}} & 0 & \frac{2}{\sqrt{3}} \\
-\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} \\
-\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}}
\end{bmatrix} \begin{bmatrix}
I_{P\alpha} \\
I_{P\beta} \\
I_{P\phi}
\end{bmatrix}
\]

Waveforms for the magnitude and direction are defined as:

\[ I_{pm} = \sqrt{I_{P\alpha}^2 + I_{P\beta}^2} \]
\[ I_{p\phi} = \text{atan2}(I_{P\beta}, I_{P\alpha}) \]

Where:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I_1)</td>
<td>Component of the phase 1 signal</td>
</tr>
<tr>
<td>(I_2)</td>
<td>Component of the phase 2 signal</td>
</tr>
<tr>
<td>(I_3)</td>
<td>Component of the phase 3 signal</td>
</tr>
<tr>
<td>(I_m)</td>
<td>Magnitude</td>
</tr>
<tr>
<td>(I_\phi)</td>
<td>Direction</td>
</tr>
<tr>
<td>(I_{A\alpha})</td>
<td>The amplitude-invariant “alpha” input value</td>
</tr>
<tr>
<td>(I_{A\beta})</td>
<td>The amplitude-invariant “beta” input value</td>
</tr>
<tr>
<td>(I_{A\phi})</td>
<td>The amplitude-invariant “zero(system)” input value</td>
</tr>
<tr>
<td>(I_{Am})</td>
<td>The amplitude-invariant “magnitude”</td>
</tr>
<tr>
<td>(I_{A\phi})</td>
<td>The amplitude-invariant “direction” (in radians)</td>
</tr>
<tr>
<td>(I_{P\alpha})</td>
<td>The power-invariant “alpha” input value</td>
</tr>
<tr>
<td>(I_{P\beta})</td>
<td>The power-invariant “beta” input value</td>
</tr>
<tr>
<td>(I_{P\phi})</td>
<td>The power-invariant “zero(system)” input value</td>
</tr>
<tr>
<td>(I_{pm})</td>
<td>The power-invariant “magnitude”</td>
</tr>
<tr>
<td>(I_{p\phi})</td>
<td>The power-invariant “direction” (in radians)</td>
</tr>
</tbody>
</table>
Notes:

- The Transformation parameter is optional and has the default value 0="amplitude".
- The input waveforms (Alpha, Beta & Zero(System)) all required the same sampling frequency.
- This function is only available in the Formula database sheet (see "Formula database sheet" on page 13) if the eDrive option (1-PERC-OP-EDR) is part of your license.

Example

Given that the three-phase input signals are available as Formula. SVATa, Formula. SVATb, Formula. SVATz, the following Space Vector amplitude transformation waveforms can be defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase1</td>
<td>@SpaceVectorInverseTransformation(Formula.SVATa; Formula.SVATb; Formula.SVATz; 0; 0)</td>
</tr>
<tr>
<td>Phase2</td>
<td>@SpaceVectorInverseTransformation(Formula.SVATa; Formula.SVATb; Formula.SVATz; 1; 0)</td>
</tr>
<tr>
<td>Phase3</td>
<td>@SpaceVectorInverseTransformation(Formula.SVATa; Formula.SVATb; Formula.SVATz; 2; 0)</td>
</tr>
<tr>
<td>Mag</td>
<td>@SpaceVectorInverseTransformation(Formula.SVATa; Formula.SVATb; Formula.SVATz; 3; 0)</td>
</tr>
<tr>
<td>Dir</td>
<td>@SpaceVectorInverseTransformation(Formula.SVATa; Formula.SVATb; Formula.SVATz; 4; 0)</td>
</tr>
</tbody>
</table>

Likewise the power-invariant variations are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase1</td>
<td>@SpaceVectorInverseTransformation(Formula.SVPTa; Formula.SVPTb; Formula.SVPTz; 0; 1)</td>
</tr>
<tr>
<td>Phase2</td>
<td>@SpaceVectorInverseTransformation(Formula.SVPTa; Formula.SVPTb; Formula.SVPTz; 1; 1)</td>
</tr>
<tr>
<td>Phase3</td>
<td>@SpaceVectorInverseTransformation(Formula.SVPTa; Formula.SVPTb; Formula.SVPTz; 2; 1)</td>
</tr>
<tr>
<td>Mag</td>
<td>@SpaceVectorInverseTransformation(Formula.SVPTa; Formula.SVPTb; Formula.SVPTz; 3; 1)</td>
</tr>
<tr>
<td>Dir</td>
<td>@SpaceVectorInverseTransformation(Formula.SVPTa; Formula.SVPTb; Formula.SVPTz; 4; 1)</td>
</tr>
</tbody>
</table>

The Zero(System) input parameter can also be specified as a value. Usually this waveform will have a constant 0 (zero) value. For example:
<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase1</td>
<td>@SpaceVectorInverseTransformation(Formula.SVPTa; Formula.SVPTb; 0; 0; 1)</td>
</tr>
</tbody>
</table>

See Also
"@SpaceVectorTransformation" on page 208
4.88  @SpaceVectorTransformation

Function
Generates a Waveform for the required Space Vector Transformation component for either amplitude-invariant or power-invariant transformations.

Syntax
@SpaceVectorTransformation(Signal1; Signal2; Signal3; Component; Transformation)

Parameters
- **Signal1**: Input waveform of phase1 signal.
- **Signal2**: Input waveform of phase2 signal.
- **Signal3**: Input waveform of phase3 signal.
- **Component**: Required output Component (0="alpha", 1="beta", 2="zero(system)", 3="magnitude", 4="angle[rad]")
- **Transformation**: Invariant Transformation Rule (0="amplitude", 1="power")

Output
The waveform of the required component and transformation.

Description
The Space Vector Transformation (also known as the Clarke transformation) is a mathematical transformation that is used to simplify the analysis of three-phase power systems.

The Vector Space Transformation is defined by the following matrix and can be applied for any three-phase quantities (e.g. voltage, currents, etc.)

The following formula is applied for amplitude-invariant transformation:

\[
\begin{bmatrix}
I_{aA} \\
I_{aB} \\
I_{a0}
\end{bmatrix}
= \frac{2}{3}
\begin{bmatrix}
1 & -1/2 & -1/2 \\
0 & \sqrt{3}/2 & -\sqrt{3}/2 \\
1/2 & 1/2 & 1/2
\end{bmatrix}
\begin{bmatrix}
I_1 \\
I_2 \\
I_3
\end{bmatrix}
\]
Waveforms for the magnitude and direction are defined as:

\[ \begin{align*}
I_{Am} &= \sqrt{I_{A\alpha}^2 + I_{A\beta}^2} \\
I_{A\phi} &= \tan^{-1}(I_{A\beta}, I_{A\alpha})
\end{align*} \]

And for the power-invariant transformation, the following formula is applied:

\[
\begin{bmatrix}
I_{P\alpha} \\
I_{P\beta} \\
I_{P\phi}
\end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix}
1 & -\frac{1}{2} & -\frac{1}{2} \\
0 & \sqrt{3}/2 & -\sqrt{3}/2 \\
1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2}
\end{bmatrix} \begin{bmatrix}
I_1 \\
I_2 \\
I_3
\end{bmatrix}
\]

Waveforms for the magnitude and direction are defined as:

\[ \begin{align*}
I_{pm} &= \sqrt{I_{P\alpha}^2 + I_{P\beta}^2} \\
I_{P\phi} &= \tan^{-1}(I_{P\beta}, I_{P\alpha})
\end{align*} \]

Where:

- \( I_1 \) Value of the phase 1 signal
- \( I_2 \) Value of the phase 2 signal
- \( I_3 \) Value of the phase 3 signal
- \( I_{A\alpha} \) The amplitude-invariant “alpha” component
- \( I_{A\beta} \) The amplitude-invariant “beta” component
- \( I_{A\phi} \) The amplitude-invariant “zero(system)” component
- \( I_{Am} \) The amplitude-invariant “magnitude”
- \( I_{A\phi} \) The amplitude-invariant “direction” (in radians)
- \( I_{P\alpha} \) The power-invariant “alpha” component
- \( I_{P\beta} \) The power-invariant “beta” component
- \( I_{P\phi} \) The power-invariant “zero(system)” component
- \( I_{Pm} \) The power-invariant “magnitude”
- \( I_{P\phi} \) The power-invariant “direction” (in radians)
Notes:

- The Transformation parameter is optional and has the default value 0="amplitude".
- The input waveforms (phase1, phase2 & phase3) all required the same sampling frequency.
- This function is only available in the Formula database sheet (see "Formula database sheet" on page 13) if the eDrive option (1-PERC-OP-EDR) is part of your license.

Example

Given that the three-phase input signals are available as Formula.Phase1, Formula.Phase2, Formula.Phase3, the following Space Vector amplitude transformation waveforms can be defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVATa</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 0; 0)</td>
</tr>
<tr>
<td>SVATb</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 1; 0)</td>
</tr>
<tr>
<td>SVATz</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 2; 0)</td>
</tr>
<tr>
<td>SVATm</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 3; 0)</td>
</tr>
<tr>
<td>SVATd</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 4; 0)</td>
</tr>
</tbody>
</table>

Likewise the power-invariant variations are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVPTa</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 0; 1)</td>
</tr>
<tr>
<td>SVPTb</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 1; 1)</td>
</tr>
<tr>
<td>SVPTz</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 2; 1)</td>
</tr>
<tr>
<td>SVPTm</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 3; 1)</td>
</tr>
<tr>
<td>SVPTd</td>
<td>@SpaceVectorTransformation(Formula.Phase1; Formula.Phase2; Formula.Phase3; 4; 1)</td>
</tr>
</tbody>
</table>

See Also

"@SpaceVectorInverseTransformation" on page 204 and 
"@DQ0Transformation" on page 78
4.89  @Sqrt

**Function**
Calculates the square root of the given parameter.

**Syntax**
@Sqrt(Par)

**Parameter**
*Par*  Input waveform or numerical value.

**Output**
Waveform or numerical value containing the square root of the input.

**Description**
This function calculates the square root of the input parameter.

If a waveform parameter is used, the square root function is calculated for each individual sample. Negative values in a waveform will result in the negative square root of the absolute value. For numerical values the square root of a negative value does not exist.

**Example**
Examples of possible square root operations are:

\[
\begin{align*}
\text{Signal} & = 4 \times @\text{SineWave}(10k; 1000; 50) \\
\text{Result} & = @\text{Sqrt}(\text{Formula.Signal}) \\
\text{Five} & = @\text{Sqrt}(25) \\
\text{NoValue} & = @\text{Sqrt}(-1)
\end{align*}
\]

**See Also**
"@Pow" on page 168
4.90  @SquareWave

Function
Generates a waveform containing a square wave.

Syntax
@SquareWave(FSampling; NSamples; NFrequency)

Parameters
FSampling  Number: sample frequency of the waveform.
NSamples  Number of samples in the waveform.
NFrequency  Number: frequency of the generated square wave.

Output
Waveform containing a square wave.

Description
This function creates a square wave. The sampling frequency, number of samples to generate and square wave frequency can be specified. There is no limit to the length of the waveform. The amplitude of the square wave is 1 V.

The possibility to generate a square wave function is used to synthesize a variety of waveforms. The simulated data can be used as input for other analysis functions.

Example
The following example generates a 100 ms waveform segment containing a simulated 50 Hz square wave of 10 V amplitude, sampled at 10 kHz with 2 V peak-to-peak noise added:

\[
\text{Signal} = 10 \times \text{@SquareWave}(10k;1000;50) + 2 \times \text{@Noise}(10k;1000)
\]

See Also
"@Noise" on page 163, "@Pulse" on page 175, "@Ramp" on page 179 and "@SineWave" on page 201
4.91 @StdDev

Function
Calculates the standard deviation of a waveform.

Syntax
@StdDev(Waveform)
@StdDev(Waveform; Begin)
@StdDev(Waveform; Begin; End)

Parameters
Waveform Input waveform for which the standard deviation is to be calculated.
Begin Number: segment begin.
End Number: segment end.

Output
The output is a numerical value.

Description
The standard deviation is a statistical characteristic of a waveform. It is approximately the root means square deviation of the samples in the waveform from the mean value of the waveform: a measure of the spread of its values.

The standard deviation is calculated using the following formula:

\[
\text{StdDev} = \sqrt{\frac{1}{N-1} \sum_{n=n_1}^{n_2} (y(n) - \bar{y})^2}
\]

in which: \( \bar{y} = \frac{1}{N} \sum_{n=n_1}^{n_2} y(n) \)

with \( N = (n_2 - n_1 + 1) \)

The segment limits (Begin and End) are used to select a range of samples. If no segment limits are specified, the complete waveform is used.

When only Begin is specified, the waveform segment from Begin to the end of the waveform is used.
Note  The **Begin** and **End** parameters have to be specified in the units of the horizontal axis (for example time) and not in samples.

Example

The following example creates a sine wave of 50 Hz and an RMS amplitude of 1 on top of a DC component of 100. From this signal the standard deviation is determined:

\[
\text{Signal} = 100 + \sqrt{2} \times \text{SineWave}(20k; 2000; 50) \\
\text{StdDev} = \text{StdDev}(\text{Formula.Signal})
\]

See Also

"@Energy" on page 82, "@Mean" on page 147 and "@RMS" on page 198
4.92 @STLX_SignalStart

Function
This function is used to recognize the start of a signal.

Syntax
@STLX_SignalStart(Waveform; Frequency; StartPos; EndPos)

Parameters
- **Waveform**: Sinusoidal input waveform.
- **Frequency**: Frequency of the sinusoidal waveform.
- **StartPos**: Optional; beginning of search. This parameter is optional if not entered the lowest possible time value is used.
- **EndPos**: Optional; end of search. This parameter is optional if not entered the highest possible time value is used.

Output
A Numerical value indicating the start of the signal.

Description
This function is using the existing @STLSignalStart() function, however, to be more accurate it is doing some extra steps:

- The function automatically rescales the input signal (similar to the @STLXRescale() function).
- The @STLSignalStart() is using the rescaled signal to find the start (Tstart1).
- Via a linear interpolation around the found start point this function tries to find a better start position (TStart2). This location is the zero crossing location of the line found via the linear interpolation.
0.5% of signal period

Figure 4.37: STLXSignal - TStart1 and TStart2 positions

See Also
TBD
4.93  @STLX_SignalEnd

Function
This function is used to recognize the end of a signal.

Syntax
@STLX_SignalEnd(Waveform; Frequency; StartPos; EndPos)

Parameters
- **Waveform**: Sinusoidal input waveform.
- **Frequency**: Frequency of the sinusoidal waveform.
- **StartPos**: Optional; beginning of search. This parameter is optional if not entered the lowest possible time value is used. Optional; end of search. This parameter is optional if not entered the highest possible time value is used.

Output
A Numerical value indicating the start of the signal.

Description
This function is using the existing @STLSignalEnd() function.

The function starts searching at “StartPos”, the search direction depends on the value of “EndPos”:
- Searches backwards if “StartPos” is greater than “EndPos”.
- Searches forwards if “EndPos” is greater than “StartPos”.

When “StartPos” and “EndPos” are not entered in the formula, the function starts searching at the beginning of the signal in forward direction until it finds an end condition or it stops when there are no more data points.

The following steps are done to get a more accurate end position:
- The function automatically rescales the input signal (similar to the @STLXRescale() function).
- The @STLSignalEnd() is using the rescaled signal to find the end location.
- Via a linear interpolation around the found end point this function tries to find a better end position. This location is the zero crossing location of the line found via the linear interpolation.

See Also
“@STLX_SignalStart” on page 215
Function
Selects a specific sweep from a multi-sweep recording.

Syntax
@Sweep(Signal; NSweep)

Parameters
- NSweep: Integer number of the sweep to be selected.

Output
Waveform containing only the selected sweep.

Description
Some recorder hardware and software settings make it possible to make multi-sweep recordings. Each valid trigger event will create a new sweep. The waveform of a channel from a multi-sweep recording contains multiple sweeps.

The @Sweep function makes it possible to select one specific sweep from the waveform. Nsweep is a numerical value in range 1 to number of sweeps.

Note
The selected waveform contains only one sweep.

Example
This example calculates the maximum value of the pressure in the second sweep of the recording.

```plaintext
PressureSweep = @Sweep
              (Active.Group1.Recorder_A.Ch_A2; 2)
MaxPressure  = @Max(Formula.PressureSweep)
```
4.95 \texttt{@SweptSineWave}

\textbf{Function}
This function generates a waveform described by the sine function sweeping from a start frequency to an end frequency.

\textbf{Syntax}
\begin{verbatim}
@SweptSineWave(FSampling; NSamples; FStart; FEnd)
@SweptSineWave(FSampling; NSamples; FStart; FEnd; Mode)
@SweptSineWave(FSampling; NSamples; FStart; FEnd; Mode; NLeadSamples)
\end{verbatim}

\textbf{Parameters}
\begin{itemize}
  \item \textit{FSampling} \hspace{1cm} Number: sampling frequency.
  \item \textit{NSamples} \hspace{1cm} Number: number of samples.
  \item \textit{FStart} \hspace{1cm} Number: start frequency, must be larger than 0.
  \item \textit{FEnd} \hspace{1cm} Number: end frequency, must be larger than the start frequency.
  \item \textit{Mode} \hspace{1cm} Number: the sweep mode, 0 = Logarithmic, 1 = Linear.
  \item \textit{NLeadSamples} \hspace{1cm} Number: number of lead samples.
\end{itemize}

\textbf{Output}
Waveform containing a swept sine wave.

\textbf{Description}
This function creates a swept sine wave with an amplitude of 1 V. The sampling frequency and number of samples can be specified.

The generated signal sweeps from start frequency to end frequency. The lead samples come prior to this. Lead samples are useful to prevent windowing effects. The frequency sweep can be either linear or logarithmic. The transition from lead-in to sweep is always at a zero-crossing.

\textbf{Example}
The following example will generate a waveform sampled at 20 kHz with a length of 80 k samples, starting at 1 Hz, logarithmic sweeping to 100 Hz from sample 30 k onwards.

\begin{verbatim}
Signal = @SweptSineWave(20k; 80k; 1; 100; 0; 30k)
\end{verbatim}
The output will look like (see Figure 4.38):

![Figure 4.38: @SweptSineWave - Example 1](image1)

The following example will generate a waveform, starting at 1 Hz, linear sweeping to 100 Hz.

\[
\text{Signal} = \text{@SweptSineWave}(20k; 80k; 1; 100; 0)
\]

The output will look like (see Figure 4.39):

![Figure 4.39: @SweptSineWave - Example 2](image2)

See Also

"@SineWave" on page 201
4.96 \(@\tan\)

**Function**
Calculates the tangent of the input parameter.

**Syntax**
\(@\tan(Par)\)

**Parameters**
- \(Par\)
  - Input waveform or numerical value.

**Output**
Waveform or numerical value containing the tangent of the input.

**Description**
The trigonometric function tangent is calculated assuming the input parameter is the angle in radians. When a waveform parameter is used, the tangent is calculated for each individual sample.

The function \(@\text{ATan}\) is the inverse trigonometric function of the function \(@\tan\).

**Example**
The following example calculates the tangent of the variable “Angle” specified in degrees:

\[
\begin{align*}
\text{Angle} &= 72 \\
\text{AngleRad} &= \text{System.Constants.Pi} \times \text{Formula.Angle} / 180 \\
\text{TanAngle} &= \@\tan(\text{Formula.AngleRad})
\end{align*}
\]

**See Also**
"\(@\text{ATan}\)" on page 59, "\(@\cos\)" on page 69 and "\(@\sin\)" on page 200
4.97 @TimeMaxAbove

Function
This function returns a number representing the time interval of the longest period that the signal is above a specified level.

Syntax
@TimeMaxAbove(Signal)
@TimeMaxAbove(Signal; Level)
@TimeMaxAbove(Signal; Level; Start)
@TimeMaxAbove(Signal; Level; Start; End)

Parameters
- **Signal**: Input waveform
- **Level**: Number: The used level default value is 0
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

Output
The output is a numerical value representing the time interval of the longest period that the signal is above a specified level.

Description
This function is used when searching for the longest period that a signal is above a specified level. In Figure 4.40 this is P1.

![Figure 4.40: TimeMaxAbove](image-url)
See Also
"@TimeMaxAboveBegin" on page 224, "@TimeMinAbove" on page 230,
"@TimeTotalAbove" on page 238 and "@TimeMaxBelow" on page 226
4.98  @TimeMaxAboveBegin

Function
This function returns a number representing the start time of the longest period that the signal is above a specified level.

Syntax
@TimeMaxAboveBegin(Signal)
@TimeMaxAboveBegin(Signal; Level)
@TimeMaxAboveBegin(Signal; Level; Start)
@TimeMaxAboveBegin(Signal; Level; Start; End)

Parameters
Signal  Input waveform
Level   Number: The used level default value is 0
Start   Number: Start of segment, default begin of signal
End     Number: End of segment, default end of signal

Output
The output is a numerical value representing the start time of the longest period that the signal is above a specified level

Description
This function is used when you want to know where the longest period that a signal is above a specified level begins. In Figure 4.41 on page 225 the longest period is P1, the start of P1 is T1, this number will be returned by the @TimeMaxAboveBegin function.
Figure 4.41: TimeMaxAboveBegin

See Also
"@TimeMaxAbove" on page 222, "@TimeMinAbove" on page 230, "@TimeTotalAbove" on page 238 and "@TimeMaxBelow" on page 226
4.99 @TimeMaxBelow

Function
This function returns a number representing the time interval of the longest period that the signal is below a specified level.

Syntax
@TimeMaxBelow(Signal)
@TimeMaxBelow(Signal; Level)
@TimeMaxBelow(Signal; Level; Start)
@TimeMaxBelow(Signal; Level; Start; End)

Parameters
Signal Input waveform
Level Number: The used level default value is 0
Start Number: Start of segment, default begin of signal
End Number: End of segment, default end of signal

Output
The output is a numerical value representing the time interval of the longest period that the signal is below a specified level.

Description
This function is used when searching for the longest period that a signal is below a specified level. In Figure 4.42 this is P1.

Figure 4.42: TimeMaxBelow
See Also
"@TimeMaxBelowBegin" on page 228, "@TimeMinBelow" on page 234, "@TimeTotalBelow" on page 242 and "@TimeMaxAbove" on page 222
4.100 \texttt{@TimeMaxBelowBegin}

Function
This function returns a number representing the start time of the longest period that the signal is Below a specified level.

Syntax
\begin{align*}
\texttt{@TimeMaxBelowBegin(Signal)} \\
\texttt{@TimeMaxBelowBegin(Signal; Level)} \\
\texttt{@TimeMaxBelowBegin(Signal; Level; Start)} \\
\texttt{@TimeMaxBelowBegin(Signal; Level; Start; End)}
\end{align*}

Parameters
\begin{align*}
\textit{Signal} & \quad \text{Input waveform} \\
\textit{Level} & \quad \text{Number: The used level default value is 0} \\
\textit{Start} & \quad \text{Number: Start of segment, default begin of signal} \\
\textit{End} & \quad \text{Number: End of segment, default end of signal}
\end{align*}

Output
The output is a numerical value representing the start time of the longest period that the signal is below a specified level.

Description
This function is used when you want to know where the longest period that a signal is below a specified level begins. In Figure 4.43 on page 229 the longest period is \( P_1 \), the start of \( P_1 \) is \( T_1 \), this number will be returned by the \texttt{@TimeMaxBelowBegin} function.
Figure 4.43: TimeMaxBelowBegin

See Also
"@TimeMaxBelow" on page 226, "@TimeMinBelow" on page 234, 
"@TimeTotalBelow" on page 242 and "@TimeMaxAbove" on page 222
4.101  @TimeMinAbove

**Function**
This function returns a number representing the time interval of the shortest period that the signal is above a specified level.

**Syntax**
- @TimeMinAbove(Signal)
- @TimeMinAbove(Signal; Level)
- @TimeMinAbove(Signal; Level; Start)
- @TimeMinAbove(Signal; Level; Start; End)

**Parameters**
- **Signal**: Input waveform
- **Level**: Number: The used level default value is 0
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

**Output**
The output is a numerical value representing the time interval of the shortest period that the signal is above a specified level.

**Description**
This function is used when searching for the shortest period that a signal is above a specified level. In the picture below this is P5.

**Figure 4.44: TimeMinAbove**
See Also
"@TimeMinAboveBegin" on page 232, "@TimeMinAbove" on page 230,
"@TimeTotalAbove" on page 238 and "@TimeMinBelow" on page 234
4.102 @TimeMinAboveBegin

Function
This function returns a number representing the start time of the shortest period that the signal is above a specified level.

Syntax
@TimeMinAboveBegin(Signal)
@TimeMinAboveBegin(Signal; Level)
@TimeMinAboveBegin(Signal; Level; Start)
@TimeMinAboveBegin(Signal; Level; Start; End)

Parameters
Signal        Input waveform
Level         Number: The used level default value is 0
Start         Number: Start of segment, default begin of signal
End           Number: End of segment, default end of signal

Output
The output is a numerical value representing the start time of the shortest period that the signal is above a specified level.

Description
This function is used when you want to know where the longest period that a signal is above a specified level begins. In Figure 4.45 on page 233 the shortest period is P₅, the start of P₅ it T₅, this number will be returned by the @TimeMinAboveBegin function.
Figure 4.45: TimeMinAboveBegin

See Also
"@TimeMinAbove" on page 230, "@TimeTotalAbove" on page 238 and "@TimeMinBelow" on page 234
4.103  \*@TimeMinBelow*

**Function**
This function returns a number representing the time interval of the shortest period that the signal is below a specified level.

**Syntax**
\[@TimeMinBelow(Signal)\]
\[@TimeMinBelow(Signal; Level)\]
\[@TimeMinBelow(Signal; Level; Start)\]
\[@TimeMinBelow(Signal; Level; Start; End)\]

**Parameters**

- **Signal**  
  Input waveform
- **Level**  
  Number: The used level default value is 0
- **Start**  
  Number: Start of segment, default begin of signal
- **End**  
  Number: End of segment, default end of signal

**Output**
The output is a numerical value representing the time interval of the shortest period that the signal is below a specified level.

**Description**
This function is used when searching for the shortest period that a signal is below a specified level. In the picture below this is \(P_4\).

**Figure 4.46: TimeMinBelow**
See Also
"@TimeMinBelowBegin" on page 236, "@TimeTotalBelow" on page 242 and
"@TimeMinBelow" on page 234
4.104 \@TimeMinBelowBegin

**Function**
This function returns a number representing the start time of the shortest period that the signal is above a specified level.

**Syntax**
\@TimeMinBelowBegin(Signal)
\@TimeMinBelowBegin(Signal; Level)
\@TimeMinBelowBegin(Signal; Level; Start)
\@TimeMinBelowBegin(Signal; Level; Start; End)

**Parameters**
- **Signal**: Input waveform
- **Level**: Number: The used level default value is 0
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

**Output**
The output is a numerical value representing the start time of the shortest period that the signal is above a specified level.

**Description**
This function is used when you want to know where the longest period that a signal is above a specified level begins. In Figure 4.47 on page 237 the shortest period is \( P_4 \), the start of \( P_4 \) is \( T_4 \), this number will be returned by the \@TimeMinBelowBegin function.
Perception Analysis

Figure 4.47: TimeMinBelowBegin

See Also
"@TimeMinBelow" on page 234 and "@TimeTotalBelow" on page 242
4.105  @TimeTotalAbove

**Function**
This function returns a number representing the total amount of time that the signal is above a specified level.

**Syntax**
@TimeTotalAbove(Signal)
@TimeTotalAbove(Signal; Level)
@TimeTotalAbove(Signal; Level; Start)
@TimeTotalAbove(Signal; Level; Start; End)

**Parameters**
- **Signal**: Input waveform
- **Level**: Number: The used level default value is 0
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

**Output**
The output is a numerical value representing the total amount of time that the signal is above a specified level.
Description
This function is used when you want to know how long a signal has been above a specified level. In Figure 4.48 this is the sum of the interval lengths of P1, P2, P3, P4 and P5.

See Also
"@TimeTotalAboveBegin" on page 240, "@TimeMinAbove" on page 230, "@TimeTotalAbove" on page 238 and "@TimeMaxBelow" on page 226
4.106  @TimeTotalAboveBegin

**Function**
This function returns a number representing the start of the first period where the signal is above the specified level.

**Syntax**
@TimeTotalAboveBegin(Signal)
@TimeTotalAboveBegin(Signal; Level)
@TimeTotalAboveBegin(Signal; Level; Start)
@TimeTotalAboveBegin(Signal; Level; Start; End)

**Parameters**
- **Signal**  
  Input waveform
- **Level**  
  Number: The used level default value is 0
- **Start**  
  Number: Start of segment, default begin of signal
- **End**  
  Number: End of segment, default end of signal

**Output**
The output is a numerical value representing the start of the first period where the signal is above the specified level.

**Description**
This function is used when you want to know where the first period where the signal is above the specified level starts. In Figure 4.49 on page 241 this is $T_1$, this number will be returned by the @TimeTotalAboveBegin function.
Figure 4.49: TimeTotalAboveBegin

See Also
"@TimeTotalAbove" on page 238, "@TimeMinAbove" on page 230 and "@TimeMaxBelow" on page 226
4.107  @TimeTotalBelow

**Function**
This function returns a number representing the total amount of time that the signal is below a specified level.

**Syntax**

@TimeTotalBelow(Signal)
@TimeTotalBelow(Signal; Level)
@TimeTotalBelow(Signal; Level; Start)
@TimeTotalBelow(Signal; Level; Start; End)

**Parameters**

- **Signal**  
  Input waveform
- **Level**  
  Number: The used level default value is 0
- **Start**  
  Number: Start of segment, default begin of signal
- **End**  
  Number: End of segment, default end of signal

**Output**
The output is a numerical value representing the total amount of time that the signal is below a specified level.

**Description**
This function is used when you want to know how long a signal has been below a specified level. In Figure 4.50 this is the sum of the interval lengths of P1, P2, P3, P4, P5 and P6.
Figure 4.50: TimeTotalBelow

See Also
"@TimeTotalBelowBegin" on page 244, "@TimeMinAbove" on page 230, "@TimeTotalBelow" on page 242 and "@TimeMaxBelow" on page 226
4.108  @TimeTotalBelowBegin

**Function**
This function returns a number representing the start of the first period where
the signal is below the specified level.

**Syntax**

@TimeTotalBelowBegin(Signal)
@TimeTotalBelowBegin(Signal; Level)
@TimeTotalBelowBegin(Signal; Level; Start)
@TimeTotalBelowBegin(Signal; Level; Start; End)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>Input waveform</td>
</tr>
<tr>
<td>Level</td>
<td>Number: The used level default value is 0</td>
</tr>
<tr>
<td>Start</td>
<td>Number: Start of segment, default begin of signal</td>
</tr>
<tr>
<td>End</td>
<td>Number: End of segment, default end of signal</td>
</tr>
</tbody>
</table>

**Output**
The output is a numerical value representing the start of the first period where
the signal is below the specified level.

**Description**
This function is used when you want to know where the first period where the
signal is below the specified level starts. In Figure 4.51 on page 245 this is T1, this number will be returned by the @TimeTotalBelowBegin function.
Figure 4.51: TimeTotalBelowBegin

See Also
"@TimeTotalBelow" on page 242, "@TimeMinAbove" on page 230 and "@TimeMaxBelow" on page 226
4.109  @TriggerTime

Function
Returns the trigger position of a selected sweep.

Syntax
@TriggerTime(Waveform)
@TriggerTime(Waveform; NSweep)

Parameters
Waveform  Waveform containing at least one sweep.
NSweep    Number: selected sweep, default = 1.

Output
A number containing the trigger position in X-Units.

Description
Some recorder hardware and software settings make it possible to make multi-sweep recordings. Each valid trigger event will create a new sweep. The waveform of a channel from a multi-sweep recording contains trigger time information for each individual sweep.

The @TriggerTime function makes it possible to get the trigger time in X-units for the specified sweep number.

This function can be used in combination with for example the “SweepReview” display. This kind of display shows only one single sweep. The display can move to each individual sweep. The trigger position of the shown sweep is always referenced to zero. Also the cursor values are referenced to the trigger position of the sweep. When a calculation is needed involving a cursor position, the actual location of the cursor in the original waveform needs to be calculated using the corresponding trigger time.

Example
This example shows the calculation of the maximum value of the voltage 20 ms around the trigger position of the second sweep from a multi-sweep recording.

\[
\begin{align*}
\text{Signal} & = \text{Active.Group1.Recorder\_A.Ch\_A2} \\
\text{TPos} & = @\text{TriggerTime(Formula.Signal; 2)}
\end{align*}
\]
MaxVoltage = @Max(Formula.Signal; Formula.TPos - 10m; Formula.TPos + 10m)

See Also
"@TriggerTimeToText" on page 248
4.110  @TriggerTimeToText

Function
Returns the trigger position of the selected sweep in a time-date formatted string.

Syntax
@TriggerTimeToText(Waveform)
@TriggerTimeToText(Waveform; NSweep)
@TriggerTimeToText(Waveform; NSweep; Decimals)
@TriggerTimeToText(Waveform; NSweep; Decimals; TimeFormat)
@TriggerTimeToText(Waveform; NSweep; Decimals; TimeFormat; DateFormat)

Parameters
Waveform  Waveform with at least one sweep
NSweep    Number: selected sweep, default = 1
Decimals  Number of decimals for the fractional part of the seconds, in the range 0 through 9. Default = 3
TimeFormat “Relative” (default)  Time since start of recording
        “Local”           Local time
        “UTC”            Coordinated Universal Time (UTC)
DateFormat “None” (default)    No date information
        “Short”          Short date string representation
        “Long”           Long date string representation

Output
The output is a string that represents the trigger position.

Description
Various recorder hardware and software settings allow you to create multi-sweep recordings. Each valid trigger event will create a new sweep. The waveform of a channel from a multi-sweep recording contains trigger time information for each individual sweep.

The @TriggerTimeToText function makes it possible to get the trigger time of a specified sweep as a text string.
This function can be used in combination with for example the Perception SweepReview display. This display shows a single sweep. The display can move to each individual sweep. The trigger position of that sweep is always referenced to zero; however with the TriggerTimeToText function it is possible to present the trigger time in other relations and in different formats.

The date and time formats are associated with the regional settings of Windows and therefore they can vary from machine to machine. Refer to your Windows Help for more details.

**Note**

*This function works only for recordings recorded with internal time base, this means that the x-axis dimension is time. The Time and Date format strings are not case-sensitive. The return value is a string that can also be used and interpreted by other programs (like Excel) as a string.*

**Example**

This example shows the trigger time of the second sweep in UTC time.

```plaintext
Signal = Active.Group1.Recorder_A.Ch_A2
TriggerUTC = @TriggerTimeToText
           (Formula.Signal; 2; 6; "UTC"; "Short")
```

The output string could be: "3-3-2008 10:40:45.301455"

When only the time is required:

```plaintext
TriggerTime = @TriggerTimeToText
             (Formula.Signal; 2; 0; "Local"; "none")
```

This could result in: "11:40:45"

**See Also**

"@TriggerTime" on page 246
4.111 @TrueFrequency

Function
Returns a waveform that represents the frequency as function of time.

Syntax
@TrueFrequency(Waveform)
@TrueFrequency(Waveform; Cycles)
@TrueFrequency(Waveform; Cycles; Fc)
@TrueFrequency(Waveform; Cycles; Fc; Hysteresis)
@TrueFrequency(Waveform; Cycles; Fc; Hysteresis; Begin)
@TrueFrequency(Waveform; Cycles; Fc; Hysteresis; Begin; End)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform</td>
<td>Waveform for the Frequency calculation.</td>
</tr>
<tr>
<td>Cycles</td>
<td>Number of cycles to be used for the calculating of the frequency, the number of cycles has to be an integer value (1, 2, 3 .. etc.). The default value is 1 cycle.</td>
</tr>
<tr>
<td>Fc</td>
<td>Cut-off frequency of 2nd order phase less lowpass Bessel filter. This filter is used to filter the input signal before looking at the zero crossings. If not entered or 0 then no filtering is done.</td>
</tr>
<tr>
<td>Hysteresis</td>
<td>Number: Hysteresis in % of the difference between the maximum and minimum value of the input signal. This hysteresis is used to find the correct zero crossing values. The default value is 20%.</td>
</tr>
<tr>
<td>Begin</td>
<td>Number: Start of segment, default begin of signal</td>
</tr>
<tr>
<td>End</td>
<td>Number: End of segment, default end of signal</td>
</tr>
</tbody>
</table>

Output
Waveform containing the Frequency value as function of time started at Begin until End.
**Description**  
This function will create a new waveform with the same sample rate as the original signal. The actual value of the created waveform is the value of the frequency at specific times of the supplied waveform. The parameter Cycles determines how many cycles have to be used for calculating the frequency in a given point, by default this is one cycle. The function uses the zero crossing to determine a cycle. The hysteresis value is used to determine the correct zero crossings and eliminate incorrect zero crossings caused by noise. An internal 2nd order phase less lowpass Bessel filter can be used to filter the signal before the zero crossings are detected. If the parameter Fc has not been entered or is equal to 0 then no filter is used.

**Example**  
The following example creates a waveform showing the true Frequency value of the voltage over each cycle. The maximum frequency is also calculated.

\[
\text{Frequency} = @\text{TrueFrequency} \\
\quad \text{(Active.Group1.Recorder_Voltage.Voltage; 5; 100)} \\
\text{MaxFreq} = @\text{Max(Formula.Frequency)}
\]
4.112  @TrueRMS

**Function**
Returns a waveform that represents the true RMS (Root Mean Square) per number of cycles.

**Syntax**
@TrueRMS(Waveform)
@TrueRMS(Waveform; Cycles; Begin; End)

**Parameters**
- **Waveform**: Waveform for the RMS calculation.
- **Cycles**: Number of cycles to be used for the RMS calculation.
- **Begin**: Number: segment begin
- **End**: Number: segment end

**Output**
Waveform containing the RMS value over the requested number of cycles.

**Description**
This function will create a new waveform with the same sample rate as the original signal. The actual value of the created waveform is the value of the true RMS of the supplied waveform.
The default value is 1 cycle. Other values will be converted to a multiple of 0.5 cycles. The function uses the zero crossing to determine a cycle. Use the Cycles parameter to set the number of cycles - in multiples of 0.5 - for the calculation.

**Example**
The following example creates a waveform showing the true RMS value of the voltage over each cycle. The average RMS over the complete signal is calculated.

```plaintext
RMS_Signal = @TrueRMS(Active.Group1.Recorder_Voltage.Voltage;1)
AverageRMS = @Mean(Formula.RMS_Signal)
```

**See Also**
"@RMS" on page 198
4.113  @TrueRMSRef

Function
Returns a waveform that represents the true RMS (Root Mean Square) per number of cycles. A reference waveform can also be used to calculate the cycles by determining the zero crossing values of the reference waveform.

Syntax
@TrueRMSRef(Waveform)
@TrueRMSRef(Waveform; Reference)
@TrueRMSRef(Waveform; Reference; Cycles)
@TrueRMSRef(Waveform; Reference; Cycles; Begin)
@TrueRMSRef(Waveform; Reference; Cycles; Begin; End)

Parameters
- Waveform: Waveform for the RMS calculation.
- Reference: Reference waveform to be used to determine the zero crossings.
- Cycles: Number of cycles to be used for the RMS calculation.
- Begin: Number: segment begin
- End: Number: segment end

Output
Waveform containing the RMS value over the requested number of cycles.

Description
This function will create a new waveform with the same sample rate as the original signal. The actual value of the created waveform is the value of the true RMS of the supplied waveform.
A reference waveform can be used to calculate the cycles by determining the zero crossing values from this reference waveform. If this reference waveform is not defined, the waveform itself will then be used to determine the cycles. The default number of cycles is 1. Other values will be converted to a multiple of 0.5 cycles. The function uses the zero crossing to calculate a cycle. Use the Cycles parameter to set the number of cycles - in multiples of 0.5 - for the calculation.

Example
The following example creates a waveform showing the true RMS value of the voltage over each cycle. The average RMS over the complete signal is calculated.
RMS_Signal = @TrueRMSRef
(Active.Group1.Recorder_Voltage.Voltage;
Active.Group1.Recorder_Voltage.RefSignal;
1)
AverageRMS = @Mean(Formula.RMS_Signal)

See Also
"@RMS" on page 198 and "@TrueRMS" on page 252
4.114  @Value

**Function**
Returns the amplitude value of a waveform at a specified x-position.

**Syntax**
@Value(Waveform; XPos)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform</td>
<td>Waveform whose value is to be determined at the specified x-position.</td>
</tr>
<tr>
<td>XPos</td>
<td>Number: x-position at which the value of the waveform is to be determined.</td>
</tr>
</tbody>
</table>

**Output**
Amplitude value of the waveform at the specified x-position.

**Description**
The value of a waveform at a specified x-position is determined. The x-position is to be specified as x-coordinate, not as a sample number. If the x-coordinate is between two sampling points, linear interpolation is used to determine the value at this x-position.

If the x-position is before the first sample or after the last sample of the waveform data set, the function returns no value.

**Example**
The following example searches a signal named Marker for a rising level crossing with level 2.5 and stores the time found in the variable TimeMarker. Then the value of another signal is determined at this position. Real-world signals are assumed.

```plaintext
Marker = Active.Group1.Recorder_A.Ch_A1
OtherSignal = Active.Group1.Recorder_A.Ch_A2
TimeMarker = @NextLvlCross(Formula.Marker; 0; 2.5; 1)
ValueAtMark = @Value(Formula.OtherSignal; Formula.TimeMarker)
```
4.115  @XDelta

**Function**
Returns the sampling interval (the time between samples) of a waveform.

**Syntax**
@XDelta(Waveform)

**Parameters**
Waveform  Input waveform.

**Output**
The output is a numerical value.

**Description**
This function returns the sampling interval (period) of a waveform. The sampling interval is the reciprocal (inverse) value of the sampling frequency and represents the difference in x units between adjacent samples.

**Example**
The following example creates a sine wave with a sampling frequency of 1 kHz. The variable dx receives the sampling interval of this signal (1 ms). The variable sf receives the sampling frequency:

\[
\text{Signal} = 25 \times \text{@SineWave}(1k; 1000; 50) \\
\text{dx} = \text{@XDelta(Formula.Signal)} \\
\text{sf} = 1.0 / \text{@XDelta(Formula.Signal)}
\]

**See Also**
"@Length" on page 136, "@XFirst" on page 259 and "@XLast" on page 260
4.116  @XDeltaHigh

**Function**
Determines the **highest** sampling interval of a waveform.

**Syntax**
@XDeltaHigh(Waveform)

**Parameters**
- **Waveform**
  Waveform for which the highest sampling interval is determined.

**Output**
The output is a numerical value: the difference in x units between two samples.

**Description**
When working with multi-time base waveforms, the highest sampling interval of the waveform is determined. The sampling interval is the reciprocal value of the sampling frequency and represents the difference in x units between adjacent samples.

**Example**
This example creates a waveform containing 2 different sample rates. The XDelta functions are used to determine the sampling frequency.

```
SignalLow = @SineWave(10k;10k;50)
SignalHigh = @SineWave(100k;10k;50)
Signal_LHL = @Join(Formula.SignalLow; Formula.SignalHigh; Formula.SignalLow)
XDelta = @XDelta(Formula.Signal_LHL)
XHigh = @XDeltaHigh(Formula.Signal_LHL)
XLow = @XDeltaLow(Formula.Signal_LHL)
```

**See Also**
"@XDelta" on page 256 and "@XDeltaLow" on page 258
4.117  @XDeltaLow

Function
Determines the lowest sampling interval of a waveform.

Syntax
@XDeltaLow(Waveform)

Parameters
Waveform Waveform for which the highest sampling interval is determined.

Output
The output is a numerical value: the difference in x units between two samples.

Description
When working with multi-time base waveforms, the lowest sampling interval of the waveform is determined. The sampling interval is the reciprocal value of the sampling frequency and represents the difference in x units between adjacent samples.

Example
This example creates a waveform containing 2 different sample rates. The XDelta functions are used to determine the sampling frequency.

\[
\begin{align*}
\text{SignalLow} & = @\text{SineWave}(10k;10k;50) \\
\text{SignalHigh} & = @\text{SineWave}(100k;10k;50) \\
\text{Signal\_LHL} & = @\text{Join}(\text{SignalLow}; \text{SignalHigh}; \text{SignalLow}) \\
\text{XDelta} & = @\text{XDelta}(\text{Signal\_LHL}) \\
\text{XHigh} & = @\text{XDeltaHigh}(\text{Signal\_LHL}) \\
\text{XLow} & = @\text{XDeltaLow}(\text{Signal\_LHL})
\end{align*}
\]

See Also
"@XDelta" on page 256 and "@XDeltaHigh" on page 257
4.118  @XFirst

Function
Returns the x-coordinate of the first sample in a waveform.

Syntax
@XFirst(Waveform)

Parameters
Waveform  Input waveform.

Output
The output is a numerical value.

Description
The x-coordinate of the first sample is returned. The x-coordinate is a value in terms of the horizontal units of the waveform (typically time).

Example
The following example creates a sine wave. Generated signals always start at time 0. The waveform is shifted horizontally by 100 ms. When the x-coordinate of the first sample of this shifted waveform is determined, it will be 100 ms.

\[
\begin{align*}
\text{Signal} & = @\text{SineWave}(10k; 1000; 50) \\
\text{Shifted} & = @\text{XShift}(\text{Formula.Signal}; 100m) \\
\text{Start} & = @\text{XFirst}(\text{Formula.Shifted})
\end{align*}
\]

See Also
"@Length" on page 136, "@XDelta" on page 256, "@XLast" on page 260 and "@XShift" on page 261
4.119  @XLast

Function
Returns the x-coordinate of the last sample in a waveform.

Syntax
@XLast(Waveform)

Parameters
Waveform  Input waveform.

Output
The output is a numerical value.

Description
The x-coordinate of the last sample is returned. The x-coordinate is a value in terms of the horizontal units of the waveform (typically time).

Example
The following example creates a sine wave. Generated signals always start at time 0. The example shows two ways of determining the x-coordinate of the last sample in the waveform:

```
Signal = @SineWave(10k; 1000; 50)
XEnd1    = @XFirst(Formula.Signal) + 
          (@Length(Formula.Signal) - 1)* 
          @XDelta(Formula.Signal)
XEnd2    = @XLast(Formula.Signal)
```

See Also
"@Length" on page 136, "@XDelta" on page 256, "@XFirst" on page 259 and "@XShift" on page 261
4.120  @XShift

**Function**
*Shifts* a waveform horizontally by a specified amount of time.

**Syntax**
@XShift(Waveform, Shift)

**Parameters**
- *Waveform*: Waveform to be shifted horizontally.
- *Shift*: Number: amount of shift.

**Output**
Original waveform shifted in time.

**Description**
This function does not alter the contents of a waveform but changes the horizontal scaling information in such a way that the waveform is shifted horizontally by the specified amount of time (or other units). Positive values shift the waveform to the right, negative values shift the waveform to the left.

**Example**
The following example makes sure the waveform’s horizontal axis always starts at zero. We assume a real-world signal.

```plaintext
Signal = Active.Group1.Recorder_A.Ch_A1
Shift = -1 * @XFirst(Formula.Signal)
CorrSignal = @XShift(Formula.Signal; Formula.Shift)
```

**See Also**
"@XDelta" on page 256, "@XFirst" on page 259 and "@XLast" on page 260
4.121 @XYArray

Function
Creates a waveform using a two-dimensional array.

Syntax
@XYArray(X1; Y1; ...; ...; XN; YN)

Parameters
X1  X value of first sample.
Y1  Y value of first sample.
XN  X value of last sample; N>= 2.
YN  Y value of last sample; N>= 2.

Output
Waveform containing all the points of the array.

Description
This function creates a waveform containing all the points passed by the
parameters X1, Y1, X2, Y2 etc. The entered points should be ordered in time.

The output waveform contains equidistant data points. This function will
evaluate the input X values and define the best possible new sample rate. For
the sample rate the 1, 2, 5 series is used.

Assume X1 = 0 ms, X2 = 30 ms, X3 = 50 ms. This yields a sampling interval of
20 ms.

Assume X1 = 25 ms, X2 = 30 ms, X3 = 50 ms. This yields a sampling interval
of 5 ms.

Assume X1 = 2 s, X2 = 12 s, X3 = 15 s. This yields a sampling interval of 2 s.

Note
The output samples are equidistant; the number of samples can be larger than
the number of input samples.

Example
The following example creates a waveform containing 8 data points, while only
5 points were entered. The three new points are generated by the XYArray
function.
NewWave = @XYArray(0; 0; 1; 2; 2; 1; 5; -3; 7; 0)

See Also
"@YArray" on page 264
4.122  @YArray

Function
This function creates a waveform using a single dimensional array as data input.

Syntax
@YArray(Fsampling; Y1; ...; YN)

Parameters
- **FSampling**  
  Number: Sampling frequency
- **Y1**  
  Number: amplitude (Y-value) of first data point
- **YN**  
  Number: amplitude (Y-value) of last data point. N >= 2

Output
The output is a waveform containing all data points of the array.

Description
This function creates a waveform containing all the points passed by the parameters Y1 through YN. The first parameter defines the sampling frequency (= 1 / sampling period)

Example
The following example creates a waveform containing 9 data points with a sampling frequency of 100 Hz.

\[ \text{Signal} = @YArray(100; 0; 1; 2; 2; 1; -1; -3; 2; 0) \]

See Also
"@XYArray" on page 262
5 Reference Guide - HIC

5.1 Introduction - HIC

HIC functions are used for analyzing crash test data. HIC is an abbreviation of Head Injury Criterion.
5.2 @Con3ms

Function
This function returns the highest acceleration level with a continuous duration of at least 3 milliseconds.

Syntax
@Con3ms(Signal)
@Con3ms(Signal; Start)
@Con3ms(Signal; Start; End)

Parameters
Signal: Input waveform
Start: Number: Start of segment, default begin of signal
End: Number: End of segment, default end of signal

Output
The output is a numerical value containing highest acceleration level with a cumulative duration of at least 3 milliseconds.

Description
Con3ms injury criterion calculates the highest acceleration level with a continuous duration of at least 3 milliseconds. For an implementation we refer to the document: Crash Analysis Criteria Description Version 2.1.1 from the Data Processing Vehicle Safety Workgroup, chapter 6 page 6-2. In this document they call it Xms.
The Con3ms value can be calculated with one peak, as shown in Figure 5.1, or with several peaks, as shown in Figure 5.2.

**Figure 5.1:** Con3ms value (One peak)

**Figure 5.2:** Con3ms value (Several peaks)

**See Also**
"@Cum3ms" on page 269, "@Cum3ms_T" on page 271 and "@Con3ms_T" on page 268
5.3 \texttt{@Con3ms\_T}

\textbf{Function}
This function returns the start time of the highest acceleration level with a \textit{continuous} duration of at least \textbf{3 milliseconds}.

\textbf{Syntax}

\begin{align*}
\texttt{@Con3ms\_T(Signal)} \\
\texttt{@Con3ms\_T(Signal; Start)} \\
\texttt{@Con3ms\_T(Signal; Start; End)}
\end{align*}

\textbf{Parameters}

\begin{align*}
\textit{Signal} & \quad \text{Input waveform} \\
\textit{Start} & \quad \text{Number: Start of segment, default begin of signal} \\
\textit{End} & \quad \text{Number: End of segment, default end of signal}
\end{align*}

\textbf{Output}
The output is a numerical value containing start time of the highest acceleration level with a \textit{continuous} duration of at least 3 milliseconds.

\textbf{Description}
See also the Con3ms description on page 266.

\textbf{See Also}
"\texttt{@Con3ms}" on page 266, "\texttt{@Cum3ms}" on page 269 and "\texttt{@Cum3ms\_T}" on page 271
5.4 @Cum3ms

Function
This function returns the highest acceleration level with a cumulative duration of at least 3 milliseconds.

Syntax
@Cum3ms(Signal)
@Cum3ms(Signal; Start)
@Cum3ms(Signal; Start; End)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>Input waveform</td>
</tr>
<tr>
<td>Start</td>
<td>Number: Start of segment, default begin of signal</td>
</tr>
<tr>
<td>End</td>
<td>Number: End of segment, default end of signal</td>
</tr>
</tbody>
</table>

Output
The output is a numerical value containing highest acceleration level with a cumulative duration of at least 3 milliseconds.

Description
Cum3ms injury criterion calculates the highest acceleration level with a cumulative duration of at least 3 milliseconds. For an implementation we refer to the document: Crash Analysis Criteria Description Version 2.1.1 from the Data Processing Vehicle Safety Workgroup, chapter 6 page 6-2. In this document they call it Xms.

In the cumulative calculation, separate periods of the measurement signal are added, until 3 milliseconds are reached.
The Cum3ms value can be calculated with one peak, as shown in Figure 5.3, or with several peaks, as shown in Figure 5.4.

**Figure 5.3:** Cum3ms value (One peak)

The special case shown in the second figure.

**Figure 5.4:** Cum3ms value (Several peaks)

See Also
"@Con3ms" on page 266, "@Cum3ms_T" on page 271 and "@Con3ms_T" on page 268
5.5  \( \text{@Cum3ms}_T \)

**Function**
This function returns the start time of the highest acceleration level with a cumulative duration of at least 3 milliseconds.

**Syntax**
\[
\text{@Cum3ms}_T(Signal) \\
\text{@Cum3ms}_T(Signal; \ Start) \\
\text{@Cum3ms}_T(Signal; \ Start; \ End)
\]

**Parameters**
- **Signal**: Input waveform
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

**Output**
The output is a numerical value containing start time of the highest acceleration level with a cumulative duration of at least 3 milliseconds.

**Description**
See also the Cum3ms description on page 269.

**See Also**
"@Con3ms" on page 266, "@Cum3ms" on page 269 and "@Con3ms_T" on page 268
5.6  @HIC

Function
The HIC functions calculate the Head Injury Criterion of a waveform.

Syntax
@HIC(Signal)
@HIC(Signal; Start)
@HIC(Signal; Start; End)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>Input waveform</td>
</tr>
<tr>
<td>Start</td>
<td>Number: Start of segment, default begin of signal</td>
</tr>
<tr>
<td>End</td>
<td>Number: End of segment, default end of signal</td>
</tr>
</tbody>
</table>

Output
The output is a numerical value containing the calculated HIC value.

Description
HIC is the abbreviation for Head Injury Criterion. For an implementation we refer to the document: Crash Analysis Criteria Description Version 2.1.1 from the Data Processing Vehicle Safety Workgroup, chapter 2 page 2-2.

The Head Injury Criterion is a measure of the likelihood of head injury arising from an impact. The HIC can be used to assess safety related to vehicles, personal protective gear, and sport equipment.

Normally the variable is derived from the acceleration/time history of an accelerometer mounted at the centre of gravity of a dummy’s head, when the dummy is exposed to crash forces.

The HIC functions calculate the Head Injury Criterion of a waveform. The parameters **Start** and **End** are optional and define the part of the input signal to be used for the HIC calculations. The HIC value is calculated with the following formula:

\[
HIC = \sup_{t_1, t_2} \left\{ \left( \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} a \, dt \right)^{2.5} (t_2 - t_1) \right\} \\
\]

\[
a = \sqrt{a_x^2 + a_y^2 + a_z^2}
\]
The HIC function determines the interval defined by $t_1$ and $t_2$ where the HIC attains a maximum value. The variable $a$ is the resultant acceleration of the center of gravity of the head in units of acceleration of gravity ($1g = 9.81 \text{ m/s}^2$).

**See Also**

"@HICStartTime" on page 274 and "@HICEndTime" on page 275
5.7  @HICStartTime

Function
This function returns the start of the interval \( t_1 \) as found during the HIC calculations.

Syntax
@HICStartTime(Signal)
@HICStartTime(Signal; Start)
@HICStartTime(Signal; Start; End)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>Input waveform</td>
</tr>
<tr>
<td>Start</td>
<td>Number: Start of segment, default begin of signal</td>
</tr>
<tr>
<td>End</td>
<td>Number: End of segment, default end of signal</td>
</tr>
</tbody>
</table>

Output
The output is a numerical value containing the start of the interval \( t_1 \) as found during the HIC calculations.

Description
The HICStartTime function returns the start in time of the interval \( t_1 \) where the HIC has its maximum value. See also the HIC description on page 272.

See Also
"@HIC" on page 272 and "@HICEndTime" on page 275
5.8 @HICEndTime

Function
This function returns the end of the interval ($t_2$) as found during the HIC calculations.

Syntax
@HICEndTime(Signal)
@HICEndTime(Signal; Start)
@HICEndTime(Signal; Start; End)

Parameters
Signal Input waveform
Start Number: Start of segment, default begin of signal
End Number: End of segment, default end of signal

Output
The output is a numerical value containing the end of the interval ($t_2$) as found during the HIC calculations.

Description
The @HICEndTime function returns the end in time of the interval ($t_2$) where the HIC has its maximum value. See also the HIC description on page 272.

See Also
"@HIC" on page 272 and "@HICStartTime" on page 274
5.9  @HIC15

Function
This function returns the maximum HIC value when working with a fixed 15 millisecond interval.

Syntax
@HIC15(Signal)
@HIC15(Signal; Start)
@HIC15(Signal; Start; End)

Parameters
- **Signal**: Input waveform
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

Output
The output is a numerical value containing the HIC value when working with a fixed 15 millisecond interval.

Description
See also the HIC function on page 272.

See Also
"@HIC36" on page 279, "@HIC15StartTime" on page 277 and "@HIC15EndTime" on page 278
5.10  @HIC15StartTime

Function
This function returns the start of the interval \( t_1 \) as found during the HIC15 calculations.

Syntax
@HIC15StartTime(Signal)
@HIC15StartTime(Signal; Start)
@HIC15StartTime(Signal; Start; End)

Parameters
- **Signal**: Input waveform
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

Output
The output is a numerical value containing the start of the interval \( t_1 \) as found during the HIC15 calculations.

Description
The @HIC15StartTime function returns the start in time of the interval \( t_1 \) where the HIC has its maximum value and the interval has a maximum of 15 ms. See also the HIC description on page 272.

See Also
"@HIC15" on page 276 and "@HIC15EndTime" on page 278
5.11  @HIC15EndTime

Function
This function returns the end of the interval ($t_2$) as found during the HIC15 calculations.

Syntax

@HIC15EndTime(Signal)
@HIC15EndTime(Signal; Start)
@HIC15EndTime(Signal; Start; End)

Parameters

Signal    Input waveform
Start     Number: Start of segment, default begin of signal
End       Number: End of segment, default end of signal

Output
The output is a numerical value containing the end of the interval ($t_2$) as found during the HIC15 calculations.

Description
The HIC15EndTime function returns the end in time of the interval ($t_2$) where the HIC has its maximum value and the interval has a maximum of 15 ms. See also the HIC description on page 272.

See Also
"@HIC15" on page 276 and "@HIC15StartTime" on page 277
5.12  @HIC36

Function
This function returns the maximum HIC value when working with a fixed 36 millisecond interval.

Syntax
@HIC36(Signal)
@HIC36(Signal; Start)
@HIC36(Signal; Start; End)

Parameters
Signal     Input waveform
Start      Number: Start of segment, default begin of signal
End        Number: End of segment, default end of signal

Output
The output is a numerical value containing the HIC value when working with a fixed 36 millisecond interval.

Description
See also the HIC function on page 272.

See Also
"@HIC" on page 279, "@HIC36StartTime" on page 280 and "@HIC36EndTime" on page 281
5.13  @HIC36StartTime

**Function**
This function returns the start of the interval \( t_1 \) as found during the HIC36 calculations.

**Syntax**
@HIC36StartTime(Signal)
@HIC36StartTime(Signal; Start)
@HIC36StartTime(Signal; Start; End)

**Parameters**
- *Signal*: Input waveform
- *Start*: Number: Start of segment, default begin of signal
- *End*: Number: End of segment, default end of signal

**Output**
The output is a numerical value containing the start of the interval \( t_1 \) as found during the HIC36 calculations.

**Description**
The @HIC36StartTime function returns the start in time of the interval \( t_1 \) where the HIC has its maximum value and the interval has a maximum of 36 ms. See also the HIC description "Description" on page 272.

**See Also**
"@HIC36" on page 279 and "@HIC36EndTime" on page 281
5.14 @HIC36EndTime

Function
This function returns the end of the interval ($t_2$) as found during the HIC36 calculations.

Syntax
@HIC36EndTime(Signal)
@HIC36EndTime(Signal; Start)
@HIC36EndTime(Signal; Start; End)

Parameters
- **Signal**: Input waveform
- **Start**: Number: Start of segment, default begin of signal
- **End**: Number: End of segment, default end of signal

Output
The output is a numerical value containing the end of the interval ($t_2$) as found during the HIC36 calculations.

Description
The @HIC36EndTime function returns the end in time of the interval ($t_2$) where the HIC has its maximum value and the interval has a maximum of 36 ms. See also the HIC description on page 272.

See Also
"@HIC36" on page 279 and "@HIC36StartTime" on page 280
6 Reference Guide - Cycle Math

6.1 Introduction - Cycle Math

A characteristic of a sinusoid waveform is a cycle (or period). A full cycle is defined as the time segment where the waveform crosses its center amplitude in the same direction (rising or falling) twice.

The cycle math formulas calculate a characteristic value of a waveform on a cycle-per-cycle basis. In order to determine which input samples to use for the calculation of this characteristic value, the formulas employ a cycle detection algorithm, which determines and collects the times of center-amplitude crossings (or simply called zero crossings, if the center amplitude is zero).

The cycle detection algorithm principle works as follows (see Figure 6.1):

- Assume that the input signal is \( y(n) \), which is a sinusoid oscillating around a center amplitude.
- Let the Hysteresis Band be the amplitude range from the -threshold level to the +threshold level (user-defined).
- Find and store the sample at index \( k_1 \) where the signal is:
  - Entering the Hysteresis Band, or
  - Leaving the Hysteresis Band
- Find the next sample at index \( k_2 \), where the signal is: (symmetrically to the previous step)
  - Leaving the Hysteresis Band, or
  - Entering the Hysteresis Band
- Calculate the center amplitude crossing time by linear interpolation between the samples at indices \( k_1 \) and \( k_2 \). This crossing time detection can occur in two ways: either a rising edge crossing (point A) or a falling edge crossing (point B). See Figure 6.1.
- Finally, store all center-amplitude crossings; these can be then used by specific formulas listed in this section to perform cycle-based calculations.
A cycle formula works as follows:

- First, the cycle detection algorithm will determine the center amplitude crossing times.
- Each pair of crossing times, such as times A and B of Figure 6.1, defines a cycle time segment.
- The formula will gather the samples of the input waveform that belong to this time segment and apply a math function on them to calculate the cycle value.
- The formula’s output waveform will assign this value to all samples that belong to the respective time segment.

The cycle math formulas have the versatility of applying a math function on input samples for a duration of a full cycle, for a half-cycle, or for a multiple number of cycles. For instance, a user might want to see the RMS value of the samples that correspond to two consecutive cycles of the input sine wave.
Linear Interpolation

In order to calculate the level crossing time precisely, the cycle detection algorithm employs linear interpolation in the following manner:

For a signal $y(n)$, assume that a -threshold crossing is detected at sample index $k_1$ (time $t_1$), and a +threshold crossing is detected at sample index $k_2$ (time $t_2$). The time of the center amplitude crossing $t_C$ is then determined by the linear interpolation equation:

$$t_C = t_1 + (CA - y(k_1)) \cdot \frac{t_2 - t_1}{y(k_2) - y(k_1)}$$

where $CA$ = Center Amplitude level

If the input signal contains noise, this calculation may result in a $t_C$ value that differs significantly from the actual time $t_P$ where the signal crosses the center amplitude, as shown in Figure 6.2. Therefore, use the $t_C$ time in this case, not the $t_P$ time.

![Figure 6.2: Linear Interpolation](image-url)
Performance
The cycle math formulas use a reference waveform, which is used to detect the level crossings. Depending on the defined reference waveform, cycle detection is performed either implicitly or explicitly. The reference waveform can be one of the following:

A A @CycleDetect function on a waveform, which is defined on page 291. The current formula will use the level crossing times found in the @CycleDetect function directly to determine the output cycles and their values.

B A recorded (sinusoid) signal. In this case, a @CycleDetect function will be created implicitly, with this signal as its input, in order to determine the level crossing times. The current formula will then use this @CycleDetect function, as in case (A). The disadvantage of this option is that the user cannot explicitly input a desired center level and threshold to the @CycleDetect function.

C A @CycleInterval function, which is defined on page 303. The current formula will use the level crossing times calculated in the @CycleInterval function directly to determine the output cycles and their values.

D No value. When the parameter is omitted, a @CycleDetect function applied on the input waveform will be created implicitly. The current formula will use this @CycleDetect function as in case (A).

Formula defaults
The cycle formulas may have optional parameters. The default behavior of the formulas is defined as the behavior when the optional parameters are not entered by the user.

Continuous vs. Sweep recordings
- If a signal contains a single sampling frequency (which is the simple behavior for continuous recordings), the cycle formulas process the whole length of the waveform. The user might notice a delay in the representation of a cycle formula for relatively long recordings.
- If a recorded signal is sampled at more than one frequency, the cycle formulas will operate only on each segment of the signal which has a separate sampling frequency.
- If a recorded signal contains sweeps (which are not continuous in time), the cycle formulas will operate on each sweep separately.

The output of a cycle formula has the same sampling frequency as its input, and contains a number of samples which is equal to the number of samples for the input waveform.
6.2 @CycleArea

Function
Calculates the area of every cycle detected on a reference signal.

Syntax
@CycleArea(Waveform; Cycles; ReferenceWaveform)

Parameters
- **Waveform**: Input waveform to be used for calculation of math function.
- **Cycles**: Number of cycles of the reference waveform used to calculate one output value.
- **ReferenceWaveform**: Waveform to be used as a reference to detect cycles.

Output
A waveform which has a value of zero before the first level crossing, and after the end of the last cycle.

Description
For a range of input samples \( y(k_1) \) to \( y(k_2) \), where \( k_1 \) and \( k_2 \) are the indices defined by the detected cycle, the output samples \( y'(k_1) \) to \( y'(k_2) \) will be assigned the value:

\[
\text{Area} = \left( \sum_{k=k_1}^{k_2} |y(k)| \right) \cdot \Delta x
\]

\( \Delta x = x\)-difference between two consecutive samples

The start and end indices \( (k_1 \text{ and } k_2) \) for these segments ("cycles") are calculated from the level crossing times determined by the CycleDetect algorithm, which is applied on the ReferenceWaveform.

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.
If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@Area" on page 56
6.3 @CycleCount

Function
Counts the number of cycles detected on a reference signal.

Syntax
@CycleCount(Waveform; Cycles; Level; Threshold; SuppressTime)

Parameters
- Waveform: Input waveform to be used for calculation of math function.
- Cycles: Number of cycles of the reference waveform used to calculate one output value.
- Level: Number: Y-value used as the center of oscillation of the periodic waveform.
- Threshold: Number: Y-value used to indicate a level crossing.
- SuppressTime: Minimum required duration (in seconds) of a half cycle, all half cycles with duration below this value are suppressed.

Output
At any given offset of time t in the recorded data for the input Waveform, the formula’s output will be equal to the number of cycles detected on the Waveform signal from the start of the recorded signal until time t.

Description
If the Cycles parameter is N, the output is incremented by one for each N cycles of the Waveform.

Note
For very long recordings, the output value of this formula may increase to a high number; to see the output on Perception’s TimeDisplay correctly, it might be necessary to apply a different scale on the Y-axis.

The Cycles, Level, and Threshold parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

See Also
"@Cycles" on page 74
6.4  @CycleCrestFactor

Function
Calculates the Crest Factor of every cycle detected on a reference signal. The “Crest Factor” is the ratio between the absolute maximum and the RMS of that cycle.

Syntax
@CycleCrestFactor(Waveform; Cycles; ReferenceWaveform)

Parameters
Waveform  Input waveform to be used for calculation of math function.
Cycles  Number of cycles of the reference waveform used to calculate one output value.
ReferenceWaveform  Waveform to be used as a reference to detect cycles.

Output
A waveform which has a value of zero before the first level crossing, and after the end of the last cycle.

Description

Figure 6.3: Cycle crest factor
CyleCrestFactor = \frac{\text{Abs(CycleMax)}}{\text{CycleRMS}}

N = k_2 - k_1 + 1

The start and end indices (k_1 and k_2) for these segments ("cycles") are calculated from the level crossing times determined by the CycleDetect algorithm, which is applied on the ReferenceWaveform.

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@CycleDetect" on page 291
6.5 @CycleDetect

Function
Performs level crossing detection on the input waveform.

Syntax
@CycleDetect(Waveform; Level; Threshold; HoldOffTime)

Parameters
- **Waveform**: Waveform to be used as a reference to detect cycles.
- **Level**: Number: Y-value used as the center of oscillation of the periodic waveform.
- **Threshold**: Number: Y-value used to indicate a level crossing.
- **HoldOffTime**: Minimum required duration (in seconds) of a half cycle, all half cycles with duration below this value are suppressed.

Output
A square waveform which has a positive value between two consecutive level crossing times when the respective input samples value is above the threshold; it has a negative value between two consecutive level crossing times when the respective input samples are below the threshold.

Description
Put together, the Threshold and Level parameters indicate a hysteresis band, which is an amplitude range positioned symmetrically around the Level value. The hysteresis band's upper and lower levels are used to detect cycles. Each input Waveform sample is compared against the lower and upper levels of the hysteresis band. The respective output samples are calculated based on these comparisons.

If the Level and/or Threshold parameters are omitted, they are calculated based on the input Waveform. The Threshold value is calculated as 5% of the input waveform's Display Range; the Level value is calculated as the middle of the Display Range.

The output is set to 0 for all samples before the first level crossing.

When it is known that the input waveform contains a DC offset, using this DC offset value as the detection level is recommended.
For better performance (faster calculation), it is also recommended to explicitly define the Level and Threshold parameters – so that the formula function does not need to calculate them.

**Example**

```plaintext
cycledetect = @CycleDetect(Active.Group1.Recorder_Vrms.Vrms;0;150)
```

This example creates the square waveform that shows the times of zero crossings in the Vrms signal, using a threshold of 150 V. The output is the red trace in Figure 6.4. The hysteresis band is indicated by the dashed lines, which are located at +150 V and -150 V.

Figure 6.4 shows a segment of a signal which is a sine wave with noise (blue trace). The red trace is the generated waveform of the @CycleDetect formula.

![Figure 6.4: Example of CycleInterval output](image)

**See Also**

"@CycleInterval" on page 303
6.6 @CycleEnergy

**Function**
Calculates the **energy** of every **cycle** detected on a reference signal.

**Syntax**
@CycleEnergy(Waveform; Cycles; ReferenceWaveform)

**Parameters**
- **Waveform**: Input waveform to be used for calculation of math function.
- **Cycles**: Number of cycles of the reference waveform used to calculate one output value.
- **ReferenceWaveform**: Waveform to be used as a reference to detect cycles.

**Output**
A waveform which has a value of zero before the first level crossing, and after the end of the last cycle.

**Description**
For a range of input samples \( y(k_1) \) to \( y(k_2) \), where \( k_1 \) and \( k_2 \) are the indices defined by the detected cycle, the output samples \( y'(k_1) \) to \( y'(k_2) \) will be assigned the value:

\[
\text{Energy} = \left( \sum_{k=k_1}^{k_2} (y(k))^2 \right) \cdot \Delta x
\]

\( \Delta x = x \)-difference between two consecutive samples

The start and end indices \( (k_1, k_2) \) for these segments ("cycles") are calculated from the level crossing times determined by the CycleDetect algorithm, which is applied on the ReferenceWaveform.

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.
If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@Energy" on page 82
6.7 **@CycleFrequency**

**Function**
Calculates the frequency of all the cycles detected on the reference signal.

**Syntax**
@CycleFrequency(Waveform; Cycles; ReferenceWaveform)

**Parameters**
- **Waveform**: Input waveform to be used for calculation of math function.
- **Cycles**: Number of cycles of the reference waveform used to calculate one output value.
- **ReferenceWaveform**: Waveform to be used as a reference to detect cycles.

**Output**
The frequency in Hz that is calculated on the ReferenceWaveform. The frequency is the inverse of the time difference between two level crossings that determine the duration of a sinus cycle.

**Description**
If the ReferenceWaveform and the input Waveform are the same signal, and the Cycles parameter is 1, then the output is the actual frequency of the Waveform.

The frequency is calculated with the formula:

\[
F = \frac{1}{\Delta T}
\]

where \(\Delta T\):

- Is the duration of half of a period of the Reference Waveform, when Cycles = 0.5.
- Is the duration of 1 period of the Reference Waveform, when Cycles = 1.
- Is the duration of 2 periods of the Reference Waveform, when Cycles = 2.
- etc.

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.
If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

Example

\[ \text{freq} = @\text{CycleFrequency(Group1.Recorder_A.Ch_A1; 1)} \]

The formula above omits the ReferenceWaveform parameter, so it will detect zero crossings in the Ch_A1 waveform. Then, for each full cycle detected on Ch_A1, it will calculate the frequency and assign this value to all output samples that belong to the cycle.

See Also

"@Frequency" on page 115, "@TrueFrequency" on page 250
6.8 @CycleFundamental

**Function**
Generates the Waveform of the **Cycle Fundamental** of every cycle detected on a reference signal.

**Syntax**
@CycleFundamental(Waveform; Cycles; ReferenceWaveform)

**Parameters**
- **Waveform**: Input waveform to be used for the calculation of the Fundamental waveform.
- **Cycles**: Number of cycles of the reference waveform used to calculate one output value.
- **ReferenceWaveform**: Waveform to be used as a reference to detect cycles.

**Output**
Waveform of the Fundamental Frequency.

**Description**
By performing harmonic analysis on an input signal it is possible to determine the basic waveforms, that is sines and cosines that compose the original input signal. As such any waveform $f(t)$ is represented by the Fourier series as:

\[
f(t) = A_0 + \sum_{n=1}^{\infty} (A_n \cos\left(\frac{2.\pi.n}{T} t\right) + B_n \sin\left(\frac{2.\pi.n}{T} t\right))
\]

The Fundamental frequency is the component of the above expression when $n$ takes the value of 1; that is the term:

\[
A_1 \cos\left(\frac{2.\pi}{T} t\right) + B_1 \sin\left(\frac{2.\pi}{T} t\right)
\]
Where

\[
A_1 = \frac{2}{T} \int_{t_1}^{t_2} F(t).\cos\left(\frac{2\pi}{T} t\right) \, dt
\]

\[
B_1 = \frac{2}{T} \int_{t_1}^{t_2} F(t).\sin\left(\frac{2\pi}{T} t\right) \, dt
\]

For each detected cycle, in the period \(t_1, t_2\), \(@\text{CycleFundamental}@\) will determine the values \(A_1, B_1\) so that it can generate the waveform given by the expression:

\[
f(t) = A_1\cos\left(\frac{2\pi}{T} t\right) + B_1\sin\left(\frac{2\pi}{T} t\right)
\]

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

Example
Given that the input signal is available as \(\text{Formula.input}\), the function \(@\text{CycleDetect}@\) determines the zero level crossings which are used by the \(@\text{CycleFundamental}@\) function to determine the Fundamental waveform for the detected cycle:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd</td>
<td>(@\text{CycleDetect}(\text{Formula.input}; : 0.0, : 0.1))</td>
</tr>
<tr>
<td>fund</td>
<td>(@\text{CycleFundamental}(\text{Formula.input}; : 1; : \text{Formula.cd}))</td>
</tr>
</tbody>
</table>

See Also
"@\text{CycleDetect}" on page 291
6.9  @CycleFundamentalPhase

Function
Returns a waveform representing the phase difference between the cycle fundamentals of two signals for each detected cycle in the cycle detect signal. The phase difference is in radians from 0 to 2π.

Syntax
@CycleFundamentalPhase(Waveform1; Waveform2; CycleDetect)

Parameters
Waveform1  Input waveform 1 to be used for the calculation of the cycle fundamental waveform 1.
Waveform2  Input waveform 2 to be used for the calculation of the cycle fundamental waveform 2.
Cycle detect  The waveform to be used as reference. This must be a predefined @CycleDetect or @CycleInterval formula.

Output
A waveform representing the phase difference between the cycle fundamentals of two signals for each detected cycle in the cycle detect signal. It has a value of zero before the first and after the last detected cycle.

Description
The CycleFundamentalPhase function determines the phase of the cycle fundamentals of two input signals with a cycle detect signal as reference. The phase difference \( \theta \) is defined as:

\[
\theta = \theta_1 - \theta_2 \text{ (rad)}
\]

where:
- \( \theta \) (rad) = Phase difference in radians from 0 to 2π.
- \( \theta_1 \) (rad) = Phase of the cycle fundamental of input signal 1
- \( \theta_2 \) (rad) = Phase of the cycle fundamental of input signal 2

Example
The following example calculates the fundamental phase difference between two sine waves. The sine wave Signal_2 is 90 degrees behind sine wave Signal_1. The cycle detect of Signal_1 is used as reference.
<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal_1</td>
<td><code>@SineWave(1000;1000;5; 0)</code></td>
</tr>
<tr>
<td>Signal_2</td>
<td><code>@SineWave(1000;1000;5; -90)</code></td>
</tr>
<tr>
<td>Reference</td>
<td><code>@CycleDetect(Formula.Signal_1)</code></td>
</tr>
<tr>
<td>CycleFP</td>
<td><code>@CycleFundamentalPhase(Formula.Signal_1; Formula.Signal_2; Formula.Reference)</code></td>
</tr>
</tbody>
</table>

Result:

![Graph showing the result of the CycleFundamentalPhase example](image)

**Figure 6.5**: Result of the CycleFundamentalPhase example

**See Also**

"@CycleDetect" on page 291 and "@CycleFundamental" on page 297
6.10  \texttt{@CycleFundamentalRMS}

\textbf{Function}

Returns a waveform representing the RMS value of the \texttt{fundamental} of the signal for each detected \texttt{cycle} in the \texttt{cycle} detect signal.

\textbf{Syntax}

\texttt{@CycleFundamentalRMS(Waveform; CycleDetect)}

\textbf{Parameters}

- \textit{Waveform}  \texttt{Input} waveform to be used for the calculation of the cycle fundamental RMS.
- \textit{Cycle detect}  \texttt{The} waveform to be used as reference. This must be a predefined \texttt{@CycleDetect} or \texttt{@CycleInterval} formula.

\textbf{Output}

Returns a waveform representing the RMS value of the fundamental of the signal for each detected cycle in the cycle detect signal.

\textbf{Description}

The CycleFundamentalRMS function calculates the RMS value of the fundamental of the signal for each detected cycle in the cycle detect signal. It has a value of zero before the first and after the last detected cycle.

\textbf{Example}

The following example calculates the RMS value fundamental RMS value of sine wave. The cycle detect of Signal_1 is used as reference.

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal_1</td>
<td>\texttt{@SineSquare(1000;1000;5; 0)}</td>
</tr>
<tr>
<td>Reference</td>
<td>\texttt{@CycleDetect(Formula.Signal_1)}</td>
</tr>
<tr>
<td>CycleFR</td>
<td>\texttt{@CycleFundamentalRMS(Formula.Signal_1;</td>
</tr>
<tr>
<td></td>
<td>Formula.Reference)}</td>
</tr>
</tbody>
</table>
Result:

![Diagram of CycleFundamentalRMS example]

**Figure 6.6:** Result of the CycleFundamentalRMS example

**See Also**
"@CycleDetect" on page 291 and "@CycleFundamental" on page 297
6.11  @CycleInterval

Function
Generates a square waveform with a specified interval between the high and low states.

Syntax
@CycleInterval(Waveform; Interval; StartOffset)

Settings
Waveform  Waveform to be used as a reference.
Interval   Number: time in seconds representing the half-period of the output square waveform.
StartOffset Number: time in seconds indicating a shift of the output signal on the X-axis. The default value is 0.

Output
A square waveform with 50% duty cycle. The duration between the start of a high pulse and the end of the following low pulse is equal to 2*Interval.

The output of this formula is equivalent to the output of the @SquareWave formula when the @CycleInterval formula has been defined with the appropriate parameters.

Description
The reference Waveform parameter is only used to determine the start time of the signal and to use it as the start time of the output waveform.

The StartOffset parameter is optional. If the StartOffset is defined, the start time of the output waveform is calculated by adding the StartOffset to the start time of the reference Waveform.

This formula can be used as the reference waveform parameter for the other Cycle formulas. The result is that all the cycles of the calculated math function have the same duration, i.e. they constantly operate on the same amount of samples.

Example
Assume that signal Ch_A1 is recorded at a sampling frequency of 1 MS/s, and it contains 1 MSamples. Then the formula
ci = @CycleInterval(Group1.Recorder_A.Ch_A1; 0.1)

is equivalent to the formula:

sw = @SquareWave(1000000; 1000000; 5)

Assuming that Ch_A1 starts at 0 s, this formula will indicate that the cycle start times are located at times 0.2 s, 0.4 s, 0.6 s, 0.8 s, etc.

The formula can be used to calculate a math function on the signal Ch_A1, using a constant cycle length of 2*100 = 200 ms from the start of the recording – which is equivalent to 200 kSamples ( = 1 MS/s * 0.2 s) of the input signal.

See Also
"@CycleDetect" on page 291
6.12  @CycleLevel

Function
Returns a waveform that uses the value of the input signal at the start of a cycle of the reference signal and repeats that value (sample and hold) during half, one or multiple cycles of that reference signal.

Syntax
@CycleLevel(Waveform; Cycles; ReferenceWaveform)

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveform</td>
<td>Input waveform to be used for the input samples.</td>
</tr>
<tr>
<td>Cycles</td>
<td>Number of cycles of the reference waveform to repeat the same output value.</td>
</tr>
<tr>
<td></td>
<td>Possible values: 0.5, 1, 2, 3, etc..</td>
</tr>
<tr>
<td>ReferenceWaveform</td>
<td>Waveform to be used as a reference to detect cycles.</td>
</tr>
</tbody>
</table>

Output
A waveform that uses the value of the input signal at the start of a cycle of the reference signal and repeats that value during half, one or multiple cycles of that reference signal. The waveform value is zero before the first cycle begins and after the last one ends.

Description
For a range of input samples y(k1) to y(k2), where k1 and k2 are the indices defined by the detected cycle, the output samples y'(k1) to y'(k2) will be assigned the value of y(k1).

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1. A value of 0.5 indicates half a cycle of the reference signal.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

In order to improve performance in case the same cycle reference signal is needed for multiple functions, you can generate the block cycle signal yourself using the CycleDetect function, and pass it as a reference waveform.
Example
The following example calculates an output signal where the ‘y’ value is set to a new ‘y’ value at each zero crossing of the second signal (sig2). The ‘y’ value is set to the ‘y’ value of the first signal (sig1) at the zero crossing point. The waveform value is zero before the first cycle begins and after the last one ends.

\[
\begin{align*}
\text{sig1} &= \text{@SineWave}(1k;1k+1;3;90) \\
\text{sig2} &= \text{@SineWave}(1k;1k+1;12) \\
\text{cd} &= \text{@CycleDetect}(\text{Formula.sig2};0;0.1) \\
\text{cl} &= \text{@CycleLevel}(\text{Formula.sig1};1;\text{Formula.cd})
\end{align*}
\]

Result:

Figure 6.7: Result of CycleLevel example

See Also
"@CycleMax" on page 307, "@CycleMean" on page 308 and "@CycleMin" on page 310.
**6.13  @CycleMax**

**Function**
Calculates the maximum value in every cycle detected on a reference signal.

**Syntax**
@CycleMax(Waveform; Cycles; ReferenceWaveform)

**Parameters**
- **Waveform**
  Input waveform to be used for calculation of math function.
- **Cycles**
  Number of cycles of the reference waveform used to calculate one output value.
- **ReferenceWaveform**
  Waveform to be used as a reference to detect cycles.

**Output**
A waveform which contains the maximum value of every detected cycle. The waveform value is zero before the first cycle begins and after the last one ends.

**Description**
For a range of input samples y(k₁) to y(k₂), where k₁ and k₂ are the indices defined by the detected cycle, if sample y(k) is the maximum (where k₁≤k≤k₂), the output samples y'(k₁) to y'(k₂) will be assigned the value of y(k).

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

**See Also**
"@Max" on page 142
6.14  @CycleMean

**Function**
Calculates the mean value in every cycle detected on a reference signal.

**Syntax**
@CycleMean(Waveform; Cycles; ReferenceWaveform)

**Parameters**
- **Waveform**: Input waveform to be used for calculation of math function.
- **Cycles**: Number of cycles of the reference waveform used to calculate one output value.
- **ReferenceWaveform**: Waveform to be used as a reference to detect cycles.

**Output**
A waveform which contains the mean value of every detected cycle. The waveform value is zero before the first cycle begins and after the last one ends.

**Description**
The formula performs the following operation:

For a range of input samples \( y(k_1) \) to \( y(k_2) \), where \( k_1 \) and \( k_2 \) are the indices defined by the detected cycle, the output samples \( y'(k_1) \) to \( y'(k_2) \) will be assigned the value:

\[
\text{Mean} = \frac{1}{N} \sum_{k = k_1}^{k_2} y(k)
\]

\[
N = k_2 - k_1 + 1
\]

The start and end indices \((k_1, k_2)\) for these segments ("cycles") are calculated from the level crossing times determined by the CycleDetect algorithm, which is applied on the ReferenceWaveform.

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.
If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@Mean" on page 147
6.15  @CycleMin

Function
Calculates the minimum value in every cycle detected on a reference signal.

Syntax
@CycleMin(Waveform; Cycles; ReferenceWaveform)

Parameters
Waveform  Input waveform to be used for calculation of math function.
Cycles    Number of cycles of the reference waveform used to calculate one output value.
ReferenceWaveform  Waveform to be used as a reference to detect cycles.

Output
A waveform which contains the minimum value of every detected cycle. The waveform value is zero before the first cycle begins and after the last one ends.

Description
For a range of input samples \( y(k_1) \) to \( y(k_2) \), where \( k_1 \) and \( k_2 \) are the indices defined by the detected cycle, if sample \( y(k) \) is the minimum (where \( k_1 \leq k \leq k_2 \)), the output samples \( y'(k_1) \) to \( y'(k_2) \) will be assigned the value of \( y(k) \).

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@Min" on page 152
6.16 @CyclePeriod

Function
Calculates the period of every cycle detected on a reference signal.

Syntax
@CyclePeriod(Waveform; Cycles; ReferenceWaveform)

Parameters

- **Waveform**
  Input waveform to be used for calculation of math function.

- **Cycles**
  Number of cycles of the reference waveform used to calculate one output value.

- **ReferenceWaveform**
  Waveform to be used as a reference to detect cycles.

Output
A waveform which contains the period of every detected cycle. The waveform value is zero before the first cycle begins and after the last one ends.

Description
This function is the inverse of the @CycleFrequency function. If the parameters given for the two functions are identical and the output of @CycleFrequency is F at time t, the output of @CyclePeriod at time t will then be T = 1/F.

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@Period" on page 166,"@CycleFrequency" on page 295
@CyclePhase

Function
Calculates the phase difference $\theta$ between two waveforms based on the rising edge level crossings of the two waveforms.

Syntax
@CyclePhase(InputWaveform; ReferenceWaveform)

Parameters
- \textit{InputWaveform} The waveform to be used to determine the time $t_\theta$. This can be an analog or digital channel or a predefined @CycleDetect formula.
- \textit{ReferenceWaveform} The waveform to be used as reference for the calculation of level crossing times and cycle period $T_\theta$. This can be an analog or digital channel or a predefined @CycleDetect or @CycleInterval formula.

Output
A waveform containing the phase difference for each detected cycle of the reference waveform. It has a value of zero before the first and after the last rising edge level crossing of the reference waveform. If for a cycle no rising edge level crossing is found for the input waveform the phase difference is undefined for this cycle.

Description
The CyclePhase function calculates the phase difference $\theta$ between two waveforms based on the rising edge level crossings of the two waveforms.

The phase difference $\theta$ is defined as:

$$\theta = 2 \pi \cdot \frac{t_\theta}{T_\theta} \text{ (rad)}$$

where:
- $T_\theta$ (s) = Cycle period of the reference signal.
- $t_\theta$ (s) = Time between a rising edge level crossing of the reference signal and the first following rising edge level crossing of the input signal which lies in $T_\theta$.

For faster results, use manually created @CycleDetect or @CycleInterval formulas for the signal and the reference signal and use these as input signals for the CyclePhase function.
Example
The following example calculates the phase difference between two sine waves. The sine wave Signal is 90 degrees behind sine wave Reference.

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>@SineWave(1000;1000;5; -90)</td>
</tr>
<tr>
<td>Reference</td>
<td>@SineWave(1000;1000;5)</td>
</tr>
<tr>
<td>CyclePhase</td>
<td>@CyclePhase(Formula.Signal; Formula.Reference)</td>
</tr>
</tbody>
</table>

Result:

![Figure 6.8: Result of the CyclePhase example](image)

**See Also**
"@CycleDetect" on page 291
6.18  @CycleRPM

Function
Calculates the revolutions per minute (RPM) of every cycle detected on a reference signal.

Syntax
@CycleRPM(Waveform; Cycles; ReferenceWaveform)

Parameters
Waveform  Input waveform representing angular position in degrees.
Cycles    Number of cycles of the reference waveform used to calculate one output value.
ReferenceWaveform  Waveform to be used as a reference to detect cycles.

Output
Waveform representing the RPM value for each detected cycle of the source channel.

Description
The revolutions per minute (RPM) is calculated by determining the amount of angular rotation of the waveform that represents the angular position over the time periods that are defined by the detected cycles.

The Cycles and ReferenceWaveform parameters are optional.
If the Cycles parameter is omitted, the default value is 1.
If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

Example
Given that the input signal is available as Formula.input, the function @CycleDetect determines the zero level crossings which are used by the @CycleRPM function to determine the RPM value for each detected cycle:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd</td>
<td>@CycleDetect(Formula.input; 0.0, 0.1)</td>
</tr>
<tr>
<td>rpm</td>
<td>@CycleRPM(Formula.input; 1; Formula.cd)</td>
</tr>
</tbody>
</table>

See Also
"@CycleDetect" on page 291
6.19  @CycleRMS

**Function**
Calculates the root mean square value of every cycle detected on a reference signal.

**Syntax**
@CycleRMS(Waveform; Cycles; ReferenceWaveform)

**Parameters**
- **Waveform**: Input waveform to be used for calculation of math function.
- **Cycles**: Number of cycles of the reference waveform used to calculate one output value.
- **ReferenceWaveform**: Waveform to be used as a reference to detect cycles.

**Output**
A waveform which contains the RMS value of every detected cycle. The waveform value is zero before the first cycle begins and after the last one ends.

**Description**
For a range of input samples \( y(k_1) \) to \( y(k_2) \), where \( k_1 \) and \( k_2 \) are the indices defined by the detected cycle, the output samples \( y'(k_1) \) to \( y'(k_2) \) will be assigned the value:

\[
RMS = \sqrt{\frac{1}{N} \sum_{k = k_1}^{k_2} (y(k))^2}
\]

\[N = k_2 - k_1 + 1\]

The start and end indices (\( k_1 \) and \( k_2 \)) for these segments (“cycles”) are calculated from the level crossing times determined by the CycleDetect algorithm, which is applied on the ReferenceWaveform.

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.
If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@RMS" on page 198
6.20 \texttt{@CycleStdDev}

\textbf{Function}  
Calculates the \textbf{standard deviation} of every \textbf{cycle} detected on a reference signal.

\textbf{Syntax}  
@CycleStdDev(Waveform; Cycles; ReferenceWaveform)

\textbf{Parameters}  
- \textit{Waveform} \hspace{1cm} Input waveform to be used for calculation of math function.
- \textit{Cycles} \hspace{1cm} Number of cycles of the reference waveform used to calculate one output value.
- \textit{ReferenceWaveform} \hspace{1cm} Waveform to be used as a reference to detect cycles.

\textbf{Output}  
A waveform which contains the standard deviation from the mean value of every detected cycle. The waveform value is zero before the first cycle begins and after the last one ends.

\textbf{Description}  
For a range of input samples $y(k_1)$ to $y(k_2)$, where $k_1$ and $k_2$ are the indices defined by the detected cycle, the output samples $y'(k_1)$ to $y'(k_2)$ will be assigned the value:

$$\text{StdDev} = \sqrt{\frac{1}{N} \sum_{k=k_1}^{k_2} (y(k) - \bar{y})^2}$$

$$\bar{y} = \frac{1}{N} \sum_{k=k_1}^{k_2} y(k)$$

$$N = k_2 - k_1 + 1$$

The start and end indices ($k_1$ and $k_2$) for these segments ("cycles") are calculated from the level crossing times determined by the CycleDetect algorithm, which is applied on the ReferenceWaveform.
The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

See Also
"@StdDev" on page 213
6.21 @CycleTHD

Function
Calculates the Total Harmonic Distortion (THD) of every cycle detected on a reference signal.

Syntax
@CycleTHD(Waveform; Cycles; ReferenceWaveform)

Parameters
Waveform Input waveform to be used for the calculation of the Cycle THD.
Cycles Number of cycles of the reference waveform used to calculate one output value.
ReferenceWaveform Waveform to be used as a reference to detect cycles.

Output
Waveform that contains the THD value of each detected cycle as a percentage. The waveform value is zero before the first and after the last detected cycles.

Description
The total harmonic distortion (THD) is a measurement of the harmonic distortion that is present in a signal. The measurement can be applied both signals that represent voltage as well as current levels.

The THD for voltage is defined as:

\[ THD_v = \sqrt{V_{rms}^2 - V_1^2} / V_1 \]

Where
\[ V_{rms} = \text{Root Mean Square Voltage} \]
\[ V_1 = \text{Root Mean Square Voltage of the 1st Harmonic (Fundamental frequency)} \]

The THD for current is defined as:

\[ THD_i = \sqrt{I_{rms}^2 - I_1^2} / I_1 \]
Where

\[ I_{rms} = \text{Root Mean Square Current} \]
\[ I_1 = \text{Root Mean Square Current of the 1st Harmonic (Fundamental frequency)} \]

The Cycles and ReferenceWaveform parameters are optional.

If the Cycles parameter is omitted, the default value is 1.

If the ReferenceWaveform parameter is omitted, a CycleDetect formula is created for the input Waveform.

Example
Given that the input signal is available as \textit{Formula.input}, the function \texttt{@CycleDetect} determines the zero level crossings which are used by the \texttt{@CycleTHD} function to determine the THD value for each detected cycle:

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{cd}</td>
<td>@CycleDetect(Formula.input; 0.0, 0.1)</td>
</tr>
<tr>
<td>\texttt{thd}</td>
<td>@CycleTHD(Formula.input; 1; Formula.cd)</td>
</tr>
</tbody>
</table>

See Also
"\texttt{@CycleDetect}" on page 291 and "\texttt{@CycleFundamental}" on page 297.
7 Pulse Measurement and Analysis

7.1 General

IEEE Standard Pulse Terms and Definitions
The contents of this section are based on the IEEE Std 194-1977 and IEEE Std 181-1977.

For more information visit the website of the IEEE Standards Association: www.standards.ieee.org/

Wave, Pulse and Transition

- **Wave**: A modification of the physical state of a medium which propagates in the medium as a function of time as a result of one or more disturbances.
- **Pulse**: A wave which departs from a first nominal state, attains a second nominal state, and ultimately returns to the first nominal state. Throughout the remainder of this document the term pulse is included in the term wave.
- **Transition**: A portion of a wave or pulse between a first nominal state. Throughout the remainder of this document the term transition is included in the terms pulse and wave.

Refer to the following diagram for an overview of the single pulse waveform.
The Single Pulse

Figure 7.1: Diagram of the single pulse waveform

1 Time origin line
   A line of constant and specified time which, unless otherwise specified, has a time equal to zero and passes through the first datum time \(t_0\) of a waveform epoch.

2 Pulse start line
   Pulse Start Line. The time reference line at pulse start time.

3 Top center point
   A specified time referenced point or magnitude referenced point on a pulse waveform top. If no point is specified, the top center point is time referenced point at the intersection of a pulse waveform and the top center line.

4 Pulse stop line
   Pulse Stop Line. The time reference line at pulse stop time.

5 Top magnitude
   The magnitude of the top as obtained by a specified procedure or algorithm.

6 Pulse amplitude
   The algebraic difference between the top magnitude and the base magnitude.
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Proximal (First transition point)</td>
</tr>
<tr>
<td>8</td>
<td>Mesial (First transition point)</td>
</tr>
<tr>
<td>9</td>
<td>Distal (First transition point)</td>
</tr>
<tr>
<td>10</td>
<td>Top center line</td>
</tr>
<tr>
<td>11</td>
<td>Distal (Last transition point)</td>
</tr>
<tr>
<td>12</td>
<td>Mesial (Last transition point)</td>
</tr>
<tr>
<td>13</td>
<td>Proximal (Last transition point)</td>
</tr>
<tr>
<td>14</td>
<td>Top line</td>
</tr>
<tr>
<td>15</td>
<td>Distal line</td>
</tr>
<tr>
<td>16</td>
<td>Mesial line</td>
</tr>
<tr>
<td>17</td>
<td>Proximal line</td>
</tr>
<tr>
<td>18</td>
<td>Baseline</td>
</tr>
<tr>
<td>19</td>
<td>First base point</td>
</tr>
<tr>
<td>20</td>
<td>Last base point</td>
</tr>
<tr>
<td>21</td>
<td>Base magnitude</td>
</tr>
<tr>
<td>22</td>
<td>First transit duration</td>
</tr>
</tbody>
</table>

A magnitude referenced point at the intersection of a waveform and a proximal line.

A magnitude referenced point at the intersection of a waveform and a mesial line.

A magnitude referenced point at the intersection of a waveform and a distal line.

The time reference line at the average of pulse start and pulse stop time.

A magnitude referenced point at the intersection of a waveform and a distal line.

A magnitude referenced point at the intersection of a waveform and a mesial line.

A magnitude referenced point at the intersection of a waveform and a proximal line.

The magnitude reference line at the top magnitude.

A magnitude reference line at a specified magnitude in the distal region of a pulse waveform. Unless otherwise specified, the distal line is at the 90 percent reference magnitude.

A magnitude reference line at a specified magnitude in the mesial region of a pulse waveform. Unless otherwise specified, the mesial line is at the 50 percent reference magnitude.

A magnitude reference line at a specified magnitude in the proximal region of a pulse waveform. Unless otherwise specified, the proximal line is at the 10 percent reference magnitude.

The magnitude reference line at the base magnitude.

Unless otherwise specified, the first datum point in a pulse epoch.

Unless otherwise specified, the last datum point in a pulse epoch.

The magnitude of the base as obtained by a specified procedure or algorithm. Unless otherwise specified, both portions of the base are included in the procedure or algorithm.

The transit duration of the first transition waveform in a pulse waveform.
<table>
<thead>
<tr>
<th>No.</th>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Last transit duration</td>
<td>The transit duration of the last transition waveform in a pulse waveform.</td>
</tr>
<tr>
<td>24</td>
<td>Pulse start time</td>
<td>The instant specified by a magnitude referenced point on the first transition of a pulse waveform. Unless otherwise specified, the pulse start time is at the mesial point on the first transition.</td>
</tr>
<tr>
<td>25</td>
<td>Pulse duration</td>
<td>The duration between pulse start time and pulse stop time.</td>
</tr>
<tr>
<td>26</td>
<td>Offset</td>
<td>The algebraic difference between two specified magnitude reference lines. Unless otherwise specified, the two magnitude reference lines are the waveform baseline and the magnitude origin line.</td>
</tr>
<tr>
<td>27</td>
<td>Pulse stop time</td>
<td>The instant specified by a magnitude referenced point on the last transition of a pulse waveform. Unless otherwise specified, the pulse stop time is at the mesial point on the last transition.</td>
</tr>
<tr>
<td>28</td>
<td>Magnitude origin line</td>
<td>A line of specified magnitude which, unless otherwise specified, has a magnitude equal to zero and extends through the waveform epoch.</td>
</tr>
<tr>
<td>29</td>
<td>Pulse waveform epoch</td>
<td>The span of time for which waveform data are known or knowable. A waveform epoch manifested by equations may extend in time from – infinity to + infinity or, like all waveform data, may extend from a first datum time to a second datum time.</td>
</tr>
</tbody>
</table>
8 IIR Filters

8.1 Introduction

The Perception formula database contains a series of IIR filters. These filters implement a class that filters the input sequence using the direct form IIR filter specified by the feedback and feedforward coefficients. IIR is an abbreviation of Infinite Impulse Response. The relation between the output and input signal looks like the following formula Figure 8.1.

\[
y(n) = \sum_{k=0}^{N} b_k x(n - k) - \sum_{k=1}^{N} a_k y(n - k)
\]

**Figure 8.1: Infinite Impulse Response – output, input**

where:

- \( N \) = Filter order
- \( b_k \) = Feedforward filter coefficients, often also called the zeros
- \( a_k \) = Feedback filter coefficients, often also called the poles
- \( x(n) \) = Input signal
- \( y(n) \) = Output signal

The block diagram looks like figure Figure 8.2:
All the filter formulas start with "@Filter". The complete list of filter functions can be find below.

- @FilterButterworthLP
- @FilterButterworthHP
- @FilterButterworthBP
- @FilterButterworthBS
- @FilterBesselLP
- @FilterBesselHP
- @FilterBesselBP
- @FilterBesselBS
- @FilterChebyshevLP
- @FilterChebyshevHP
- @FilterChebyshevBP
- @FilterChebyshevBS

Three different types of filters are available:

- Bessel
- Butterworth
- Chebyshev

For each type there is a Lowpass (LP), Highpass (HP), Bandpass (BP) and Bandstop (BS) implementation.

### 8.1.1 Bessel

A Bessel filter is a type of linear filter with a maximally flat group delay (maximally linear phase response). Bessel filter has a good step response with very little overshoot; however the frequency response is worse than that of Butterworth filter. Bessel filters have near linear phase response. The applications of Bessel are where the phase relations are critical.

This filter is best used in the time domain.

It is also recommended to use a Bessel filter when the signal to be filtered contains non sinusoidal signals like for example square waveforms or noise and you want to use the filtered signal for timing purposes. Bessel is then the best choice because all of the different frequency components of the original signal will have almost the same delay after they have been filtered.

**Advantages**

Best step response-very little overshoot or ringing.
Disadvantages
Slower initial rate of attenuation beyond the pass-band than Butterworth.

8.1.2 Butterworth
The Butterworth filter is a type of signal processing filter designed to have as flat a frequency response as possible in the passband so that it is also termed a maximally flat magnitude filter. It has a good frequency response with no ripple; however the phase response may be quite nonlinear especially for high order filters.

This filter is best used in the frequency domain.

If the signal to be filtered is a sinusoidal signal than Butterworth is often the best choice, the nonlinear phase response doesn’t matter too much because we are dealing with only one dominant frequency in the input signal, therefore the deformation of the output signal will be minimal.

Advantages
- Maximally flat magnitude response in the pass-band.
- Good all-around performance.
- Pulse response better than Chebyshev.
- Rate of attenuation better than Bessel.

Disadvantages
- Some overshoot and ringing in step response.

8.1.3 Chebyshev (Type I)
Chebyshev filters have a steeper roll-off and more passband ripple (type I) than Butterworth filters. Chebyshev filters have the property that they minimize the error between the idealized and the actual filter characteristic over the range of the filter, but with ripples in the passband.

Compared to a Butterworth filter, a Chebyshev filter can achieve a sharper transition between the passband and the stopband with a lower order filter.

This filter is best used in the frequency domain.

Advantages:
- Better rate of attenuation beyond the pass-band than Butterworth.

Disadvantages:
- Ripple in pass-band.
- Considerably more ringing in step response than Butterworth.
8.1.4 Magnitude Spectrum
The following Figure 8.3 shows the magnitude spectrums of the three different 2nd order lowpass IIR filters. The Bessel filter has the worsted frequency response in the stop band. The Chebyshev has the steepest roll-off, but has a ripple in the passband.

Figure 8.3: Magnitude spectrum of lowpass filters
Figure 8.4 shows the passband of the three different filters.

Figure 8.4: Magnitude Spectrum Passband of filters
Figure 8.5 shows the stopband of the three filters.

Figure 8.5: Magnitude Spectrum – stopband
8.1.5 Impulse Response

The following Figure 8.6 shows the Impulse response of the three different 2\textsuperscript{nd} order lowpass IIR filters. The Bessel filter provides the best impulse response.

![Impulse response of lowpass filters](image)

**Figure 8.6:** Impulse response of lowpass filters
8.1.6 Step Response

Figure 8.7 below shows the step response of the three different 2nd order lowpass IIR filters.

![Step response of lowpass filters](image)

**Figure 8.7:** Step response of lowpass filters

Again the Bessel filter gives the best step response. The above impulse and step responses can be reconstructed by using the following formulas:

![Filter response table](image)

**Figure 8.8:** Filter response
The above formulas can also be used to examine the step and impulse responses of any filter you can create with the current filter formulas of Perception.

8.1.7 Phaseless Filtering
Phaseless filtering is done by using the reverse order technique. This technique filters the signal twice. The order of the used filter is half of the entered order value \((\text{Order}/2)\). During the first step the signal will be filtered as usual, during the second step the filtered output will be filtered again using the same filter but the data points will go through the filter in reverse order. Because we use the filter twice the overall filter order will be equal to the entered order value \((\text{Order})\). If the entered order is an odd value than the filter order will be the first even value below this odd value. If for example the entered order value is 7 then the filter will have an order value of 6. The phase shift of the filtered signal will be zero, see Figure 8.9.

![Phase Response](image)

**Figure 8.9: Phase response of filters**

There will be a difference between the magnitude spectrums of the phaseless and not phaseless filters. This difference is because the phaseless filter is the result of filtering with a filter with half the order twice, therefore the attenuation at the cutoff frequency is not -3 dB but -6 dB, see Figure 8.10.
8.1.8 Importance of sampling rate and cutoff frequency

It is important to know the influence of the sampling rate of a discrete signal to be filtered and the filter cutoff frequency. The sampling rate must be at least twice the cutoff frequency, this is also known as the Nyquist frequency.

The filter magnitude curve will also be influenced by the sampling frequency. A second order filter has a roll-off of 12 dB/Octave or 40 dB/Decade, however for frequencies near half the sampling rate the filter behaves different, see Figure 8.11. You should be aware of this difference if you want to compare digital filtering with analog filtering. Analog filters roll off with a constant slope for the complete stopband.
**Figure 8.11:** Frequencies at half sample rate
Index

A
Abs ................................................................. 53
ACosine .......................................................... 54
Analysis .......................................................... 11
  Constants ..................................................... 16
  Formula sheet .............................................. 13
  Functions .................................................. 17
  Load Formulas ........................................... 30
  Modifying layout ....................................... 18
  Operators .................................................. 13
  Print Formulas ........................................... 31
  Save Formulas .......................................... 31
  Tools .......................................................... 13
  Variables .................................................. 16
And .................................................................. 55
Area .................................................................. 56
Arithmetic Operations
  Addition ....................................................... 43
  Division ....................................................... 49
  Modulo ........................................................ 52
  Multiplication .......................................... 47
  Subtraction ............................................... 45
  Unary minus .............................................. 51
  ASine ........................................................... 58
  ATan ............................................................. 59
  ATan2 ........................................................... 60

B
BlockFFT .......................................................... 63

C
Clip .................................................................... 65
Comparator ...................................................... 67
Cos ..................................................................... 69
Creating formulas .......................................... 20
  Entering comment .................................... 25
  Entering formulas ...................................... 26
  Function Creator ....................................... 20
  Problems in formulas ............................... 27
  CurveFitting .............................................. 70
  Cut ................................................................. 72

Cycle Math
  Continuous vs. Sweep recordings ............... 285
  Formula defaults ........................................ 285
  Linear Interpolation ................................... 284
  Performance .............................................. 285
Cycle Math Formulas
  CycleArea .................................................... 286
  CycleCount .................................................. 288
  CycleCrestFactor ....................................... 289
  CycleDetect ............................................... 291
  CycleEnergy .............................................. 293
  CycleEnergy .............................................. 295
  CycleFundamental ...................................... 297
  CycleFundamentalPhase ......................... 299
  CycleFundamentalRMS ............................ 301
  CycleInterval ............................................ 303
  CycleLevel .................................................. 305
  CycleMax ................................................... 307
  CycleMean .................................................. 308
  CycleMin .................................................... 310
  CyclePeriod .............................................. 311
  CycleRMS ................................................... 315
  CycleRPM ................................................... 314
  CycleStdDev .............................................. 317
  CycleTHD .................................................... 319
Cycles ............................................................. 74

D
Diff .................................................................... 76
DQ0Transformation ...................................... 78

E
Energy ................................................................ 82
EqualTo ............................................................. 84
Exception ..................................................... 33
Exp ................................................................. 85
ExpWave ........................................................ 86

F
FallTime ........................................................... 87
FilterBesselBP ............................................. 101
| N | NextHillPos ............................................................. 157 |
| N | NextLvlCross ............................................................ 159 |
| N | NextValleyPos .......................................................... 161 |
| N | Noise ................................................................ 163 |
| N | Not .................................................................... 164 |

| O | Or ...................................................................... 165 |
| O | Overview ............................................................. 34 |

| P | Period ............................................................... 166 |
| P | Pow .................................................................. 168 |
| P | PrevHillPos ....................................................... 169 |
| P | PrevLvlCross .................................................... 171 |
| P | PrevValleyPos .................................................. 173 |
| P | Print Formulas ...................................................... 31 |
| P | Pulse .............................................................. 175 |
| P | PulseWidth ......................................................... 177 |

| R | Ramp ................................................................. 179 |
| R | ReadAsciiFile ..................................................... 181 |
| R | ReadLogFile ....................................................... 183 |
| R | Reduce ........................................................... 187 |
| R | RefCheck .......................................................... 188 |
| R | RelativeTime2Local ............................................. 190 |
| R | RelativeTime2UTC ............................................... 192 |
| R | RemoveGlitch .................................................... 194 |
| R | Res2 ............................................................... 195 |
| R | RiseTime .......................................................... 196 |
| R | RMS .................................................................. 198 |

| S | SAEJ211 ............................................................. 199 |
| S | Sin .................................................................. 200 |
| S | SineWave .......................................................... 201 |
| S | Smooth ............................................................ 202 |
| S | SpaceVectorInverseTransformation ......................... 204 |
| S | SpaceVectorTransformation .................................. 208 |

| Sqrt | ................................................................. 211 |
| Sqrt | ................................................................. 211 |
| SquareWave | .............................................................. 212 |
| StdDev | .............................................................. 213 |
| STLX_SignalEnd | .......................................................... 217 |
| STLX_SignalStart | ......................................................... 215 |
| Sweep | .............................................................. 218 |
| SweptSineWave | ......................................................... 219 |

| T | Tan .................................................................. 221 |
| T | The Single Pulse .............................................. 322 |
| T | TimeMaxAbove .................................................. 222 |
| T | TimeMaxAboveBegin .......................................... 224 |
| T | TimeMaxBelow .................................................. 226 |
| T | TimeMaxBelowBegin .......................................... 228 |
| T | TimeMinAbove .................................................. 230 |
| T | TimeMinAboveBegin .......................................... 232 |
| T | TimeMinBelow .................................................. 234 |
| T | TimeMinBelowBegin .......................................... 236 |
| T | TimeTotalAbove ................................................ 238 |
| T | TimeTotalAboveBegin ........................................ 240 |
| T | TimeTotalBelow ................................................ 242 |
| T | TimeTotalBelowBegin ........................................ 244 |
| T | TriggerTime ...................................................... 246 |
| T | TriggerTimeToText ............................................. 248 |
| T | TrueFrequency .................................................. 250 |
| T | TrueRMS .......................................................... 252 |
| T | TrueRMSRef ...................................................... 253 |

| U | Update the key information ................................ 11 |

| V | Value ................................................................ 255 |

| W | Warranty ............................................................ 3 |
| W | Wave, Pulse and Transition .................................. 321 |

| X | XDelta ............................................................. 256 |
| X | XDeltaHigh ........................................................ 257 |
Perception Analysis

XDeltaLow ........................................................ 258
XFirst ................................................................. 259
XLast ................................................................. 260
XShift ................................................................. 261
XYArray ............................................................ 262

Y

YArray ............................................................... 264