

BECKHOFF New Automation Technology

Manual | EN

TE3510

TwinCAT 3 | Analytics Vision

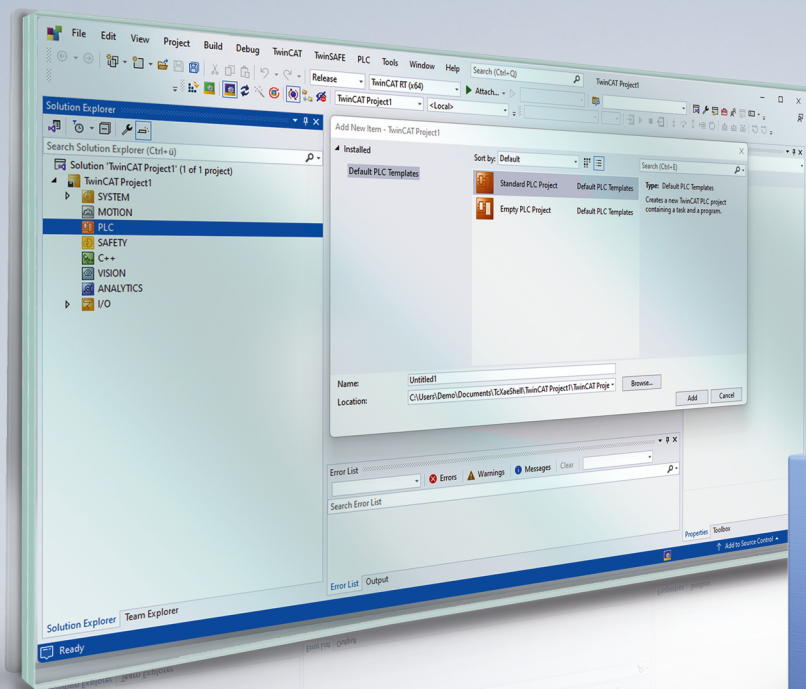


Table of contents

1 Foreword	5
1.1 Notes on the documentation	5
1.2 For your safety	6
1.3 Notes on information security	7
2 Overview	8
3 Installation	9
3.1 System requirements	9
3.2 Installation	9
4 Technical introduction	10
4.1 Basic concept	10
5 Configuration	12
5.1 Editor Basics	12
5.1.1 Networks	12
5.1.2 Smart Link	18
5.1.3 Enable Disable	22
5.1.4 Static Values	24
5.1.5 Global parameters	25
5.1.6 Additional analysis windows	26
5.2 Algorithms	35
5.2.1 Analytics algorithms	37
5.2.2 C++ lambda functions	240
5.2.3 Vision	270
5.3 Interaction with Scope	293
5.3.1 Scope configuration in the network template	298
5.4 Working with Historical Data	301
5.5 Runtime deployment	303
5.5.1 Algorithm properties	310
5.5.2 PLC Code	312
5.6 HMI One-Click Dashboard	369
5.6.1 Manage dashboard structure and content in Analytics project	372
5.6.2 Use customized and own controls	382
5.6.3 Configure user management and access rights	401
5.6.4 Dashboard Configuration	404
5.6.5 Modifying a generated dashboard in HMI engineering	411
5.6.6 Configuration of the dashboard at runtime in the web browser	411
5.6.7 Creating charts in the web browser	424
5.6.8 Switching multiple machines in the HMI Dashboard	448
5.6.9 Integration of a language switch	454
5.6.10 Using the interactive functions in standard TwinCAT HMI projects	457
5.7 Reporting Integration	459
5.7.1 Basic concept	459
5.7.2 Analytics Reporting	460
5.7.3 TwinCAT Reporting Server	468

5.8 Comparison of analysis results 478

5.8.1 Loading and saving project configurations 480

6 Appendix 482

6.1 FAQ - Frequently asked questions and answers 482

7 Support and Service 483

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.

The documentation and the following notes and explanations must be complied with when installing and commissioning the components.

The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without notice.

Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.



EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: <https://www.beckhoff.com/trademarks>.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.

Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

DANGER

Hazard with high risk of death or serious injury.

WARNING

Hazard with medium risk of death or serious injury.

CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product



This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

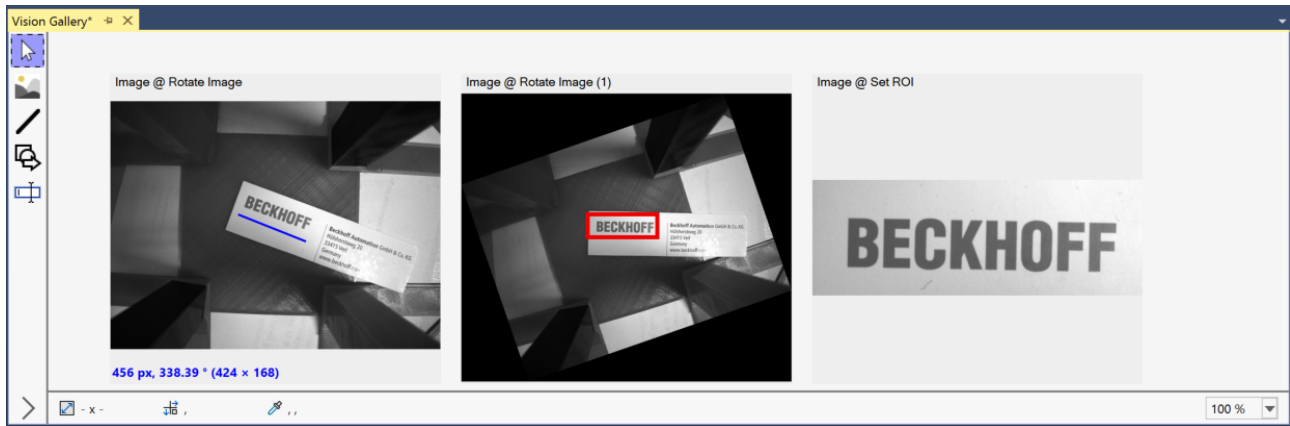
The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our <https://www.beckhoff.com/secguide>.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at <https://www.beckhoff.com/secinfo>.

2 Overview



TE3510 Analytics Vision is an engineering product for simplified commissioning of TwinCAT Vision applications and is suitable for testing initial vision concepts.

In this Analytics Engineering package, which can be combined with other packages from the TE351x product series, the focus is on the vision algorithms from TF7xxx. In addition, all Analytics Base algorithms from TF3510 can also be used. You can import image data via ADS, Analytics Logger (TF3500) as an MQTT stream or Analytics file, as well as directly from a folder on your hard disk into the analysis interface to perform an image analysis.

There are additional windows such as the Module Watcher, which quickly visualizes input and output images of each function block, or the Vision Gallery, in which many tools can be used to determine regions of interest or angles.

3 Installation

3.1 System requirements

The following system requirements must be met for TwinCAT Analytics Engineering packages of the TE351x product series to function properly.

Supported operating systems

Windows 10, Windows 11

TwinCAT

TwinCAT 4026 builds are supported. All necessary components are automatically installed via the TwinCAT Package Manager.

Development environment

If the development environment is not already installed, it is installed by the product packages. We recommend the use of TwinCAT XAE Shell 64.

3.2 Installation

Depending on the TwinCAT version and operating system used, this TwinCAT 3 Function can be installed in different ways, which are described in more detail below.



Update installation

An update installation always uninstalls the previous installation.

- Create a backup of your configuration files before the update.

TwinCAT 3 Package Manager: Installation (TwinCAT 3.1 Build 4026)

If you are using TwinCAT 3.1 Build 4026 (and higher) on the Microsoft Windows operating system, install this function via the TwinCAT Package Manager (TcPkg). Detailed instructions on installing products can be found in the chapter [Managing TwinCAT Software](#) in the [TwinCAT 3.1 Build 4026 installation instructions](#).

NOTICE

Unprepared TwinCAT restart can cause data loss

The installation of this function may result in a TwinCAT restart.

- Make sure that no critical TwinCAT applications are running on the system or shut them down in an orderly manner first.

TwinCAT Package Manager UI

Install the following workload via the TcPkg User Interface (UI) to use the product:

- **TE3510 | TwinCAT 3 Analytics Vision**

TwinCAT Package Manager CLI

You can use the [TcPkg Command Line Interface \(CLI\)](#) to display the available workloads on the system:

```
tcpkg list -t workload
```

Use the following command to install the Workload:

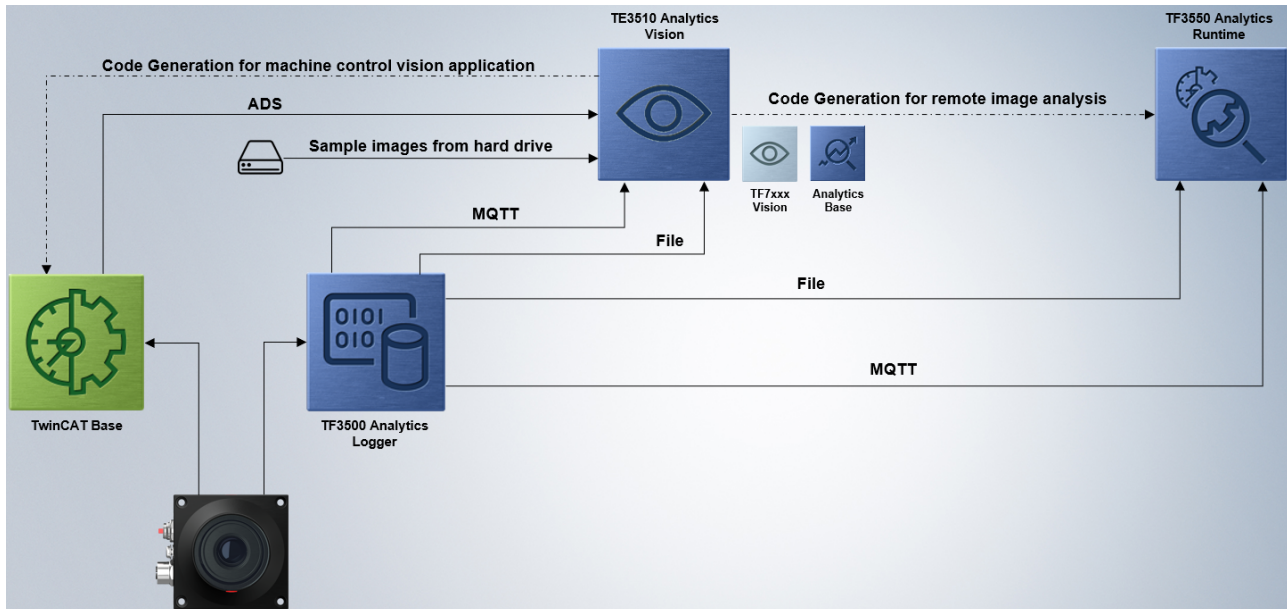
```
tcpkg install TE3510.AnalyticsVision.XAE
```

4 Technical introduction

In the technical introduction to TE3510 Analytics Vision, you will find various basic topics that are important for understanding and using the product.

4.1 Basic concept

The following image shows the basic concept of TE3510 Analytics Vision. It shows the product embedded in a workflow that you can use as you wish. The other products are optionally linked and can be used.



The TE3510 Analytics Vision is an engineering product of the TwinCAT Analytics product family. It has been specially developed for simple image data analysis.

Image data sources

You can import image data from different sources in Engineering.

- **ADS** - Images can be selected via PLC symbols or directly via VnTask in the Target Browser.
- **Analytics Logger** - Image data can be sent via so-called image streams via MQTT or written directly locally to an Analytics file.
- **Hard disk** - Image data stored on the hard disk can be transferred to the analysis software via a File Source. The following formats are supported: jpg, png, jpeg and tiff.

Algorithms

The image data can be interconnected with a variety of vision algorithms in engineering and evaluated step by step. The algorithms provided are all from the TwinCAT Vision products of the TF7xxx family. The algorithms cover a wide range from filtering, measurement technology and code detection to drawing, algebraic functions and OCR with machine learning.

Optimized engineering

Two essential tool windows make it easier to work with image data. One is the Module Watcher, which quickly visualizes the input and output image for an individual algorithm. And secondly, the Vision Gallery, in which you can view several images from different algorithms in parallel and edit them with tools.

Code generation

The result is automatic PLC and HMI code generation. TE3510 makes it possible to automatically generate PLC code based on the TwinCAT Vision libraries (TF7xxx) from the created block-based configuration. As can be seen in the picture above, two different approaches are offered.

1. Classic analytics approach: The image data comes via Analytics Logger using MQTT via a message broker. If the Analytics Vision configuration is to be converted into PLC code, transparent PLC code is traditionally generated, which is deployed to a remote device in order to carry out continuous image data evaluation on this device. Optionally with your own HMI dashboard, which can be generated directly.
2. Vision application: The prerequisite for the second case is a controller with a camera, which is configured in the TwinCAT System Manager. PLC code can be dispensed with initially, as this can be generated by TwinCAT Analytics Vision itself. You simply take the images from the VnTask and interconnect them in the Analytics configurator. Then deploy the configuration to the original source controller. In this case, the necessary FB_VN_CameraControl is generated automatically.
3. Vision application 2: In another scenario, the task is not integrated via ADS, but a File Source with image data from the local hard disk. This automatically uses the FB_VN_FileSourceControl in the generated PLC code.

Extensions with other packages

All TE351x products can be combined with each other. For example, you can combine TE3510 Analytics Vision with TE3512 Analytics Condition Monitoring. Alternatively, purchase TE3500 Analytics Workbench. This product contains all algorithms and offers all functions.

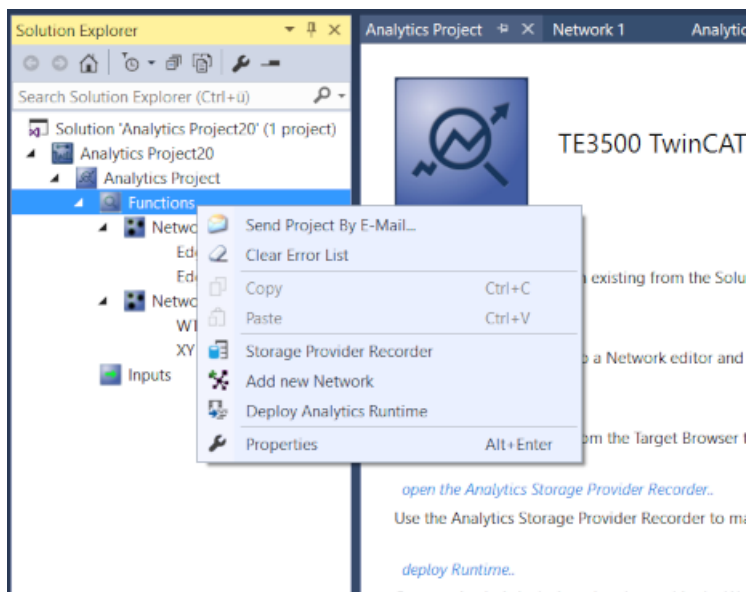
5 Configuration

5.1 Editor Basics

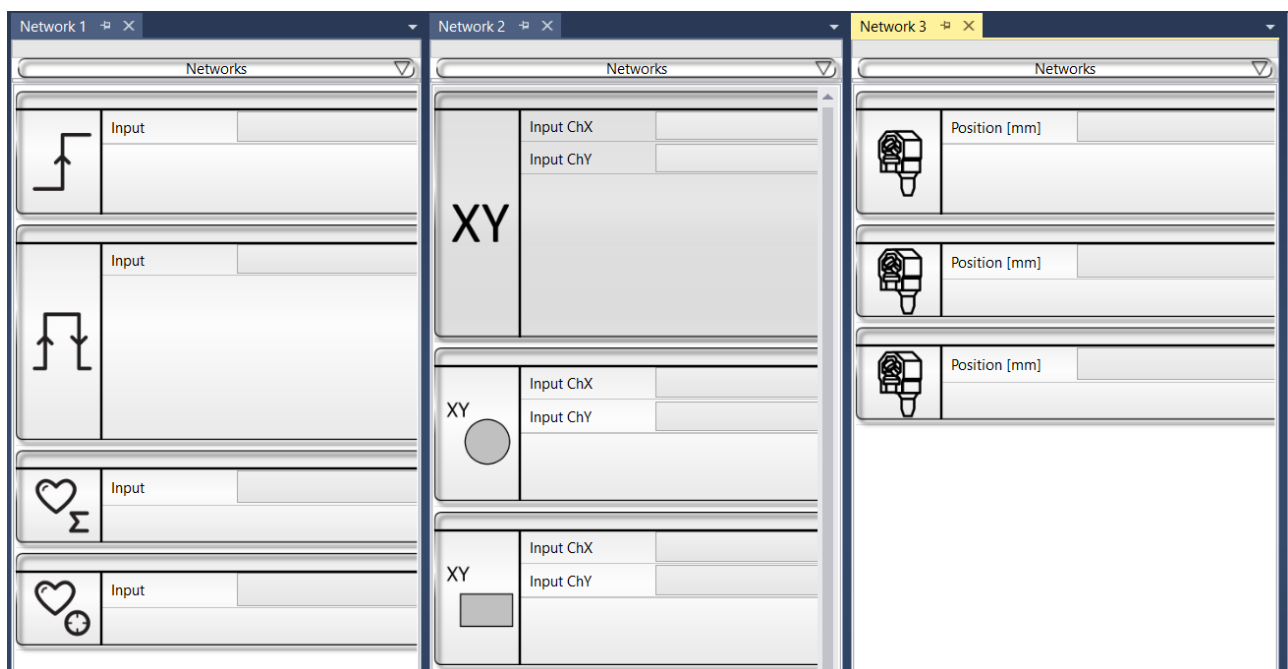
5.1.1 Networks

The networks are suitable for organizing and structuring an analysis. This significantly increases the clarity. Furthermore, they can also serve as containers for algorithms, which you can save as templates.

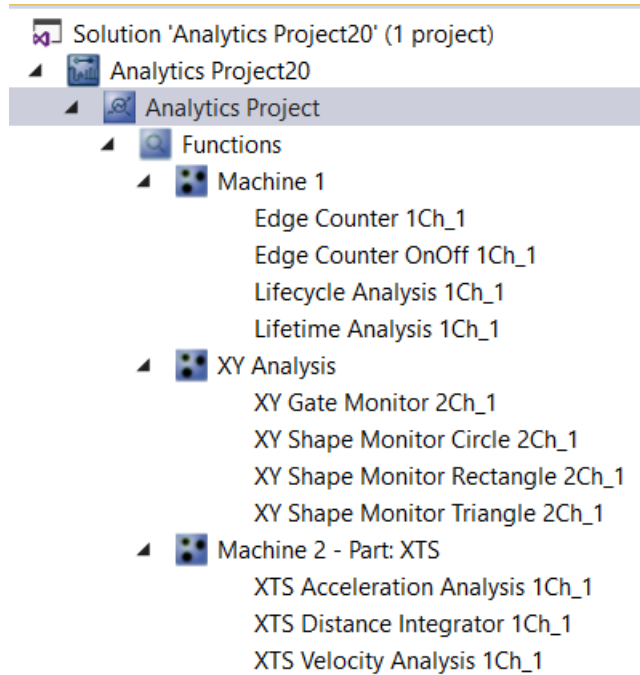
A network can be added directly from the Analytics Project home page or from the context menu.



Each network is displayed in an individual tab. In this way, the networks can be displayed separately, i.e. side by side or one above the other.



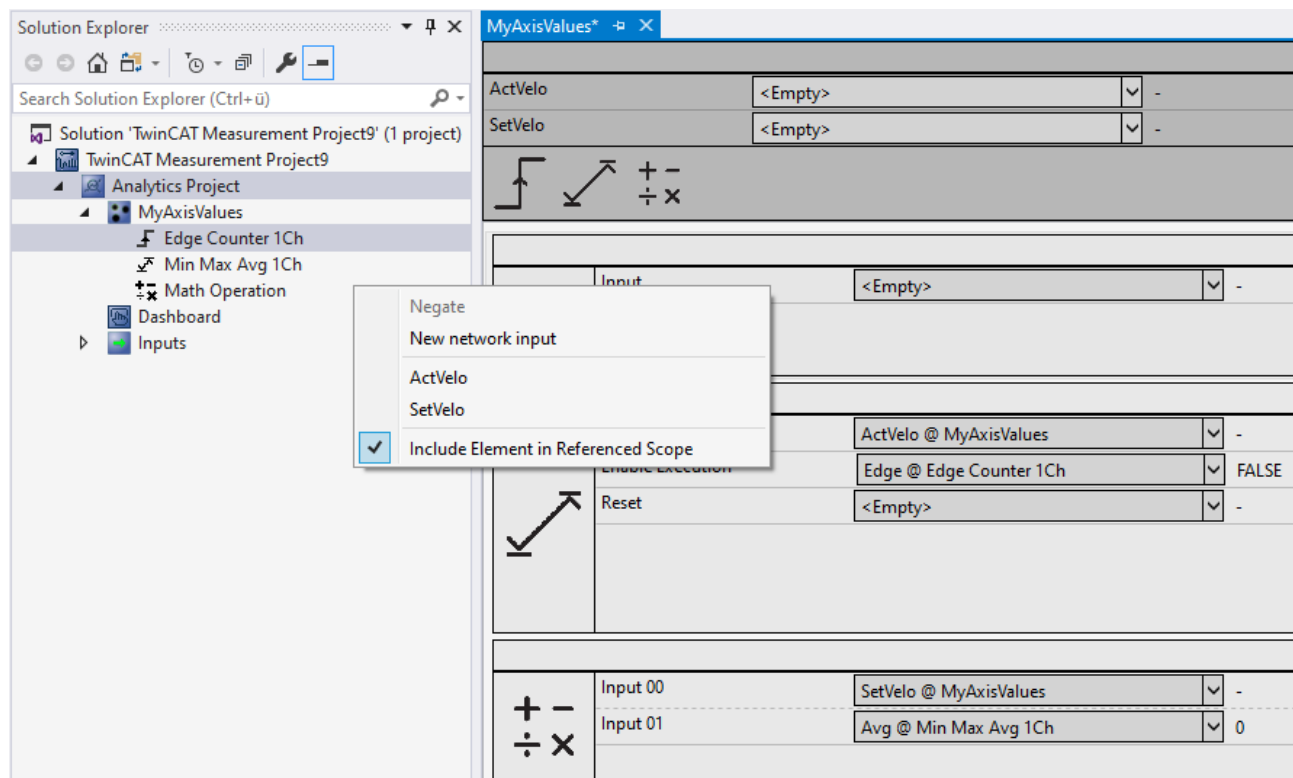
Furthermore, you have the option to rename the networks (F2 on the selected network element in the Solution Explorer) to create networks for different machines, machine parts or other content-related connections, for example.



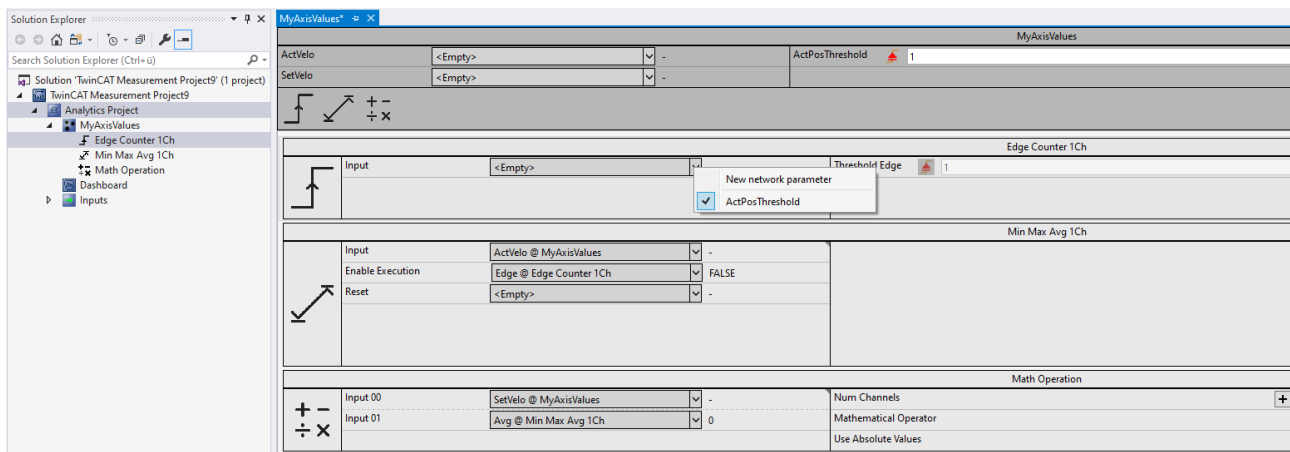
5.1.1.1 Networks as template

Inputs, parameters or even outputs of algorithms within a network can be pinned directly to the outside of the network. Thus, the network itself has inputs, parameters and outputs. This makes it possible to save recurring analyses as templates and to instantiate them several times.

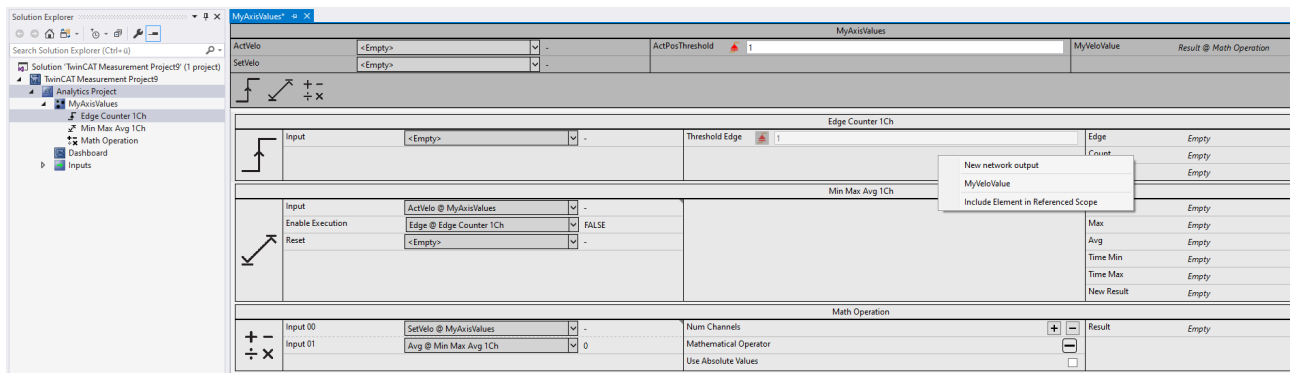
Once you have created an analysis, you must first select which of the available inputs should be visible to the outside of the network. To do this, select the desired input and switch to the context menu by right-clicking. There you can link the input to an existing network input or alternatively define a new network input.



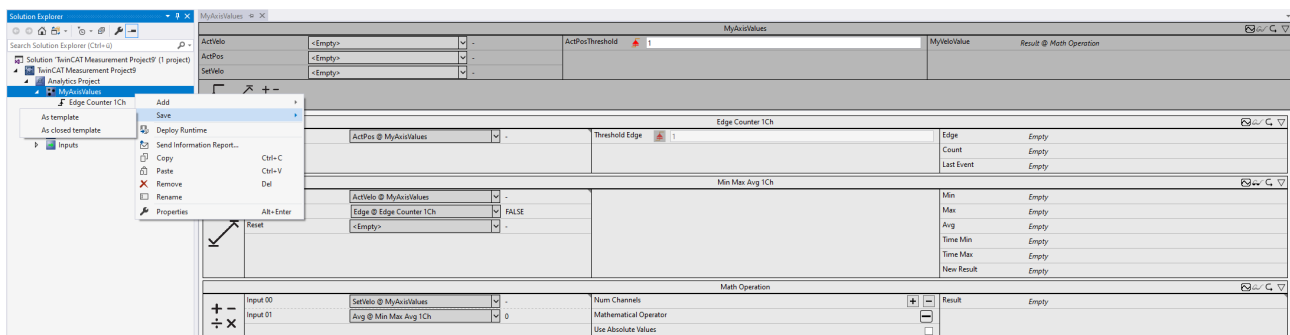
The same procedure is available for parameters and outputs.



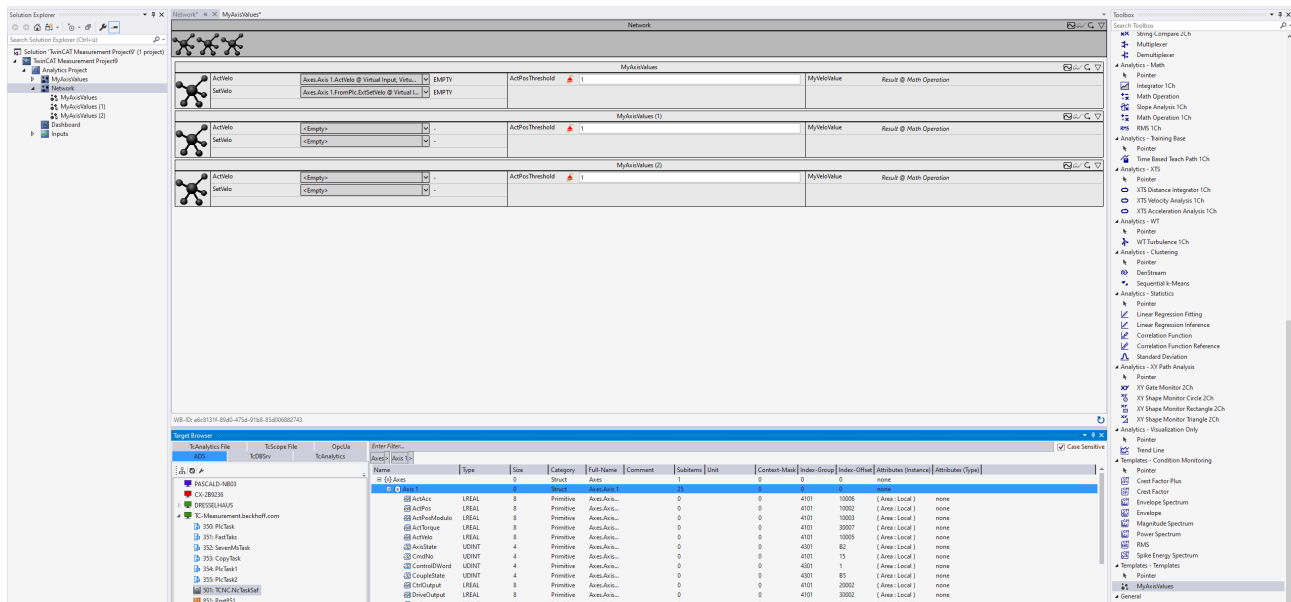
In addition to these inputs and outputs, it is also possible to specify dynamic inputs and outputs. These are automatically offered to you when one of the selected algorithms supports this function. The background to this is the option of increasing the number of inputs or outputs on an algorithm via a parameter, e.g. the inputs of the Math Operation algorithm.



Once the definition of the network to be used as a template is complete, you can save it accordingly. To do this, go to Solution Explorer and right-click on the network. Use the Save option in the context menu. You can choose between Template and Closed Template. In a simple template you can look inside after instantiation and see the interconnection of the basic algorithms and also change them. This option is not available for a closed network. However, this does not offer know-how protection! The internal logic is also visible in a possible PLC code generation by the Analytics Workbench.



After saving, the template is selectable in the toolbox and can be used for recurring analyses.



The Target Browser offers a special function.

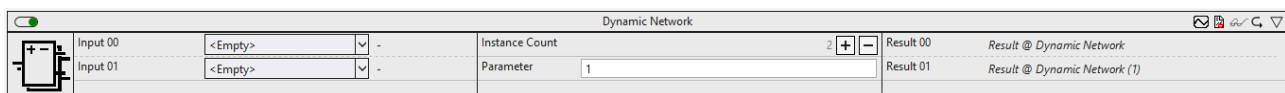
If the names have been chosen for the network in such a way that they correspond exactly to variables of a structure, you can drag a complete structure such as an axes structure directly onto one of the x-inputs of a network. All matching names of all network inputs and structure variables are mapped automatically.

Scope configuration stored in network template

A created Scope configuration can be saved together with the associated network in a [network template](#) [► 298], in order to automatically obtain the same Scope configuration when a network is used again.

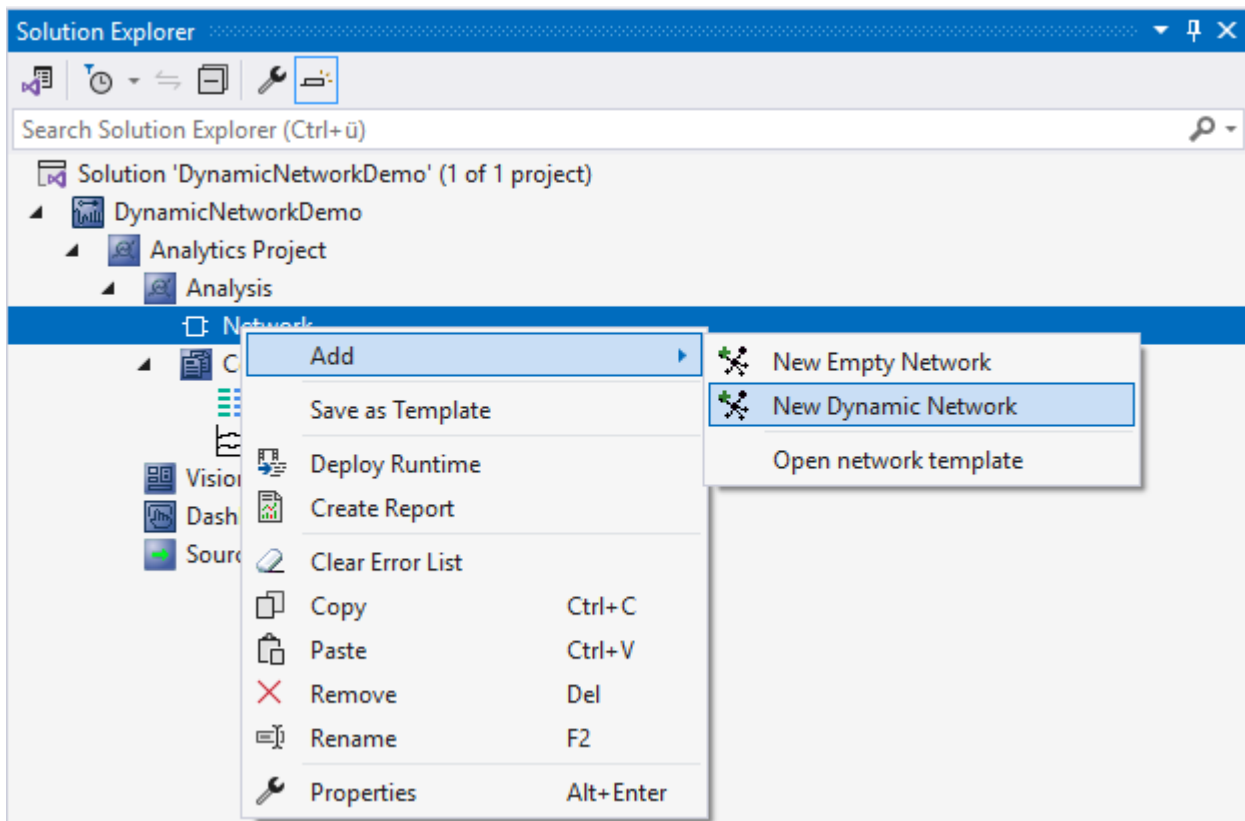
5.1.1.2 Dynamic networks

Dynamic networks offer extended functions for the scalability of the analysis chain. For this purpose, a dynamic network always contains a parameter with the title "Instance Count". This makes it possible to duplicate the current network configuration in order to apply the same analysis function blocks to several inputs.



There are two ways to create dynamic networks:

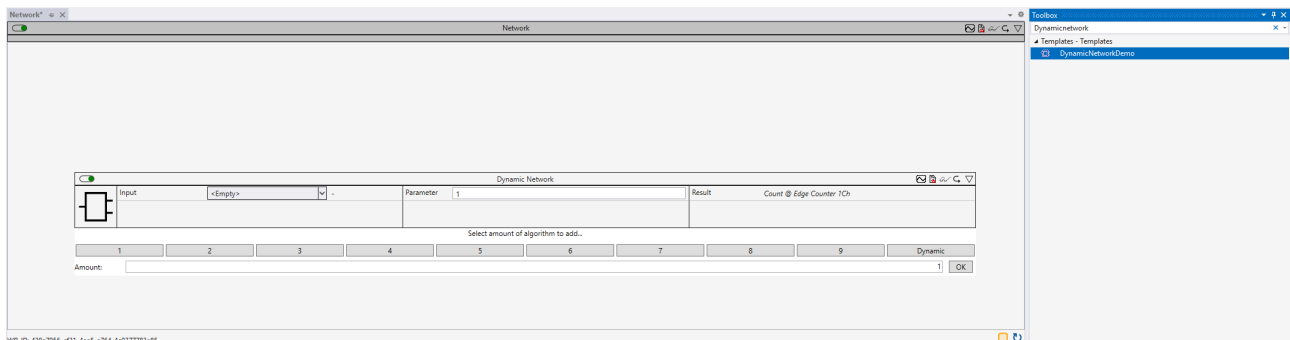
1. Via the context menu in the Solution Explorer:



An empty dynamic network is created that can be freely configured.

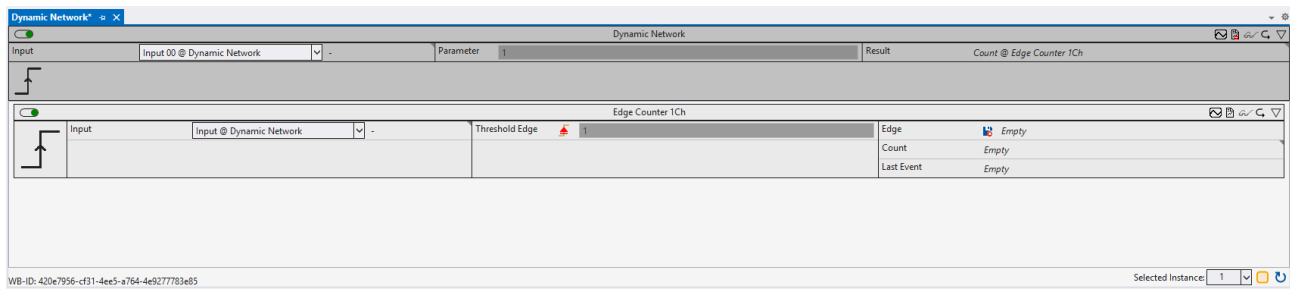
2. From a saved network template from the toolbox:

Choose the desired network template in the toolbox. Hold down the Ctrl key while dragging the template into the editor. A preview of the selected network appears. Click on the **Dynamic** button to add a dynamic instance of the template.



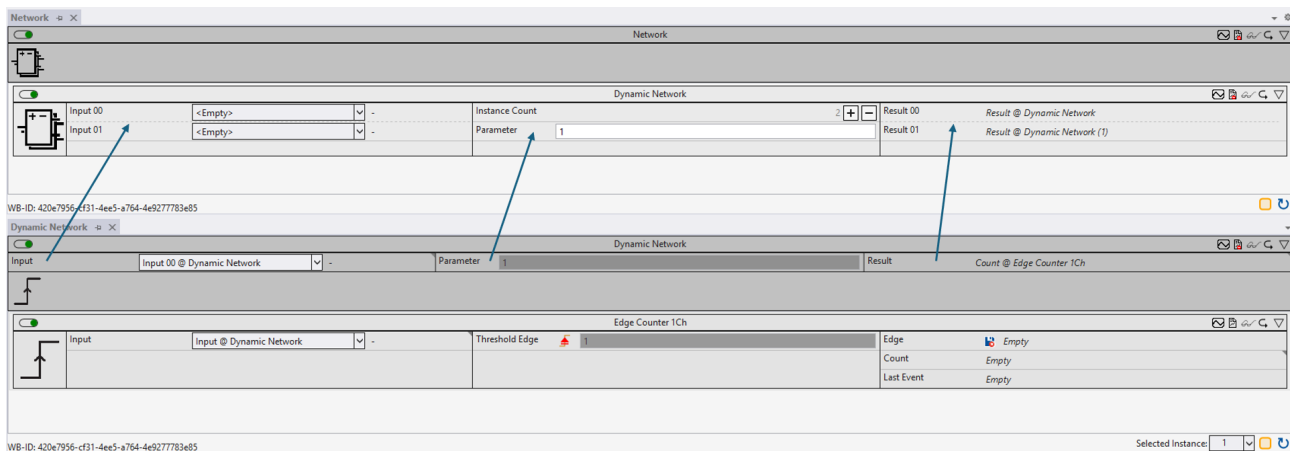
Editing dynamic networks

Dynamic networks can be opened and edited in the editor just like “normal” networks (with the exception of dynamic networks of closed templates). As a special feature, only one instance of the dynamic network is displayed in the editor at a time. All changes made (e.g. the addition or deletion of algorithms) are automatically applied to all instances, thus guaranteeing the same processing of all input data. To inspect the data flow during the analysis, an instance can be selected at the bottom right of the open editor. The values displayed at the inputs and outputs of the algorithms always refer to the selected instance.



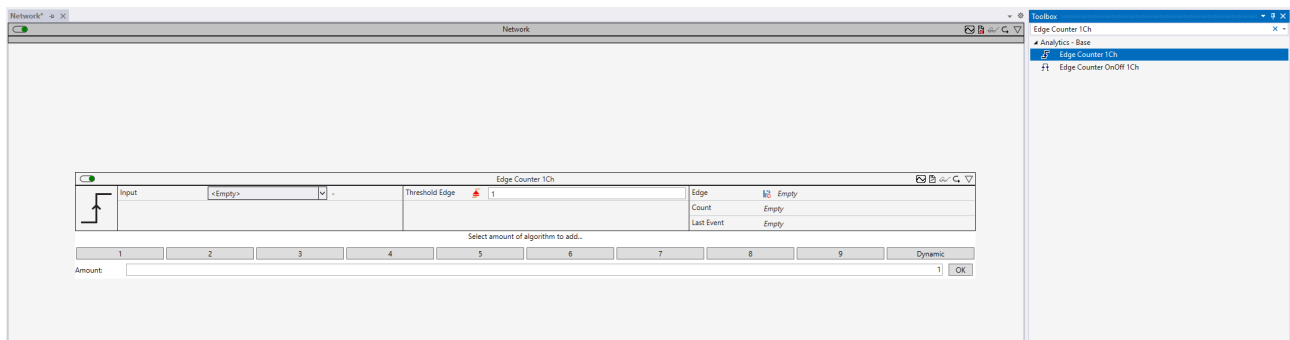
Inputs and outputs of dynamic networks

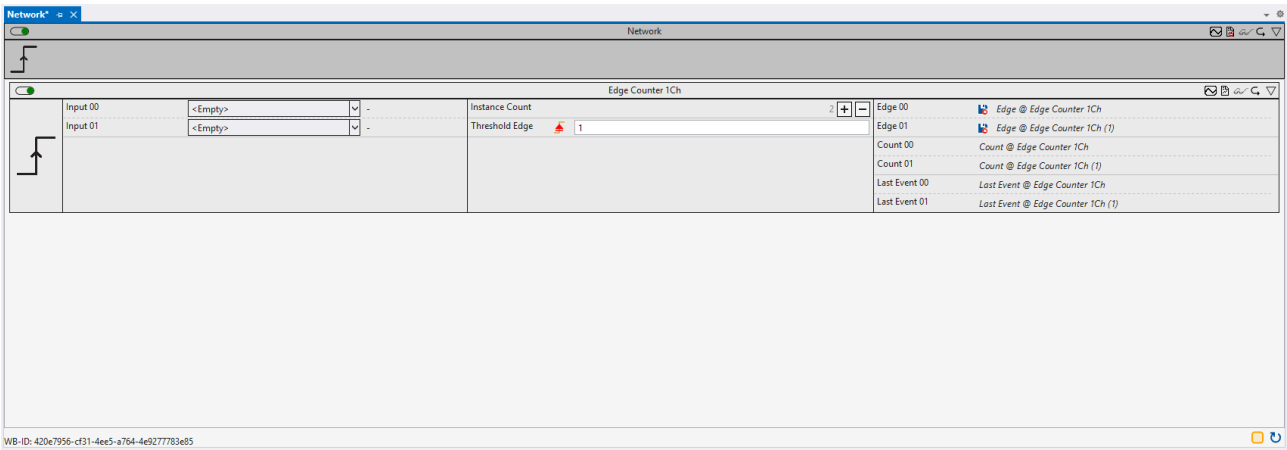
Inputs and outputs of the analysis chain can also be defined in dynamic networks. These do not differ internally from the inputs or outputs of other networks. Externally, however, these automatically become dynamic inputs or outputs, so that the inputs can be linked for each instance and the results can be processed further. The [Smart Link \[► 18\]](#) functions are particularly helpful for the simple linking of dynamic inputs and outputs.



Dynamic functions

It is also possible to create dynamic instances of functions. To do this, hold down the Ctrl key when adding the functions from the editor and select the **Dynamic** button in the preview. The number of instances can also be selected for dynamic functions. Editing is not possible.

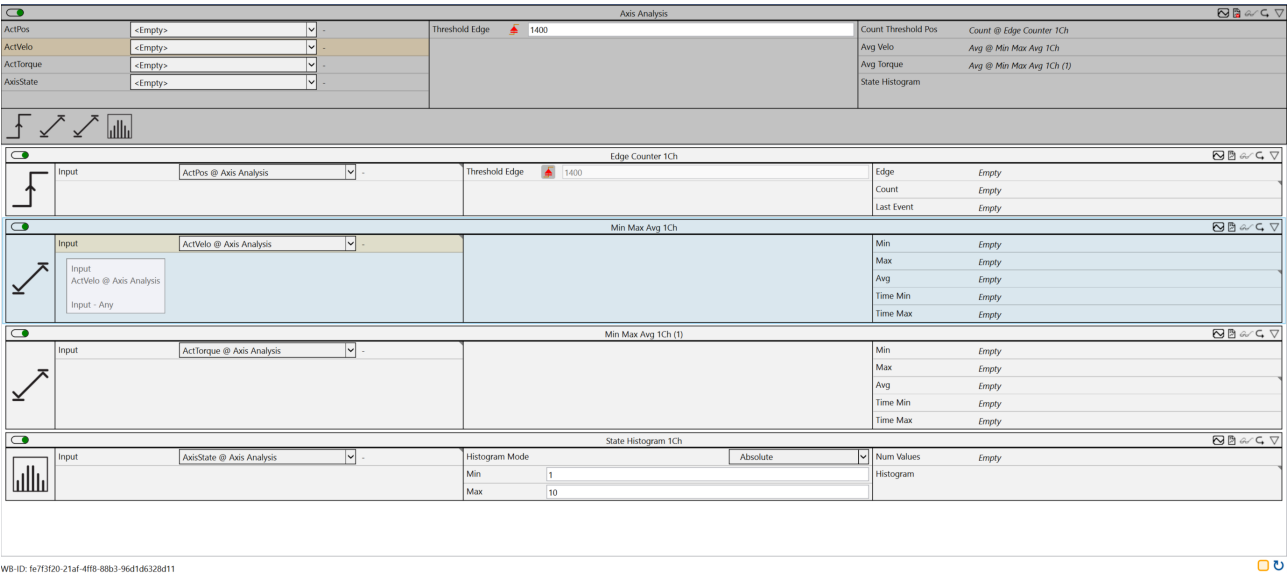




5.1.2 Smart Link

The Smart Link functions in TwinCAT Analytics make engineering easier. Modules can be linked to variables and other values in groups very quickly using a few key/mouse combinations.

A first variant of the Smart Link function is the linking of structure members from the Target Browser. This is particularly useful if you use Analytics network templates. This requires that you name your network inputs exactly the same as the members that you want to link from a structure.



In the figure shown, these are the names of the known NC structure NcToPlc. The inputs are given exactly the same names as the members in the structure.

The screenshot displays the Beckhoff Configuration Editor interface. At the top, there are three network templates labeled 'Axis 1 Analysis', 'Axis 2 Analysis', and 'Axis 3 Analysis'. Each template has a table of inputs and outputs. The 'Axis 3 Analysis' template has a red arrow pointing to its 'AxisState' input, which is currently set to '<Empty>'. Below the templates is the 'Target Browser' window, which shows a tree view of the project structure. A red arrow points from the 'Axis[2]' entry in the 'MAIN' folder to the 'AxisState' input of the 'Axis 3 Analysis' template.

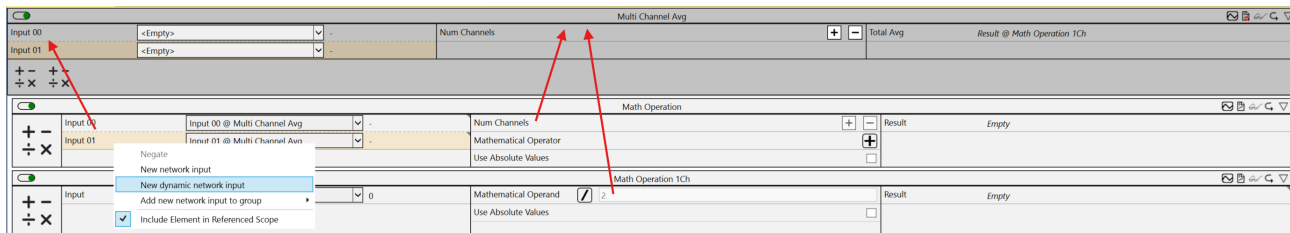
The drive axis structures can be easily linked in the following step. It is sufficient to link the desired structure to one of the inputs of the network template using drag and drop. All members are automatically linked to the inputs.

Another Smart Link function makes it easy to link outputs to inputs while taking their names into account. For example, build a network template and name the inputs in the same way as the outputs of the modules to be linked.

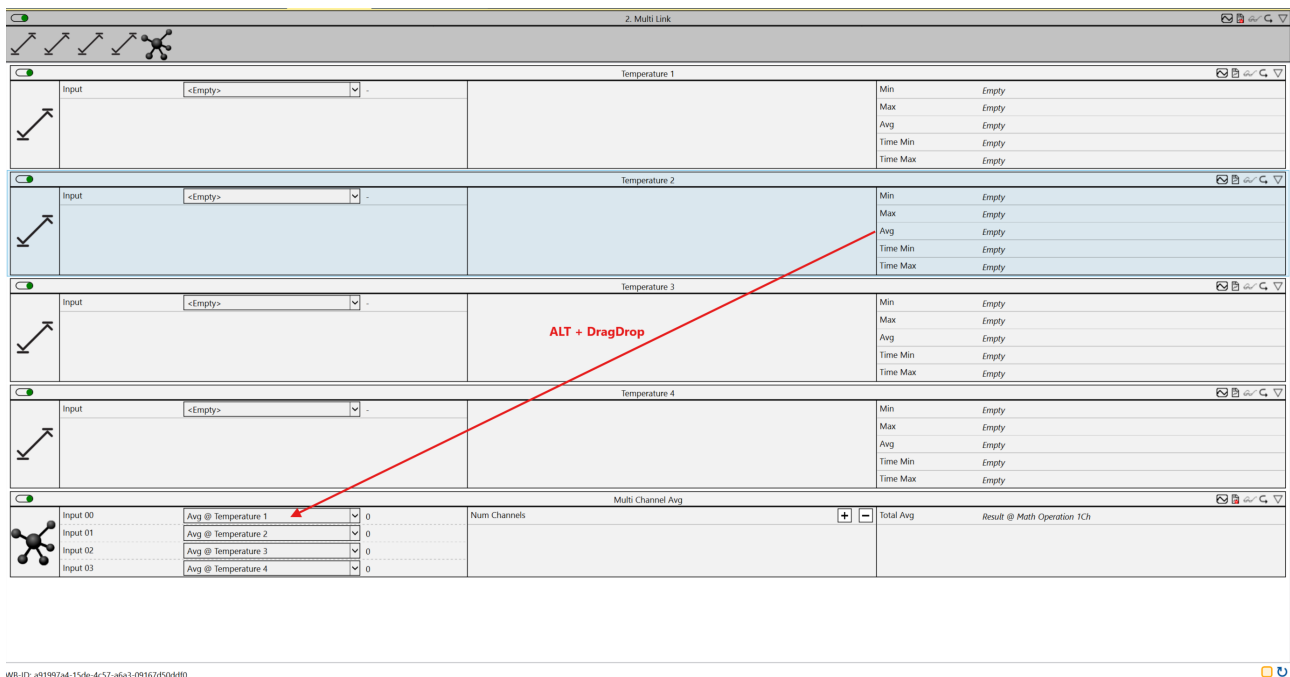
The screenshot shows a network template with three input sections. The first two sections are 'Min Max Avg Temperature' and 'Min Max Avg Pressure'. The third section is 'MyTemplate_Instance1'. A red arrow points from the 'Max' output of 'MyTemplate_Instance1' to the 'Max' input of 'MyTemplate_Instance2'. The text 'ALT + DragDrop' is written in red, indicating the Smart Link function.

This means that, by holding down the Alt key while dragging and dropping an output onto an input of the template, all other outputs can automatically link to their corresponding inputs.

The next Smart Link function is combined with a search function in the modules of the current network. For example, it creates a network template with dynamic inputs, which allows the number of inputs to be set via the parameters during subsequent use. The Num-Channel parameter, which can be seen as an example in the screenshot, is also linked as a network parameter. The same applies to the operand, which grows with the number of channels.

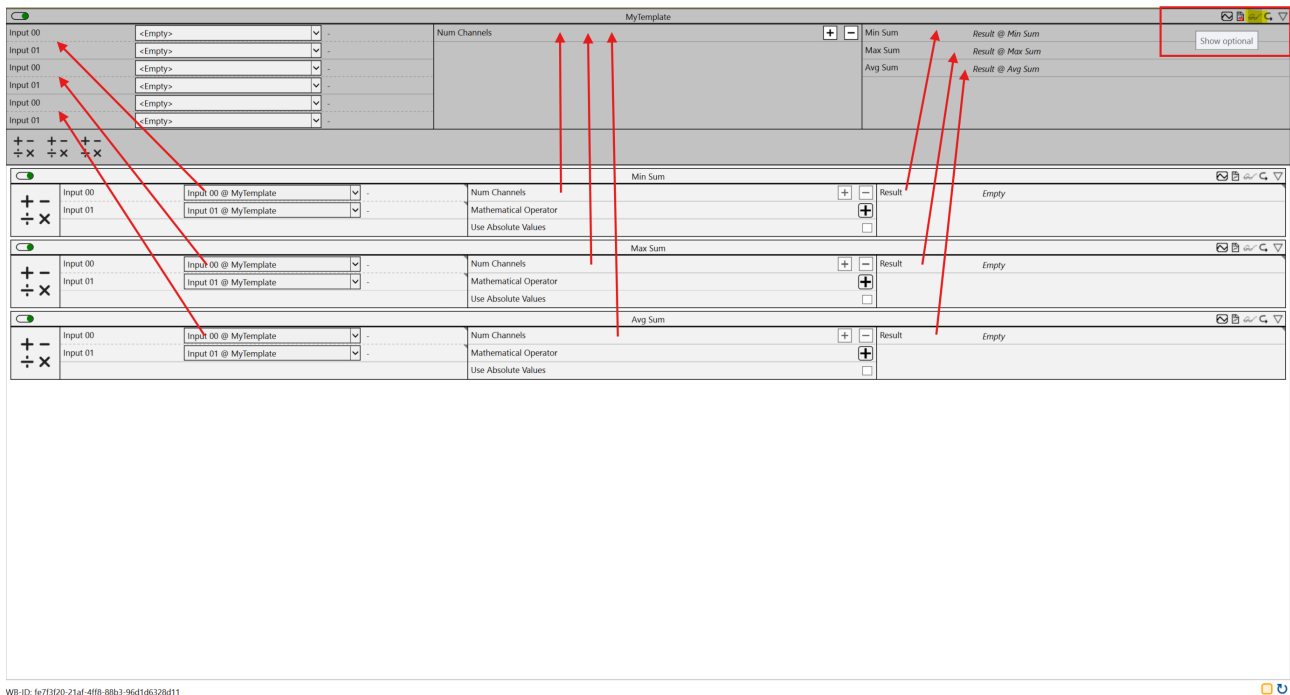


The example should simply create the sum of the X inputs and calculate the average.

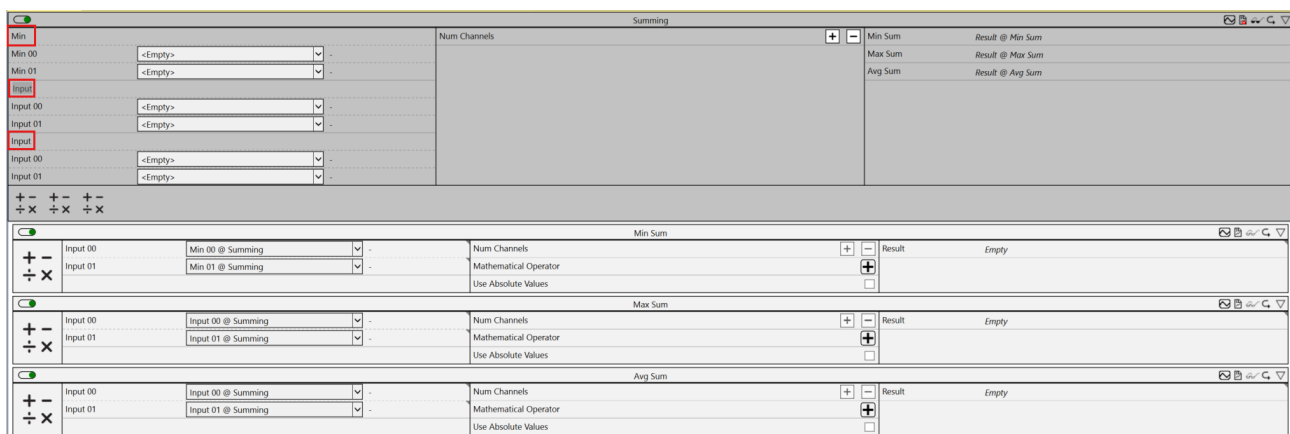


During use, the number of inputs is entered in the template. Then, while holding down the ALT key, drag an output with a corresponding name in the local network to one of the template inputs. This feature searches the local network for outputs with the same name and automatically links them to the template's inputs.

The Smart Link function shown can also be extended to group linking. To do this, first define dynamic inputs, parameters and corresponding outputs in your template. In the example, the sum of all minimum, maximum, and average values of a network is to be calculated.

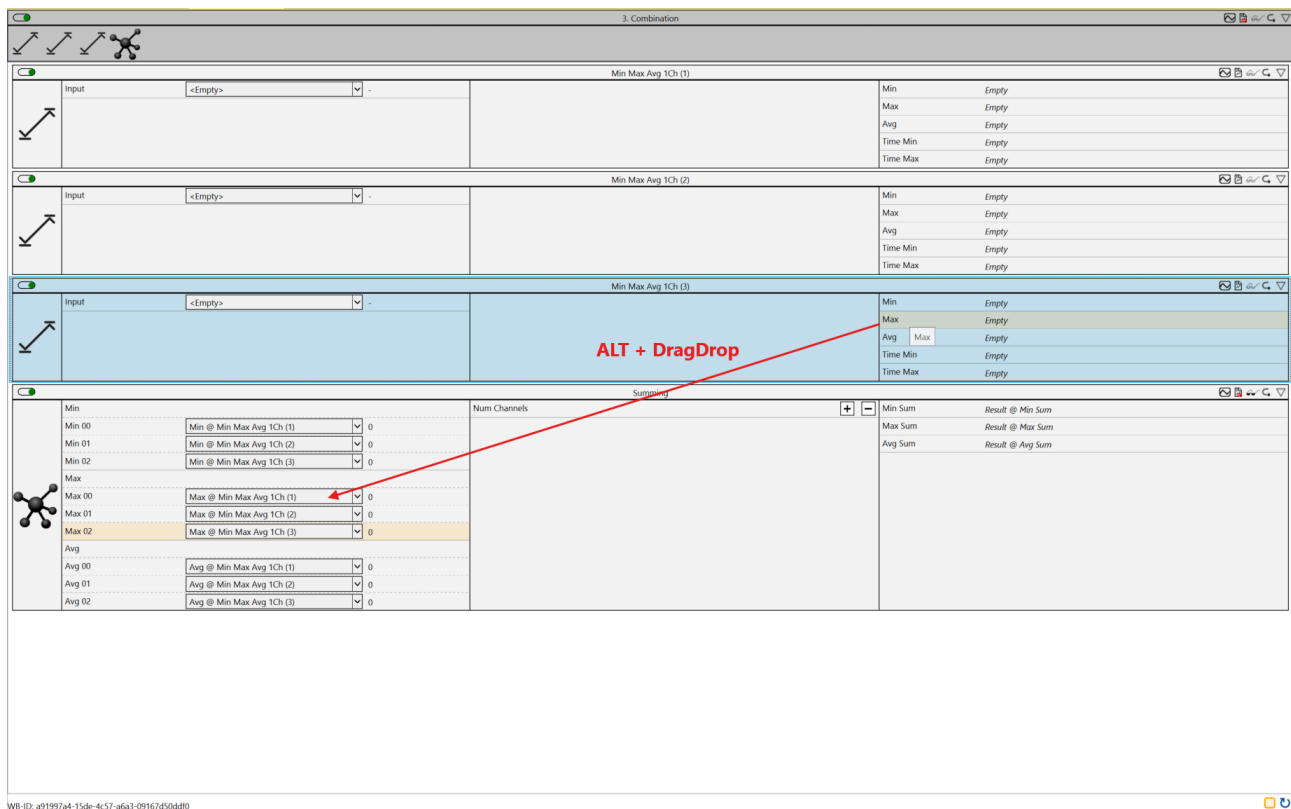


To make the group link work, click on the glasses icon in the top right corner at the network level to reveal more details.



You will now see group names at the network inputs. You assign these as the exact names of the subsequent outputs. In our example, you form the groups Min, Max and Avg.

When using the template, it is again a single operation with the ALT key pressed and a drag and drop from one of the outputs to an input of the template. This searches all algorithms in the network and automatically links all outputs with the same name as the group names.



5.1.3 Enable Disable

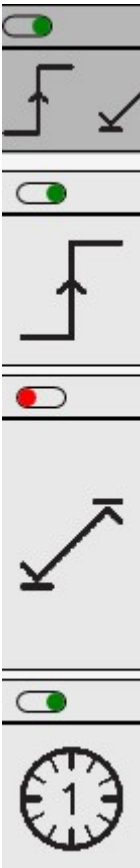
The TwinCAT Analytics Workbench offers the functionality to switch individual algorithms or even complex networks on or off in a data analysis. This document explains the use and advantages of this function.

The function allows users to specifically optimize or even deactivate individual parts of an analysis in order to influence the behavior of the analysis as a whole.

Advantages and application examples for the deactivation of modules in an analysis:

- **Improved precision:**
The ability to turn algorithms and networks on or off in the data analysis allows users to improve the accuracy of the analysis. If an algorithm is not working optimally or a certain network area is not relevant, it can be disabled to improve the overall precision of the analysis.
- **Optimizing resource use:**
Turning algorithms or networks on or off can also help optimize resource use. For example, if an algorithm requires high computational power, it can be turned off as needed to focus resources on other areas of analysis.
- **Flexibility:**
The ability to turn algorithms and networks on or off in data analysis provides users with flexibility and control over analysis. Users can customize the analysis to meet the specific needs of their data.
- **Debugging:**
The function also allows users to debug problems in the analysis. If a particular part of the analysis is faulty, the affected algorithm or network can be disabled to isolate and fix the problem.

Switching modules on or off is controlled by the control elements in the upper left corner of the respective module.



Alignment	Color	Status
Right	Green	Enabled
Left	Red	Disabled

For linked modules, the chain of links is passed through and inputs linked to a disabled module are indicated by a red X at the input.

Disabled Link between two analysis algorithms:

Input

fStairs, Virtual Inp...

Threshold Edge

1

Edge

Empty

Count

Empty

Last Event

Empty

Input

Count @ Zählwerk

Min

Empty

Max

Empty

Avg

Empty

Time Min

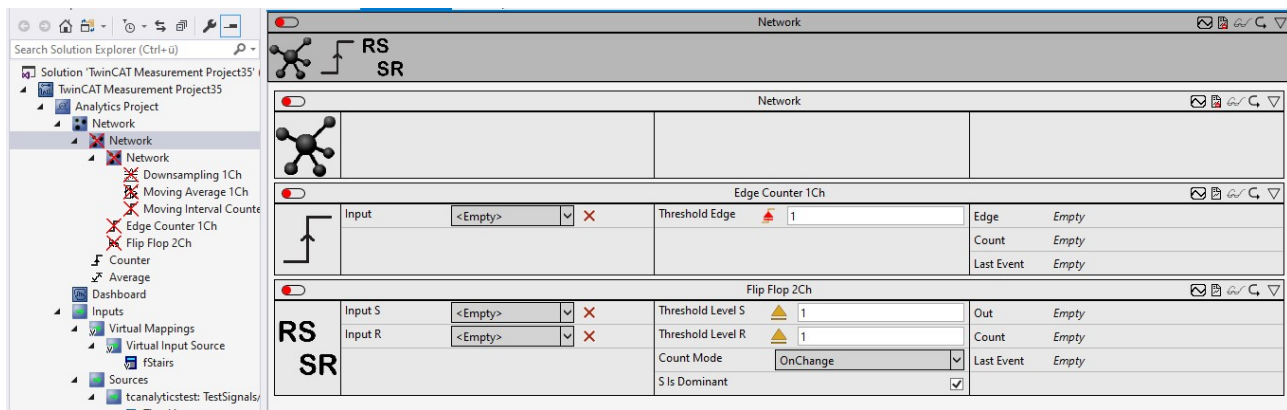
Empty

Time Max

Empty

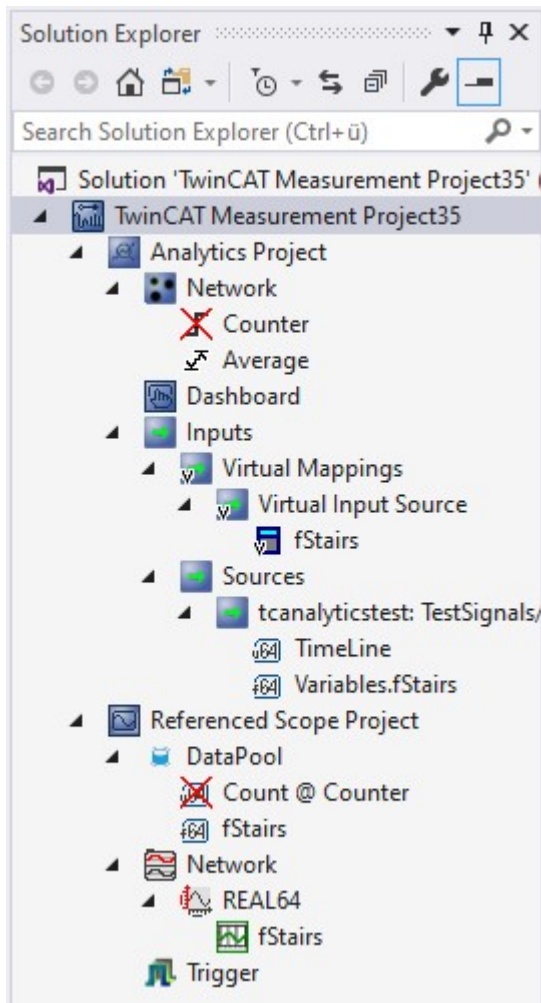
If a network is deactivated, the processing of all algorithms below the network and its subnetworks (recursive) is also deactivated.

Deactivated network elements with their subelements:



If you use a Referenced Scope (see [Interaction with Scope](#) [► 293]), linked elements of the Scope are also affected by the deactivation.

Deactivating the algorithm also deactivates the associated Scope acquisition:

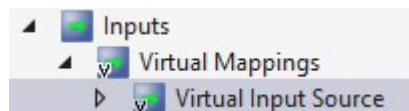


5.1.4 Static Values

Static values are often used in analysis algorithms to store constants that should remain unchanged during the execution of the algorithm. These static values can serve, for example, as thresholds, scaling factors, or other mathematical constants used in the algorithm.

The use of static values in analysis algorithms offers several advantages. First, the performance of the algorithm can be improved by avoiding computations that otherwise, would have to be performed during each execution of the algorithm. Instead, the algorithm can access the static value set during the initialization of the algorithm. On the other hand, the readability and comprehensibility of the algorithm can be improved by structuring the input value more clearly and providing constants with meaningful designations.

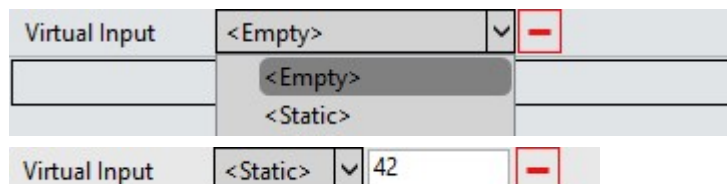
Static values are sorted in the TwinCAT Analytics Workbench under the Virtual Inputs in order to generate them at a central location and manage them later.



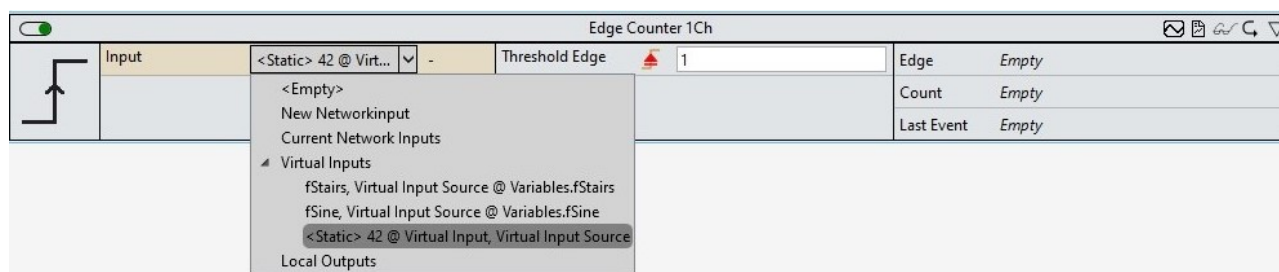
In the editor of the Virtual Input Source you can create a new Virtual Input or edit an existing one by using **+**.



Afterwards, the virtual input can be set to a static value (here 42) via the drop-down menu.



This static value is now selectable throughout the configuration via the Input module.



Static values can also be used on network inputs. Here the linking with the dependent inputs of the individual modules would run exactly the same as the behavior with incoming data inputs.

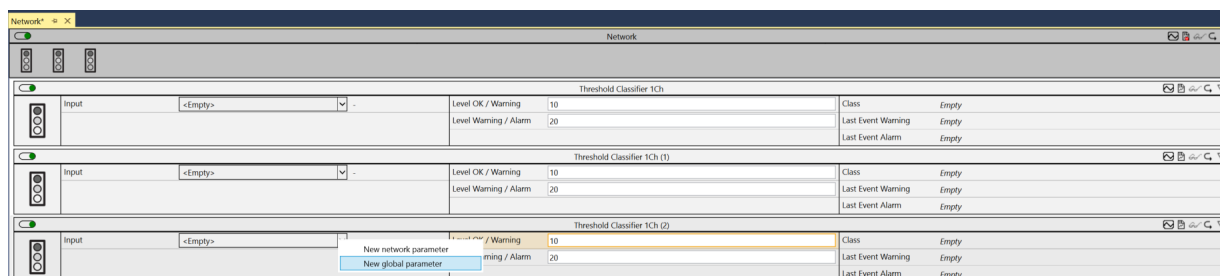


Static values in analysis algorithms should be used cautiously. If the data or parameters to which the algorithm is applied change, it may be necessary to adjust the static values to ensure optimal performance of the algorithm. Therefore, static values should be checked regularly and adjusted if necessary to ensure that the algorithm is working correctly and effectively.

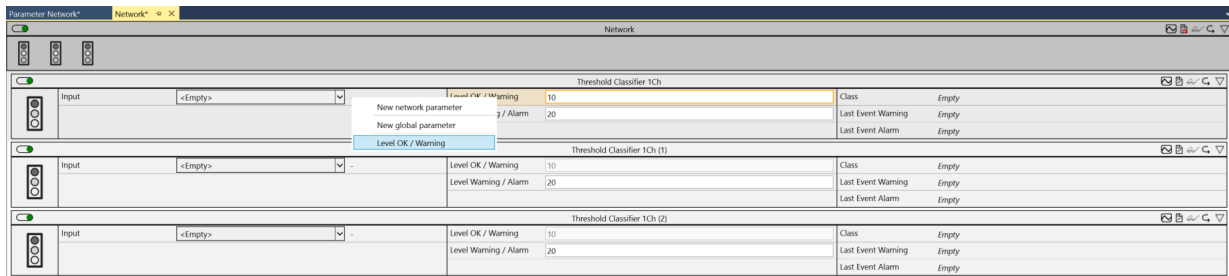
5.1.5 Global parameters

With the option of global parameters, fixed values such as a cost factor for energy management can be defined, which can occur multiple times in different algorithms.

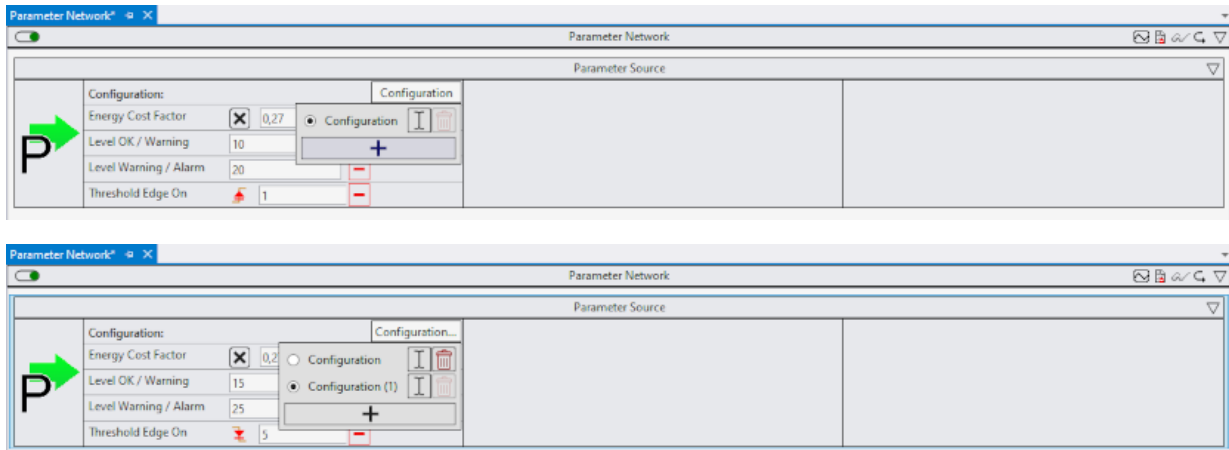
1. To create a new global parameter, right-click on an existing parameter and select the **New global parameter** option.



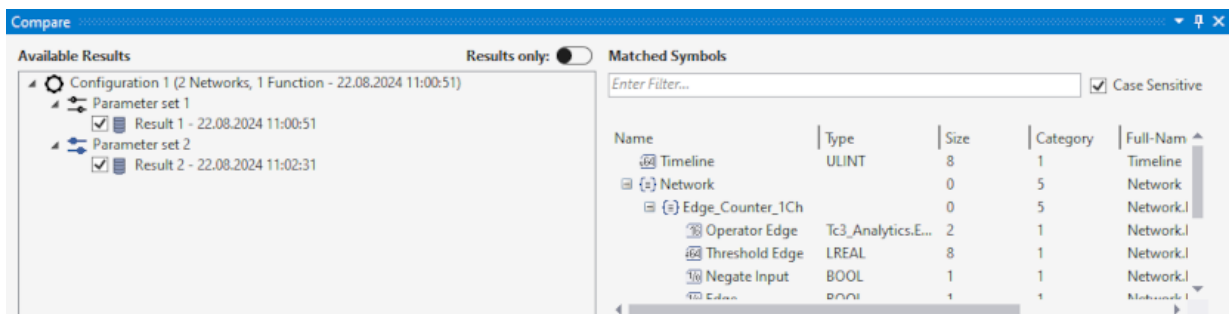
2. Existing global parameters can then be selected via the same context menu on other algorithms.



3. As soon as the first global parameter has been created, a new **Parameter Network** is created. This allows you to set parameters in one central location. It is also possible to define several configurations of parameters that can be switched.



4. If the project is executed successively with different parameter configurations, a new **Parameter Set** with the corresponding result data set is created in the [Analytics Diary](#) [► 478] under the existing project configuration.



- ⇒ No further changes may be made to the project, otherwise a new project configuration will also be created.

5.1.6 Additional analysis windows

In order to precisely evaluate the results of an analysis chain and efficiently adjust parameters, a detailed representation of the image data is required. As large-format image views would limit the clarity in the network editor, two specialized windows have been implemented for working with vision modules: Module Watcher and Vision Gallery.

1. Module Watcher

The “Module Watcher” window is used for direct visual control during the configuration process.

- **Function:** Displays the images currently used in the module in an enlarged view parallel to the network editor.
- **Area of application:** Quick checking of parameter settings and direct viewing of image data without interrupting the workflow.



This window is purely for display purposes; the images are not saved permanently.

2. Vision Gallery

The “Vision Gallery” is designed as a toolbox for in-depth analysis and documentation.

- **Long-term viewing:** Images can be stored over a longer period of time and saved in a targeted manner.
- **Comparison function:** It is possible to save reference images from previous analysis times in order to compare them directly with current results.
- **Interaction:** The tool offers the option of making markings in the images to highlight special features or sources of error.

5.1.6.1 Module Watcher

The “Module Watcher” window is used for direct visual control during the configuration process.

- **Function:** Displays the images currently used in the module in an enlarged view parallel to the network editor.
- **Area of application:** Quick checking of parameter settings and direct viewing of image data without interrupting the workflow.

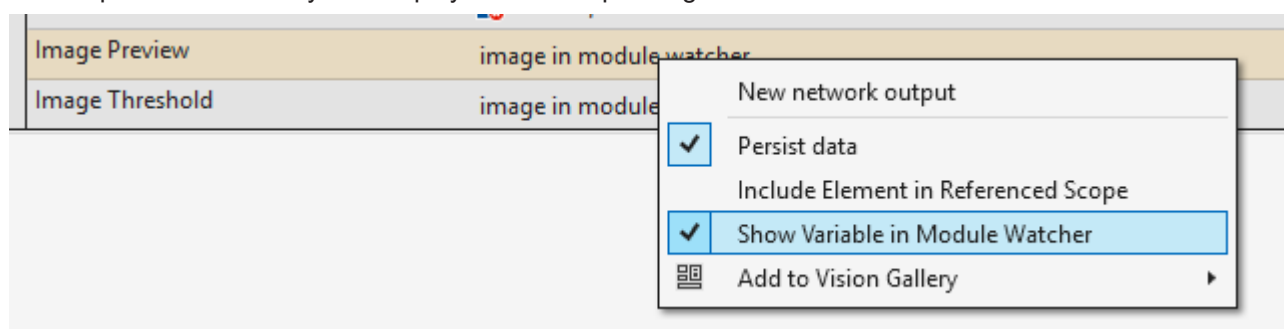


This window is purely for display purposes; the images are not saved permanently.

Open

There are two ways to call up the “Module Watcher” window:

1. **Via the main menu**
Navigate in the Visual Studio menu bar to:
Visual Studio menu: TwinCAT > Analytics > Module Watcher
2. **Via the network editor (context menu):**
In the network editor, right-click on an **In variable** or **Out variable**. In the context menu that appears, select the **Show Variable in Module Watcher** command (see illustration below). The window then opens automatically and displays the corresponding variable.



Dynamic display and interaction

The Module Watcher is directly synchronized with the active network editor. The display is context-dependent: The window automatically visualizes the data of the module that you have currently chosen in the editor.

- **Automated update:** As soon as you click on another vision module in your analysis, the Module Watcher updates the view and displays the images processed in it.
- **Workflow advantage:** This direct coupling allows you to quickly switch between different modules and compare their image data without manually reloading.

- **Visible values:** The eyeglasses icon, which you will find with each module, also have an effect on the Module Watcher.

Navigation and detailed view

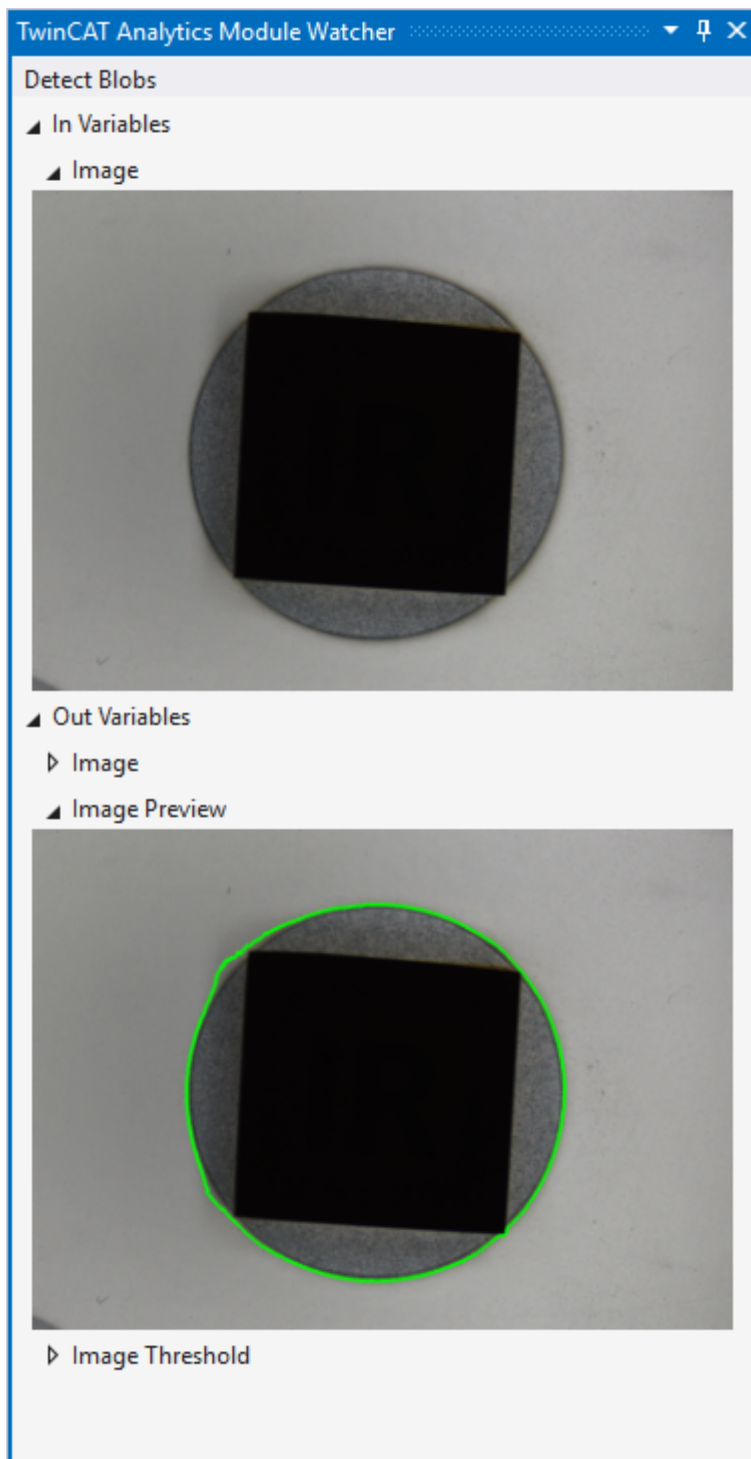
The user interface is structured hierarchically to ensure a clear representation.

Hierarchical tree view

Below the module name, all available data is visualized in a tree structure.

- **Structure:** The display is divided into groups. The In variables are listed first, followed by the Out variables.
- **Grouping:** Within these categories you will find the individual variables that are assigned to the module.

Module Watcher - Complete function:

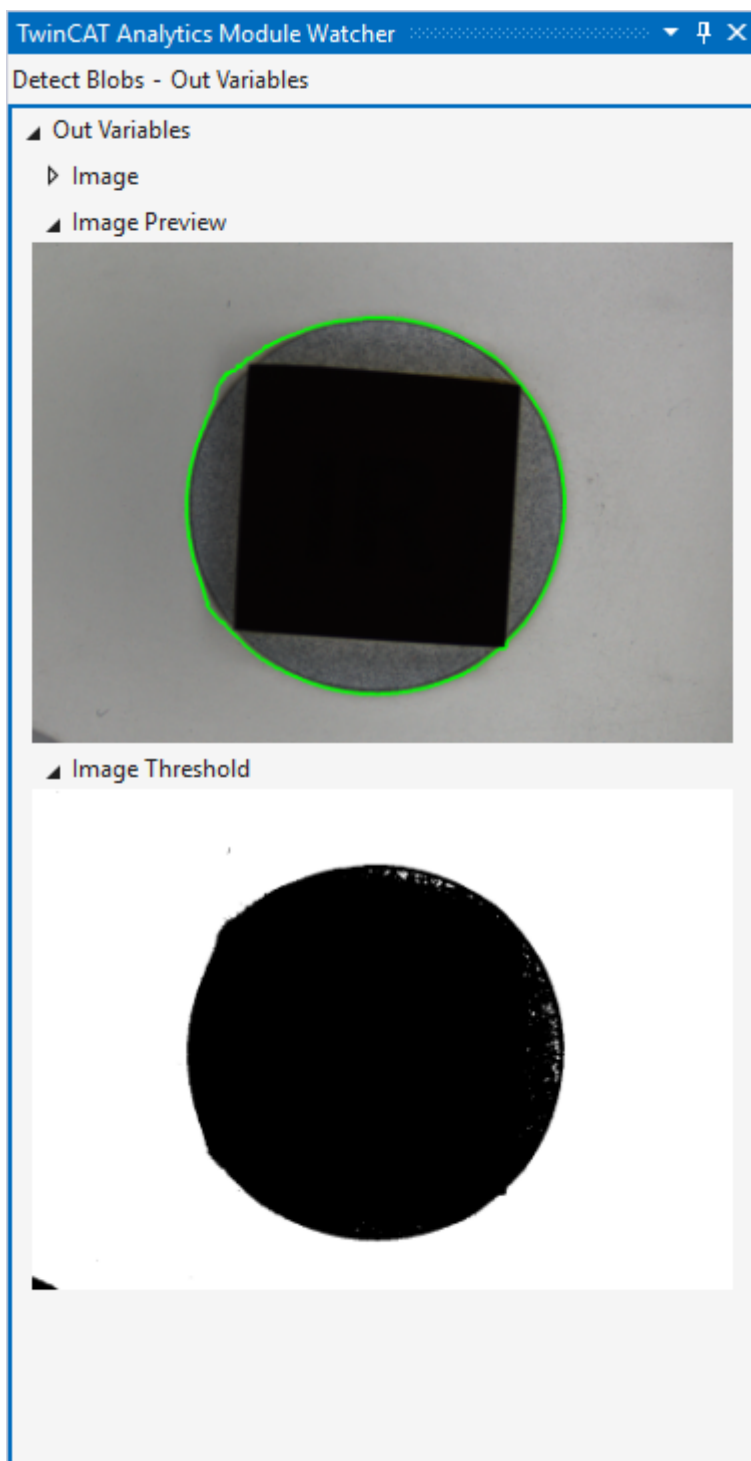


Focus mode (root navigation)

You can use the focus view to limit the display to relevant information:

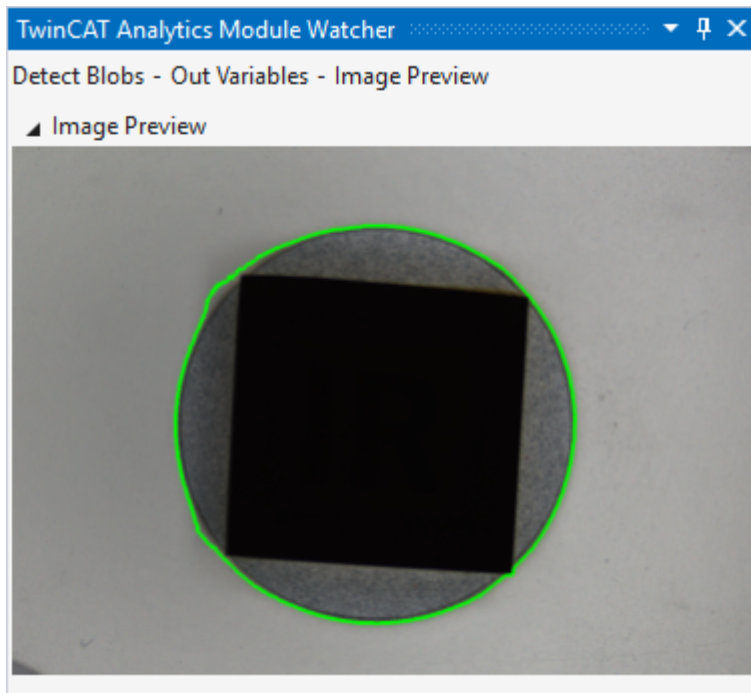
- **Switch levels:** Double-click on an item (e.g. the "Out Variables" group) to make it the new starting point (root node) of the view. All other items are hidden in order to make optimum use of the available space for the relevant images. Double-clicking an image produces the same result as selecting the "Open" option from the context menu.
- **Breadcrumb navigation:** As soon as you are at a lower level, a breadcrumb bar appears in the title bar. This shows your current path (e.g. module name > Out Variables). You can switch back to a higher hierarchical level by clicking on the entries.

Module Watcher - Out Variable Focus:

**Example: Individual display of images**

For example, if you double-click on a specific variable such as “Image Preview”, only this image will be displayed in large format. This is particularly helpful for analyzing the effects of parameter changes in detail without being distracted by other image data.

Module Watcher - Image Preview:



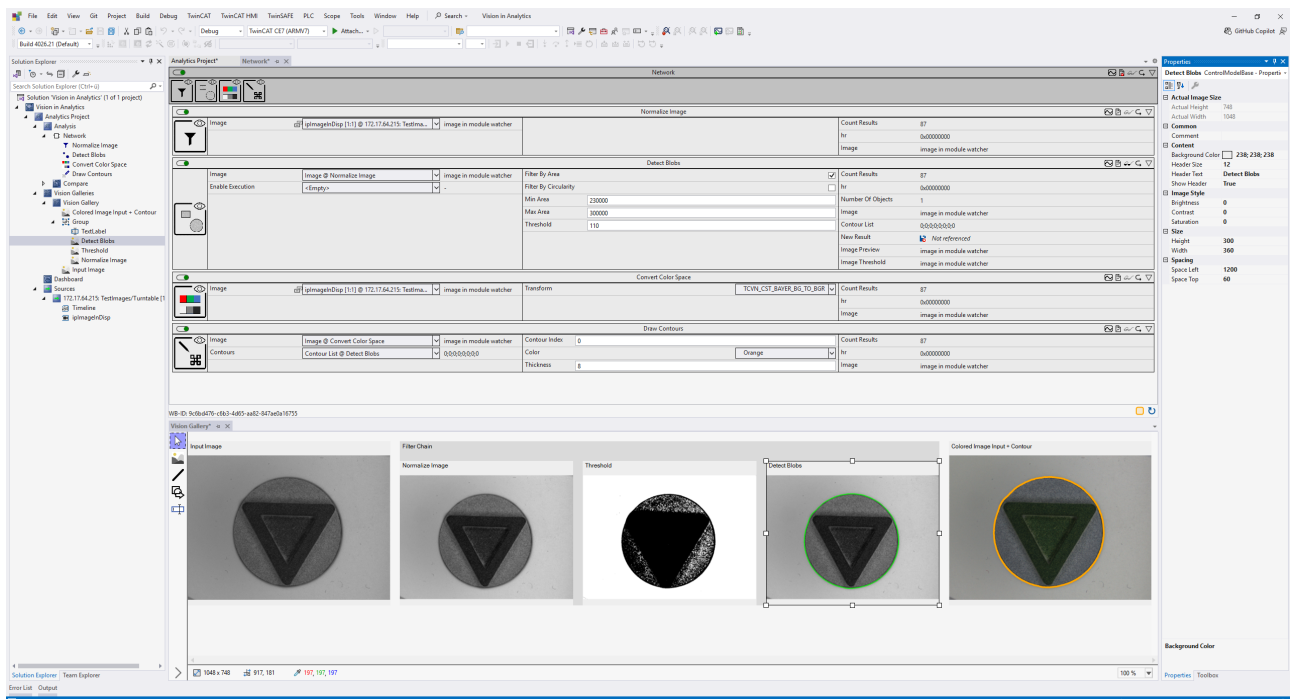
5.1.6.2 Vision Gallery

The “Vision Gallery” is designed as a toolbox for in-depth analysis and documentation.

- **Long-term viewing:** Images can be stored over a longer period of time and saved in a targeted manner.
- **Comparison function:** It is possible to save reference images from previous analysis times in order to compare them directly with current results.
- **Interaction:** The tool offers the option of making markings in the images to highlight special features or sources of error.

Integration in Visual Studio

The following figure shows the integration within the Visual Studio environment:



All instances of the Gallery are managed centrally in the Solution Explorer under the dedicated “Vision Galleries” node. This enables quick access to analysis layouts that have already been created within your solution.

There are several ways to add a new Vision Gallery:

1. **Via the Solution Explorer:**

Right-click on the “**Vision Galleries**” node and select the command to create a new gallery from the context menu.

2. **Directly from the analysis (context menu):**

You can also create a new gallery directly from the workflow. To do this, right-click on any Image In or Out variable in the Network Editor or in the Module Watcher and select the corresponding option in the context menu.

The fine-tuning of the Gallery and the objects it contains is done via the standard Visual Studio Properties window. As soon as you select an item within the Gallery, all editable parameters are displayed there.

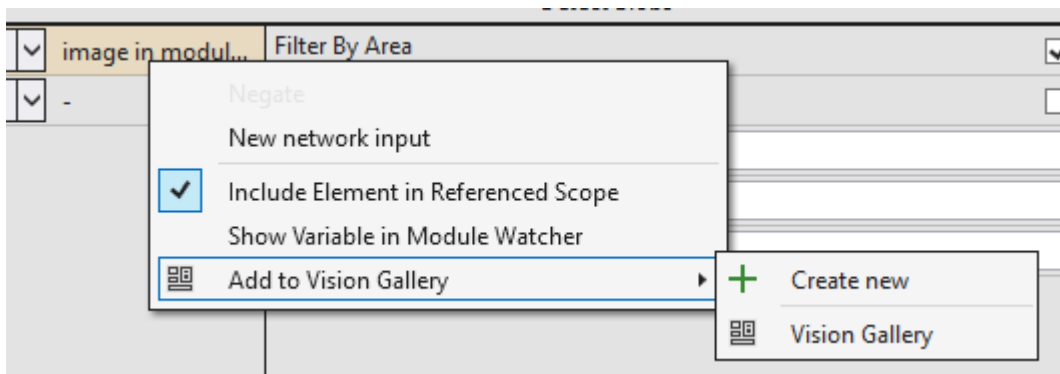
By linking to the Analytics Timeline, you can also navigate specifically through the analysis period in order to view historical image data in detail.

Adding analysis images

There are two methods available for transferring image data to a Gallery:

1. **Via the context menu**

Right-click on the desired image variable in the network editor. Using the context menu (see illustration below), you can either assign the image to an already open Gallery or create a new Gallery instance directly.



2. **Drag and drop:**

Drag the variable with the mouse directly from the network editor into the Gallery window. The decisive advantage here is that the image is placed exactly at the position at which you release the mouse button. This allows you to design your layout intuitively.

Toolbox

The toolbar is located on the left-hand side of the Gallery. To activate a Toolbox, select the corresponding icon in the list. Two creation methods are available for all Toolboxes that create new objects (e.g. shapes, lines or additional images):

1. **Double-click to create:**

Double-click at the point in the Gallery where the object is to be placed. The item is then automatically created in a predefined standard size (default) at this position.

2. **Creation by dragging:**

To define the size of the object directly when creating it, click on the desired start point and drag the item while holding down the mouse button. As soon as the desired dimensions are reached, release the mouse button.

Below you will find an overview of the Toolboxes available for customizing and annotating your Gallery:

	Cursor (selection tool): This is the standard tool of the Gallery. It is automatically active if no other drawing tool has been chosen. When the cursor is active, you can select existing objects in the Gallery by clicking on them. A selected object can then be moved freely on the workspace or its size can be adjusted (scaled) using the handles on the corners and edges.
	Image tool: Allows you to manually insert additional graphics. The property grid can be used to load images from the hard disk, for example to store static reference images for direct comparison in the Gallery.
	Line tool: Used to mark and measure image areas. When drawing on an image, the exact length and angle of the line are automatically displayed below the image. Particularly efficient: The determined angle can be transferred directly to parameters in the Workbench (e.g. for the Rotate Image function block). The line thus functions as an interactive configuration aid for image alignment.
	Shape tool: Enables the insertion of geometric objects, whereby a rectangle is created by default. The property grid can be used to switch the type flexibly between rectangle, circle and arrow. Particularly useful: For rectangles, the position and dimensions can be adopted directly as parameters for an ROI function block in the Workbench via the context menu, which considerably simplifies the precise definition of image areas.
	Text tool: Allows you to add individual texts and descriptions within the Gallery. This allows analysis results to be documented directly in the view or prepared for other users in an understandable way.

Object manipulation

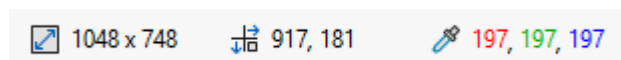
The handling of objects within a gallery follows common standards of advanced graphic tools, which enables intuitive operation.

The following functions are available for all placed items:

- **Move:** Selected objects can be freely positioned on the workspace using the mouse.
- **Scaling:** The size of an object can be individually adjusted at any time using the handles on the corners and edges.
- **Grouping:** Several objects can be grouped together so that they can be moved or aligned together. Management is carried out in the Solution Explorer either via the context menu or intuitively via drag and drop.
Particularly important: Objects that are placed on a picture are automatically grouped with it. In this case, the coordinates of the object (e.g. position of a rectangle) do not refer to the Gallery area, but to the native pixel coordinates of the image. This means that markings remain absolutely true to position even when the image is scaled or moved.

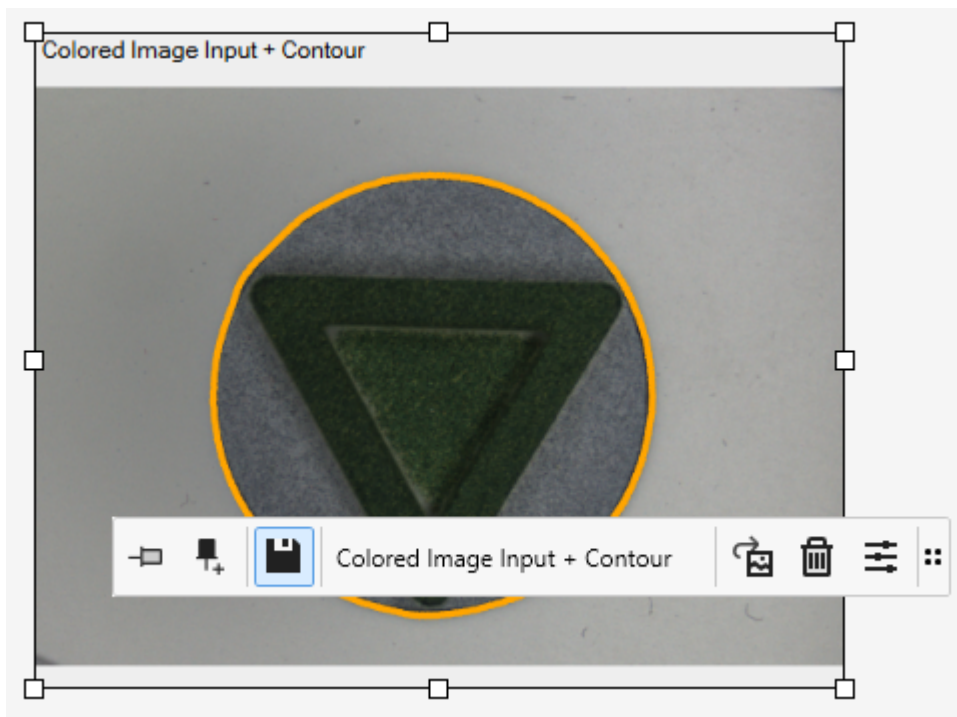
Additional interactions

The information bar is located at the bottom of the Gallery (see illustration below). All values displayed there refer to the original dimensions of the image and are independent of the current scaling or display size within the Gallery.



- **Image dimensions (left):** Shows the actual resolution of the currently selected source image in pixels.
- **Cursor position (center):** Specifies the exact pixel coordinates (X/Y) of the current mouse position within the image that is currently under the cursor.
- **Color value display (right):** Displays the RGB color value of the image pixel located directly under the mouse cursor. This enables a quick check of color and brightness values during the analysis.

Click on an image in the Gallery to open the annotation popup.



	Pin function (time fixing): This function can be used to freeze an image to a specific point in time. If the image is “pinned”, it is no longer updated by successor data from the live analysis and remains unchanged even when scrolling through the timeline. For better orientation, the selected time is also visualized by a marking directly in the Analytics Timeline.
	Pin & Copy: This function extends regular pinning by creating a copy of the current image and inserting it into the Gallery as a new, independent object. While the original view can still be updated live, the newly created image remains fixed to the selected time. This enables the simultaneous comparison of different analysis states within a gallery.
	Persistent function: Controls whether image data remains available for the review in the timeline. • Activated: When scrolling through the timeline, you can access past recordings of the connected output and view them in the Gallery. • Disabled: Only the current image is available. When moving the timeline position, the object does not display any data for past points in time. • Standard: To optimize the system load, intermediate screens in modules are predefined as non-persistent by default.
Name	The name assigned here is displayed directly above the image in the Gallery for quick identification.
	Export: You can use this button and the submenu to export the selected image. Details on the options can be found in the following section.
	Delete: Removes the currently selected image from the Gallery.
	Adjustments: Opens the Toolbox for image editing. The available parameters are explained in detail below.

Image adjustments

The following illustration shows the available adjustments with which you can optimize the representation of an image to make details more visible.



These adjustments only affect the visual representation within the Gallery. They have no influence on the image data in the current analysis. The function is used to carry out tests quickly or to facilitate the placement of additional Toolboxes through improved visibility.



Saturation	Saturation: Adjusts the intensity of the colors. Increasing the color intensity makes the colors more vivid, while reducing the color intensity makes the image grayscale.
Brightness	Brightness: Controls the general light intensity of the image. This can be used to brighten underexposed images or darken areas that are overexposed.
Contrast	Contrast: Controls the difference between light and dark areas of the image. A higher contrast enhances the definition of details, while a lower contrast makes the image appear flatter.

Export

Gallery items can be flexibly exported and saved locally. You have the choice of saving either individual images or entire groups. The export process can be started in two ways:

- **Context menu in the Solution Explorer:**
Right-click the desired item in the Solution Explorer. Several images can also be saved simultaneously via a multiselect in the Solution Explorer.
- **Annotation pop-up:**
Use the export button directly within the image view.

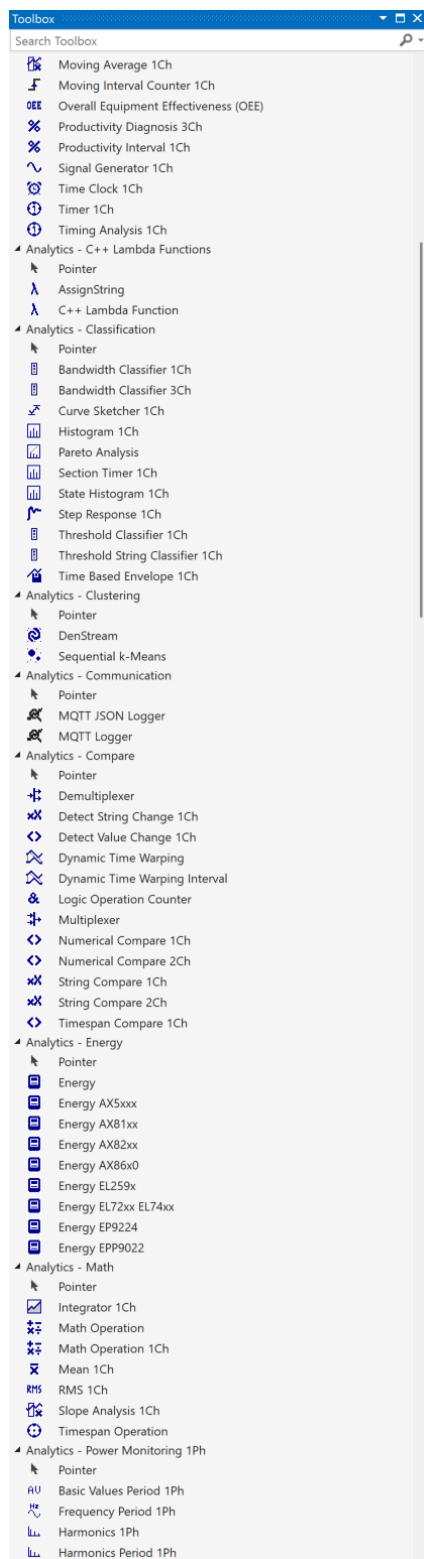
A distinction is made between four options when exporting:

- **Export Group**
Exports the entire group including all contained items. The result corresponds to the resolution configured in the Gallery.
- **Export Original Image**
Saves the original image in its full native resolution without additional overlays.
- **Export with Markups (full Resolution)**
Saves the image in full resolution, whereby all annotations (markups) created are drawn directly into the image.
- **Export with Title**
Saves the image together with the title and the line information below the representation. The resolution corresponds to the current display in the Gallery.

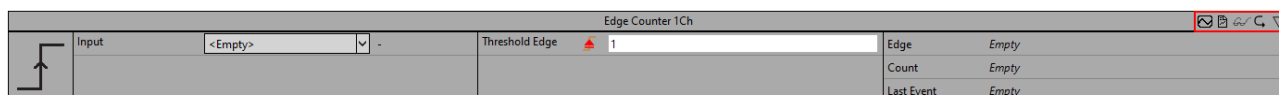
5.2 Algorithms

TwinCAT Analytics Engineering offers you a graphical user interface for easy use of mathematical algorithms from various categories. Depending on the installed workload and activated license, the algorithms of some or all categories are visible in the toolbox.

If none of the many algorithms fit your requirements, you can develop application-specific algorithms yourself using the TwinCAT Analytics Lambda function.



All algorithms have the same five icons in the top right-hand corner:



- **Include in Referenced Scope:** If you click on the *Included in Referenced Scope* icon, the algorithm outputs are added to a referenced Scope project.
- **Include in Report:** If you click on the *Include in Report* icon, the algorithm with the configured visualization will be integrated into a manual report.

- **Eyeglasses:** If you click on the eyeglasses icon you can see the optional parameter *Enable Execution*. You can select a Boolean signal for this parameter, so that the algorithm is just active, if the value of the selected signal is *TRUE*.
- **Reset arrow:** If you click on the arrow the output values of the specific algorithm will be reset.
- **Minimize arrow:** If you click on the minimize arrow on the right, the algorithm will be collapsed.

The various categories and their algorithms are described in more detail below.

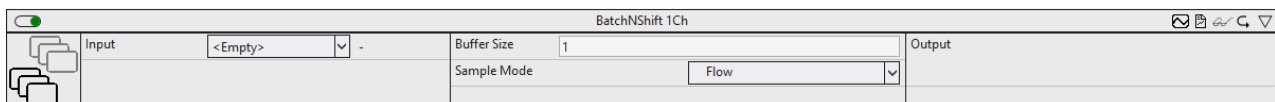
5.2.1 Analytics algorithms

The basic algorithms of the TwinCAT Analytics Engineering Tools originate from the PLC library TF3500 Analytics Library.

5.2.1.1 Analytics - Base

The algorithms of the category *Analytics-Base* provide base functionalities for analyzing process and application data. For example threshold detection, timing analysis or calculation of minimum, maximum and average values.

5.2.1.1.1 BatchNShift 1Ch



The *BatchNShift 1Ch* buffers the values of the input signal according to the buffer size and the sample mode. The number of output channels in which the buffered input values are stored corresponds to the buffer size. With the help of the sample mode it is possible to distinguish between two different operating modes of the algorithm. If the sample mode *Flow* is selected, a ring buffer or shift register is realized (Shift). The values are written to the buffer one after the other and shifted by one position of the buffer in each cycle. If the buffer is full, the last value falls out. In the *Wait* mode, the buffer is instead completely emptied and filled with new values whenever it is completely full, so that the values are processed in the form of batches (batch). At the beginning of an analysis, the system also waits until the buffer is completely filled before writing the values to the buffer. Therefore, the function block supplies valid values only from the cycle (*BufferSize + 1*).

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Buffer Size:** specifies the size of the buffer and thus the number of values that are stored. The number of output channels equals the buffer size.
- **Sample Mode:** the values from the buffer can be passed to the output channels in two different modes:
 - *Flow:* the buffer is filled like a ring buffer. At the start of the analysis all output values are set to zero. Each change to the ring buffer is transferred to the output channels immediately. The New Result flag is set to *TRUE*, once all output channels got assigned a value and is always true, when a new value is saved in the buffer.
 - *Wait:* at the start of the analysis or after reset all output channels are set to zero. Only when the internal buffer is full, these values are transferred to the output channels and the New Result flag is set to *TRUE*. These values stay as output values until all the values in the internal buffer are renewed. Only then they are transferred to the output channels.

Output values

- **Output Value 00..n:** results of the BatchNShift buffer according to the Sample Mode. Each output channel represents a buffer storage space.

Standard HMI Controls

For the *BatchNShift 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control or Multivalue Control visualizes the output values *Output Value 00..n*

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the *BatchNShift 1Ch* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.2 Bitmask

Bitmask			
1 0	Input	<Empty>	-
0 1			

The *Bitmask* module can be used to evaluate individual bits. To do this, the individual bits of the input value are output as Boolean values at the output.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

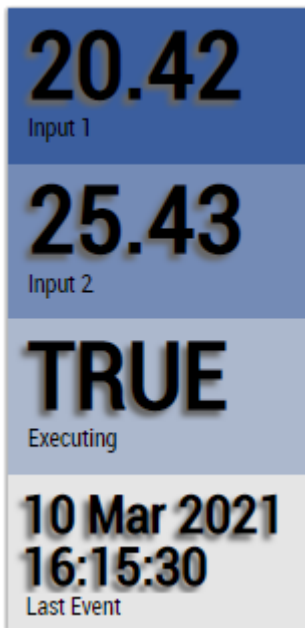
- **Bits:** Array with Boolean elements. The value of an element is *TRUE* if the respective bit in the input value is set. The size of the output array depends on the data type of the linked input.

The following human-machine interface controls are available for the algorithm to generate an Analytics Dashboard:

The Table Control or List Control visualizes all output values.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be linked using the [Mapping Wizard](#) [► 382].

5.2.1.1.3 Calendar

Calendar			
	Timestamp	Recurrence Mode	Daily
		Time	hh 1 mm 0 ss 0
		Event	Empty
		Next Event	Empty

The *Calendar* creates a recurring event at certain times. The repeat mode can be set individually. The modes of daily, weekly, monthly or annual repetition are available.

A timestamp is required as a reference value, as the algorithm needs a time context in which to work. This reference timestamp is set automatically, if there is another algorithm in the configuration. Therefore, it is not possible to use the *Calendar* individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Recurrence Mode:**
 - Daily:* Daily recurrence. The time of the event can be set.
 - Weekly:* Weekly recurrence. The time of the event and the days of the week on which the event is to be generated can be set.
 - Monthly:* Monthly recurrence. The time of the event and the recurrence type can be set.
 - Yearly:* Annual recurrence. The time of the event, the recurrence type and the months in which the event is to be created can be set.
- **Time:** Time of the event
- **Day Of Week Mask:** Days of the week on which an event is to be created.
- **Recurrence Specification:**
 - Specific Day:* The event should be created on a specific day. This can be set using the *Day* parameter.
 - First:* The event should be created on the first specified day of the month. The specification is made via the *Day Of Week* parameter.
 - Last:* The event should be created on the last specified day of the month. Specification takes place via the *Day Of Week* parameter
- **Day:** Day on which an event is to be created.
- **Day Of Week:** Day of the week of the event
 - Unspecified:* The event is created on the first or last day of the month, depending on the *Recurrence Specification* parameter.
 - Monday..Sunday:* The event is generated on the first or last set weekday of the month, depending on the *Recurrence Specification* parameter.

- **Month Of Year Mask:** Months in which an event is to be created.

Output values

- **Event:** Indicates *TRUE* at the time of the event, otherwise *FALSE*.
- **Next Event:** Displays the time remaining until the next event.

Standard HMI Controls

For the Calendar algorithm, the following HMI controls are available for generating an Analytics Dashboard:

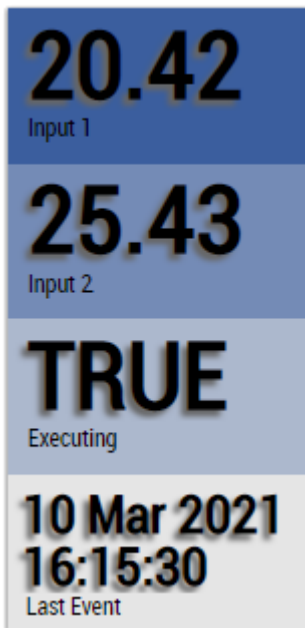
1. The Table Control or Multivalue Control visualizes all output values: Event, Next Event.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

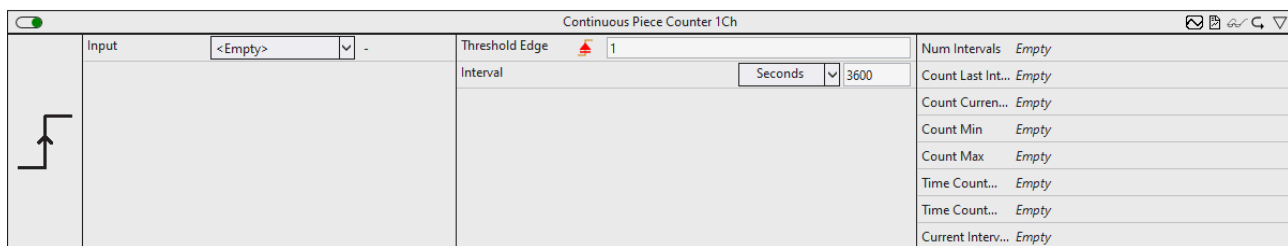
DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the *Calendar* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.4 Continuous Piece Counter 1Ch



The *Continuous Piece Counter 1Ch* counts the number of pieces within the configured interval. The counter is increased when the signal of the input channel passes the configured edge at a specific threshold. The calculation restarts when the time of the interval has elapsed. The algorithm provides the amount of pieces, the minimal and the maximal number of pieces as well as the time values of minimum and maximum.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The counter increments when the signal passes this threshold.
- **Interval:** Time interval in which the values are to be calculated.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

Output Values

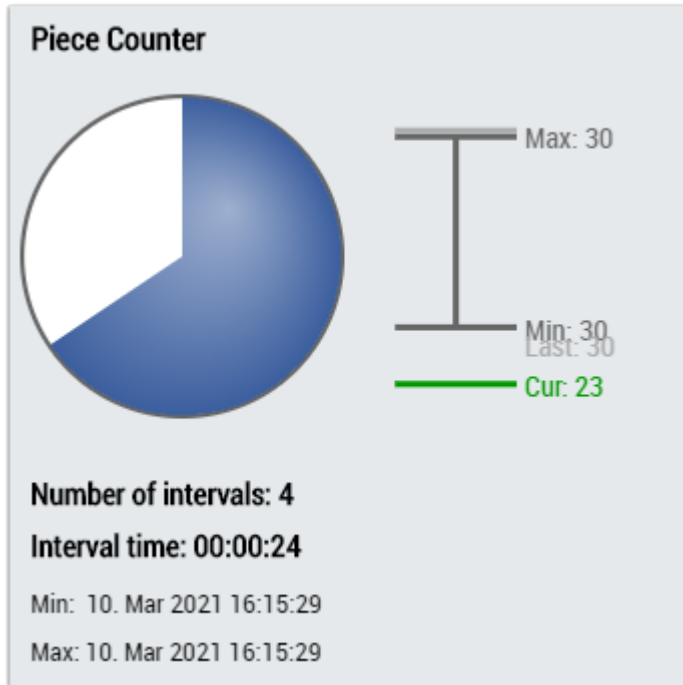
- **Num Interval:** Shows the number of intervals.
- **Count Last Interval:** Shows the amount of pieces in the last interval.
- **Count Current Interval:** Shows the amount of pieces in the current interval.
- **Count Min:** Shows the minimal number of pieces in an interval.
- **Count Max:** Shows the maximal number of pieces in an interval.
- **Time Count Min:** Shows the time value of the minimum.
- **Time Count Max:** Shows the time value of the maximum.

- **Current Interval Time:** Shows the time of the current interval.

Standard HMI Controls

For the Continuous Piece Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

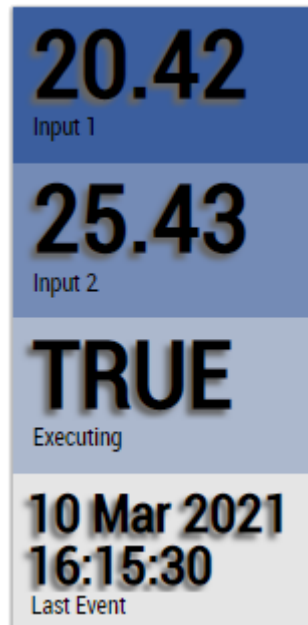
1. The PieceCounter control visualizes the output values Num Intervals, Count Last Interval, Count Current Interval, Count Min, Count Max, Time Count Min, Time Count Max and Time Current Interval.



2. The Table Control or Multivalue Control visualizes all output values: Num Intervals, Count Last Interval, Count Current Interval, Count Min, Count Max, Time Count Min, Time Count Max, Time Current Interval.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Continuous Piece Counter 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.5 Downsampling 1Ch

Downsampling 1Ch				
	Input	<Empty>	Downsampling Factor	1
			Out	Empty
			Time Last Sa...	Empty

Downsampling 1Ch processes the values of the input channel with a configurable downsampling factor. This achieves downsampling so that the output signal is a representation of the input signal at a lower sampling rate. This can be useful, for example, to better identify trends or to perform subsequent compression of highly sampled signals if only lower sampling rates are required within the analysis. This is a simple way to increase the performance of the analysis.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Downsampling Factor:** the factor used for downsampling. For example, if the downsampling factor is 100, only every 100th value is saved. The sample time is thus 100 times the original cycle time, which lies between two sampled data points. If the downsampling factor is set to one, all values are buffered.

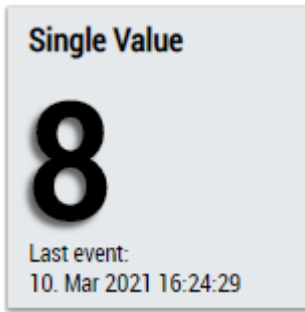
Output values

- **Output Value:** output signal with the sampling rate lower by the downsampling factor.

Standard HMI Controls

For the *Downsampling 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SingleValue Control visualizes the output value *Output Value*.



Alternatively, customer-specific HMI controls can be mapped in the *Downsampling 1Ch* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.6 Edge Counter 1Ch

Edge Counter 1Ch			
Input	<Empty>	Threshold Edge	1
		Edge	Empty
		Count	Empty
		Last Event	Empty

The *Edge Counter 1Ch* counts the amount of raised events. An event is raised when the signal of the input channel passes the configured edge at a specific threshold.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

Output values

- **Edge:** Indicates *TRUE* at the time the event is triggered, otherwise *FALSE*.
- **Count:** Counts the number of triggered events.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the Edge Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SingleValue control visualizes the output values Count and Last Event.



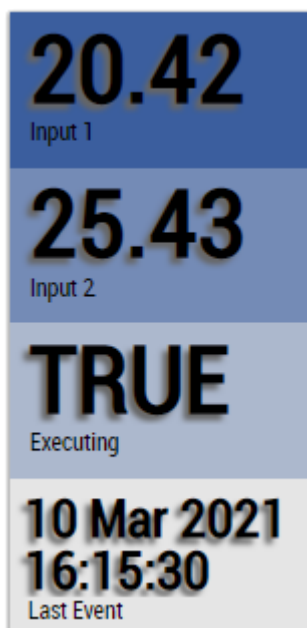
2. The Table Control or Multivalue Control visualizes all output values: Edge, Count, Last Event.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, custom HMI controls can be mapped in the Edge Counter 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.7 Edge Counter On Off 1Ch

Edge Counter OnOff 1Ch			
	Input	<Empty>	-
	Threshold Edge On	1	
	Threshold Edge Off	1	
	Is On	Empty	
	Edge On	Empty	
	Edge Off	Empty	
	Count On	Empty	
	Count Off	Empty	
	Last Event	Empty	

The *Edge Counter On Off 1Ch* counts the amount of raised on- and off-events. An on-event is raised when the signal of the input channel passes the configured edge at a specific on-threshold and an off-event is raised when the off-threshold is passed by the same signal.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold On:** Threshold of the signal at the respective edge. The On event is triggered when the signal passes this threshold.
- **Threshold Off:** Threshold of the signal at the respective edge. The Off event is triggered when the signal passes this threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

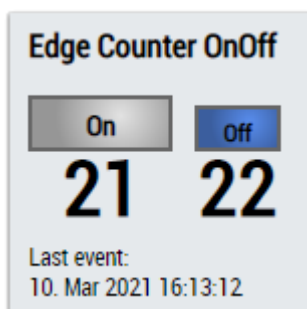
Output values

- **Is On:** Indicates *TRUE* within the timespan between the On event and the Off result, otherwise *FALSE*.
- **Edge On:** Indicates *TRUE* on a rising edge.
- **Edge Off:** Indicates *TRUE* on a falling edge.
- **Count On:** Counts the number of triggered On events.
- **Count Off:** Counts the number of triggered Off events.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

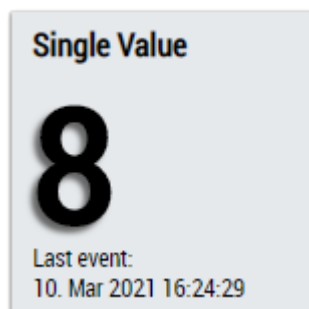
Standard HMI Controls

For the Edge Counter On Off 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

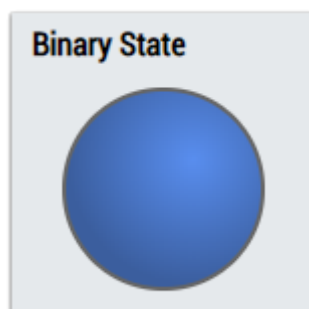
1. The EdgeCounterOnOff control visualizes the output values Is On, Count On, Count Off and Last Event.



2. The SingleValue control visualizes the output values Count On and Last Event.



3. The BinaryState control visualizes the output value Is On.



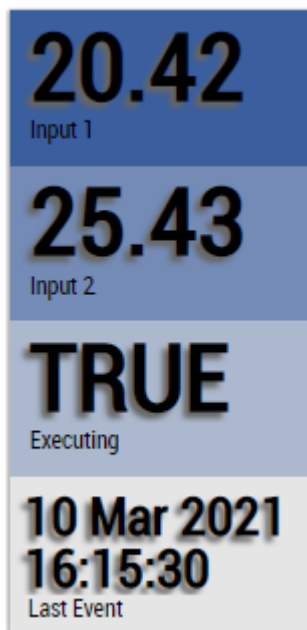
4. The Table Control or Multivalue Control visualizes all output values: Flanks (Is On, Edge On, Edge Off), Count On, Count Off, Last Event.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, custom HMI controls can be mapped in the Edge Counter On Off 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.8 Edge Counter On Off 2Ch

Edge Counter OnOff 2Ch									
	Input Ch1	<Empty>	▼	-	Threshold Edge On	1	Is On	Empty	
	Input Ch2	<Empty>	▼	-	Reset On Multiple On	☐	Edge On	Empty	
					Threshold Edge Off	1	Edge Off	Empty	
							Count On	Empty	
							Count Off	Empty	
							Last Event	Empty	

The *Edge Counter On Off 2Ch* counts the amount of raised on- and off-events. An on-event is raised when the signal of the first input channel passes the configured edge at a specific on-threshold and an off-event is raised when the off-threshold is passed by the signal of the second channel.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold On:** Threshold of the signal at the respective edge. The On event is triggered when the signal passes this threshold.
- **Reset On Multiple On:** If the checkbox is checked, the "Count On" counter increments with every On event. Otherwise, the On events are only counted after a counter reset (Off event).
- **Threshold Off:** Threshold of the signal at the respective edge. The Off event is triggered when the signal passes this threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

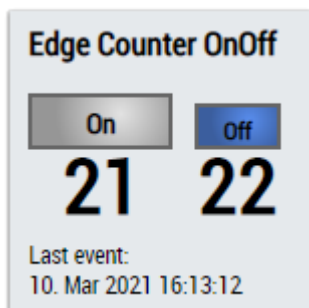
Output values

- **Is On:** Indicates *TRUE* within the timespan between the On event and the Off result, otherwise *FALSE*.
- **Edge On:** Indicates *TRUE* on a rising edge.
- **Edge Off:** Indicates *TRUE* on a falling edge.
- **Count On:** Counts the number of triggered On events.
- **Count Off:** Counts the number of triggered Off events.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the Edge Counter On Off 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

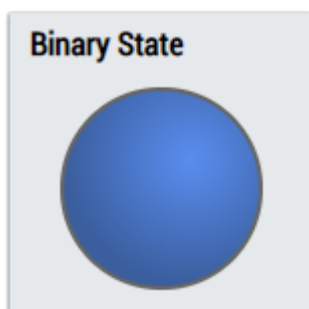
1. The EdgeCounterOnOff control visualizes the output values Is On, Count On, Count Off and Last Event.



2. The SingleValue control visualizes the output values Count On and Last Event.



3. The BinaryState control visualizes the output value Is On.



4. The Table Control or Multivalue Control visualizes all output values: Flanks (Is On, Edge On, Edge Off), Count On, Count Off, Last Event.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, custom HMI controls can be mapped in the Edge Counter On Off 2Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.9 Event Timing Analysis 1Ch

Event Timing Analysis 1Ch			
	Input	<Empty>	-
	Threshold Edge On		1
	Threshold Edge Off		1
	Init With Threshold	☐	
	Single Edge	☐	
	Is On	Empty	
	Current Interval	Empty	
	Last On Interval	Empty	
	Last Off Interval	Empty	
	On Min	Empty	
On Max	Empty		
On Avg	Empty		
On Total	Empty		
Off Min	Empty		
Off Max	Empty		
Off Avg	Empty		
Off Total	Empty		
Count On	Empty		

The *Event Timing Analysis 1Ch* measures time differences between on- and off-event and counts the amount of raised events. An on-event is raised when the signal of the input channel passes the configured edge at a specific on-threshold and an off-event is raised when the off-threshold is passed by the same signal.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** indicates whether the edge counter should react to a rising or a falling edge.
- **Threshold On:** threshold of the signal at the respective edge. The On event is triggered when the signal passes this threshold.
- **Threshold Off:** threshold of the signal at the respective edge. The Off event is triggered when the signal passes this threshold.
- **Init With Threshold:** if the value is *TRUE*, the algorithm uses a threshold to initialize the internal state instead of waiting for an edge.
- **Single Edge:** if the value is *TRUE*, the algorithm is executed only on the basis of the edges of the On event.
- **Tolerance (optional):** tolerance value for the Equal / NotEqual comparisons.

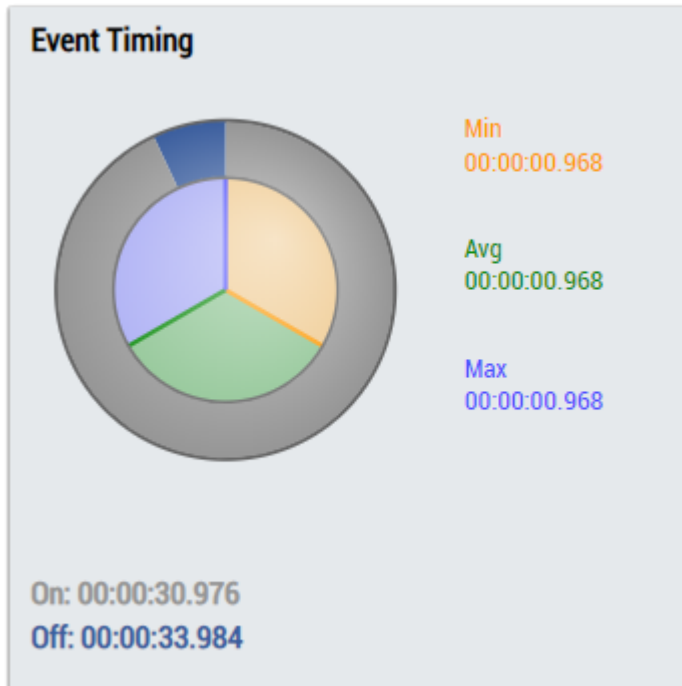
Output values

- **Is On:** Indicates *TRUE* within the timespan between the On event and the Off result, otherwise *FALSE*.
- **Current Interval:** Indicates the time of the current interval.
- **Last On Interval:** Timespan of the last completed interval between On event and Off event.
- **Last Off Interval:** Timespan of the last completed interval between Off event and the On event.
- **On Min:** Minimum time between On event and Off event.
- **On Max:** Maximum time between On event and Off event.
- **On Avg:** Average time between On event and Off event.
- **On Total:** Total time between On events and Off events.
- **Off Min:** Minimum time between Off event and On event.
- **Off Max:** Maximum time between Off event and On event.
- **Off Avg:** Average time between Off event and On event.
- **Off Total:** Total time between Off events and On events.
- **Count On:** Counts the number of triggered On events.

Standard HMI Controls

For the Event Timing Analysis 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The EventTiming control visualizes the output values Is On, Count On, Current Interval, On Min, On Max, On Avg, On Total, Off Min, Off Max, Off Avg and Off Total.



2. The SingleValue control visualizes the output value Count On.



3. The Table Control or Multivalue Control visualizes all output values: Is On, Count On, Current Interval, On Min, On Max, On Avg, On Total, Off Min, Off Max, Off Avg, Off Total.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42
Input 1

25.43
Input 2

TRUE
Executing

10 Mar 2021
16:15:30
Last Event

Alternatively, customer-specific HMI controls can be mapped in the Event Timing Analysis 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.10 Event Timing Analysis 2Ch

Event Timing Analysis 2Ch				
	Input Ch1	<Empty>	-	
	Input Ch2	<Empty>	-	
	Threshold Edge On	 1	Is On	Empty
	Reset On Multiple On	☐	Current Interval	Empty
	Threshold Edge Off	 1	Last On Interval	Empty
	Init With Threshold	☐	Last Off Interval	Empty
			On Min	Empty
			On Max	Empty
			On Avg	Empty
			On Total	Empty
		Off Min	Empty	
		Off Max	Empty	
		Off Avg	Empty	
		Off Total	Empty	
		Count On	Empty	

The *Event Timing Analysis 2Ch* measures time differences between on- and off-event and counts the amount of raised events. An on-event is raised when the signal of the first input channel passes the configured edge at a specific on-threshold and an off-event is raised when the off-threshold is passed by the signal of the second channel.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold On:** Threshold of the signal at the respective edge. The On event is triggered when the signal passes this threshold.

- **Reset On Multiple On:** If the checkbox is checked, the "Count On" counter increments with every On event. Otherwise, the On events are only counted after a counter reset (Off event).
- **Threshold Off:** Threshold of the signal at the respective edge. The Off event is triggered when the signal passes this threshold.
- **Init With Threshold:** If the value is TRUE, the algorithm uses a threshold to initialize the internal state, instead of waiting for an edge.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

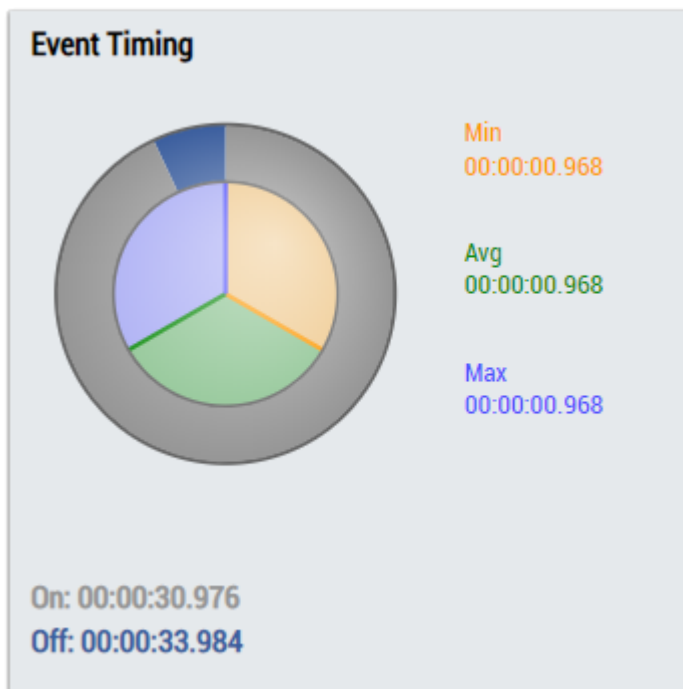
Output values

- **Is On:** Indicates *TRUE* within the timespan between the On event and the Off result, otherwise *FALSE*.
- **Current Interval:** Indicates the time of the current interval.
- **Last On Interval:** Timespan of the last completed interval between On event and Off event.
- **Last Off Interval:** Timespan of the last completed interval between Off event and the On event.
- **On Min:** Minimum time between On event and Off event.
- **On Max:** Maximum time between On event and Off event.
- **On Avg:** Average time between On event and Off event.
- **On Total:** Total time between On events and Off events.
- **Off Min:** Minimum time between Off event and On event.
- **Off Max:** Maximum time between Off event and On event.
- **Off Avg:** Average time between Off event and On event.
- **Off Total:** Total time between Off events and On events.
- **Count On:** Counts the number of triggered On events.

Standard HMI Controls

For the Event Timing Analysis 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The EventTiming control visualizes the output values Is On, Count On, Current Interval, On Min, On Max, On Avg, On Total, Off Min, Off Max, Off Avg and Off Total.



2. The SingleValue control visualizes the output value Count On.

Single Value

8

Last event:
10. Mar 2021 16:24:29

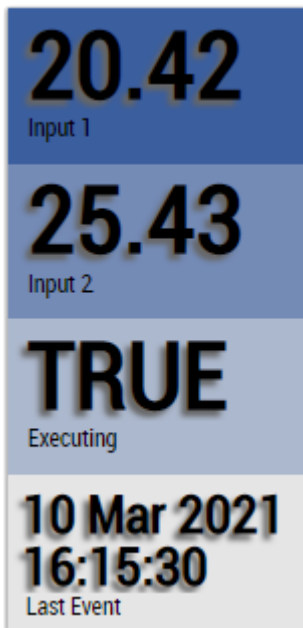
3. The Table Control or Multivalue Control visualizes all output values: Is On, Count On, Current Interval, On Min, On Max, On Avg, On Total, Off Min, Off Max, Off Avg, Off Total.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Event Timing Analysis 2Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.11 Flip Flop 2Ch

Flip Flop 2Ch							
RS SR	Input S	<Empty>	-	Threshold Level S	▲ 1	Out	Empty
	Input R	<Empty>	-	Threshold Level R	▲ 1	Count	Empty
				Count Mode	OnChange	Last Event	Empty
				S Is Dominant	☒		

The *Flip Flop 2Ch* implements a bistable flip-flop. The dominance for setting (RS) or resetting (SR) the output value can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Operator:** specifies whether the input value should be greater than, greater than or equal to, less than or equal to, less than or not equal to the threshold.
- **Threshold Level S:** threshold of the signal for setting.
- **Threshold Level R:** threshold of the signal for resetting.
- **Count Mode:** mode of the result counter.
OnChange: the counter counts every time the result changes to *TRUE*.
Cyclic: the counter increments every cycle when the condition is *TRUE*.
- **S Is Dominant:** the dominance of setting (RS) or resetting (SR) can be configured.
- **Tolerance (optional):** tolerance value for the Equal / NotEqual comparisons.

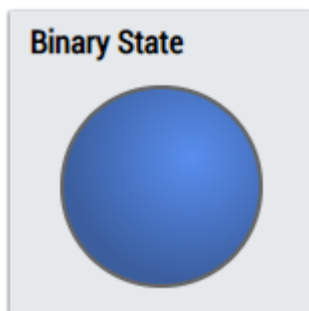
Output values

- **Out:** Result of the bistable flip-flop.
- **Count:** Incremented when the output value is *TRUE*. The behavior depends on the parameter *Count Mode*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the *Flip Flop 2Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The BinaryState control visualizes the output value *Out*.



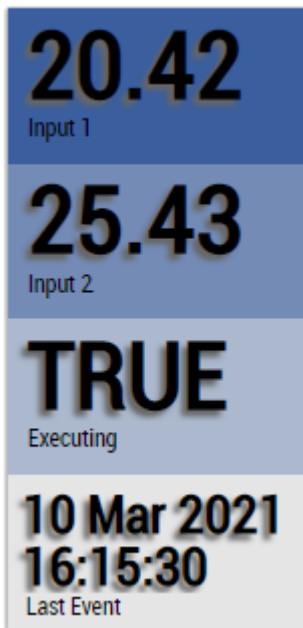
2. The Table Control and Multivalue control visualize all output values: *Out*, *Count*, *Last Event*.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the *Flip Flop 2Ch* algorithm using the Mapping Wizard [► 382].

5.2.1.1.12 Interval Piece Counter 1Ch

Interval Piece Counter 1Ch			
	Input	<Empty>	-
	Start Interval	<Empty>	-
	Threshold Edge		1
	Reset On Multiple Start		☐
Interval		Seconds	1
Executing Int...		Empty	
Num Intervals		Empty	
Count Last Int...		Empty	
Count Curren...		Empty	
Count Min		Empty	
Count Max		Empty	
Time Count...		Empty	
Time Count...		Empty	
Current Interv...		Empty	

The *Interval Piece Counter 1Ch* counts the amount of raised events within a configured interval, which starts when the value of the start interval flag is *TRUE*. An event is raised when the signal of the input channel passes the configured edge at a specific threshold. The calculation restarts when the time of the interval has elapsed and the value of the start interval flag is *True* again.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.
- **Reset On Multiple Start:** If the checkbox is checked, the interval restarts when the Start Interval flag becomes *TRUE* again. Otherwise, the interval restarts automatically when the time has elapsed.
- **Interval:** Time interval in which the values are to be calculated.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

Output values

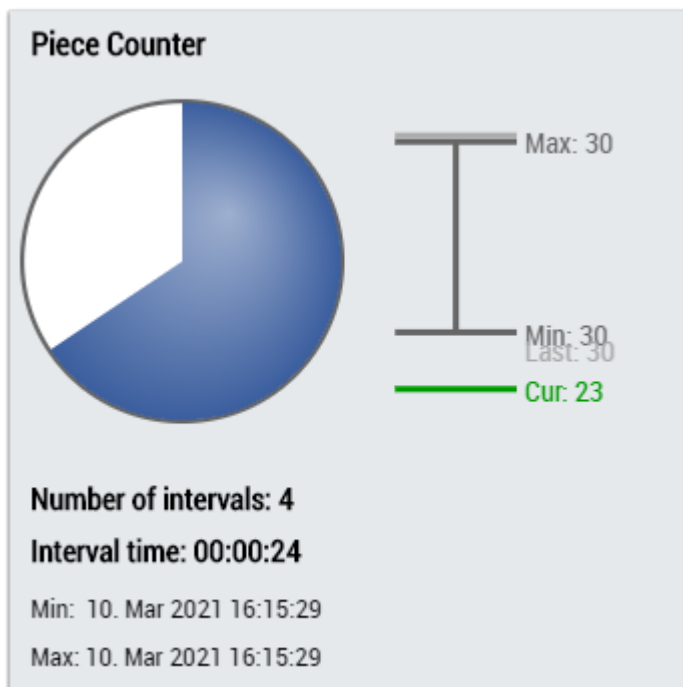
- **Executing Interval:** Indicates *True* if the calculation is active and the interval is running, otherwise *False*.
- **Num Intervals:** Indicates the number of intervals.
- **Count Last Interval:** Indicates the number of triggered events in the last interval.

- **Count Current Interval:** Indicates the number of triggered events in the current interval or, if the calculation is currently inactive, the number of triggered events in the last interval.
- **Count Min:** Indicates the minimum of triggered events in an interval.
- **Count Max:** Indicates the maximum of triggered events in an interval.
- **Time Count Min:** Indicates the time value of the minimum → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Time Count Max:** Indicates the time value of the maximum → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Current Interval Time:** Indicates the time of the current interval → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the Interval Piece Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The PieceCounter control visualizes the output values Num Intervals, Count Last Interval, Count Current Interval, Count Min, Count Max, Time Count Min, Time Count Max and Time Current Interval.



2. The Table Control or Multivalue Control visualizes all output values: Num Intervals, Count Last Interval, Count Current Interval, Count Min, Count Max, Time Count Min, Time Count Max, Time Current Interval.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021 16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Interval Piece Counter 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.13 Latching Switch 1Ch

Latching Switch 1Ch				
	Input	<Empty>	Threshold Edge	1
	Out	Empty	Count	Empty
	Last Event	Empty		

The *Latching Switch 1Ch* realizes a virtual impulse switch. The output alternates between TRUE and FALSE on each edge detected at the input.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether to react to a rising or falling edge at the input
- **Threshold Edge:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

Output values

- **Out:** Result of the virtual impulse switch.
- **Count:** incremented when the output value changes.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

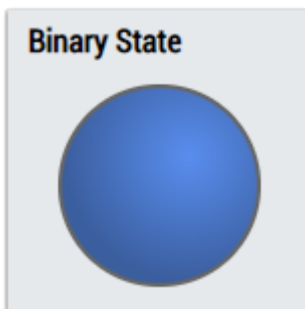
Standard HMI Controls

For the *Latching Switch 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SingleValue control visualizes the output values Count and Last Event.



2. The BinaryState control visualizes the output value *Out*.



3. The Table Control and Multivalue Control visualize all output values.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021 16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped using the [Mapping Wizard](#) [► 382].

5.2.1.1.14 Lifecycle Analysis 1Ch

Lifecycle Analysis 1Ch				
	Input	<Empty>	Threshold Edge	1
			Estimated Cycles	1000
			Cycles Elapsed	Empty
			Cycles Remai...	Empty

The *Lifecycle Analysis 1Ch* calculates the elapsed and the estimated remaining cycles of a device. When the signal of the input channel passes the configured edge at a specific threshold, the elapsed cycles are increased and the remaining cycles are decreased.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. An event is triggered when the signal passes this threshold.
- **Estimated Cycles:** Estimated cycles over the lifetime of the respective device.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

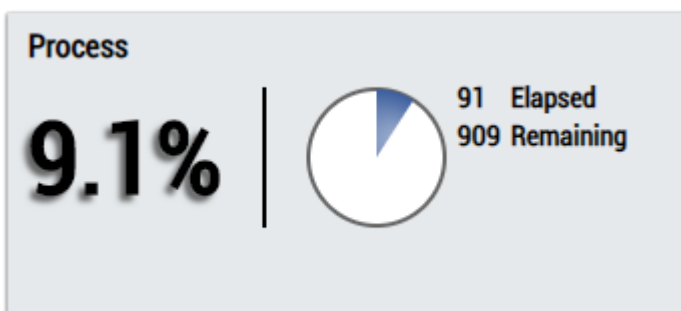
Output Values

- **Elapsed Cycles:** Counts the amount of cycles which are already elapsed.
- **Remaining Cycles:** Shows the remaining cycles of the device as the difference of estimated and elapsed cycles.

Standard HMI Controls

For the Lifecycle Analysis 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Process control visualizes the output values Elapsed Cycles and Remaining Cycles.



2. The Table Control or Multivalue Control visualizes all output values: Elapsed Cycles, Remaining Cycles.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the Lifecycle Analysis 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.15 Lifetime Analysis 1Ch

Lifetime Analysis 1Ch				
	Input	<Empty>	Threshold Level	1
			Estimated Lifetime	Seconds 3600
				Time Elapsed Empty
				Time Remains... Empty

The *Lifetime Analysis 1Ch* calculates the elapsed and the estimated remaining lifetime of a device. If the input value met the configured condition the lifetime will be reduced.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Operator:** Specifies whether the input value should be greater than, greater than or equal to, less than or equal to, less than or not equal to the threshold.
- **Threshold:** Signal threshold.
- **Estimated Lifetime:** Estimated lifetime of the respective device.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

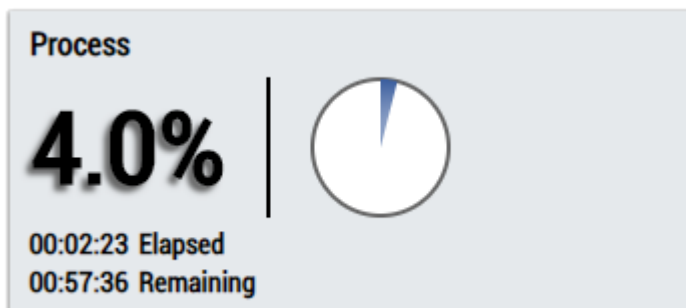
Output Values

- **Elapsed Lifetime:** Shows the lifetime which is already elapsed.
- **Remaining Lifetime:** Shows the remaining lifetime of the device as the difference of estimated and elapsed lifetime.

Standard HMI Controls

For the Lifetime Analysis 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Process control visualizes the output values Elapsed Lifetime and Remaining Lifetime.



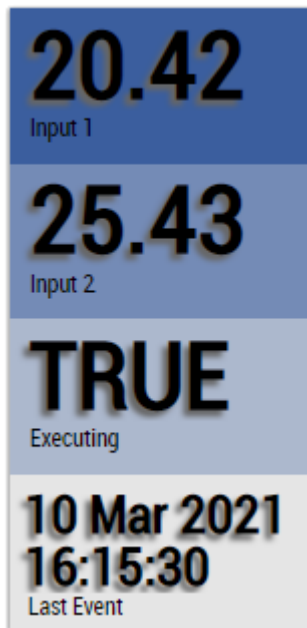
2. The Table Control or Multivalue Control visualizes all output values: Elapsed Lifetime, Remaining Lifetime.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Lifetime Analysis 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.16 Min Max Avg

Min Max Avg												
Input 00	<Empty>	-	Num Channels	1	+	-		Min	Max	Avg	Time...	Time...
							Overall Re...	Empty	Empty	Empty	Empty	Empty
							Min					
							Max					
							Avg					

The *Min Max Avg* calculates the minimum, maximum and the average of the input values from the beginning of the analysis up to the current moment. Compared to [Min Max Avg 1Ch](#) [► 69], any number of input channels are supported. The calculations are performed separately for each input channel. Additional output values are also calculated across all input channels.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** The number of operands that are inputs to the function.

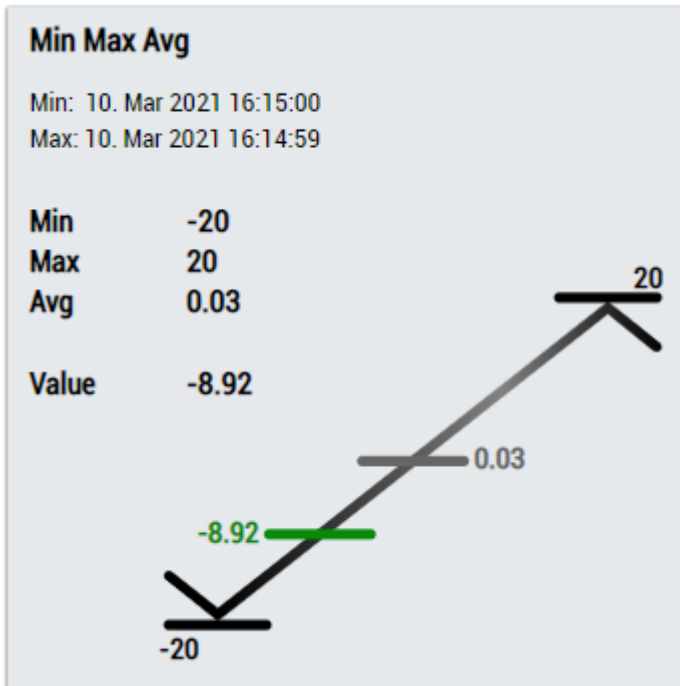
Output values

- **Overall Result:** Result across all input channels
 - **Min:** Indicates the minimum of the input values.
 - **Max:** Indicates the maximum of the input values.
 - **Avg:** Indicates the average of the input values.
 - **Time Min:** Indicates the time value of the minimum → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
 - **Time Max:** Indicates the time value of the maximum → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Min:** Indicates the minimum of the input values per channel.
- **Max:** Indicates the maximum of the input values per channel.
- **Avg:** Indicates the average of the input values per channel.

Standard HMI Controls

For the algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Average Chart Control visualizes the output values Min, Max, Avg, Time Min and Time Max.



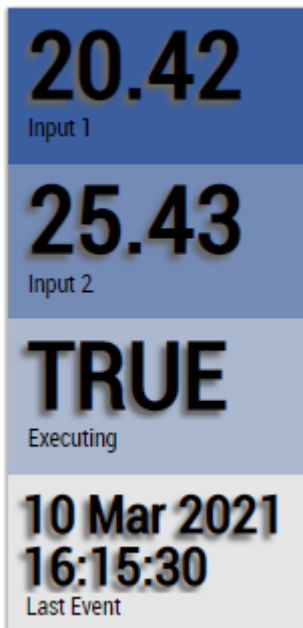
2. The Table Control or List Control visualizes all output values.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



5.2.1.1.17 Min Max Avg 1Ch

Min Max Avg 1Ch			
Input	<Empty>	Min	Empty
		Max	Empty
		Avg	Empty
		Time Min	Empty
		Time Max	Empty

The *Min Max Avg 1Ch* calculates the minimum, maximum and the average of the input values from the beginning of the analysis up to the current moment. Furthermore, the time values of minimum and maximum are shown.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

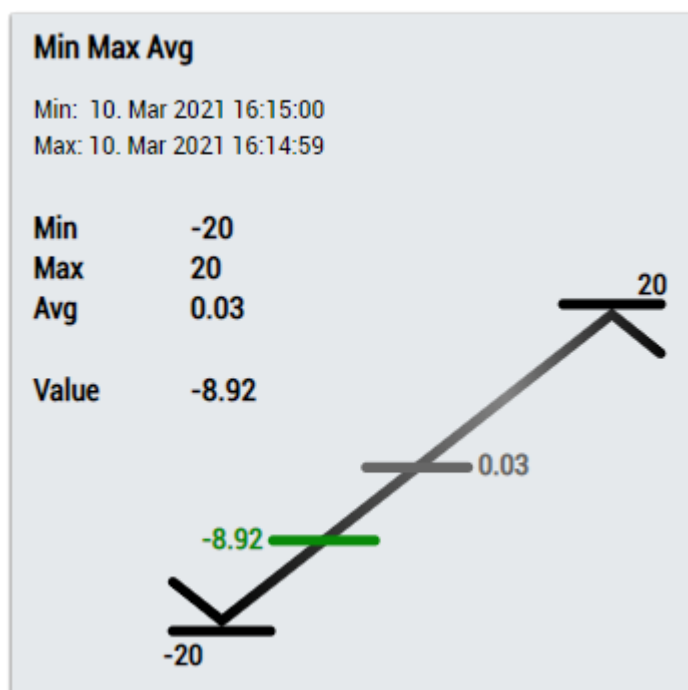
Output values

- **Min:** Indicates the minimum of the input values.
- **Max:** Indicates the maximum of the input values.
- **Avg:** Indicates the average of the input values.
- **Time Min:** Indicates the time value of the minimum → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Time Max:** Indicates the time value of the maximum → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

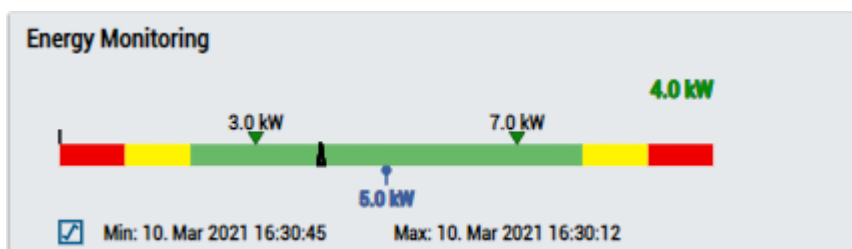
Standard HMI Controls

For the Min Max Avg 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

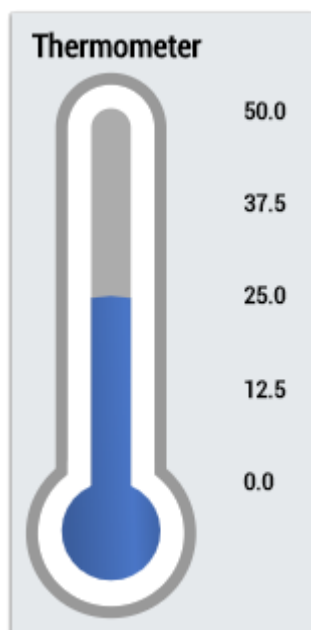
1. The MinMaxAvg control visualizes the output values Min, Max, Avg, Time Min and Time Max as well as the input value of the data.



2. The EnergyMonitoring control visualizes all output values: Min, Max, Avg, Time Min, Time Max and the current input value.



3. The Thermometer control visualizes the average (Avg) temperature.



4. The Table Control or Multivalue Control visualizes all output values: Min, Max, Avg, Time Min, Time Max.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Min Max Avg 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.18 Min Max Avg Interval 1Ch

Min Max Avg Interval 1Ch			
	Input:	<Empty>	-
	Interval:		Seconds 1
	Min	Empty	
	Max	Empty	
	Avg	Empty	
	Time Min	Empty	
	Time Max	Empty	
Current Interv...		Empty	

The *Min Max Avg Interval 1Ch* calculates the minimum, maximum and the average of the input values for the time period of the configured Interval. Furthermore the time values of minimum and maximum are shown. Note that all values are from the relative last interval and that they will only be updated when the interval is over. The calculation restarts when the time of the interval has elapsed.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

If only individual samples within the configured time signal are to be included in the calculation of the output values, this can optionally be implemented via the Boolean input *Add Sample*. All input values are considered for which the input *Add Sample* has the value *TRUE*.

Configuration Options

- **Interval:** Time Interval in which the values should be calculated.

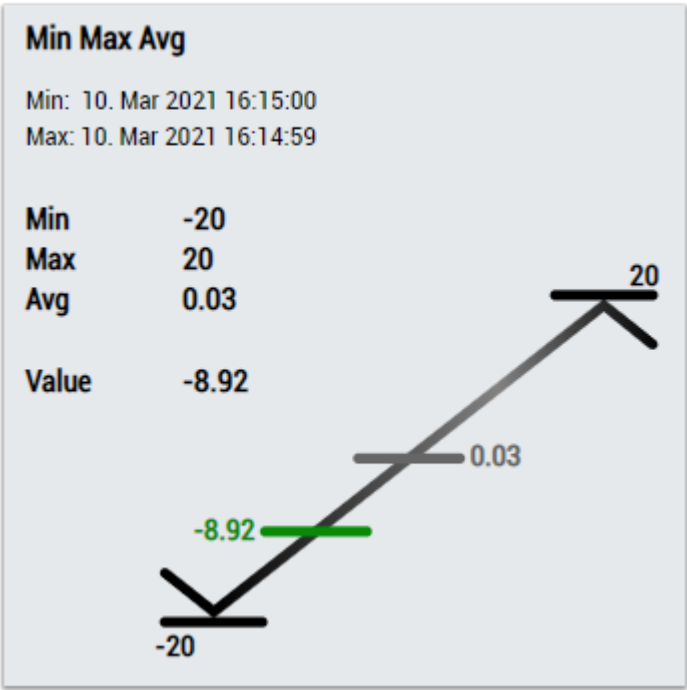
Output values

- **Min:** Indicates the minimum of the input values in the last time interval.
- **Max:** Indicates the maximum of the input values in the last time interval.
- **Avg:** Indicates the average of the input values in the last time interval.
- **Time Min:** Indicates the time value of the minimum in the last time interval → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Time Max:** Indicates the time value of the maximum in the last time interval → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Current Interval Time:** indicates the amount of time that has already elapsed from the current interval.

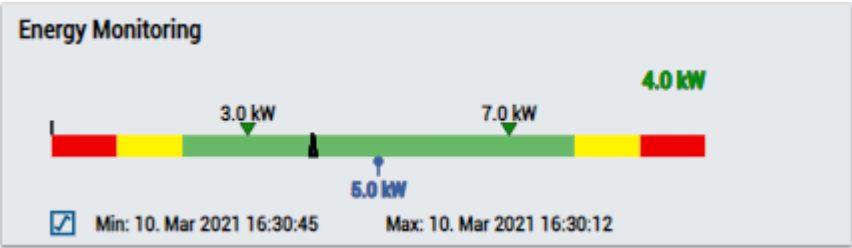
Standard HMI Controls

For the Min Max Avg Interval 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

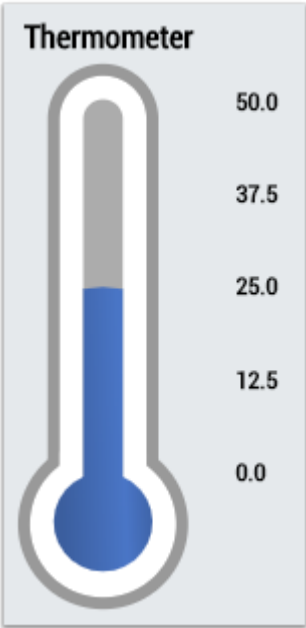
1. The MinMaxAvg control visualizes the output values Min, Max, Avg, Time Min and Time Max as well as the input value of the data.



2. The EnergyMonitoring control visualizes the output values: Min, Max, Avg, Time Min, Time Max and the current input value.



3. The Thermometer control visualizes the average (Avg) temperature.



4. The Table Control or Multivalue Control visualizes all output values: Min, Max, Avg, Time Min, Time Max, Current Interval Time.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Min Max Avg Interval 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.19 Moving Average 1Ch

Moving Average 1Ch				
	Input	<Empty>	Num Values	1
		-	Startup Behaviour	AvgOverExisting
				Moving Avg <i>Empty</i>
				Moving Min <i>Empty</i>
			Moving Max <i>Empty</i>	

The *Moving Average 1Ch* calculates the moving average, the minimum and the maximum of the most recent input values in an interval of specified length. Furthermore the time values of minimum and maximum are shown. The calculation of the moving average depends on the configuration parameters *Num Values* and *Startup Behaviour*.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Num Values:** Amount of values which will be included in the calculation of the moving average, the minimum and the maximum.
- **Startup Behaviour:** Calculation behaviour at the beginning of the analysis before at least *Num Values* input values exist.

ZeroPadding: The missing values are filled with zeros.

UseFirstValue: The first value is used until the amount of values is equivalent to *Num Values*.

WaitUntilFilled: The first result is calculated when the amount of values is equivalent to *Num Values*.

AvgOverExisting: The average will be calculated with the already existing values until the amount of values is equivalent to *Num Values*.

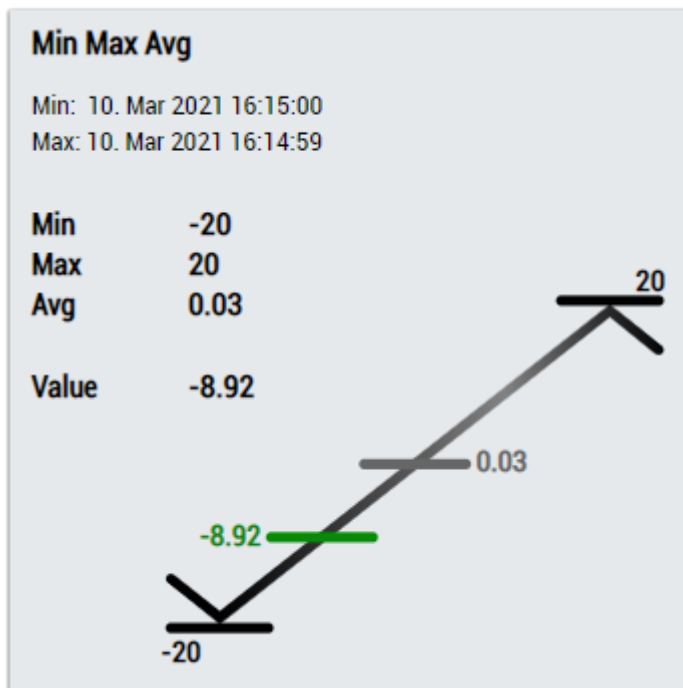
Output Values

- **Moving Avg:** Shows the current average value.
- **Moving Min:** Shows the minimum of the last n input values.
- **Moving Max:** Shows the maximum of the last n input values.

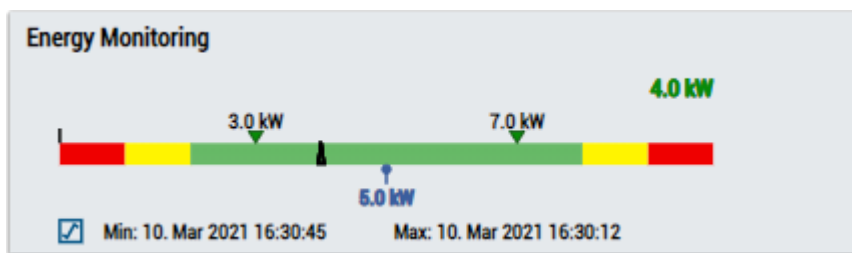
Standard HMI Controls

For the Moving Average 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

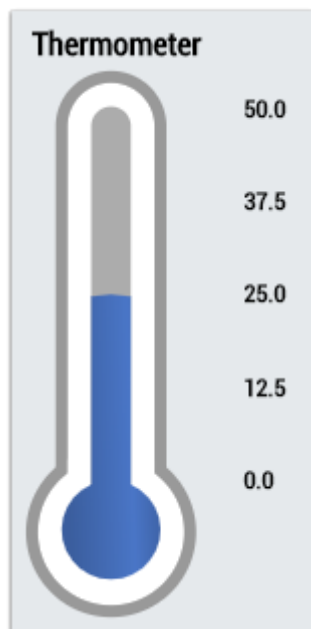
1. The MinMaxAvg control visualizes the output values Moving Min, Moving Max and Moving Avg as well as the input value of the data.



2. The EnergyMonitoring control visualizes the output values Moving Min, Moving Max and Moving Avg, and the input value of the data.



3. The Thermometer control visualizes the average (Moving Avg) temperature.



4. The Table Control or Multivalue Control visualizes all output values: Moving Min, Moving Max, Moving Avg.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the Moving Average 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.20 Moving Interval Counter 1Ch

Moving Interval Counter 1Ch										   		
	Input	<Empty>	-	Threshold Edge				1	Edge	Empty		
				Interval				Seconds	1	Limited	Empty	
				Count Limit			20		Counts In Inte...	Empty		
									Time First Co...	Empty		
									Time Last Co...	Empty		

The *Moving Interval Counter 1Ch* counts the amount of raised events within a configured interval. An event is raised when the signal of the input channel passes the configured edge at a specific threshold. The calculation restarts when the time of the interval has elapsed.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.
- **Interval:** Time interval in which the values are to be calculated.
- **Count Limit:** Limits the number of edges that can be counted in an interval.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

Output values

- **Edge:** Indicates *TRUE* at the time the event is triggered, otherwise *FALSE*.

- **Limited:** Indicates TRUE if the number of edges in the current interval exceeds the set Count Limit.
- **Counts in Interval:** Indicates the number of triggered events in the current interval.
- **Time First Count:** Indicates the time of the first triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Time Last Count:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the Moving Interval Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

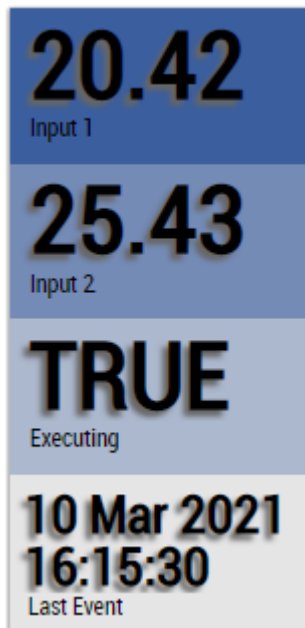
1. The MovingIntervalCounter control visualizes the output values Counts in Interval, Time First Count and Time Last Count.



2. The Table Control or Multivalue Control visualizes all output values: Edge, Counts in Interval, Time First Count, Time Last Count, Limited.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Moving Interval Counter 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.21 Overall Equipment Effectiveness (OEE)

Overall Equipment Effectiveness (OEE)									
%	Scheduled Time	<Empty>	-	Ideal Cycle Time		Seconds	1	OEE [%]	Empty
	Operating Time	<Empty>	-	Level OK / Warning	90			OEE Class	Empty
	Units Produced	<Empty>	-	Level Warning / Alarm	75			OEE Event Wa...	Empty
	Defective Units	<Empty>	-					OEE Event Ala...	Empty
								Availability [%]	Empty
								Performance...	Empty
								Quality [%]	Empty

The *Overall Equipment Effectiveness (OEE)* calculates key figures that make it possible to compare the current state of the manufacturing process with its maximum potential.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Ideal Cycle Time:** ideal cycle time for the production of one unit.
- **Level Ok / Warning:** the overall equipment effectiveness greater than the configured threshold is classified as *OK*. If the overall equipment effectiveness is less than or equal to the configured threshold, it is classified as *Warning*.
- **Level Warning / Alarm:** the overall equipment effectiveness greater than the configured threshold is classified as *Warning*. If the overall equipment effectiveness is less than or equal to the configured threshold, it is classified as *Alarm*.

Input values

- **Scheduled Time:** the operating time is calculated from the calendar time minus the scheduled non-production.
- **Operating Time:** the running time is calculated from the operating time minus the downtimes.
- **Units Produced:** corresponds to the number of units produced including defective units.
- **Defective Units:** corresponds to the number of defective units.

Output values

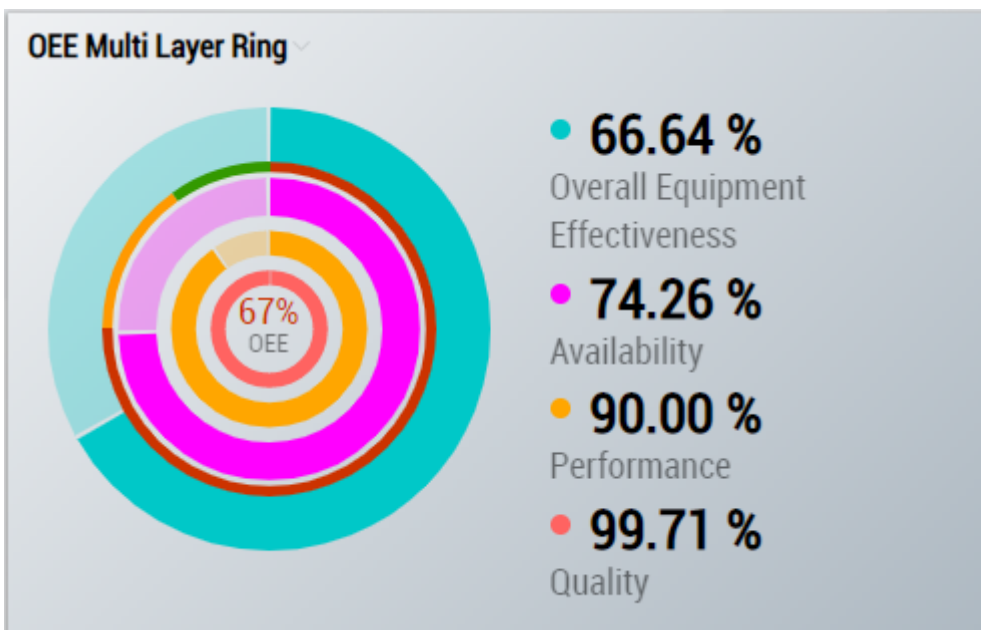
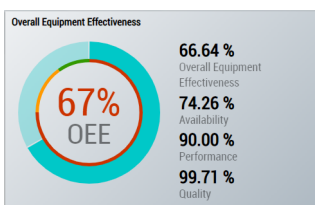
- **OEE:** Overall equipment effectiveness in percent. It is calculated by multiplying the availability factor, the performance factor and the quality factor.

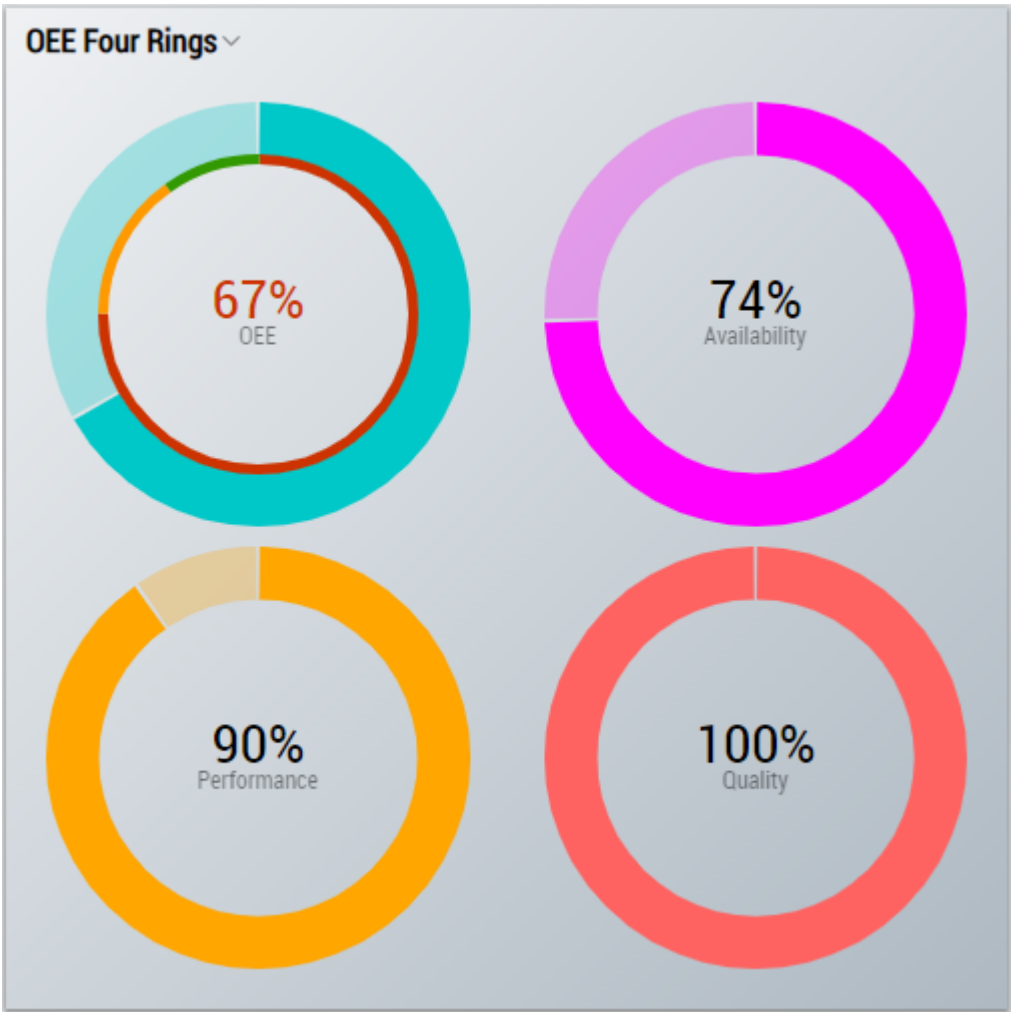
- **OEE Class:** Classification of overall equipment effectiveness.
- **OEE Event Warning:** Indicates the time of the last triggered warning event → the event can be dragged and dropped into the scope chart to display it as a trigger-event.
- **OEE Event Alarm:** Indicates the time of the last triggered alarm event → the event can be dragged and dropped into the scope chart to display it as a trigger-event.
- **Availability:** Availability factor in percent. It is calculated from the ratio between the runtime and the operating time.
- **Performance:** Performance factor in percent. It is calculated from the ratio of units actually produced and the number of units produced in the ideal case.
- **Quality:** Quality factor in percent. It is calculated as the ratio of intact produced units to produced units.

Standard HMI Controls

For the *Overall Equipment Effectiveness (OEE)* algorithm the following HMI controls are available for generating an Analytics Dashboard:

1. The OEE Control comes with three designs. It visualizes all output values as well as the parameters *Level OK / Warning* and *Level Warning / Alarm*.





2. The Table Control and Multivalue Control visualize the output values: *OEE*, *Availability*, *Performance* and *Quality*.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

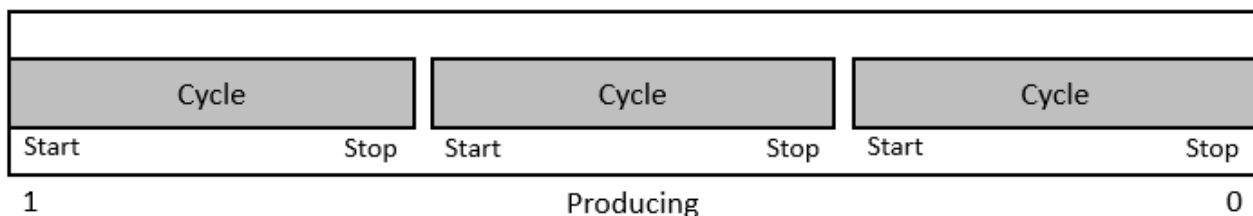
20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the *Overall Equipment Effectiveness (OEE)* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.22 Productivity Diagnosis 3Ch

Productivity Diagnosis 3Ch									
%	Is Producing	<Empty>	▼	-	Threshold Level Producing	▲	1	Producing	Empty
	Start Cycle	<Empty>	▼	-	Threshold Edge Start Cycle	▲	1	Productivity [...]	Empty
	Stop Cycle	<Empty>	▼	-	Threshold Edge Stop Cycle	▲	1	Productivity L...	Empty
					Produced Pieces		1	Expected Pro...	Empty
					Production Time		Seconds ▼ 1200	Elapsed Time	Empty
									Production C...

The *Productivity Diagnosis 3Ch* algorithm calculates the productivity of the process during a production interval. The diagram below schematically illustrates the relationship between the production process and the individual production cycles.



The production interval can be started and stopped via the input *Is Producing*. During the execution of the production interval, the production cycles are counted. Each production cycle corresponds to one piece produced. A production cycle starts with an edge at *Start Cycle* and stops with an edge at *Stop Cycle*. The productivity over the entire production interval (*Productivity*) is calculated after stopping the interval when the signal *Is Producing* no longer meets the condition for *Threshold Level Producing*. The completed production cycles and therefore all finished pieces are taken into account. Productivity is calculated as the ratio of pieces actually produced per time and the target value of pieces to be produced in a given time. The output

Productivity Last Cycle is calculated from the time required for the last production cycle in relation to the configured time for a piece. Any break times between cycles are not taken into account. The output *Expected Productivity* estimates the total productivity during the production interval. For this purpose, the previous production time is extrapolated to the total productivity for the target value of the pieces to be produced. The algorithm can be configured with the target value for the *Produced Pieces* within a configured interval (*Production Time*), e.g. 1 piece in 30 seconds or 50 pieces per hour.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the Edge:** Specifies whether the algorithm should respond to a rising or falling edge. Can be configured individually for each threshold.
- **Threshold Level Producing:** Threshold of *Input Producing* at the respective edge. The *Production Time Interval* starts when the signal passes this threshold.
- **Threshold Edge Start Cycle:** Threshold of *Input Start Cycle* at the respective edge. The production cycle starts when the signal passes this threshold.
- **Threshold Edge Stop Cycle:** Threshold of *Input Stop Cycle* at the respective edge. The production cycle stops when the signal passes this threshold.
- **Produced Pieces:** Target value for pieces produced during the configured time interval (*Production Time*).
- **Production Time:** Time interval of the production time. It can be configured in days, hours, minutes or seconds.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

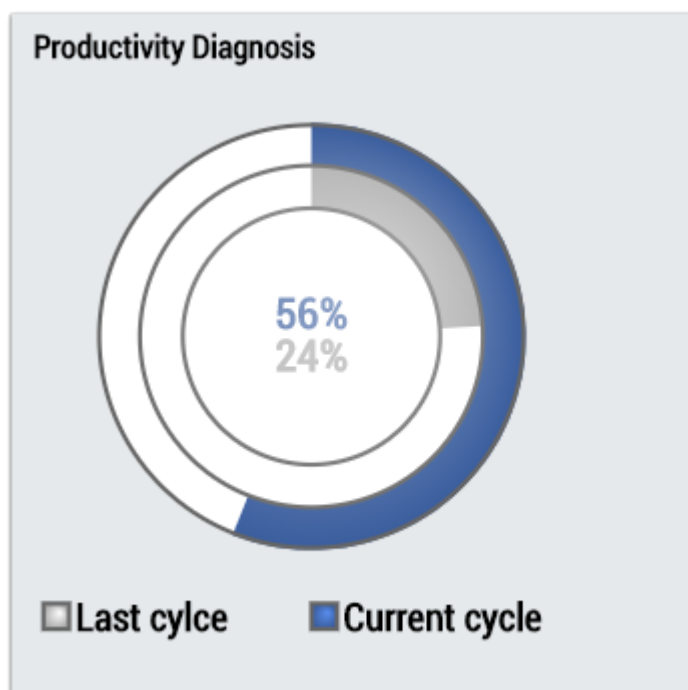
Output values

- **Producing:** Indicates whether the production interval is active.
- **Productivity:** Productivity of the entire production interval in percent.
- **Productivity Last Cycle:** Productivity of the last production cycle in percent.
- **Expected Productivity:** Estimates the productivity of the production interval. Specified in percent.
- **Elapsed Time:** Timespan since the start of the production interval.
- **Production Cycles:** Number of complete production cycles in the current production interval.

Standard HMI Controls

For the Productivity Diagnosis 3Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The ProductivityDiagnosis control visualizes the output values Productivity and Productivity Last Cycle.



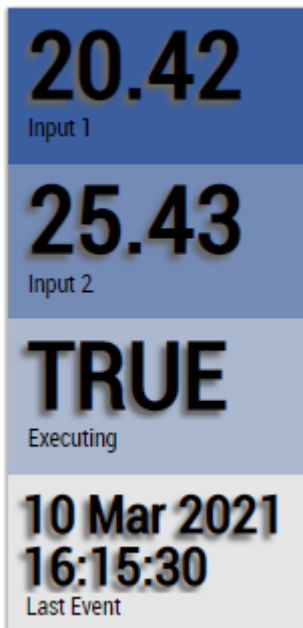
2. The Table Control or Multivalue Control visualizes all output values: New Result, Producing, Cycle Finished, Productivity, Productivity Last Cycle.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Productivity Diagnosis 3Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.23 Productivity Interval 1Ch

Productivity Interval 1Ch					
%	Input	<Empty>	-	Threshold Edge	1
	Time Start	<Empty>	-	Expected Pieces	100
	Time Stop	<Empty>	-		
					Current Times... Empty
					Interval Length Empty
					Elapsed Time Empty
					Remaining Ti... Empty
					Produced In L... Empty
					Remaining In... Empty
					Current Prod... Empty
					Expected Pro... Empty
					Last Full Perio... Empty

The algorithm *Productivity Interval 1Ch* calculates the productivity of the process during a given interval. The interval can be defined by the inputs *tTimeStart* and *tTimeStop*. The pieces produced are taken into account during execution. A produced element is counted when an edge is applied to the input. The estimated productivity of the current interval and the productivity of the last complete interval are provided as output values. The algorithm can be configured with the target value of the produced pieces within a given interval.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the Edge:** Specifies whether the piece counter should respond to a rising or falling edge.
- **Threshold Edge:** Threshold of *Input* at which a manufactured piece is counted.
- **Expected Pieces:** Specification of the pieces to be produced within the defined timespan.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

Input values

- **Input:** Input value for counting the pieces produced.
- **Time Start:** Start time of the specified interval. Allows the interval to be specified from process data. Specified as time of day with a resolution of 1 ns.
- **Time Stop:** End time of the specified interval. Allows the interval to be specified from process data. Specified as time of day with a resolution of 1 ns.

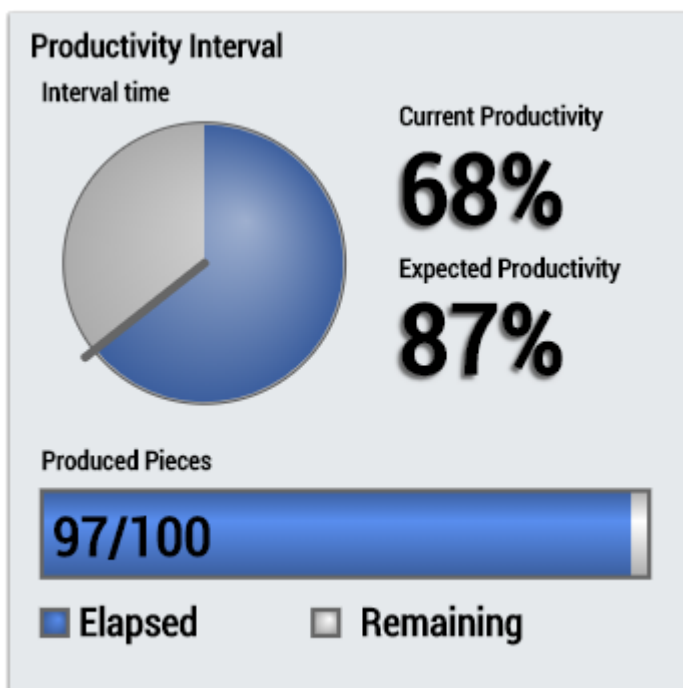
Output values

- **Within Interval:** Indicates whether the current time is within the interval.
- **Current Timestamp:** Current timestamp.
- **Interval Length:** Length of the interval.
- **Elapsed Time:** Elapsed time within the interval.
- **Remaining Time:** Remaining time within the interval.
- **Produced In Interval:** Produced pieces within the interval.
- **Remaining In Interval:** Remaining pieces within the interval.
- **Current Productivity:** Current productivity of the interval in percent. Takes into account the length of the interval, the time already elapsed, the pieces to be produced and the pieces already produced. The output is in percent.
- **Expected Productivity:** Expected productivity of the interval in percent. The production time of the last piece is used to estimate the number of pieces that can be produced in the remaining time.
- **Last Full Period Productivity:** Productivity of the last complete interval in percent. This is only calculated if the interval was fully processed.

Standard HMI Controls

For the Productivity Interval 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The ProductivityInterval control visualizes the output values: time elapsed, time remaining, number of pieces produced in the interval, number of pieces remaining in the interval, productivity.



2. The Table Control or Multivalue Control visualizes all output values: Current time, interval length, elapsed time, remaining time, number of pieces produced in the interval, number of pieces remaining in the interval, productivity.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Productivity Interval 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.24 Signal Generator 1Ch

Signal Generator 1Ch			
	Timestamp	-	
	Function Type	Sine	
	Sample Rate	1000	
	Frequency	50	
	Amplitude	1	
	Offset	0	
Signal		Empty	

Signal Generator 1Ch can be used to generate various signal curves. The signal type, the frequency, the amplitude and the offset can be set individually. A timestamp is required as a reference value because the algorithm needs a time context in which to operate. This reference timestamp is automatically set if the configuration includes another algorithm. Therefore it is not possible to use the *Signal Generator 1Ch* individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Amplitude:** Configuration of the signal amplitude.
- **Frequency:** Frequency of the generated signal.
- **Function Type:** Function type of the generated signal.
Const: constant
Rectangle: rectangle function
Sawtooth: sawtooth function
Sine: sine function
Triangle: triangle function
- **Offset:** constant offset of the generated signal.
- **Sample Rate:** sample rate of the system to be analyzed.

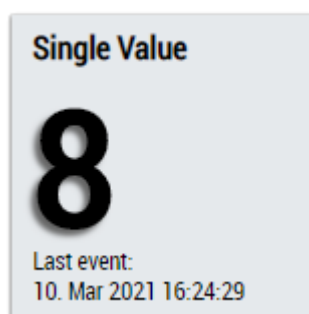
Output values

- **Signal:** outputs the generated signal.

Standard HMI Controls

For the *Signal Generator 1Ch*, the following HMI controls are available for generating an Analytics Dashboard:

1. The *SingleValue* control visualizes the signal output value.



Alternatively, customer-specific HMI controls can be mapped in the *Signal Generator 1Ch* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.25 Time Clock 1Ch

Time Clock 1Ch			
	Timestamp	-	
	Time On	hh 1 mm 0 ss 0	Is On Empty
	Time Off	hh 1 mm 0 ss 0	Next Switch Empty
	Day Of Week Mask	☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun	

Time Clock 1Ch executes a timer which can be configured with switch-on time, switch-off time and the days of the week on which the time switch should be active. A timestamp is required as a reference value because the algorithm needs a time context in which to operate. This reference timestamp is automatically set if the configuration includes another algorithm. Therefore it is not possible to use the Time Clock 1Ch individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Time On:** switch-on time.
- **Time Off:** switch-off time.
- **Day of Week Mask:** weekdays on which the timer should be active.

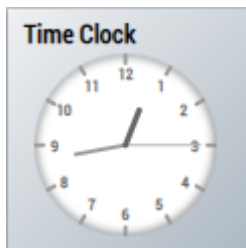
Output Values

- **Is On:** Shows *TRUE*, if the time switch is on, otherwise *FALSE*.
- **Next Switch:** Shows the remaining time up to the next switch.

Standard HMI Controls

For the Timer Clock 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Timer Control visualizes the output value Next Switch.



2. The Table Control or Multivalue Control visualizes all output values: Output, Next Switch.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the Timer Clock 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.26 Timer 1Ch

Timer 1Ch			
	Input	<Empty>	Threshold Level  0
			Timer Mode TON
			Interval Seconds 1
		Output	Empty
		Time Elapsed	Empty

The *Timer 1Ch* starts a timer which can be configured by timer mode and interval. According to the specific timer mode the timer is started, if the configured condition becomes *TRUE* (TON, TP) or the condition becomes *FALSE* (TOF).

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Operator:** Specifies whether the input value should be greater than, greater than or equal to, equal to, less than or equal to, or less than the threshold.
- **Threshold:** Signal threshold.
- **Timer Mode:** mode of the timer:
 - TON:* The TON timer is a switch-on delay timer that enables the output after the threshold condition becomes *TRUE* and the timespan specified in the interval has elapsed.
 - TOF:* The TOF timer is a switch-off delay timer that disables the output after the threshold condition becomes *FALSE* and the timespan specified in the interval has elapsed.
 - TP:* The TP timer is a pulse generator that activates the output for the time specified in the interval after the threshold condition becomes *TRUE*.
- **Interval:** Time interval of the configured timer.

- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

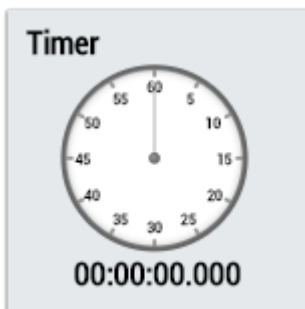
Output Values

- **Output:** Shows the output value which is affected by the configured timer.
- **Elapsed Time:** Shows the elapsed time of the timer.

Standard HMI Controls

For the Timer 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

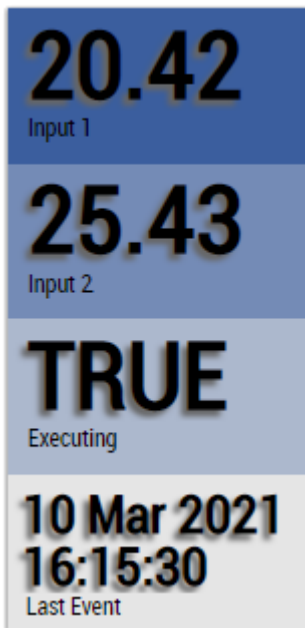
1. The Timer Control visualizes the output value Elapsed Time.



2. The Table Control or Multivalue Control visualizes all output values: Output, Elapsed Time.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Timer 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.1.27 Timing Analysis 1Ch

Timing Analysis 1Ch			
	Input	<Empty>	-
	Threshold LevelOn		1
	Is On		Empty
	Current Interval		Empty
	Last Interval		Empty
	On Total		Empty
	Off Total		Empty
Count On		Empty	

The *Timing Analysis 1Ch* measures time differences between on- and off-periods and counts the amount of on-periods. The on-period starts when the condition of operator and threshold is met.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Operator:** Specifies whether the input value should be greater than, greater than or equal to, less than or equal to, less than or not equal to the threshold.
- **Threshold:** Signal threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

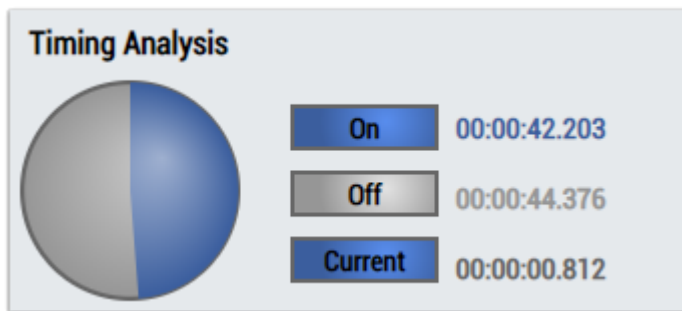
Output values

- **Is On:** Indicates *TRUE* within the timespan of the On period, otherwise *FALSE*.
- **Current Interval:** Indicates the time of the current interval.
- **Last Interval:** Timespan of the last completed interval.
- **On Total:** Indicates the total time for which the “Is On” value is *TRUE*.
- **Off Total:** Displays the total time for which the “Is On” value is *FALSE*.
- **Count On:** Counts the number of triggered On events.

Standard HMI Controls

For the Timing Analysis 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The TimingAnalysis Control visualizes the output values Is On, On Total, Off Total and Current Interval.



2. The SingleValue control visualizes the output value Count On.



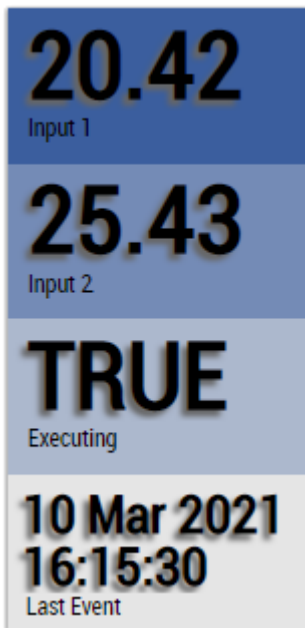
3. The Table Control or Multivalue Control visualizes all output values: Is On, Count On, Current Interval, On Total, Off Total.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Timing Analysis 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2 Analytics - Classification

The algorithms of the category *Analytics-Classification* provide functionalities for classification and state detection.

5.2.1.2.1 Bandwidth Classifier 1Ch

Bandwidth Classifier 1Ch					
	Input	<Empty>	-	Lower Bound	0
				Upper Bound	10
				Class	Empty
				Last Event	Empty

Bandwidth Classifier 1Ch determines whether the input signal is within the configured limits or is less than or greater than the limits.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Lower Bound:** Lower Bound for the comparison.
- **Upper Bound:** Upper Bound for the comparison.

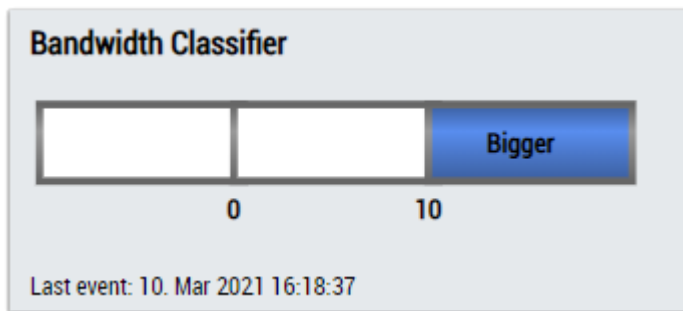
Output values

- **Class:** Indicates the class to which the input values belong (WithinBounds/Smaller/Bigger).
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

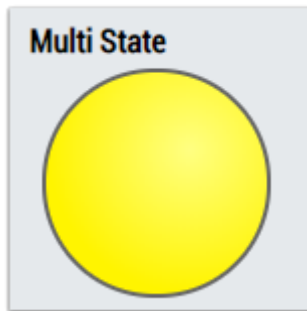
Standard HMI Controls

For the Bandwidth Classifier 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The BandwidthClassifier control visualizes the output values Class and Last Event and the configuration options Lower Bound and Upper Bound.



2. The MultiState control visualizes the output value Class.



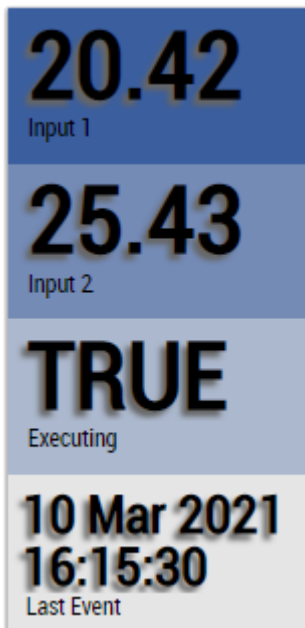
3. The Table Control or Multivalue Control visualizes all output values: Class, Last Event.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Bandwidth Classifier 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.2 Bandwidth Classifier 3 Ch

Bandwidth Classifier 3Ch					
	Input	<Empty>	▼	-	Class Empty
	Lower Bound	<Empty>	▼	-	Last Event Empty
	Upper Bound	<Empty>	▼	-	

Bandwidth Classifier 3Ch determines whether the input signal is within the limits or is less than or greater than the limits. The limits can be configured with input signals, so it is possible to use curves as lower and upper band.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

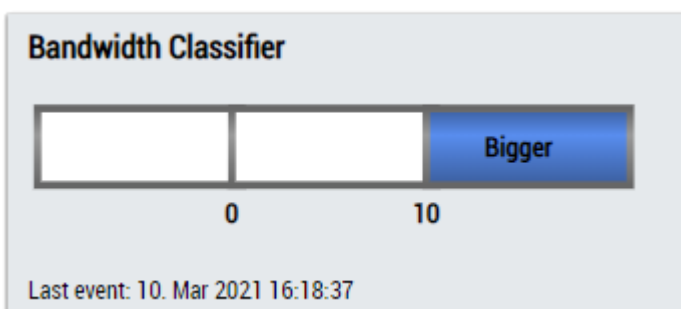
Output values

- **Class:** Indicates the class to which the input values belong (WithinBounds/Smaller/Bigger).
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

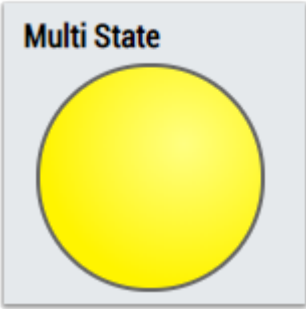
Standard HMI Controls

For the Bandwidth Classifier 3Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The BandwidthClassifier control visualizes the output values Class and Last Event and Input Lower Bound and Input Upper Bound.



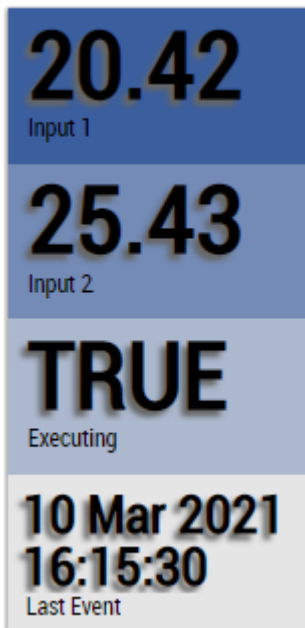
2. The MultiState control visualizes the output value Class.



3. The Table Control visualizes all output values: Class, Last Event.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Bandwidth Classifier 3Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.3 Curve Sketcher 1Ch

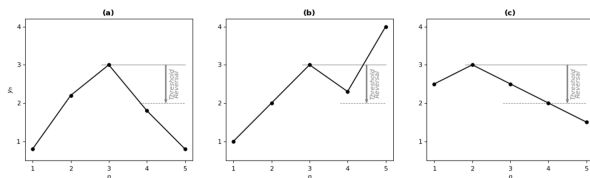
Curve Sketcher 1Ch			
↗	Input	<Empty>	-
	Threshold Reversal	0	
	Calculate Inflection	☐	
	Threshold Delta	10	
	Last Peak	Empty	
	Time Last Peak	Empty	
	Count Peaks	Empty	
	Last Valley	Empty	
	Time Last Valley	Empty	
	Count Valleys	Empty	
	Value at Max...	Empty	
	Max Delta	Empty	
	Time Max Delta	Empty	
	Count Max D...	Empty	

Curve Sketcher 1Ch identifies inversions (*peaks* and *valleys*) in an input data stream. Furthermore, local maxima of the absolute difference between two consecutive values (referred to as *Delta*) can be identified. Analogous to a continuous curve, the identified peaks and valleys correspond to local maxima and minima. Delta corresponds to the slope, so that a maximum of the absolute values of Delta can be associated with an inflection point.

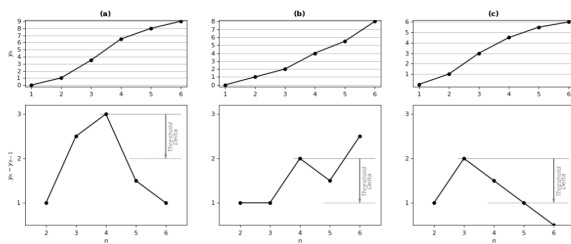
Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Calculate Inflection:** Boolean flag. Maxima of the rate of change are only identified if this flag is *True*. Otherwise, the values for *Count Max Delta*, *Max Delta*, *Time Max Delta* and *Value at Max Delta* will not be calculated.
- **Threshold Reversal:** threshold for identifying reversals. Reversals are only detected if their difference from the next reversal exceeds the value of *Threshold Reversal*. Below are three examples of peak identification using the parameter *Threshold Reversal*.
 - The value y_3 is identified as a peak immediately after processing the value y_4 because the difference between y_3 and y_4 is greater than *Threshold Reversal*.
 - The value y_3 is not identified as a peak because the difference between y_3 and y_4 is smaller than *Threshold Reversal* and the curve starts rising again after y_4 .
 - The value y_2 is identified as a peak after processing the value y_5 because the difference between y_2 and y_5 exceeds *Threshold Reversal*. The value y_2 cannot be identified as a peak beforehand because the difference between y_2 and y_3 (y_4) is less than/equal to *Threshold Reversal* and it is not known whether the values will continue to decrease.



- **Threshold Delta:** threshold for identifying Delta maxima. Maxima of the absolute difference of two successive values (delta) are detected only if the difference between successive deltas exceeds *Threshold Delta*.
Below are three examples of identifying the Delta maxima with the parameter *Threshold Delta*. The upper diagrams show the original input signals, the lower ones the corresponding delta.
- (a) The value y_4 is identified as a maximum after processing the value y_5 because the difference between the two deltas exceeds *Threshold Delta*.
- (b) No maximum is identified because the difference between the deltas is less than *Threshold Delta*.
- (c) The value y_3 is identified as a maximum after processing the value y_6 .



Regardless of *Threshold Delta*, at least one maximum of the Delta between two reversals is detected.

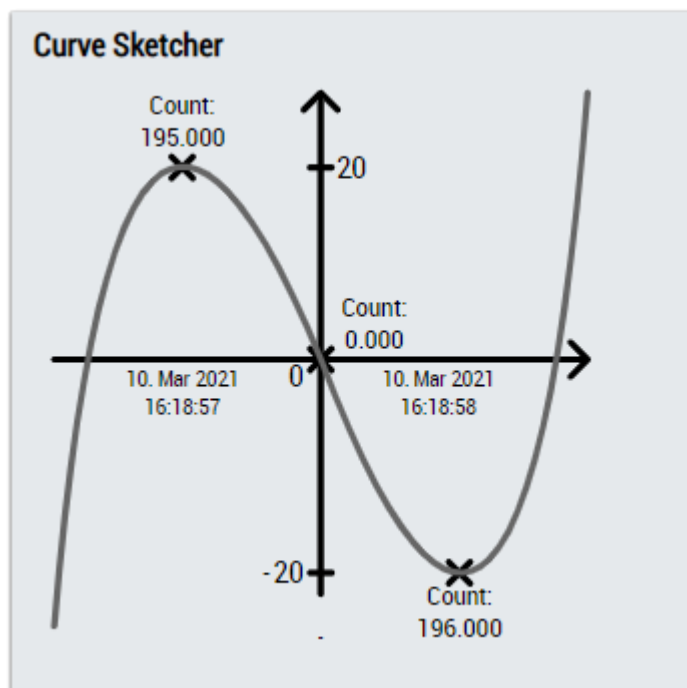
Output Values

- **Last Peak:** Indicates the y-value of the last identified peak.
- **Time Last Peak:** Indicates the timestamp of the last identified peak.
- **Count Peaks:** Indicates the total number of counted peaks.
- **Last Valley:** Indicates the y-value of the last identified valley.
- **Time Last Valley:** Indicates the timestamp of the last identified valley.
- **Count Valleys:** Indicates the total number of counted valleys.
- **Value at Max Delta:** Indicates the y-value of the analyzed stream (input variable) that is led by the last detected maximum of delta. The value delta is the difference between *Value at Max Delta* and the value that reached one cycle before.
- **Max Delta:** Indicates the last identified local maximum of the absolute difference between two successive values in the input stream.
- **Time Max Delta:** This is the timestamp of *Value at Max Delta*.
- **Count Max Delta:** Indicates the total number of counted local maxima of delta.

Standard HMI Controls

For the Curve Sketcher 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. CurveSketcher control visualizes the output values Last Peak, Time Last Peak, Count Peaks, Last Valley, Time Last Valley, Count Valley, Last Delta, Time Last Delta and Count Delta as well as the input value of the data.



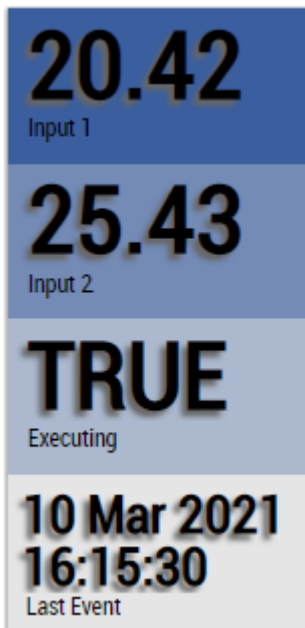
2. The Table Control visualizes all output values: Last Peak, Time Last Peak, Count Peaks, Last Valley, Time Last Valley, Count Valley, Value at Max Delta, Last Delta, Time Last Delta and Count Delta.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Curve Sketcher 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.4 Histogram 1Ch

Histogram 1Ch			
	Input	<Empty>	-
	Histogram Mode		Absolute
	Min Binned	1	
	Max Binned	10	
	Bins	9	
Num Values		Empty	
Histogram			

The *Histogram 1Ch* calculates the distribution of a single channel input value cyclically. It can be configured with minimal bin, maximal bin and the total amount of bins. The dimension of the output array is the number of bins + 2. Because values that are less than the minimal bin are stored in the first array element and values greater than the maximal bin are stored in the last array element.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Histogram Mode:** The operation mode of the histogram. A distinction is made between absolute and relative output. The latter can be used to display the percentage distribution.
- **Min Binned:** Minimum value to be analyzed.
- **Max Binned:** Maximum value to be analyzed.
- **Bins:** Total number of histogram classes to be calculated.
- **Enable Display Names:** Enable Histogram Display Names. This configuration is visible hidden in the property window.
- **Histogram Display Names:** Display names of the individual bins. This configuration is hidden in the Property window and is used for the HMI.

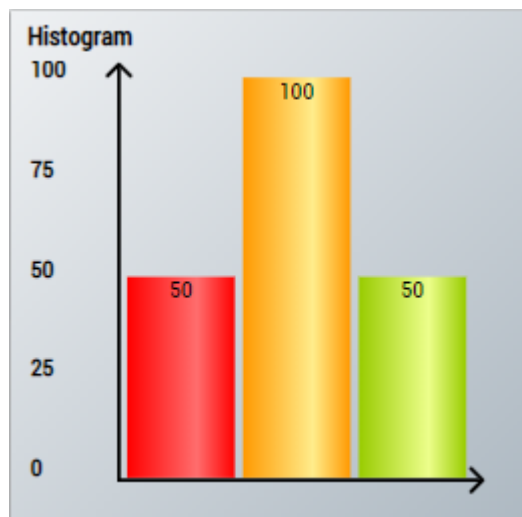
Output values

- **Num Values:** Indicates the total number of analyzed values for the distribution.
- **Histogram:** The histogram is displayed below the Num Values. If you move the cursor over the bars, you will see a tooltip with the value interval and the corresponding value of the histogram bin.

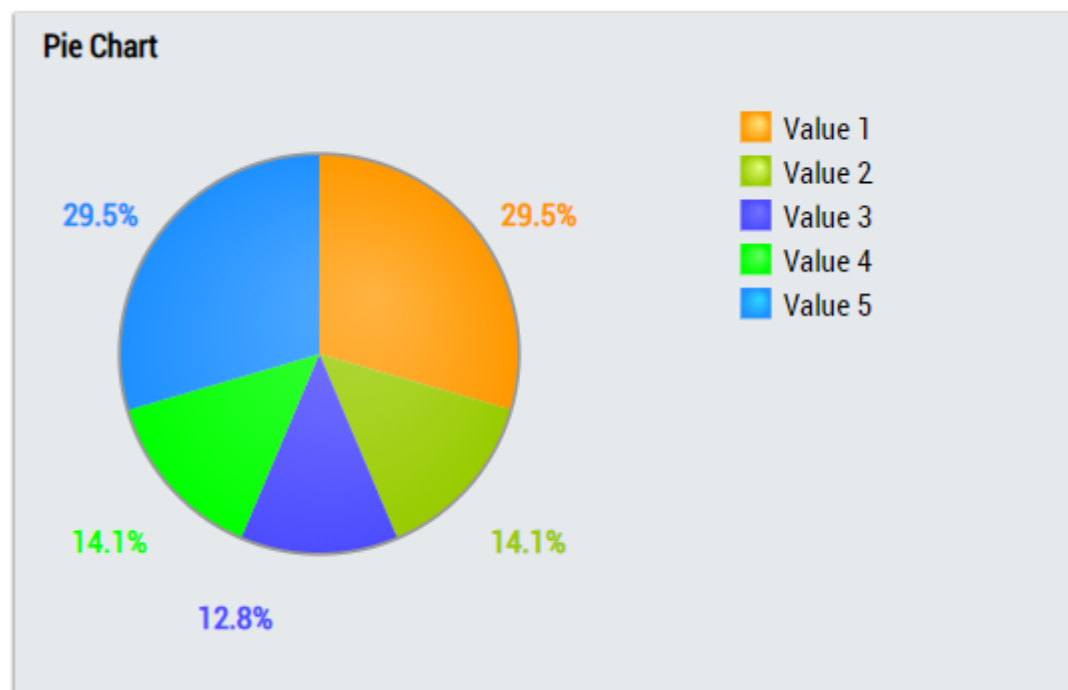
Standard HMI Controls

For the Histogram 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Histogram control visualizes the output value Num Values.



2. The PieChart control visualizes the output value Num Values.



3. The Table Control or Multivalue Control visualizes all output values: Num Values, Histogram.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Histogram 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.5 Pareto Analysis

The screenshot shows the 'Pareto Analysis' configuration window. It has a title bar with a green status indicator and standard window controls. The main area is divided into several sections: 'Input' with two channels (Input 00 and Input 01) both set to '<Empty>', 'Single Input' with an unchecked checkbox, 'Num Channels' set to 1, 'Pareto Mode' set to 'Count', 'Timespan' set to 1, and 'Output' with options for 'Output' and 'Accumulated Relative Output'. There are also fields for 'Channel 00', 'Channel 01', 'Channel 1', and 'Channel 2'.

The *Pareto Analysis* module allows you to combine different input data to display a Pareto chart. This is used in the following applications, among others:

Error analysis: Identifying the most common sources of errors

Quality management: Determining the main causes of quality problems

Time management: Diagnosis of which tasks take up most of the time

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Single Input:** Selection of whether a single input or several independent input channels should be used.
- **Num Channels:** Number of input channels if *Single Input* is *FALSE*.
- **Num Elements:** Number of elements to be compared if *Single Input* is *TRUE*.
- **Pareto Mode:** Mode for calculating the output values.
Count: The cycles in which the respective input channel is *TRUE* are counted.
Timespan: The timespans in which the input channel is *TRUE* are added up.
- **Threshold 00, Threshold 01, ..., Threshold n:** Comparison operator and threshold for monitoring input values.
- **Channel Name 00, Channel Name 01, ..., Channel Name n:** Alias of the input channel.



If the configuration option *Single Input* is set and an input of type Enumeration is linked to the input, configuration options are set automatically:

Number of input channels: 1
 Number of elements to be compared: Number of enumerations
 Threshold values: Values of the individual enumerations

Output values

- **Output:** Sorted absolute values. The output type depends on the Pareto mode.
- **Accumulated Relative Output:** Accumulated relative output values.
- **Channel Name 00, Channel Name 01, ... Channel Name n:** Sorted alias names of the input channels.

5.2.1.2.6 Section Timer 1 Ch

The screenshot shows the 'Section Timer 1Ch' configuration window. It has a title bar with a green status indicator and standard window controls. The main area is divided into several sections: 'Input' set to '<Empty>', 'Num Sections' set to 4, and a table for section borders. The table has four rows: 'First Lower Border' (1), 'Upper Border 00' (2), 'Upper Border 01' (3), and 'Upper Border 02' (4). There are also fields for 'Section' and 'Sections'.

The *Section Timer 1Ch* calculates the timespan the input is in range of each configured section. It can be configured with the amount of sections and the borders of each section. Each section is defined with lower border (greater than or equal to) and upper border (less than). The following sections lower border is set by the previous upper border. Values that are less than the minimal border are stored in the first array element. Values that are greater than or equal to the maximal border are stored in the last array element.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Sections:** This is the number of sections.
- **First Lower Border:** This is the lower border of the first section.
- **Upper Border 00, Upper Border 01, ..., Upper Border n:** These are the upper borders of all sections.
- **Enable Display Names:** Enable Section Display Names. This configuration is visible hidden in the property window.
- **Section Display Names:** Display names of the individual sections. This configuration is hidden in the Property window and is used for the HMI.

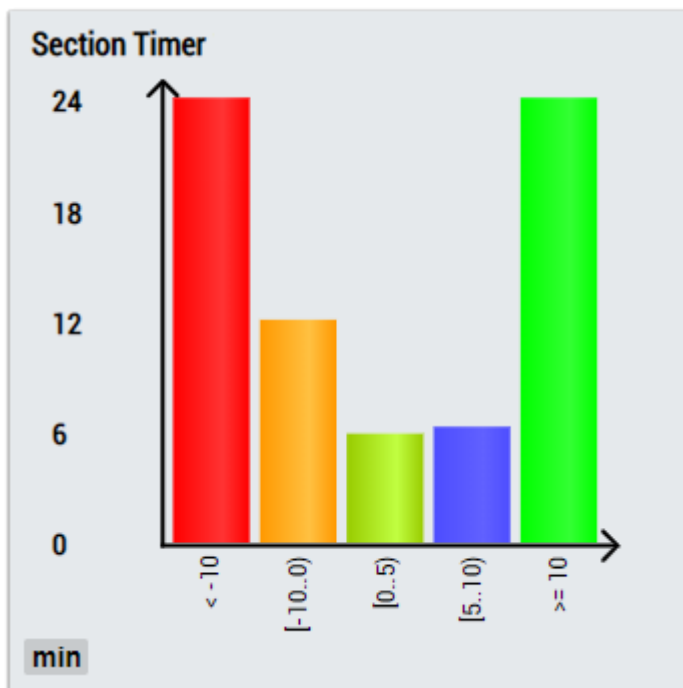
Output values

- **Section:** specifies the section of the last input value. If the input value is less than the *First Lower Border*, the return value is zero. If the input value is in the interval [*First Lower Border*, *Upper Border 00*), the return value is one, for the interval [*Upper Border 00*, *UpperBorder 01*) it is two, etc. If the input value is greater than the last specified limit *Upper Border 0n*, the return value is *NumSections+1*.
- **Array of Timespans:** total time during which the input value was sorted into each section.

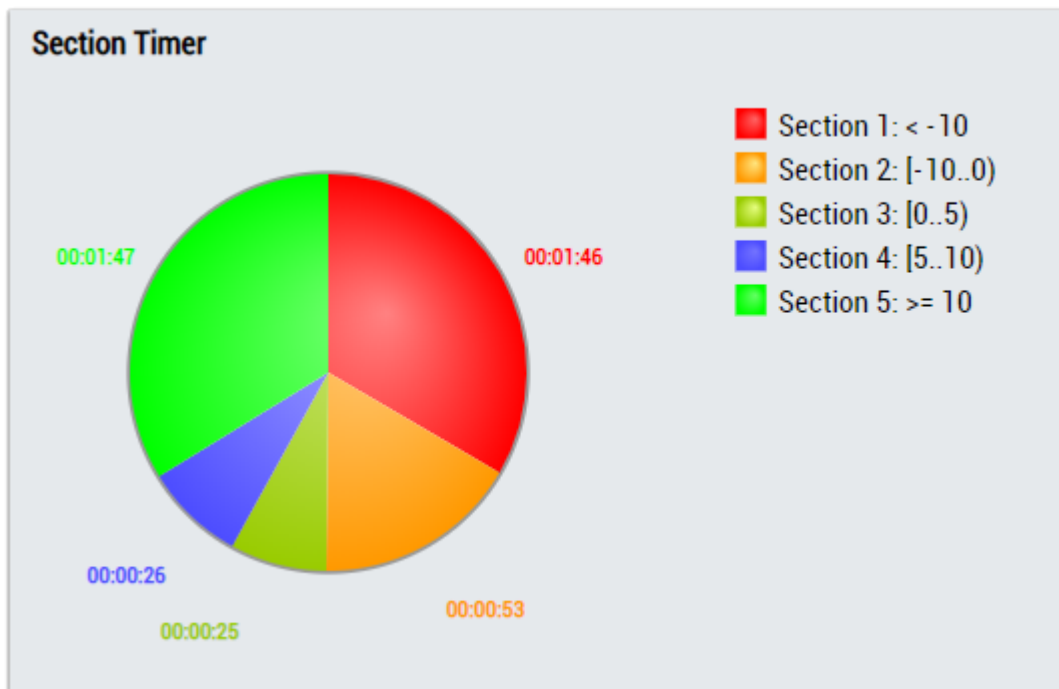
Standard HMI Controls

For the Section Timer 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SectionTimer control visualizes the input value Array of Timespans and the configuration options First Lower Border and Upper Border.



2. The PieChart control visualizes the input value Array of Timespans and the configuration options First Lower Border and Upper Border.



Alternatively, customer-specific HMI controls can be mapped in the Section Timer 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.7 State Histogram 1Ch

State Histogram 1Ch			
	Input	<Empty>	-
	Histogram Mode	Absolute	
	Num Values	Empty	
	Min	1	
	Max	10	
Histogram			

The *State Histogram 1Ch* counts how often the input signal (INT) has a specific value between the configured minimum and maximum and shows the distribution in a histogram. The first bar represents the boundary values which are smaller than the minimum and the last bar represents the boundary values which are greater than the maximum. The *State Histogram 1Ch* is suitable for state-machines to show how often the different states are executed.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Histogram Mode:** The operation mode of the histogram. A distinction is made between absolute and relative output. The latter can be used to display the percentage distribution.
- **Min:** Minimum value to be analyzed.
- **Max:** Maximum value to be analyzed.
- **Enable Display Names:** Enable Histogram Display Names. This configuration is visible hidden in the property window.
- **Histogram Display Names:** Display names of the individual bins. This configuration is hidden in the Property window and is used for the HMI.

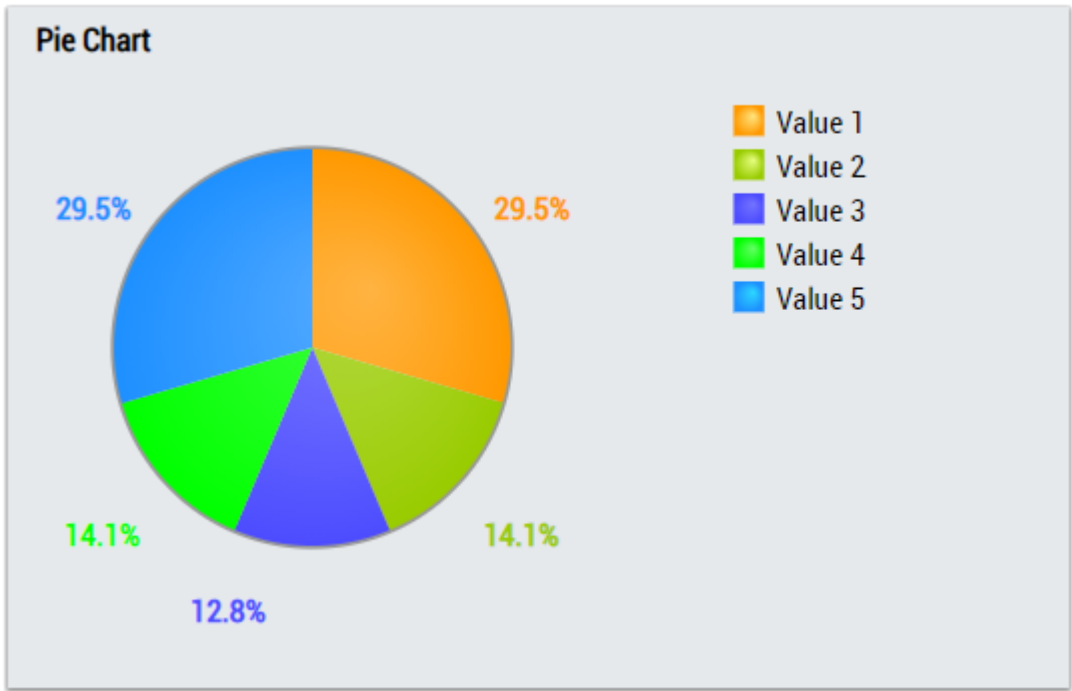
Output values

- **Num Values:** Indicates the total number of states executed between the configured borders.
- **Histogram:** The histogram is displayed below the Num Values. The respective value is indicated on each bar. If you move the cursor over the bars, you will see a tooltip with the value interval and the corresponding value of the histogram bin.

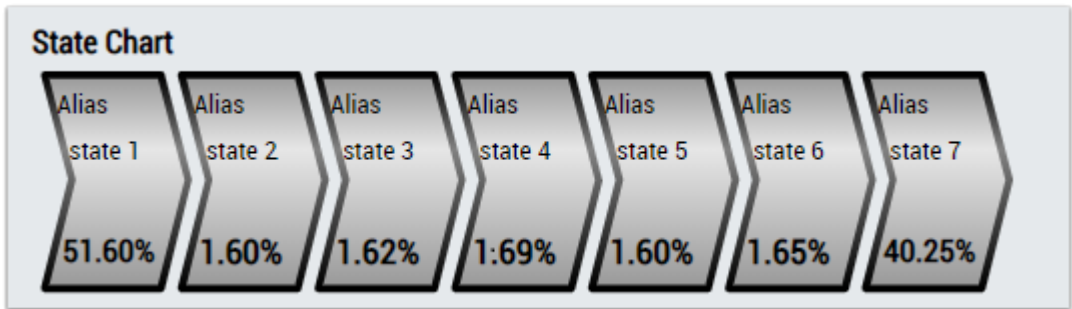
Standard HMI Controls

For the State Histogram 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The PieChart control visualizes the output value histogram.



2. The StateHistogram control visualizes the output value histogram.



3. The Table Control or Multivalue Control visualizes all output values: Num Values, Histogram.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the State Histogram 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.8 Threshold Classifier 1Ch

Threshold Classifier 1Ch					
	Input	<Empty> ▾ -	Level OK / Warning	10	Class Empty
			Level Warning / Alarm	20	Last Event Wa... Empty
					Last Event Ala... Empty

Threshold Classifier 1Ch classifies the input values into three different classes: *OK*, *Warning* and *Alarm* according to the configured thresholds.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Level Ok / Warning:** Input values that are less than the configured threshold are classified as *OK*; input values that are equal to or greater than the configured threshold are classified as *Warning*.
- **Level Warning / Alarm:** Input values that are less than the configured threshold are classified as *Warning*; input values that are equal to or greater than the configured threshold are classified as *Alarm*.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

Output values

- **Class:** Indicates the class to which the input values belong.
- **Last Event Warning:** Indicates the time of the last triggered warning event → the event can be dragged and dropped into the scope chart to display it as a trigger-event.
- **Last Event Alarm:** Indicates the time of the last triggered alarm event → the event can be dragged and dropped into the scope chart to display it as a trigger-event.

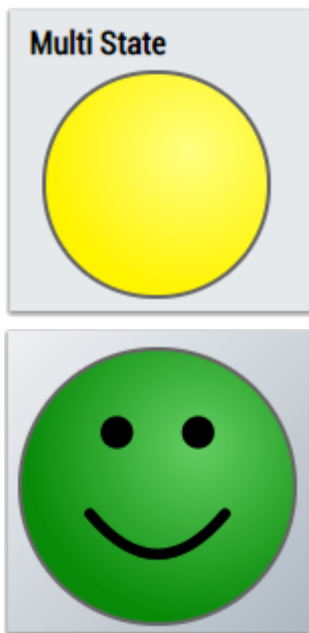
Standard HMI Controls

For the Threshold Classifier 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

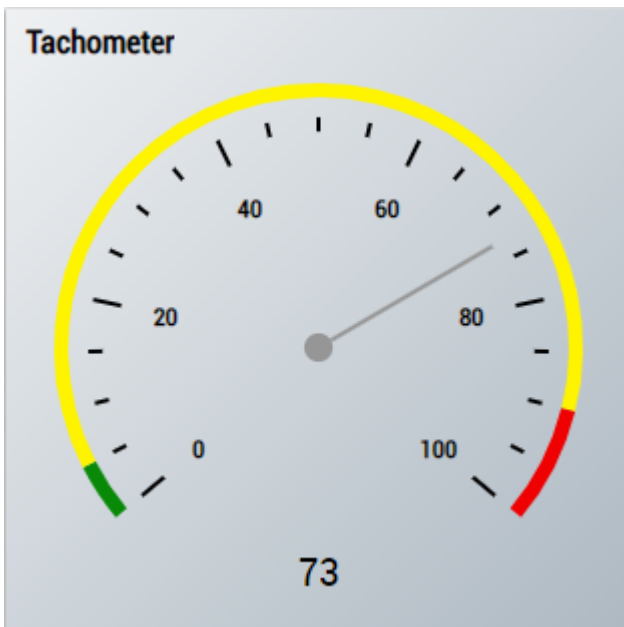
1. The TrafficLight Control visualizes the output values Class, Last Event Warning and Last Event Alarm as well as the input value of the data.



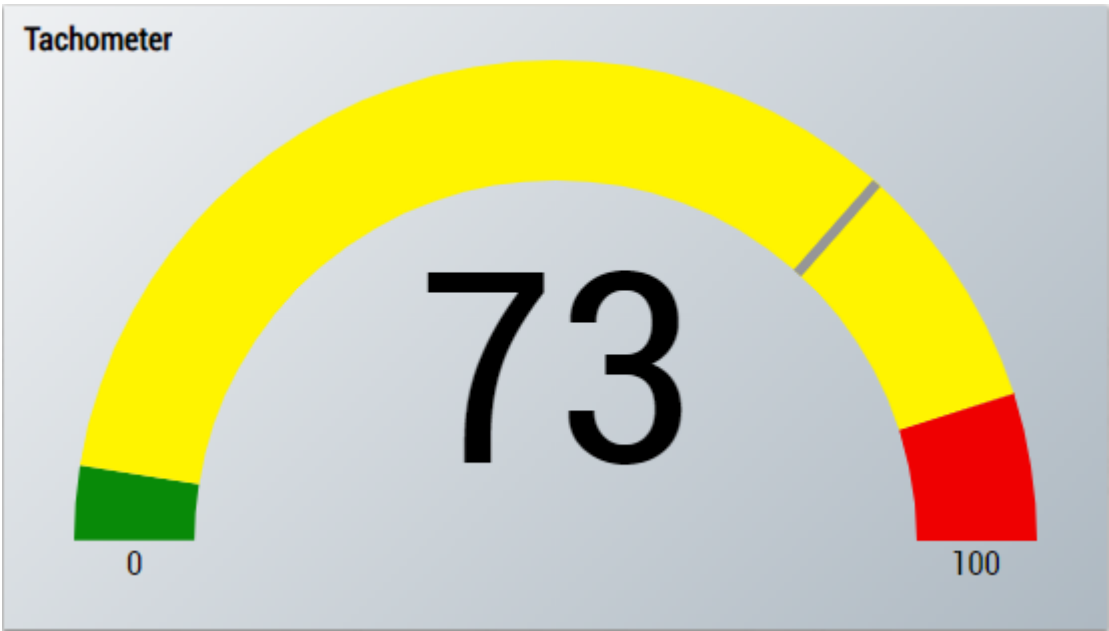
2. The MultiState Control visualizes the Class output value, optionally with a smiley.



3. The Tachometer Control visualizes the input value including the configured limit values.



4. The Radial Gauge Control visualizes the input value including the configured limit values.



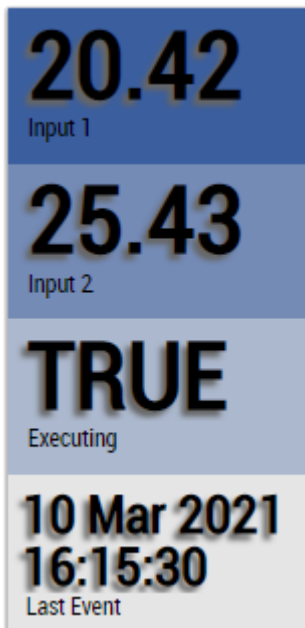
5. The Table Control or Multivalue Control visualizes all output values: Class, Last Event Warning Last Event Alarm.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Threshold Classifier 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.9 Threshold String Classifier 1Ch

Threshold String Classifier 1Ch			
	Input	<Empty>	-
	Level 1 / 2	10	
	Level 2 / 3	20	
	String Level 1	String 1	
	String Level 2	String 2	
	String Level 3	String 3	
	Output String	Empty	
	Last Event	Empty	

The *Threshold String Classifier 1Ch* algorithm classifies the input values into three different classes according to the configured thresholds. The class names (output string) can be configured individually as *String 1*, *String 2* and *String 3*.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Level 1 / 2:** Input values that are less than the configured threshold are assigned to the *first* class; input values that are equal to or greater than the configured threshold are assigned to the *second* class.
- **Level 2 / 3:** Input values that are less than the configured threshold are assigned to the *second* class; input values that are equal to or greater than the configured threshold are assigned to the *third* class.
- **String 1:** Name of the first class.
- **String 2:** Name of the second class.
- **String 3:** Name of the third class.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.

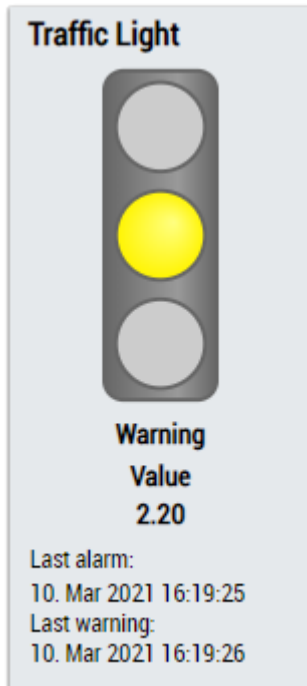
Output values

- **Output String:** Indicates the class to which the input values belong.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the Threshold String Classifier 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

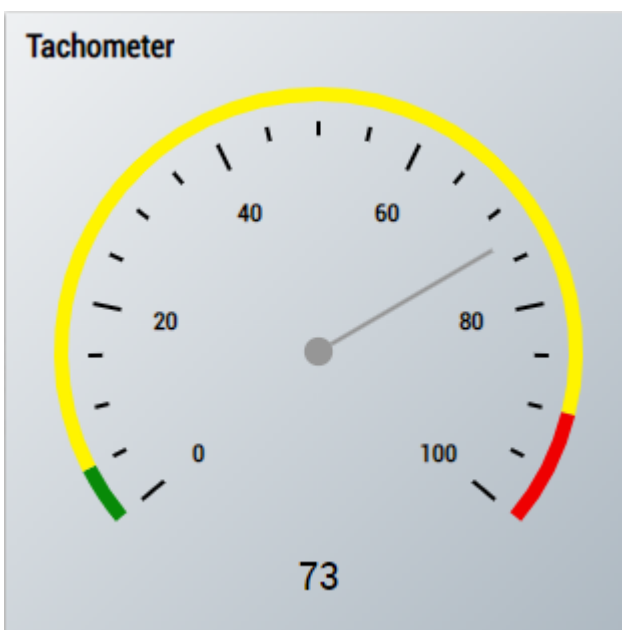
1. The TrafficLight control visualizes the output values Output String and Last Event as well as the input value of the data.



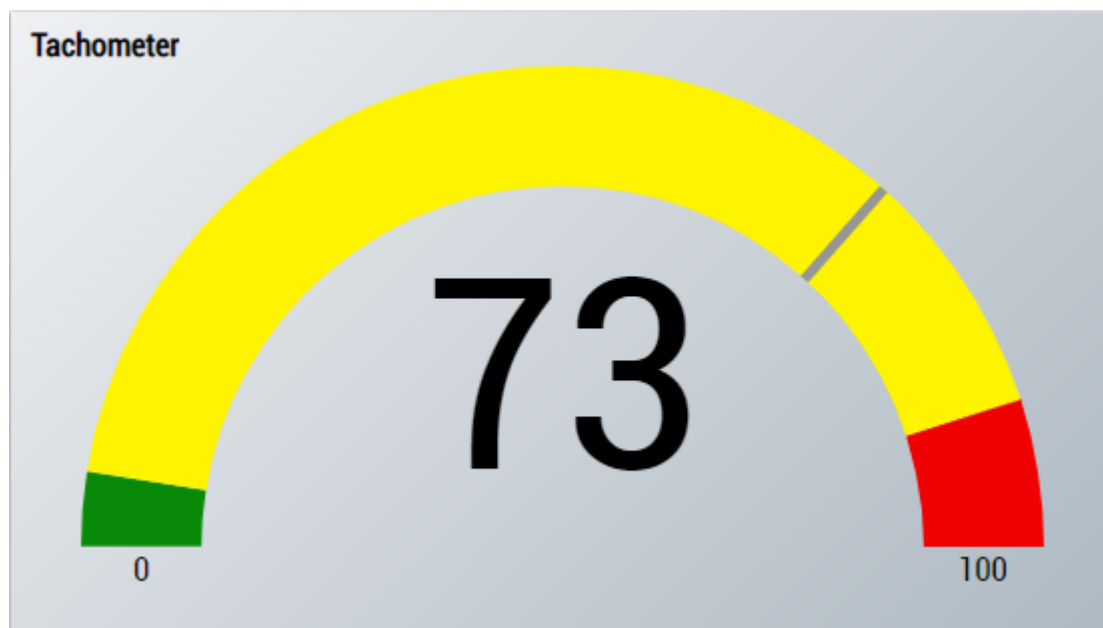
2. The SingleValue control visualizes the output value Output String.



3. The Tachometer Control visualizes the input value including the configured limit values.



4. The Radial Gauge Control visualizes the input value including the configured limit values.



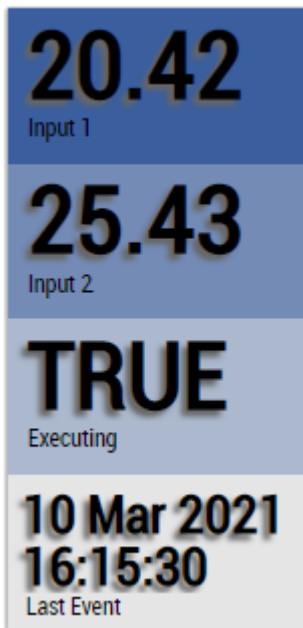
5. The Table Control or Multivalue Control visualizes all output values: Output String, Last Event.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Threshold String Classifier 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.10 Time Based Envelope 1Ch

Time Based Envelope 1Ch						
Input	<Empty>	-	File Path	C:\Temp\Analytics Project\Teach	Executing Co...	Empty
Start Period	<Empty>	-	Band Mode	Absolute	Value Read	Empty
			Band	10	Lower Band	Empty
					Upper Band	Empty
					Within Band	Empty
					Compare Res...	Empty
					Current Com...	Empty
					Count Within...	Empty
					Count Smaller	Empty
					Count Bigger	Empty
					Value Number	Empty

The *Time Based Envelope 1Ch* algorithm checks whether the periodic input data is within a configured range of values read from a file. This can be a reference signal that was previously learned with *Time Based Teach Path 1Ch* [► 219], for example. The comparison starts when the signal of the Start Period flag is *TRUE*. It is recommended not to use *Time Based Envelope 1Ch* simultaneously with *Time Based Teach Path 1Ch* [► 219] due to concurrent file access. Instead, a reference signal should first be taught in with *Time Based Teach Path 1Ch* [► 219] and only then should the evaluation be carried out with the aid of the *Time Based Envelope 1Ch*.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **File Path:** Path to the previously taught data file.
- **Band Mode:** Mode of the band operation (use absolute or relative values).
- **Band:** Bandwidth of the band operation.

Output values

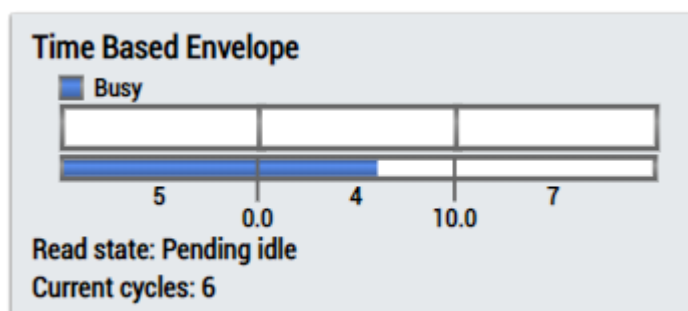
- **Executing Compare:** displays *TRUE* if the algorithm is processing the envelope, otherwise *FALSE*. The envelope process begins when the Start Period flag is *TRUE*.
- **Lower Band:** displays the value of the lower band depending on the band mode.
- **Upper Band:** displays the value of the upper band depending on the band mode.
- **Within Band:** displays *TRUE* if the current values are within the band, otherwise *FALSE*.

- **Compare Result:** result of the current comparison. Indicates whether the current values are within the band or are smaller or larger than the band.
- **Current Compared Cycles:** number of cycles that have been compared.
- **Count Within Band:** counts how often the values were within the band.
- **Count Smaller:** counts how often the values were smaller than the band.
- **Count Bigger:** counts how often the values were larger than the band.
- **Value Number:** displays the value of the data point in the file that is currently being compared.

Standard HMI Controls

For the Time Based Envelope 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

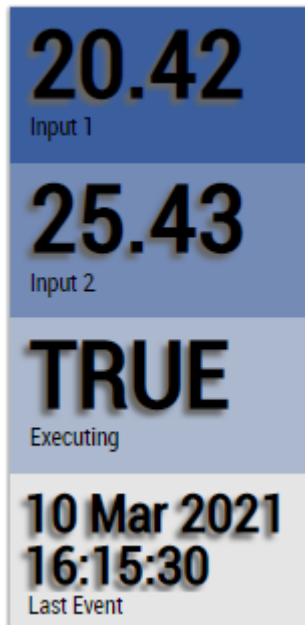
1. The TimeBasedEnvelope control visualizes the output values Executing Comparison, Lower Band, Upper Band, Within Band, Compare Result, Count Within Band, Count Smaller, Count Bigger, Current Compared Cycles, State and Value Number.



2. The Table Control or Multivalue Control visualizes all output values: Executing Comparison, State, Lower Band, Upper Band, Within Band, Compare Result, Current Compared Cycles, Count Within Band, Count Smaller, Count Bigger, Value Number.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

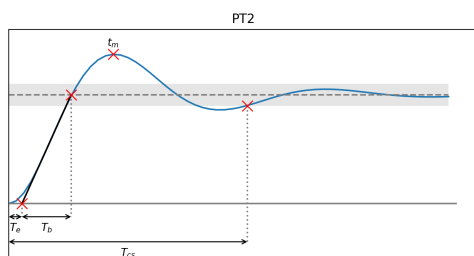


Alternatively, customer-specific HMI controls can be mapped in the Time Based Envelope algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.2.11 Step Response 1Ch

Step Response 1Ch				
	Input	<Empty>	-	Executing Empty
	Setpoint	<Empty>	-	Equivalent De... Empty
			Threshold Reversal 0.01	Equivalent Ti... Empty
			Relative Tolerance ☒	Settling Time Empty
			Error Tolerance 5	Time Max Empty
				Error Empty

Step Response 1Ch identifies parameters of the step response of a PT2 track. These include the delay time T_e , the compensation time T_b , the settling time T_{cs} , and the time of maximum t_m .



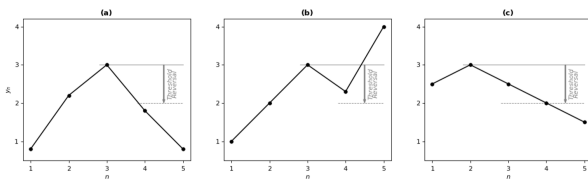
To detect whether the track is steady, a local minimum or maximum is searched for. If this is within the tolerance band (marked in gray), it is assumed that the track is steady. Only then is the settling time set.

The algorithm starts when a new setpoint is outside the previously stored tolerance band.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Threshold Reversal:** threshold for identifying reversals. Reversals are only detected if their difference from the next reversal exceeds the value of *Threshold Reversal*. Below are three examples of peak identification using the parameter *Threshold Reversal*.
 (a) The value y_3 is identified as a peak immediately after processing the value y_4 because the difference between y_3 and y_4 is greater than *Threshold Reversal*.
 (b) The value y_3 is not identified as a peak because the difference between y_3 and y_4 is smaller than *Threshold Reversal* and the curve starts rising again after y_4 .
 (c) The value y_2 is identified as a peak after processing the value y_5 because the difference between y_2 and y_5 exceeds *Threshold Reversal*. The value y_2 cannot be identified as a peak beforehand because the difference between y_2 and y_3 (y_4) is less than/equal to *Threshold Reversal* and it is not known whether the values will continue to decrease.



- **Relative Tolerance:** Boolean flag. If this flag is *True*, the parameter *Error Tolerance* refers to the setpoint at the input as a percentage. Otherwise, an absolute tolerance band is taken into account.
- **Error Tolerance:** Specifies the size of the tolerance band in relation to the parameter *Relative Tolerance*. The tolerance band is updated when parameter identification is restarted.

Output values

- **Executing:** True if parameter identification is active.
- **Equivalent Dead Time:** Delay time of the track. Timespan between the start of the step response and the intersection of the inflectional tangents with the start value.
- **Equivalent Time Constant:** Compensation time of the track. Timespan between the intersection point of the inflectional tangents with the start value and the intersection point of the inflectional tangents with the setpoint.
- **Settling Time:** Settling time of the track.
- **Time Max:** Time of the maximum overshoot.
- **Error:** Difference between input value and setpoint.

Standard HMI Controls

For the *Step Response 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

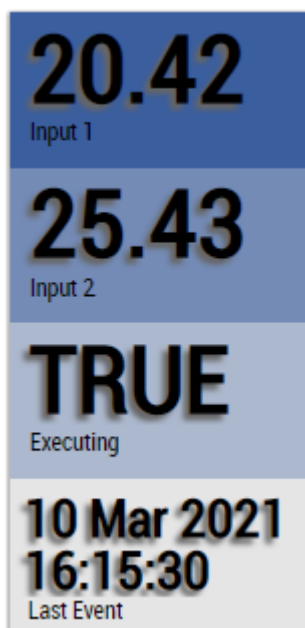
1. The Table Control and Multivalue Control visualize all output values.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped using the [Mapping Wizard](#) [► 382].

5.2.1.3 Analytics - Clustering

The algorithms in the *Clustering* category provide functions for streaming data-based clustering of input data into various clusters that are not predefined. The clusters are detected based on the structures present in the input data. Examples of such cluster algorithms are *Sequential k-Means* and *DenStream*.

5.2.1.3.1 DenStream

DenStream			
Update Micro CL...	<Empty>	-	Num Channels
Input 00	<Empty>	-	Epsilon
			Mu x Beta
			Lambda
			Epsilon (DBSCAN)
			Min Weight (DBSCAN)
			potMCs Buffer Size
			outMCs Buffer Size
			MC Buffer Ov...
			Last Event
			Last Switch
			Number of p...
			Number of o...
			Cluster Index
			Number of CL...

DenStream is an implementation of the unmonitored, density-based clustering algorithm of the same name [1]. The latter is based on the well-known clustering algorithm DBSCAN [2, 3] and is particularly suitable for data streams whose structures change over time.

The number of input channels (referred to below as n) for this algorithm can be selected by the user. These inputs form the n -dimensional feature space in which clusters can be found. In each analysis cycle, the data stream provides the algorithm with a new feature vector that can be interpreted as a data point in this feature space. Clusters are separable areas with a high density of data points in the feature space.

In the first phase of the algorithm, the incoming data points are assigned to so-called micro-clusters (MCs). These micro clusters have properties (such as center point, weight and variance) that depend on the data points they contain. Only micro-clusters whose weight exceeds a certain threshold enter the second phase and are clustered by the DBSCAN algorithm. Thus, it is not necessary to retain the information about each data point. This reduces memory requirements, because over time there are far fewer micro-clusters than data points. Also, the computing effort for the DBSCAN algorithm is much lower since it runs over the reduced set of micro-clusters rather than all the data points. It is also possible to apply a fading function to the weights of the micro clusters. In this way, old data points lose their importance to the clustering process over time. This allows the algorithm to capture changes (such as the movement of clusters or their disappearance/appearance over time).

The DenStream algorithm has further advantages over other clustering algorithms. The user does not need to know the number of micro-clusters in advance, as the DenStream algorithm determines this number automatically. In addition, the algorithm is able to detect outliers in the data that does not belong to any cluster. Since it is a density-based algorithm, it is even possible to detect separate clusters of any shape (even if they are intertwined).

Parameter setting

Here we give a short introduction to how the algorithm works, mainly to give the reader a quick introduction to the parameter setting. For a deep understanding of the algorithm and its parameters, we refer the reader to the publications mentioned. Most of the terms and parameter names used here come directly from these publications.

The parameters of the DenStream algorithm mainly affect the following properties of the algorithm:

- the coarseness of the micro-clusters,
- the maximum distance between data points/micro-clusters so that they are assigned to the same cluster,
- the minimum density so that data points are identified as clusters and not as outliers,
- The fading rate at which older data points lose their significance.

The *Parameters Epsilon, Lambda* and *Mu x Beta* belong to the first phase of the algorithm, the formation of micro-clusters.

If possible, a data point is inserted into the micro-cluster whose center is closest to the data point. For this purpose, the Euclidean distances between the data point and the center points of all micro-clusters are compared and the micro-cluster with the smallest distance is selected. The data point can only be inserted into the micro-cluster if the radius of the micro-cluster does not exceed the *Epsilon* threshold after insertion. The radius is analogous to the variance of all data points contained in the micro-cluster. This means that data points can also be integrated into a micro-cluster even if their Euclidean distance to the center of the cluster is greater than *Epsilon*, as long as there are enough other points in the micro cluster with a smaller distance.

If the data point cannot be inserted into the nearest micro-cluster, a new micro-cluster is created with that data point. The weight of the respective micro-cluster is increased by one with the insertion of a data point.

In the left-hand plot in the illustration, the assignment of the data points to the micro-clusters is sketched for two input channels as an example. 20 data points assigned to four different micro-clusters are shown. The first micro-cluster (#1, marked in red) contains six data points, the second (#2, marked in green) also contains six, the third (#3, marked in blue) contains seven and the fourth (#4, marked in gray) contains only one data point. The area around the center of the micro-cluster in which a new data point would have to be located in order to be accepted into the respective micro-cluster with the specified epsilon (marked by dashed line) is colored. This sphere of influence is greater if the micro-cluster already contains several data points and they have a lower variance (see, for example, micro-clusters #1 and #2). In addition, the spheres of influence of multiple micro-clusters can overlap and mutually influence one another through their existence, see micro-cluster #2 (green) and #3 (blue). The data points are always assigned to the closer micro-cluster, for which reason the spheres of influence are separated by a straight line. In the plot, a data point can be seen that is assigned to the micro-cluster #2, but if the latter did not exist, then it would be assigned to micro-cluster #3.

Like in the original study [1], micro-clusters are divided into potential and outlier micro-clusters depending on their weight. Only potential micro-clusters are subsequently clustered by the DBSCAN algorithm. The data points in the outlier micro-clusters are marked as outliers. However, outlier micro-clusters are also stored and updated with new data points, as they can still develop into potential micro-clusters. The weight of a micro-cluster must exceed the $Beta \times Mu$ threshold in order to be counted as a potential micro-cluster. In the left-hand sketch in the illustration, for example, the micro-cluster #4 (gray) contains only one data point, thus has a weight of less than or equal to one and would be counted as an outlier micro-cluster for $Beta \times Mu = 1$.

When a fading function is applied, the weight of the micro-clusters decreases over time. This fading rate is determined by the parameter $Lambda$. If the value is set to zero, no fading function is applied, otherwise the weights decrease by a factor of $2^{(-Lambda)}$ every second. If the weight of an outlier micro-cluster falls below an internal threshold (depending on $Mu \times Beta$ and $Lambda$), it is deleted from the memory.

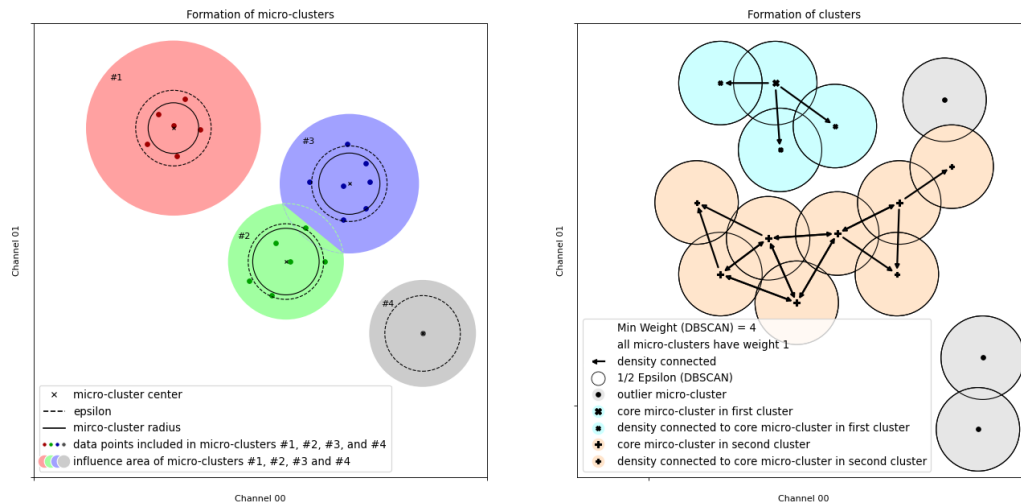
The parameters *Epsilon (DBSCAN)* and *Min Weight (DBSCAN)* affect the second phase. These parameters were adopted from the DBSCAN algorithm [3].

The DBSCAN algorithm runs over the set of potential micro-clusters and assigns them cluster designations. This can be either the index of the cluster to which they belong, or the designation outlier. The data point currently being processed is then assigned the name of the micro-cluster to which it belongs.

How does DBSCAN cluster the micro-clusters? The algorithm works according to the concept of density accessibility. Objects (in this case micro-clusters) belong to the same cluster if they are density-connected. This means that there must be a chain of micro-clusters with a maximum distance *Epsilon (DBSCAN)*. All micro-clusters forming this chain must satisfy a second condition. The sum of the weights of all micro-clusters within the distance *Epsilon (DBSCAN)* around each individual micro-cluster in this chain must exceed the threshold *Min Weight (DBSCAN)*. Micro-clusters that are not density-connected to at least one micro-cluster that meets this second condition are marked as outliers.

This is shown in the right-hand sketch in the illustration. For simplicity, it is assumed here that the weight of all micro-clusters is equal to 1. This corresponds to the case where there is exactly one data point in each micro-cluster and no fading function has been applied. The two clusters (marked with an x (turquoise) and a plus (orange)) with the two outlier micro-clusters result when the *Min Weight (DBSCAN)* parameter is set to four. The micro-clusters marked with a capital "x" or "+" are core micro-clusters. This means that at least three more micro-clusters (plus the micro cluster considered = 4) have a maximum distance of *Epsilon (DBSCAN)* to these micro-clusters. The micro-clusters marked with a small "x" or "+" are not core micro-clusters, but are located in the *Epsilon (DBSCAN)* neighborhood of a core micro-cluster and therefore belong to the same cluster. The micro-cluster in the upper right corner, marked with a small dot, is an outlier micro-cluster. Although it is located in the *Epsilon (DBSCAN)* neighborhood of a micro-cluster that is counted as belonging to a cluster, it is not a core micro-cluster.

Likewise, the two micro-clusters at the bottom right are outliers. Although they are in the immediate *Epsilon (DBSCAN)* neighborhood, there are only two of them. The *Min Weight (DBSCAN)* threshold of the weights is not exceeded.



The parameters *outMCs Buffer Size* and *potMCs Buffer Size* are specific to this implementation of the algorithm and are required because the memory for outliers and potential micro-clusters must be allocated before execution. Thus, *outMCs Buffer Size* and *potMCs Buffer Size* limit the possible number of outliers and potential micro-clusters during runtime. The user must find values for these parameters so that this limit is not exceeded.

The maximum number of outliers and potential micro-clusters during the execution of the algorithm depends on the distribution of the input data, but also on the setting of the other parameters. There are fewer micro-clusters at higher values of *Epsilon* as this results in coarser micro-clusters that can contain data points from a wider range. In general, the number of outlier micro-clusters increases at the beginning of the analysis, but decreases again when outlier micro-clusters transform into potential micro-clusters. If the patterns in the data stream do not change over time, the number of micro-clusters settles after an initial phase.

The more micro-clusters there are, the higher the computing requirements. For all outliers and potential micro-clusters we compare the distance to a data point and then all potential micro-clusters must be included in the calculation of the DBSCAN algorithm. A compromise must therefore be reached between the computing speed and the coarseness of the micro-clusters.

What happens if the values of *outMCs Buffer Size* and *potMCs Buffer Size* are set too low and at some point during the analysis more micro-clusters are required to capture the input data points? In this case, the algorithm continues to assign the data points to the existing micro-clusters and marks the data points accordingly, but the existing micro-clusters are no longer updated to prevent the buffer from overflowing. This means that the clustering of the data points continues, but with an overall stagnated feature space (older set of micro-clusters). Changes in the pattern of the data stream could then no longer be detected.

[1] F. Cao, M. Ester, W. Qian, A. Zhou. Density-Based Clustering over an Evolving Data Stream with Noise. In Proceedings of the 2006 SIAM International Conference on Data Mining, S. 326-337. SIAM.

[2] M. Ester, H.-P. Kriegel, J. Sander and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD*, 1996.

[3] J. Sander, M. Ester, H.-P. Kriegel, X. Xu. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. *Data Mining and Knowledge Discovery* 2, 169-194 (1998)

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Input values

- **Update Micro Cluster:** If *TRUE*, micro-clusters are updated by the incoming data. If *FALSE*, the existing micro-clusters remain unchanged and are only used to determine the cluster index of the incoming data points.
- **Input 01, ..., Input 0n:** These inputs form the feature space for which clustering is performed.

Configuration options

- **Num Channels:** Here you can modify the number of input channels.
- **Epsilon:** Threshold for the maximum radius of micro-clusters.
- **Mu x Beta:** Threshold for the weight of a micro-cluster between outlier and potential micro-cluster.
- **Lambda:** Specifies the fading rate of the algorithm. The weight of each data point decreases by a factor of $2^{(-\text{Lambda})}$ every second.
- **Epsilon (DBSCAN):** Specifies the epsilon parameter of the DBSCAN algorithm.
- **Min Weight (DBSCAN):** Threshold for the sum of weights in the epsilon neighborhood of a micro-cluster for the DBSCAN algorithm.
- **potMCs Buffer Size:** Maximum number of potential micro-clusters. The memory is allocated to *potMCs Buffer Size* micro-clusters.
- **outMCs Buffer Size:** Maximum number of outlier micro-clusters. The memory is allocated to *outMCs Buffer Size* micro-clusters.

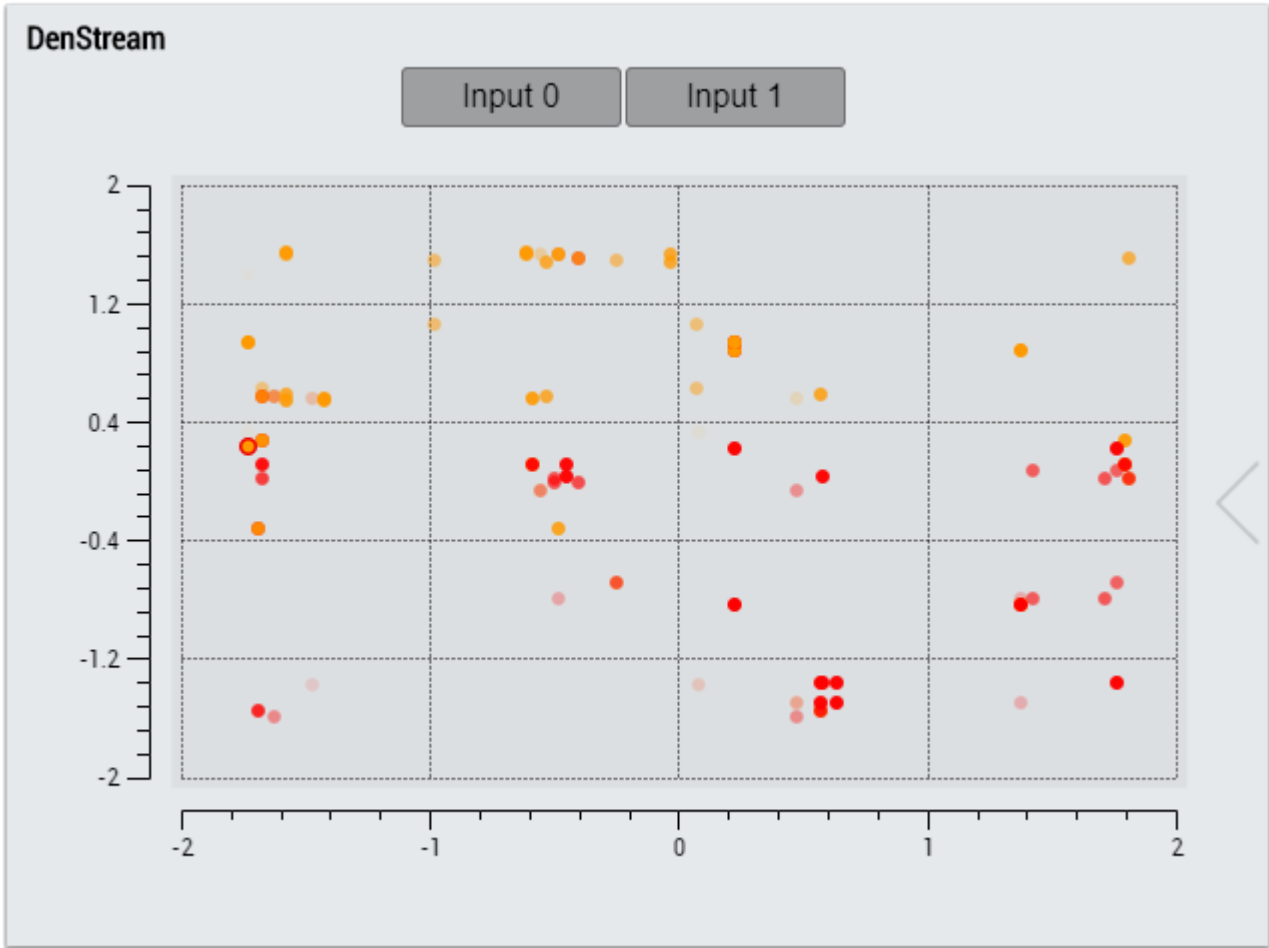
Output values

- **New Result:** Is TRUE if the new cluster index differs from the cluster index of the last cycle.
- **MC Buffer Overflow:** TRUE if the micro-cluster update is stopped to prevent overflow of *potMCs* or *outMCs Buffer Size*.
- **Last Event:** This is the timestamp of the last cycle with a change of the cluster index
- **Last Switch:** This is the timestamp of the last cycle with an alternation between updating and not updating micro-clusters (either by setting the input *Update Micro Cluster* to TRUE or by internally preventing an overflow of *potMCs/outMCs Buffer Size*).
- **Number of potMCs:** Indicates the number of potential micro-clusters currently present.
- **Number of outMCs:** Indicates the number of outlier micro-clusters currently present.
- **Cluster Index:** Specifies the cluster index that the DBSCAN algorithm outputs for the data point of the current cycle.
- **Number of Clusters:** Specifies the total number of clusters detected by the DBSCAN algorithm.

Standard HMI Controls

For the DenStream algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The DenStream control visualizes the inputs in the chart and their respective classification in a cluster (cluster index) in color. The buttons can be used to select the inputs to be displayed, up to two at a time. The slider on the right lists all output values: cluster count, number of potential micro-clusters, number of outlier micro-clusters, cluster index, last event, last switch.

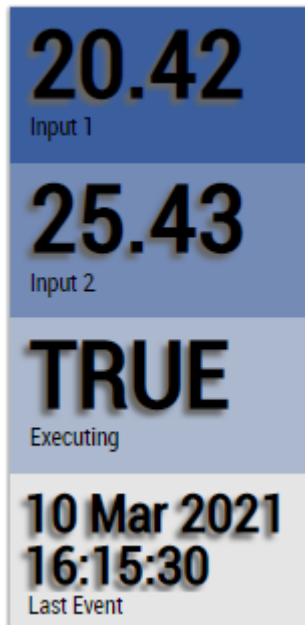


2. The Table Control or Multivalue Control visualizes all output values: cluster count, number of potential micro-clusters, number of outlier micro-clusters, cluster index, last event, last switch.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the DenStream algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.3.2 Sequential k-Means

Sequential k-Means			
	Update Cluster C...	<Empty>	-
	Input 00	<Empty>	-
	Num Channels		+ -
	Number of Clusters	2	
	Aggregation Buffer Size	10	
	Max Iterations	1	
	Initialization Mode	Equidistant	
Lower Bounds 00	-1000		
Upper Bounds 00	1000		
Cluster Index	Empty		
Distance	Empty		
Cluster Cente...	0		
Cluster Cente...	0		

The *Sequential k-Means* algorithm is an implementation of the unmonitored clustering algorithm of the same name. It is a sequential variant of the widely used k-Means clustering algorithm for streaming data. The aim of the algorithm is to find clusters based on the structure of the data, each of which contains similar data points and separates different data points from each other.

The number of input channels (referred to below as n) for this algorithm can be freely selected by the user. These inputs span the n -dimensional feature space in which the clusters are found. In each analysis cycle, the data stream provides the algorithm with a new feature vector that can be interpreted as a data point in this feature space. Data points that are close to each other in this feature space are assigned to the same cluster. The number of clusters present must be set by the user before the analysis begins and remains fixed.

In contrast to the k-Means algorithm for conventional batch analysis, the data for the *Sequential k-Means* are not fully available at the time of analysis. Instead, the data points arrive one by one in the form of streaming data. They are therefore processed sequentially and assigned to the corresponding cluster closest to them. This approach results in a number of differences, two of which are particularly relevant to the use of the algorithm as well as the parameter settings.

On the one hand, all data points and thus the value ranges of the individual features are already available at the beginning of a batch analysis, whereas this is not the case with sequential analysis, so that the value ranges are not necessarily fixed in advance. However, it is helpful to know the value ranges of the input channels in advance, even if the actual values only arrive during the course of the analysis. This is particularly important for the initialization of cluster centers. Three different approaches are available for initialization. The center points can be specified in the form of specific *values* via a parameter array. Alternatively, the center points can be set *randomly* or *equidistantly* in a defined range of values. For the initialization modes *Random* and *Equidistant* the value ranges are required and have to be set via the parameters *Lower Bounds* and *Upper Bounds* for the individual input channels.

On the other hand, in a batch analysis all data points are typically traversed multiple times to update the cluster centers until they change only minimally. This is not possible within the framework of the sequential analysis. However, in order to still be able to adjust the cluster centers and traverse data points multiple times, the algorithm *Sequential k-Means* has a buffering mechanism referred to as *Aggregation Buffer*, which makes it possible to store a limited number of values temporarily. When filling the buffer, all incoming data points are assigned to the closest cluster. The distance between a data point and the cluster centers is determined by the Euclidean norm. Only when the buffer is filled are the cluster centers updated based on the newly allocated data points in the buffer. The new cluster center corresponds to the mean value of all data points contained in the cluster. This can be calculated incrementally, so that the old data points are not needed for the calculation. The size of the buffer is set by the parameter *Aggregation Buffer Size*; the default value is 10. The parameter *Max Iterations* can be used to specify the number of iterations through the buffer. The default value is one. If the value is set to two, for example, after the first adjustment of the cluster centers the data points in the buffer are reassigned to the clusters and then the cluster centers are adjusted again. Due to the shift in cluster centers, it is possible for individual data points to be assigned to different clusters from one iteration to the next. Due to the limited computing capacity for data processing between two cycles, excessively high values should be avoided for the parameters *Aggregation Buffer Size* and *Max Iterations*, otherwise the update of the cluster centers may not be guaranteed. If the cluster centers are not updated for large values for these parameters but are updated for smaller parameter values, this is an indication that the computing capacity is insufficient for the set parameter values and smaller values should be selected.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Input values

- **Update Cluster Centers:** If *TRUE*, the centers of each cluster are updated by the incoming data. If *FALSE*, the cluster centers remain unchanged and are only used to determine the cluster index of the incoming data points.
- **Input 01, ..., Input 0n:** These inputs form the n-dimensional feature space for which clustering is performed.

Configuration options

- **Num Channels:** Determines the number of input channels.
- **Number of Clusters:** Defines the number of clusters.
- **Aggregation Buffer Size:** Specifies the size of the aggregation buffer and thus the number of cycles after which the cluster centers are updated. The input values of these cycles are stored internally (in the aggregation buffer). The default value for this parameter is 10.
- **Max Iterations:** Specifies how often to iterate over the values in the aggregation buffer. The default value for this parameter is 1.
- **Initialization Mode:** Specifies the way in which the cluster centers are initialized:
 - **Random:** The cluster centers are set randomly within the limits set by the *Lower Bounds* and *Upper Bounds*.
 - **Equidistant:** The cluster centers are distributed equidistantly in the range of values defined by the *Lower Bounds* and *Upper Bounds*.
 - **Values:** The cluster centers are initialized with the values set by the array *Initial Cluster Centers*.
- **Initial Cluster Centers:** For the *initialization mode Values* the values for the initial cluster centers are set here. The values for the individual clusters are set line by line. That is, the number of matrix rows corresponds to the *Number of Clusters* and the number of matrix columns corresponds to the *Number of Channels*. The first row contains the values for the first cluster for each input channel, and so on.

- **Lower Bounds:** For the modes *Random* and *Equidistant* the lower limits for the individual input channels are set.
- **Upper Bound:** For the modes *Random* and *Equidistant* the upper limits for the individual input channels are set.

Output values

- **Cluster Index:** Specifies the cluster index assigned to the data point of the last cycle, indicating the corresponding assigned cluster.
- **Distance:** Specifies the Euclidean distance between the data point and the assigned cluster center.
- **Cluster Centers:** Outputs the cluster centers of all clusters line by line. This corresponds to a matrix of dimension *Number of Clusters* x *Number of Channels*.

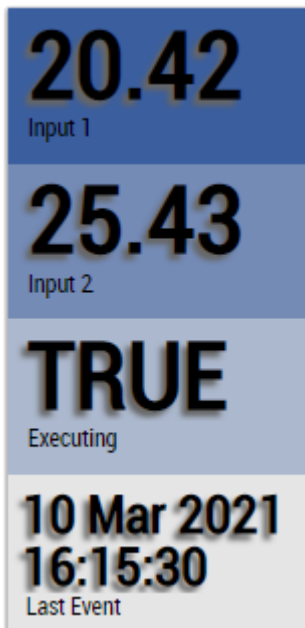
Standard HMI Controls

For the *Sequential k-Means* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control or Multivalue Control visualizes the output values: Lower Bounds, Upper Bounds and Initial Centers.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the *Sequential k-Means* algorithm using the Mapping Wizard [► 382].

5.2.1.4 Analytics - Compare

The algorithms of the category *Analytics-Compare* provide functionalities for comparative analysis and logic operations.

5.2.1.4.1 Demultiplexer

Demultiplexer			
	Input	<Empty>	Num Channels

Current Chan...	Empty
Count	Empty
Last Event	Empty
Output 00	Empty
Output 01	Empty

The demultiplexer selects an output channel based on the input value. For this purpose, the input value is interpreted as an integer. This value corresponds to the output channel. If the value is outside the configured number of channels, the output channel is set to 0.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **NumChannels:** Adds or removes an output channel.

Output values

- **Current Channel:** Indicates the number of the selected channel. The value is 0 if the selected channel is outside the configured channels.
- **Count:** Starts with 1 for the channel selected at the start of the analysis and increments each time another channel is selected.
- **Last Event:** Timestamp of the last channel change.
- **Out 00..n:** Boolean output for channels 0 (selected channel < 1 or > n) to n

Standard HMI Controls

For the Demultiplexer algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control or Multivalue Control visualizes the output values Count, Current Channel and Last Event and Out.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the Demultiplexer algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.2 Detect String Change 1Ch

Detect String Change 1Ch			
a→A	Input	<Empty>	-
	Case Sensitive		☒
	String Changed		Empty
Count		Empty	
Last Event		Empty	

The *Detect String Change 1Ch* detects and counts changes of string values. Therefore, case sensitivity can be taken into account or not.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Case Sensitivity:** If the checkbox is ticked off, case sensitivity is taken into account, otherwise not.

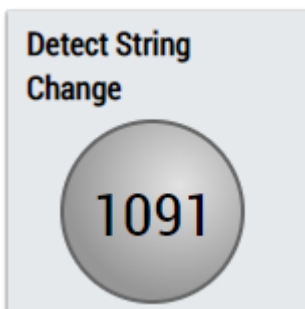
Output values

- **String Changed:** *TRUE*, if a string change was detected, otherwise *FALSE*.
- **Count:** Count up every time Boolean Switch is *TRUE*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

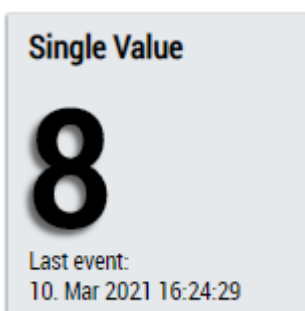
Standard HMI Controls

For the Detect String Change 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

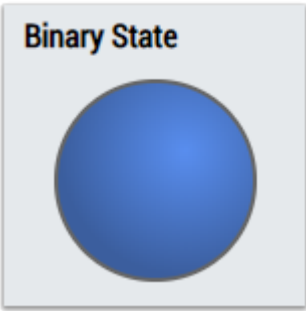
1. The Comparison control visualizes the output values Boolean Switch and Count.



2. The SingleValue control visualizes the output values Count and Last Event.



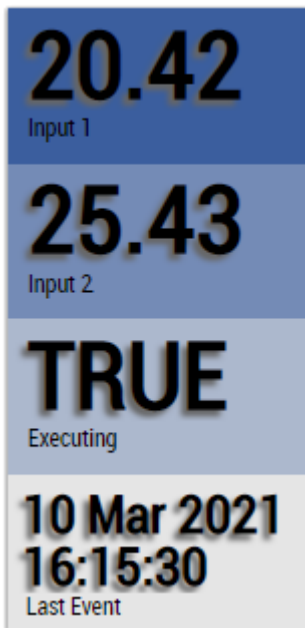
3. The BinaryState control visualizes the output value Boolean Switch.



4. The Table Control or Multivalue Control visualizes all output values: Operation Out, Count, Last Event.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Detect String Change 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.3 Detect Value Change 1Ch

Detect Value Change 1Ch			
< >	Input	<Empty> ▼	Tolerance 0
			Value Changed Empty
			Count Empty
			Last Event Empty

The *Detect Value Change 1Ch* detects and counts changes in numeric input values.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Tolerance:** The tolerance refers to the input value of the last detected value change. If the input value is outside this tolerance, a value change is detected.

Output values

- **Value Changed:** *TRUE*, if a value change was detected, otherwise *FALSE*.
- **Count:** Count up every time Boolean Switch is *TRUE*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

Standard HMI Controls

For the *Detect Value Change 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SingleValue control visualizes the output values Count and Last Event.

Single Value

8

Last event:
10. Mar 2021 16:24:29

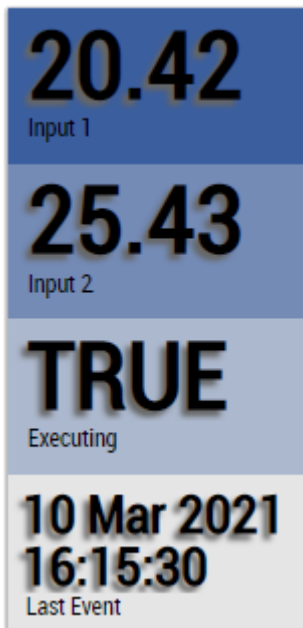
2. The Table Control and Multivalue Control visualize all output values.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped using the [Mapping Wizard](#) [► 382].

5.2.1.4.4 Dynamic Time Warping

Dynamic Time Warping					
	Input	<Empty>	-	Num Templates	+ -
	Start Period	<Empty>	-	File Path 00	C:\Analytics Project\Template1 ... 
	Stop Period	<Empty>	-	File Path 01	C:\Analytics Project\Template2 ... 
					Executing Co... Empty
					Best Match Idx Empty
					Distance 01 Empty
					Distance 02 Empty

The *Dynamic Time Warping* algorithm compares input data with previously recorded templates. The special feature of the algorithm is that signals with different speeds or even shifted signals can be compared. As a result, the distance between the input signal and the respective template is output. The smaller the distance, the more equal the compared signals are. If the distance is 0, both signals are identical. The amount of distance depends on the equality but also on the length of the signals.

The comparison starts when the signal of the Start Period flag is *TRUE*. A result is output if the signal of the Stop Period flag is *TRUE* or the Start Period flag is *TRUE* again.

It is recommended not to use *Dynamic time Warping* simultaneously with [Time Based Teach Path 1Ch](#) [► 219] due to concurrent file access. Instead, a reference signal should first be taught in with the [Time Based Teach Path 1Ch](#) [► 219] and only then should the evaluation be carried out with the aid of the *Dynamic time Warping*. The templates contain reference signals previously recorded with the [Time Based Teach Path 1Ch](#) [► 219]. As a rule, the templates are a few hundred supporting points. Therefore, a reduction of the data with the function block [Downsampling 1Ch](#) [► 44] is often useful.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Templates:** adds or removes a template for comparison.
- **File Path 01 ... n:** path to the previously recorded template.

Output values

- **Executing Compare:** displays *TRUE* if the algorithm is processing the incoming values, otherwise *FALSE*. The processing begins when the Start Period flag is *TRUE* and ends when Stop Period is *TRUE*.
- **Best Match Idx:** outputs the index of the template with the smallest distance to the input channel.
- **Distance 01 ... n:** specifies the distance between the input signal and the respective template. The smaller the distance, the more equal the compared signals are.

Standard HMI Controls

For the *Dynamic Time Warping* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

- 1. The Table Control and Multivalue Control visualize all output values: *Executing Compare*, *Best Match Idx*, *Distance*.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the *Dynamic Time Warping* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.5 Dynamic Time Warping Interval

Dynamic Time Warping Interval			
Reference Input	<Empty>	-	Num Channels
Input 01	<Empty>	-	Window Size
			Best Match Idx
			Distance 01

The *Dynamic Time Warping Interval* algorithm compares several input data with each other. The special feature of the algorithm is that signals with different speeds or even shifted signals can be compared. Only the signal interval of a configured window is considered for the comparison. New results are output after the window expires. As a result, the distance between the reference signal and the respective input signal is output. The smaller the distance, the more equal the compared signals are. If the distance is 0, both signals are identical. The amount of distance depends on the equality but also on the length of the signals.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** adds or removes an input channel.
- **Window Size:** specifies the number of cycles over which a calculation is made. The memory requirement of the algorithm is proportional to this parameter.

Output values

- **Best Match Idx:** outputs the index of the input channel with the smallest distance to the reference channel.
- **Distance 01 ... n:** specifies the distance between the reference signal and the respective input channel. The smaller the distance, the more equal the compared signals are.

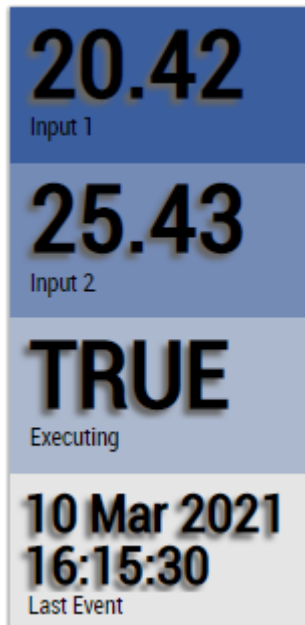
Standard HMI Controls

For the *Dynamic Time Warping Interval* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control and Multivalue Control visualize all output values: *Best Match Idx*, *Distance*.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the *Dynamic Time Warping Interval* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.6 Logic Operation Counter

Logic Operation Counter			
&	Input 00	<Empty>	-
	Input 01	<Empty>	-
	Num Channels		+ -
	Logic Operator		≥1
Count Mode		OnChange	▼
Threshold 00		▲ 1	
Threshold 01		▲ 1	
Operation Out		Empty	
Count		Empty	
Last Event		Empty	

The *Logic Operation Counter* executes a logical operation on the values of two or more channels and provides the result of this logical operation. Therefore, each input value can be combined with a threshold and an operator. Furthermore, the logic operator and the count mode can be configured individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** adds or removes an input channel.
- **Threshold 00:** threshold for the signal of the first channel.
- **Threshold 01 ..n:** threshold for the signal of the second to nth channel.
- **Logic Operator:** logical operator for the operation:

≥1 Logical OR

=1 Logical XOR (EXCLUSIVE OR)



Logical AND



Logical NAND (NOT AND)



Logical NOR (NOT OR)

- **Count Mode:** mode of the result counter.
OnChange: the counter counts every time the result changes to **TRUE**.
Cyclic: the counter increments every cycle when the condition is **TRUE**.
- **Use Absolute Values:** if the checkbox is checked, the absolute values of the input signal are used.
- **Tolerance (optional):** tolerance value for the Equal / NotEqual comparisons.

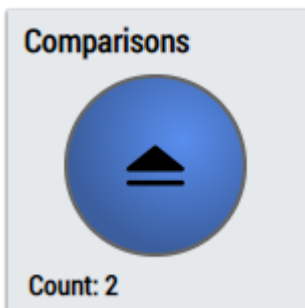
Output values

- **Operation Out:** Result of the logical operation.
- **Count:** Incremented when the output value is **TRUE**. The behavior depends on the parameter *Count Mode*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

Standard HMI Controls

For the Logic Operation Counter algorithm, the following HMI controls are available for generating an Analytics Dashboard:

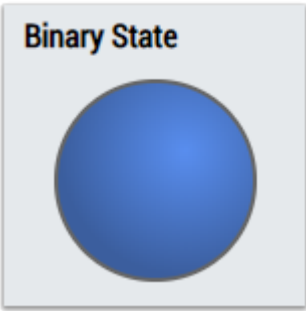
1. The Comparison control visualizes the output values Operation Out and Count as well as the configuration option Logic Operator.



2. The SingleValue control visualizes the output values Count and Last Event.



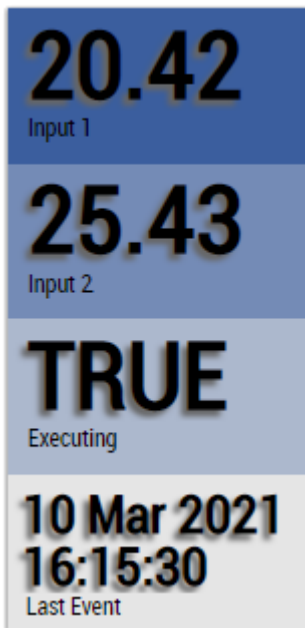
3. The BinaryState control visualizes the output value Operation Out.



4. The Table Control or Multivalue Control visualizes all output values: Operation Out, Count, Last Event.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Logic Operation Counter algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.7 Multiplexer

Multiplexer					
	Default Result	<Empty>	-	Num Channels	+ -
	Condition 00	<Empty>	-		
	Input 00	<Empty>	-		
				Result	Empty
				Current Chan...	Empty
				Count	Empty
				Last Event	Empty

The Multiplexer selects one channel out of one or more input channels. For each input channel a boolean input has to be provided additionally. The output corresponds to the first input channel, where the conditional input is **TRUE**. The priority of the configured channels is the order of configuration. If the condition is not met for any of the channels, the provided default channel is returned.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is **TRUE**.

Configuration options

- **Num Channels:** The number of channels. For each channel there are two input variables; one is a boolean condition (*Condition0n*), the other is an input value of any data type (*Input0n*) which can be passed to *Result* if the condition is met.

Output values

- **Result:** Delivers the signal *Input0n* of the selected input channel.
- **Current Channel:** Indicates the number of the selected channel. The value is 0 if the default result is selected. The input channels are numbered in the order of their configuration.
- **Count:** Starts with 1 for the channel selected at the start of the analysis and increments each time another channel is selected.
- **Last Event:** Timestamp of the last channel change.

Standard HMI Controls

For the Multiplexer algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control or Multivalue Control visualizes the output values Count, Result, Current Channel and Last Event.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Multiplexer algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.8 Numerical Compare 1Ch

Numerical Compare 1Ch					
< >	Input	<Empty>	Operator	▲ 0	Operation Out Empty
			Count Mode	OnChange	Count Empty
			Use Absolute Values	☐	Last Event Empty

The *Numerical Compare 1Ch* compares the input values with a reference value and provides the result of this comparison operation. The operator, the reference value and the count mode can be configured individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Operator:** specifies whether the input value should be greater than, greater than or equal to, less than or equal to, less than or not equal to the reference value.
- **Reference:** reference value for the comparison operation.
- **Count Mode:** mode of the result counter.
OnChange: the counter counts every time the result changes to *TRUE*.
Cyclic: the counter increments every cycle when the condition is *TRUE*.
- **Use Absolute Values:** if the checkbox is checked, the absolute value of the input signal is used.
- **Tolerance (optional):** tolerance value for the Equal / NotEqual comparisons.

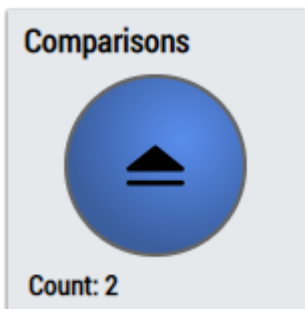
Output values

- **Operation Out:** Result of the comparison operation.
- **Count:** Incremented when the output value is *TRUE*. The behavior depends on the parameter *Count Mode*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

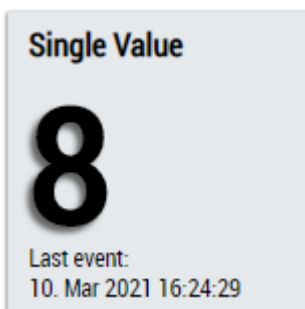
Standard HMI Controls

For the Numerical Compare 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

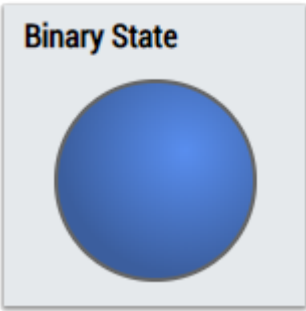
1. The Comparison Control or Multivalue Control visualizes the output values Operation Out and Count as well as the configuration option Operator.



2. The SingleValue control visualizes the output values Count and Last Event.



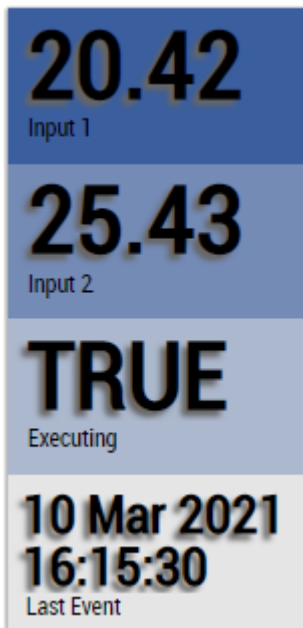
3. The BinaryState control visualizes the output value Operation Out.



4. The Table Control and Multivalue Control visualize all output values: Operation Out, Count, Last Event.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the *Numerical Compare 1Ch* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.9 Numerical Compare 2Ch

Numerical Compare 2Ch						
< >	Input Ch1	<Empty>	-	Operator (Ch1 Operator Ch2)	GreaterThanOrEqualTo	Operation Out Empty
	Input Ch2	<Empty>	-	Count Mode	OnChange	Count Empty
	Use Absolute Values				☐	Last Event Empty

The *Numerical Compare 2Ch* compares the input values of the first channel with the input values of the second channel and provides the result of this comparison operation. The operator and the count mode can be configured individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Operator:** specifies whether the input value should be greater than, greater than or equal to, less than or equal to, less than or not equal to the reference value.
- **Count Mode:** mode of the result counter.
OnChange: the counter counts every time the result changes to *TRUE*.
Cyclic: the counter increments every cycle when the condition is *TRUE*.
- **Use absolute values:** if the checkbox is checked, the absolute values of the input signal are used.
- **Tolerance (optional):** tolerance value for the Equal / NotEqual comparisons.

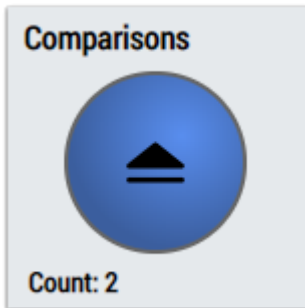
Output values

- **Operation Out:** Result of the comparison operation.
- **Count:** Incremented when the output value is *TRUE*. The behavior depends on the parameter *Count Mode*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

Standard HMI Controls

For the *Numerical Compare 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

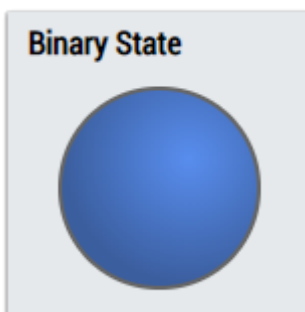
1. The Comparison Control or Multivalue Control visualizes the output values Operation Out and Count as well as the configuration option Operator.



2. The SingleValue control visualizes the output values Count and Last Event.



3. The BinaryState control visualizes the output value Operation Out.



4. The Table Control visualizes all output values: Operation Out, Count, Last Event.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42	Input 1
25.43	Input 2
TRUE	Executing
10 Mar 2021 16:15:30	Last Event

Alternatively, customer-specific HMI controls can be mapped in the Numerical Compare 1Ch algorithm using the [Mapping Wizard](#) [[▶ 382](#)].

5.2.1.4.10 String Compare 1Ch

String Compare 1Ch										   		
abc <> acb	Input	<Empty>	-	Reference String	EmptyString					String Match	Empty	
				String Compare Mode	Equals					Count	Empty	
				Case Sensitive	☒					Last Event	Empty	
				Count Mode	OnChange							

The *String Compare 1Ch* compares the input string with a reference string and counts the string matches. Therefore, case sensitivity can be taken into account or not and the count mode can be changed.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Reference String:** reference string for the comparison operation.
- **String Compare Mode:** enumeration for different string compare modes.
Equals: input string equals the reference string.
BeginsWith: input string begins with the reference string.
Contains: input string contains the reference string.
- **Case Sensitivity:** if the checkbox is checked the input is case-sensitive, otherwise it is not case-sensitive.
- **Count Mode:** mode of the result counter.
OnChange: the counter counts every time the result changes to *TRUE*.
Cyclic: the counter increments every cycle when the condition is *TRUE*.

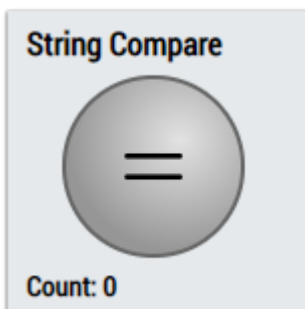
Output values

- **String Match:** *TRUE*, if the input string matches the reference string, otherwise *FALSE*.
- **Count:** Incremented when the output value is *TRUE*. The behavior depends on the parameter *Count Mode*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

Standard HMI Controls

For the String Compare 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

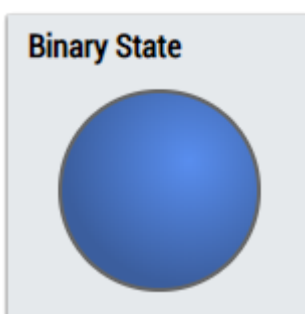
1. The Comparison control visualizes the output values String Match and Count as well as the configuration option String Compare Mode.



2. The SingleValue control visualizes the output values Count and Last Event.



3. The BinaryState Control visualizes the output value String Match.



4. The Table Control or Multivalue Control visualizes all output values: String Match, Count, Last Event.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021 16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the String Compare 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.4.11 String Compare 2Ch

		String Compare 2Ch					
abc<>acb	Input Ch1	<Empty>	-	String Compare Mode	Equals	String Match	Empty
	Input Ch2	<Empty>	-	Case Sensitive	☒	Count	Empty
				Count Mode	OnChange	Last Event	Empty

The *String Compare 2Ch* compares the values of the first input string with the values of the second string and counts the string matches. Therefore case sensitivity can be taken into account or not and the count mode can be changed.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **String Compare Mode:** enumeration for different string compare modes.
Equals: input string equals the reference string.
BeginsWith: input string begins with the reference string.
Contains: input string contains the reference string.
- **Case Sensitivity:** if the checkbox is checked the input is case-sensitive, otherwise it is not case-sensitive.
- **Count Mode:** mode of the result counter.
OnChange: the counter counts every time the result changes to *TRUE*.
Cyclic: the counter increments every cycle when the condition is *TRUE*.

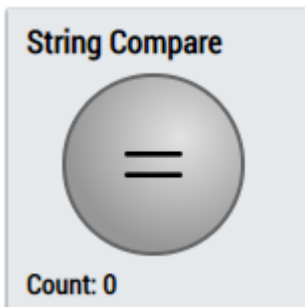
Output values

- **String Match:** *TRUE*, if the value of the first input string matches the value of the second input string, otherwise *FALSE*.
- **Count:** Incremented when the output value is *TRUE*. The behavior depends on the parameter *Count Mode*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

Standard HMI Controls

For the String Compare 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

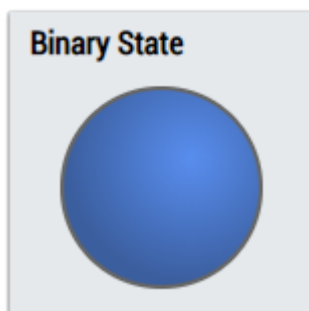
1. The Comparison control visualizes the output values String Match and Count as well as the configuration option String Compare Mode.



2. The SingleValue control visualizes the output values Count and Last Event.



3. The BinaryState Control visualizes the output value String Match.



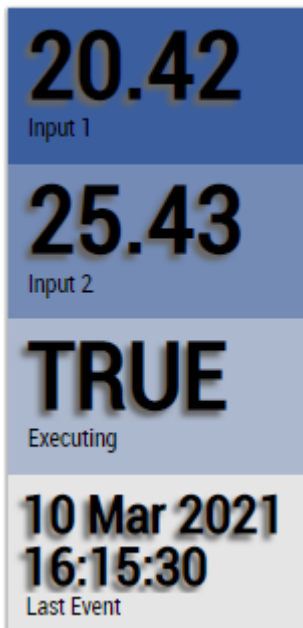
4. The Table Control or Multivalue Control visualizes all output values: String Match, Count, Last Event.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the String Compare 2Ch algorithm using the Mapping Wizard [► 382].

5.2.1.4.12 Timespan Compare 1Ch

Timespan Compare 1Ch					
< >	Input	<Empty>	-	Reference	Seconds 1
				Count Mode	OnChange
				Use Absolute Values	☐
				Operation Out	Empty
				Count	Empty
				Last Event	Empty

The *Timespan Compare 1Ch* compares the input value with a reference value and returns the result of this comparison operation. A timespan with a resolution of 1 ns is supported as an input. The operator, the reference value and the count mode can be configured individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Operator:** specifies whether the input value should be greater than, greater than or equal to, less than or equal to, less than or not equal to the reference value.
- **Reference:** Reference timespan for the comparison operation.
- **Count Mode:** Mode of the result counter.
OnChange: The counter counts every time the result changes to *TRUE*.
Cyclic: The counter increments every cycle when the condition is *TRUE*.
- **Use Absolute Values:** If the checkbox is checked, the absolute value of the input signal is used.

Output values

- **Operation Out:** Result of the comparison operation.
- **Count:** Incremented when the output value is *TRUE*. The behavior depends on the parameter *Count Mode*.
- **Last Event:** Indicates the time of the last triggered event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event

Standard HMI Controls

For the Timespan Compare 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control visualizes all output values: Operation Out, Count, Last Event.

Alternatively, customer-specific HMI controls can be mapped in the *Timespan Compare 1Ch* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.5 Analytics - Energy

5.2.1.5.1 Energy

Energy					
	Voltage	<Empty>	-	Sample Rate	1000
	Current	<Empty>	-	Voltage Scale Factor	1
				Current Scale Factor	1
				Electrical Pow...	Empty
				Electrical Ener...	Empty

The *Energy module* is suitable for energy monitoring of various electrical components. The input data for current and voltage are averaged over a configurable interval and then calculated.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Voltage Scale Factor:** Scale factor of the input voltage.
- **Current Scale Factor:** Scale factor of the input current.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **Voltage:** RMS value of the voltage in V.
- **Current:** RMS value of the current in A.

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.2 Energy AX5xxx

Energy AX5xxx					
	DC Bus Voltage	<Empty>	-	Sample Rate	1000
	DC Bus Current	<Empty>	-	Peak Torque	0
	Velocity Feedback...	<Empty>	-		
	Torque Feedback...	<Empty>	-		
	Internal Braking...	<Empty>	-		
	External Braking...	<Empty>	-		
				Electrical Pow...	Empty
				Mechanical P...	Empty
				Braking Powe...	Empty
				Electrical Ener...	Empty

The *Energy AX5xxx module* is suitable for energy monitoring of servo drives from the AX5xxx series. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Peak Torque:** Peak value of the torque in Nm
Process data: CoE: P-0-0094
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **DC Bus Voltage:** DC bus voltage in 0.1 V.
Process data: CoE: S-0-0016. Add that of the CoE S-0-0380 "DC bus voltage" to the PDO content.

- **DC Bus Current:** Input current to the DC link in mA.
Process data: CoE: S-0-0016. Add that of CoE S-0-0381 "DC bus current" to the PDO content.
- **Velocity Feedback Value:** Motor velocity.
Process data: CoE: S-0-0016. Add that of the CoE S-0-0040 "Velocity feedback 1 value" to the PDO content.
- **Torque Feedback Value:** Torque-forming current.
Process data CoE: S-0-0016. Add that of CoE S-0-0084 "Torque/force feedback value" to the PDO content.
- **Internal Braking Resistor Power:** Braking power output via the internal braking resistor in W.
Process data: CoE: S-0-0016. Add that of the CoE P-0-0209 "Internal braking resistor actual averaged power" to the PDO content.
- **External Braking Resistor Power:** Braking power output via the external braking resistor in W.
Process data: CoE: S-0-0016. Add the CoE P-0-0210 "External braking resistor actual averaged power" to the PDO content.

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Mechanical Power [W]:** Mechanical power in W.
- **Braking Power [W]:** Braking power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.3 Energy AX81xx

Energy AX81xx				
	Electrical Power	<Empty>	-	Sample Rate
	Mechanical Power	<Empty>	-	1000
	DC Link Voltage	<Empty>	-	
	Motor Brake Cur...	<Empty>	-	
				Electrical Pow... Empty
				Mechanical P... Empty
				Braking Powe... Empty
				Electrical Ener... Empty

The *Energy AX81xx* module is suitable for energy monitoring of servo drives from the AX81xx series. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **Electrical power:** Electrical power in W.
Process data: Enable PDO input 0x1A32 and add CoE 0x3242:12 "Electrical power" to the PDO content.
- **Mechanical Power:** Mechanical power in W.
Process data: Enable PDO input 0x1A32 and add CoE 0x3242:13 " Mechanical power" to the PDO content.
- **DC link voltage:** DC link voltage in mV.
Process data: Enable PDO input 0x1A00 and add CoE 0x6079 "DC link circuit voltage" to the PDO content.
- **Motor Brake Current:** Brake current of the motor in A.
Process data: Enable PDO input 0x1A20 and add CoE 0x3001:01 "Actual motor brake current" to the PDO content.

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Mechanical Power [W]:** Mechanical power in W.

- **Braking Power [W]:** Braking power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.4 Energy AX82xx

Energy AX82xx					
	Electrical Power...	<Empty>	▼	-	Sample Rate
	Mechanical Pow...	<Empty>	▼	-	1000
	Electrical Power...	<Empty>	▼	-	
	Mechanical Pow...	<Empty>	▼	-	
	DC Link Voltage	<Empty>	▼	-	
	Motor Brake Cur...	<Empty>	▼	-	
	Motor Brake Cur...	<Empty>	▼	-	
					Electrical Pow... Empty
					Mechanical P... Empty
					Braking Powe... Empty
					Electrical Ener... Empty

The *Energy AX82xx* module is suitable for energy monitoring of servo drives from the AX82xx series. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **Electrical Power ChA:** Electrical power of the first channel in W.
Process data: Enable PDO input 0x1A32 and add CoE 0x3242:12 "Electrical power" to the PDO content.
- **Mechanical Power ChA:** Mechanical power of the first channel in W.
Process data: Enable the PDO input 0x1A32 and add the CoE 0x3242:13 "Mechanical power" to the PDO content.
- **Electrical Power ChB:** Electrical power of the second channel in W.
Process data: Enable PDO input 0x1A52 and add CoE 0x3642:12 "Electrical power" to the PDO content.
- **Mechanical Power ChB:** Mechanical power of the second channel in W.
Process data: Enable the PDO input 0x1A52 and add the CoE 0x3642:13 "Mechanical power" to the PDO content.
- **DC link voltage:** DC link voltage in mV.
Process data: Enable PDO input 0x1A00 and add CoE 0x6079 "DC link circuit voltage" to the PDO content.
- **Motor Brake Current ChA:** Brake current of the motor of the first channel in A.
Process data: Enable PDO input 0x1A20 and add CoE 0x3001:01 "Actual motor brake current" to the PDO content.
- **Motor Brake Current ChB:** Brake current of the motor of the second channel in A.
Process data: Enable PDO input 0x1A40 and add CoE 0x3401:01 "Actual motor brake current" to the PDO content.

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Mechanical Power [W]:** Mechanical power in W.
- **Braking Power [W]:** Braking power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.5 Energy AX86x0

Energy AX86x0					
	DC Link Voltage	<Empty>	-	Sample Rate	1000
	DC Link Current	<Empty>	-		
	Brake Resistor C...	<Empty>	-		
					Electrical Pow... Empty
					Braking Powe... Empty
					Electrical Ener... Empty

The *Energy AX86xx* module is suitable for energy monitoring of AX86xx series power supply modules. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **DC Link Voltage:** DC link voltage in mV.
Process data: Enable the PDO input 0x1A05.
- **DC Link Current:** Input current to the DC link in 0.01 A.
Process data: Enable the PDO input 0x1A06.
- **Brake Resistor Continuous Power:** Power output via the braking resistor in W.
Process data: Enable the PDO input 0x1A07.

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Braking Power [W]:** Braking power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.6 Energy EL72xx EL74xx

Energy EL72xx EL74xx					
	Actual Velocity	<Empty>	-	Sample Rate	1000
	Actual Torque	<Empty>	-	Rated Current	1000
	DC Link Voltage	<Empty>	-	Torque Constant	0
				Feature Bits	0
					Electrical Pow... Empty
					Mechanical P... Empty
					Electrical Ener... Empty

The *Energy AX72xx EL74xx* module is suitable for energy monitoring of EL72xx and EL74xx EtherCAT Terminals. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Rated Current:** Nominal current in mA.
Process data: CoE: 0x8011:12.
- **Torque Constant:** Torque constant in mNm / A.
Process data: CoE: 0x8011:16.
- **Feature bits:** Feature bits for specifying the current type.
Process data: CoE: 0x8010:53.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **Actual velocity:** Actual velocity value.
Process data: Enable the PDO input 0x1A02.

- **Actual Torque:** Actual value of the torque.
Process data: Enable the PDO input 0x1A03.
- **DC link voltage:** DC link voltage in mV.
Process data: Enable PDO input 0x1A04 and select "DC link voltage (mV)" in CoE 0x8010:39.

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Mechanical Power [W]:** Mechanical power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.7 Energy EL259x

Energy EL259x					
	Supply Voltage	<Empty>	-	Sample Rate	1000
	Supply Current	<Empty>	-		
					Electrical Pow... Empty
					Electrical Ener... Empty

The *Energy EL259x* module is suitable for energy monitoring of EL259x EtherCAT Terminals. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **Supply Voltage:** Supply voltage in 0.01 V.
Process data: Enable PDO input 0x1A01 and select CoE 0xF900:05 "Supply voltage" in CoE 0x8002:11.
- **Supply Current:** Input current in mA.
Process data: Enable PDO input 0x1A01 and select CoE 0xF900:06 "Supply current" in CoE 0x8002:19.

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.8 Energy EP9224

Energy EP9224					
	Device Voltage Us	<Empty>	-	Sample Rate	1000
	Device Current Us	<Empty>	-		
	Device Voltage Up	<Empty>	-		
	Device Current Up	<Empty>	-		
					Electrical Pow... Empty
					Electrical Ener... Empty

The *Energy EP9224* module is suitable for the energy monitoring of EP9224 EtherCAT Box modules. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **Device Voltage Us:** Control voltage in V.
Process data: PDO Inputs Device
- **Device Current Us:** Control current in A.
Process data: PDO Inputs Device
- **Device Voltage Up:** Load voltage in V.
Process data: PDO Inputs Device
- **Device Current Up:** Load current in A.
Process data: PDO Inputs Device

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.5.9 Energy EPP9022

Energy EPP9022			
	Voltage Us	<Empty>	-
	Current Is	<Empty>	-
	Voltage Up	<Empty>	-
	Current Ip	<Empty>	-
Sample Rate		1000	
			Electrical Pow... Empty
			Electrical Ener... Empty

The *Energy EPP9022* module is suitable for the energy monitoring of EPP9022 EtherCAT Box modules. The input data is averaged over a configurable interval and then calculated. To ensure that the required input data is available, it must be added to the process image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Interval (optional):** By default, the input data is averaged over an interval of one second.

Input values

- **Device Voltage Us:** Control voltage in V.
Process data: PDO Inputs Device
- **Device Current Is:** Control current in A.
Process data: PDO Inputs Device
- **Device Voltage Up:** Load voltage in V.
Process data: PDO Inputs Device
- **Device Current Ip:** Load current in A.
Process data: PDO Inputs Device

Output values

- **Electrical Power [W]:** Electrical power in W.
- **Electrical Energy [Wh]:** Electrical energy in Wh.

5.2.1.6 Analytics - Math

The algorithms of the category *Analytics-Math* provide functionalities for mathematical operations such as basic arithmetic operation, integration or slope analysis.

5.2.1.6.1 Integrator 1Ch

Integrator 1Ch			
	Input	<Empty>	-
	Integration Mode		
	Factor	1	
			Result Empty

The *Integrator 1Ch* integrates the input value over time with a base unit of one second and provides the result of this integration operation. For the approximation of this integral the trapezoidal rule is used. The

trapezoidal $T(t_n, t_{n+1})$ between two subsequent timestamps t_n and t_{n+1} with the values y_n and

y_{n+1} is calculates as

$$T(t_n, t_{n+1}) = (t_{n+1}[s] - t_n[s]) \cdot \frac{y_n + y_{n+1}}{2}$$

If the integration mode "absolute" ("|x|") is chosen in the configuration, y_n and y_{n+1} are substituted by their absolute values in the above equation.

In each cycle the trapezoidal between the current and the last timestamp is calculated and added to the sum of trapezoids starting from the beginning of the analysis. Additionally, this sum can be scaled by a factor that can be configured individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Integration Mode:** You can select an integration mode. "→": the input value is will be integrated directly. "|x|": The absolute values of the input signal will be integrated.
- **Factor:** With this factor the integral is multiplied.

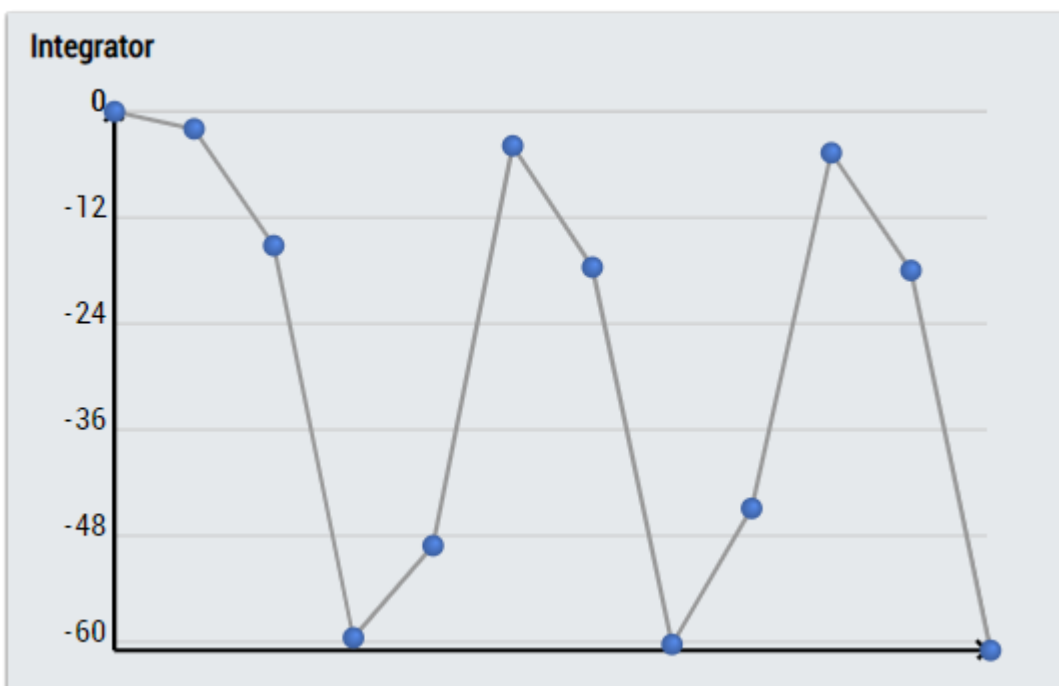
Output Values

- **Result:** Shows the result of the integration operation.

Standard HMI Controls

For the Integrator 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Integrator Control visualizes the last x output values Result.



2. The Table Control or Multivalue Control visualizes all output values: Result.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the Integrator 1Ch algorithm using the Mapping Wizard [► 382].

5.2.1.6.2 Math Operation

Math Operation							
+ - ÷ ×	Input 00	<Empty>	-	Num Channels	+ -	Result	Empty
	Input 01	<Empty>	-	Mathematical Operator	+ -		
					Use Absolute Values	☐	

The *Math Operation* executes a mathematical operation on two or more different input channels and provides the result of the mathematical operation. The operator is the same for all operands.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** The number of operands that are inputs to the function.
- **Mathematical Operator:** Mathematical operator of the operation.
 - "+" *Addition*
 - "-" *Subtraction*
 - "x" *Multiplication*
 - "/" *Division*
 - "xy" *PowerOf*
 - "%" *Modulo*
 - "v" *Minimum*
 - "^" *Maximum*
- **Use Absolute Values:** If the checkbox is checked, the absolute values of the input signal are used.

Output Values

- **Result:** Result of the mathematical operation.

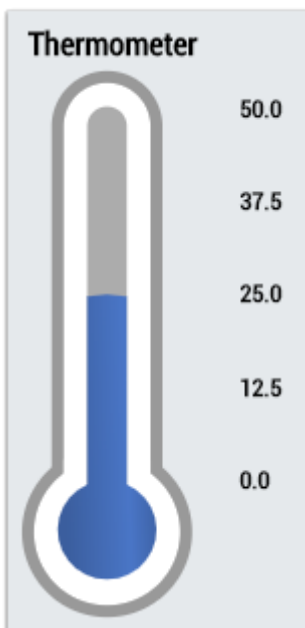
Standard HMI Controls

For the Math Operation algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SingleValue control visualizes the output value Result.



2. The Thermometer control visualizes the Result output value and can be used for temperature displays.



3. The Table Control or Multivalue Control visualizes all output values: Result.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the Math Operation algorithm using the Mapping Wizard [► 382].

5.2.1.6.3 Math Operation 1Ch

Math Operation 1Ch						
+ - ÷ ×	Input	<Empty>	Mathematical Operand	0	Result	Empty
			Use Absolute Values	☐		

The Math Operation 1Ch executes a mathematical operation on the signal of the input channel and a reference value. The algorithm provides the result of the mathematical operation and the operator can be configured individually.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Mathematical Operator:** Mathematical operator of the operation.
 - "+" *Addition*
 - "-" *Subtraction*
 - "x" *Multiplication*
 - "/" *Division*
 - "xy" *PowerOf*
 - "%" *Modulo*
 - "v" *Minimum*
 - "^" *Maximum*
- **Mathematical Operand:** Mathematical operand for the operation.
- **Use Absolute Values:** If the checkbox is checked, the absolute values of the two operands are used.

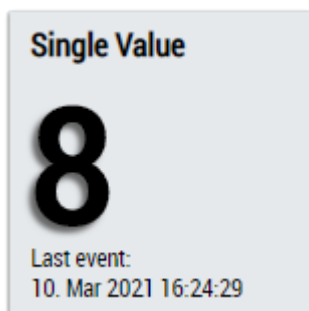
Output Values

- **Result:** Result of the mathematical operation.

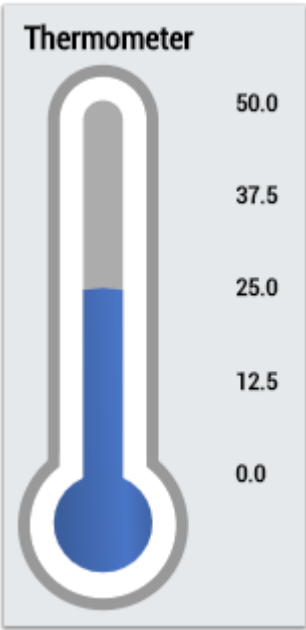
Standard HMI Controls

For the Math Operation algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SingleValue control visualizes the output value Result.



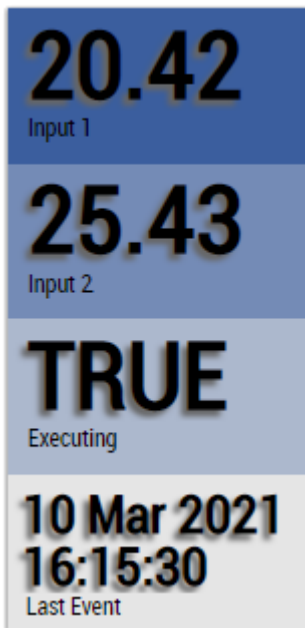
2. The Thermometer control visualizes the Result output value and can be used for temperature displays.



3. The Table Control or Multivalue Control visualizes all output values: Result.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Math Operation algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.6.4 Mean 1Ch

Mean 1Ch			
$\bar{x}$	Input	<Empty>	-
	Sample Rate	1000	
	Startup Behaviour	UsePreviousCascadeValue	
	Num Cascades	+ -	
	Cascades	Cascade 00: 0:00:00:01.000	
	Mean 00	Empty	

The *Mean 1Ch* calculates the mean value over the input values according to the formula

$$\bar{x} = \frac{1}{N} \sum_{n=1}^N x$$

The number of samples N that are included in the calculation can be configured by specifying a time interval. A cascaded output can be configured to realize a long-term mean value in a resource-saving way and to pick up intermediate results. The time interval of the configured cascade must correspond to an integer multiple of the time interval of the previous cascade.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Cascades:** Number of output cascades
- **Cascades:** Configuration of the output cascades. The time interval of the configured cascade must correspond to an integer multiple of the time interval of the previous cascade.
- **Sample Rate:** Sample rate of the system to be analyzed in Hz.
- **Startup Behavior:**
WaitUntilFilled waits until the configured timespan of the cascade has elapsed. The result and the "NewResult" Boolean flag are only set for the first time after the timespan has expired.
UsePreviousCascadeValue: The cascades whose configured timespan has not yet expired use the next smallest result that has already been set.

Output values

- **Mean:** Output array in which the results of the mean value calculations are saved. The dimension corresponds to the number of set cascades. The startup behavior can be set via the *Startup Behavior* parameter.

Standard HMI Controls

For the Mean algorithm, the following HMI controls are available for generating an Analytics Dashboard:

- 1. The Table Control or Multivalue Control visualizes all output values: Mean Results.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Mean algorithm using the [Mapping Wizard](#) [[▶ 382](#)].

5.2.1.6.5 RMS 1Ch

RMS 1Ch			
RMS	Input	<Empty>	-
	Sample Rate	1000	
	Startup Behaviour	UsePreviousCascadeValue	
	Num Cascades	+ -	
	Cascades	Cascade 00:	0:00:00:01.000
			RMS 00 Empty

RMS 1Ch calculates the root mean square over the input values according to the formula

$$\text{RMS} = \sqrt{\frac{1}{N} \sum_{n=1}^N x[n]^2}$$

The number of samples N that are included in the calculation can be configured by specifying a time interval. A cascaded output can be configured to realize a long-term RMS in a resource-saving way and to pick up intermediate results. The time interval of the configured cascade must correspond to an integer multiple of the time interval of the previous cascade.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Cascades:** Number of output cascades
- **Cascades:** Configuration of the output cascades. The time interval of the configured cascade must correspond to an integer multiple of the time interval of the previous cascade.
- **Sample Rate:** Sample rate of the system to be analyzed in Hz
- **Startup Behaviour:**
WaitUntilFilled waits until the configured timespan of the cascade has elapsed. The RMS result and the Boolean flag "NewResult" are only set for the first time after the timespan has elapsed.
UsePreviousCascadeValue The RMS cascades whose configured timespan has not yet expired use the next smallest RMS result already set.

Output values

- **RMS:** Output array in which the results of the RMS calculations are stored. The dimension corresponds to the number of set cascades. The startup behavior can be set via the parameter *Startup Behaviour*.

Standard HMI Controls

For the RMS algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control or Multivalue Control visualizes all output values: RMS Results.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the RMS algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.6.6 Slope Analysis 1Ch

Slope Analysis 1Ch			
	Input	<Empty>	-
	Num Values		2
	Slope		Empty
	Slope Min		Empty
	Slope Max		Empty
	Delta Value		Empty
	Delta Time		Empty
	Time Slope		Empty
	Time Slope Min		Empty
	Time Slope M...		Empty

The *Slope Analysis 1Ch* calculates the slope between two values of the input stream. One of those two values is the current input value and the second value is the input value that occurred a defined number (configured by the parameter *Num Values*) of cycles before in the input stream. The difference between these two values is returned as *Delta Value*.

The corresponding distance on the time-coordinate is calculated as the difference of the timestamps of these two values and is provided as the output value *Delta Time*. Note that the value *Delta Time* is displayed in nanoseconds, but for the calculation of the slope it is scaled to a second as base unit.

The *Slope* is then calculated as the fraction of *Delta Value* and *Delta Time* (scaled to seconds) and estimates the gradient for the timestamp in the center of the two timestamps used in the calculation of *Delta Time*. This is the value returned as *Time Slope* if it corresponds to a timestamp of the input stream. For configurations, where *Num Values* is an uneven number there is no input value matching the exact centre timestamp. In this case the timestamp of the value that directly succeeded the calculated centre timestamp is returned as *Time Slope*.

Further, the algorithm provides the minimal slope, the maximal slope and the time values of minimum and maximum.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

Num Values: Number of cycles that are in between the values used for the calculation of the slope.

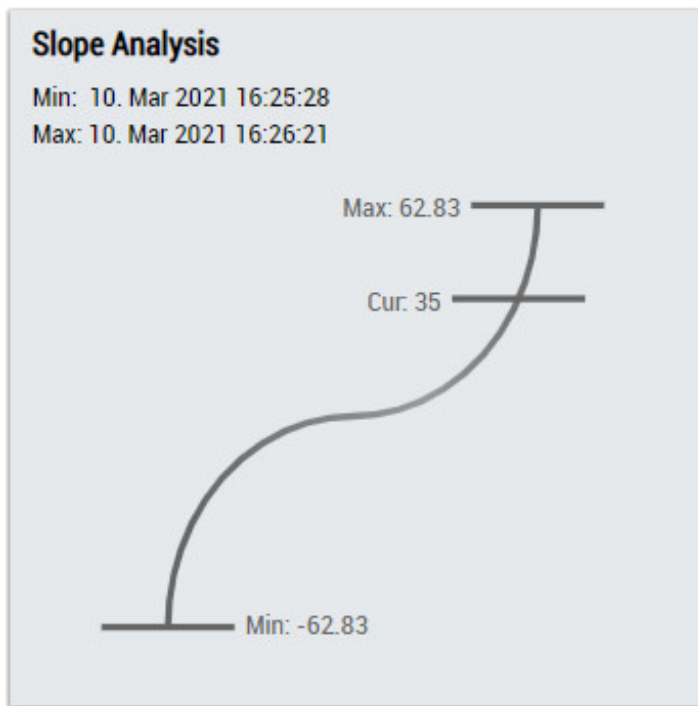
Output values

- **Slope:** Indicates the current slope value.
- **Slope Min:** Indicates the minimum of the slope values.
- **Slope Max:** Indicates the maximum of the slope values.
- **Delta Value:** Indicates the difference between the two values used to calculate the latest slope.
- **Delta Time:** Indicates the time period used to calculate the latest slope.
- **Time Slope:** Indicates the time value of the latest slope value.
- **Time Slope Min:** Indicates the time value of the minimum of the slope → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **Time Slope Max:** Indicates the time value of the maximum of the slope → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the *Slope Analysis 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SlopeAnalysis control visualizes the output values Slope, Slope Min, Slope Max, Time Slope Min and Time Slope Max.



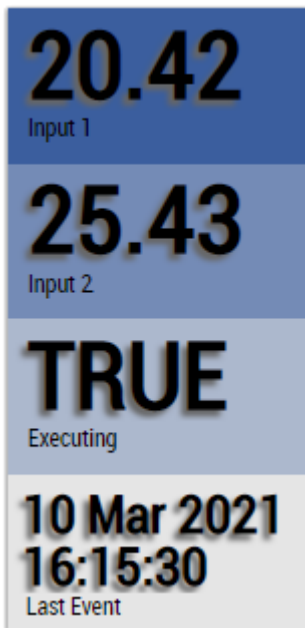
2. The Table Control or Multivalue Control visualizes all output values: Slope, Slope Min, Slope Max, Delta Value, Delta Time, Time Slope, Time Slope Min, Time Slope Max.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

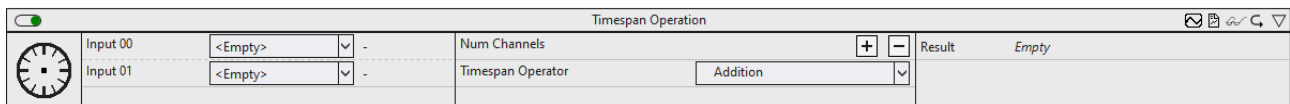
DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.6.7 Timespan operation



The *Timespan Operation* performs a mathematical operation on two or more different input channels and returns the result of the mathematical operation. Timespans with a resolution of 1 ns are supported as inputs.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** The number of operands that are inputs to the function.
- **Timespan Operator:** Mathematical operator of the operation (*addition / subtraction*).

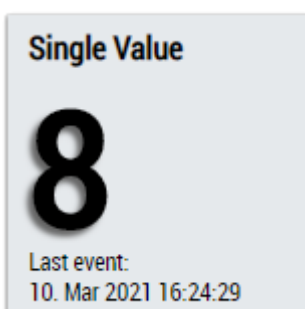
Output values

- **Result:** Result of the mathematical operation as timespan.

Standard HMI Controls

For the Timespan Operation algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The SingleValue control visualizes the output value Result.



2. The Table Control or Multivalue Control visualizes all output values: Result.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Timespan Operation algorithm using the Mapping Wizard [[▶ 382](#)].

5.2.1.7 Analytics - Preprocessing

5.2.1.7.1 Scaler Fitting

Scaler Fitting 1Ch					
	Input	<Empty>	-	Scaling Mode	Normalization
	Fit	<Empty>	-	File Path	C:\Analytics Project\Teach\Scaler.json
				Scaling Para...	Empty
				Scaling Para...	Empty

The *Scaler Fitting* module can be used to determine the parameters for feature scaling and save them in a file. These parameters can then be used in the *Scaler* module.

The parameters required for scaling are determined depending on the operation mode.

Normalization:

$$x' = \frac{x - \text{low}(x)}{\text{high}(x) - \text{low}(x)}$$

Standardization:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Scaling Mode:** Mode for scaling.
Normalization
Standardization
- **File Path:** Specifies the file path for the file in which the scaling parameters are saved. File type: *json*.

Output values

- **Scaling Param 1:** The value of the first scaling parameter depends on the scaling mode set.
Normalization: Smallest value of the input signal
Standardization: Average value of the input signal
- **Scaling Param 2:** The value of the second scaling parameter depends on the scaling mode set.
Normalization: Largest value of the input signal
Standardization: Standard deviation of the input signal

5.2.1.7.2 Scaler

Scaler 1Ch					
	Input	<Empty>	-	Scaling Mode (Result: Norm.: 0..+1 Stand.: -1..+1)	Normalization
				Scaling Param 1 (Norm.: Xlow Stand.: u)	0
				Scaling Param 2 (Norm.: Xhigh Stand.: sigma)	1
				Result	Empty

The *Scaler* module can be used to perform feature scaling.

Depending on the operation mode, the scaling parameters are used in different ways.

Normalization

$$x' = \frac{x - \text{low}(x)}{\text{high}(x) - \text{low}(x)}$$

Standardization:

$$x' = \frac{x - \bar{x}}{\sigma}$$

The scaling parameters can be configured directly. Alternatively, they can be imported from a file.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Scaling Mode:** Mode for scaling.
Normalization: Result has a value range of 0..+1
Standardization: Result has a value range of -1..+1
- **Scaling Param 1:** The value of the first scaling parameter depends on the scaling mode set.
Normalization: Smallest value of the input signal
Standardization: Average value of the input signal
- **Scaling Param 2:** The value of the second scaling parameter depends on the scaling mode set.
Normalization: Largest value of the input signal
Standardization: Standard deviation of the input signal
- **File Path (optional):** File path to a file with previously determined parameters. The file is created by the *Scaler Fitting* module. File type: *json*.

Output values

- **Result:** Outputs the calculated value. This is calculated from the two scaling parameters depending on the scaling mode.
Normalization: Result has a value range of 0..+1
Standardization: Result has a value range of -1..+1

5.2.1.8 Analytics - Reporting

24/7 reporting can be implemented in TwinCAT Analytics using the algorithms in the *Reporting* category. The reporting collectors collect the data and send it to the reporting server. The reporting triggers trigger the creation of a report.

5.2.1.8.1 Reporting Collector

The Reporting Collectors collect data and send it to the Reporting Server in a data message after an event.

5.2.1.8.1.1 Reporting Collector Edge

Reporting Collector - Edge			
Edge	<Empty>	Threshold Edge	1
Channel 00	<Empty>	Report Name	Beckhoff Report Template
		Data Key	DataKey101
		Buffer Size	1
		Buffer on event	☒
		Num Channels	+ -
		Alias Name 00	Channel 1
		New Result	Empty
		Last Message	Empty
		Buffer Count	Empty

The Reporting Collector Edge collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the signal of the input channel passes the configured edge at a certain threshold.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.

- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.
- **Data Key:** the data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
- **Buffer Size:** specifies the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
- **Buffer on event:** specifies how the data is to be collected and buffered.
If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer.
If the parameter is False, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the TcReportingServer.
- **Num Channels:** the number of input channels from which the data will be collected. And the number of parameters to allow a description of the input channels.
- **Alias Name:** serves as a description of the input channel. BufferSize = 1: the alias name serves as a key in the key-value structure. BufferSize > 1: the alias name serves as the heading of the table column.
- **Include Timestamps (optional):** inserts a column with the timestamps.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Buffer Count:** specifies the number of elements in the buffer.

5.2.1.8.1.2 Reporting Collector Interval

The Reporting Collector Interval collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the timespan of the configured interval has expired.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Interval:** time interval at which the data should be sent to the Reporting Server.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.
- **Data Key:** the data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
- **Buffer Size:** specifies the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
- **Buffer on event:** specifies how the data is to be collected and buffered.
If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer.

If the parameter is `False`, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the `TcReportingServer`.

- **Num Channels:** the number of input channels from which the data will be collected. And the number of parameters to allow a description of the input channels.
- **Alias Name:** serves as a description of the input channel. `BufferSize = 1`: the alias name serves as a key in the key-value structure. `BufferSize > 1`: the alias name serves as the heading of the table column.
- **Include Timestamps (optional):** inserts a column with the timestamps.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Current Interval Time:** indicates the amount of time that has already elapsed from the current interval.
- **Buffer Count:** specifies the number of elements in the buffer.

5.2.1.8.1.3 Reporting Collector Time

The Reporting Collector Time collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the configured switch-on time is reached. The switch-on time and the days of the weeks can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Time On:** switch-on time.
- **Day of Week Mask:** weekdays on which the timer should be active.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.
- **Data Key:** the data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
- **Buffer Size:** specifies the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
- **Buffer on event:** specifies how the data is to be collected and buffered.
If the parameter is `True`, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the `TcReportingServer`.
If the parameter is `False`, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the `TcReportingServer`.
- **Num Channels:** the number of input channels from which the data will be collected. And the number of parameters to allow a description of the input channels.
- **Alias Name:** serves as a description of the input channel. `BufferSize = 1`: the alias name serves as a key in the key-value structure. `BufferSize > 1`: the alias name serves as the heading of the table column.
- **Include Timestamps (optional):** inserts a column with the timestamps.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Next Message:** indicates the remaining time until the next message.
- **Buffer Count:** specifies the number of elements in the buffer.

5.2.1.8.2 Reporting Trigger

The Reporting Triggers send a trigger message to the Reporting Server after an event and thus trigger the creation of a report.

5.2.1.8.2.1 Reporting Trigger Edge

Reporting Trigger - Edge			
Timestamp	-	Num Channels	+ -
Channel 00	<Empty>	Threshold Edge	1
Report Name		Beckhoff Report Template	
Last Trigger		Empty	
Edge Overview		Empty	

The Reporting Trigger Edge triggers the creation of a report after an event is triggered. An event is triggered when the input channel signal exceeds the configured edge at a specified threshold. Internally, the inputs that were once True remain True. The inputs are only reset to False as soon as all inputs were True at least once. This allows the output bNewResult to be used as one input by multiple Reporting Collectors and once all Reporting Collectors have sent a data message, a trigger message is sent.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.

Output values

- **Last Trigger:** indicates when the last message was sent to the Reporting Server.
- **Edge Overview:** indicates which input channels were True at least once.

5.2.1.8.2.2 Reporting Trigger Interval

Reporting Trigger - Interval			
Timestamp	-	Interval	Seconds 1
Report Name		Beckhoff Report Template	
Last Trigger		Empty	
Current Interv...		Empty	

The Reporting Trigger Interval triggers the creation of a report after an event has been triggered. An event is triggered when the timespan of the configured interval has expired.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Interval:** time interval at which the data should be sent to the Reporting Server.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.

Output values

- **Last Trigger:** indicates when the last message was sent to the Reporting Server.
- **Current Interval Time:** indicates the amount of time that has already elapsed from the current interval.

5.2.1.8.2.3 Reporting Trigger Time

Reporting Trigger - Time														
	Timestamp	Time On				hh	1	mm	0	ss	0	Last Trigger	Empty	
		Day Of Week Mask				☒ Mon	☒ Tue	☒ Wed	☒ Thu	☒ Fri	☒ Sat	☒ Sun	Next Trigger	Empty
		Report Name				Beckhoff Report Template								

The Reporting Trigger Time triggers the creation of a report after an event has been triggered. An event is triggered when the configured switch-on time is reached. The switch-on time and the days of the weeks can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Time On:** switch-on time.
- **Day of Week Mask:** weekdays on which the timer should be active.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Next Trigger:** indicates the remaining time until the next message.

5.2.1.9 Analytics - Specific

5.2.1.9.1 XTS

The algorithms of the category *Analytics-XTS* provide special functionalities for the Beckhoff XTS system. For example analysis of distance, velocity and acceleration.

5.2.1.9.1.1 XTS Acceleration Analysis 1Ch

XTS Acceleration Analysis 1Ch			
	Position [mm]	<Empty>	-
	XTS Length [mm]	4000	
			Acceleration [... Empty]

The *XTS Acceleration Analysis 1Ch* calculates the current acceleration of a XTS mover. For this purpose, the length of the XTS in millimeters must be declared and as input signal the mover position is required.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **XTS Length [mm]:** Length of the given XTS system in millimeters.

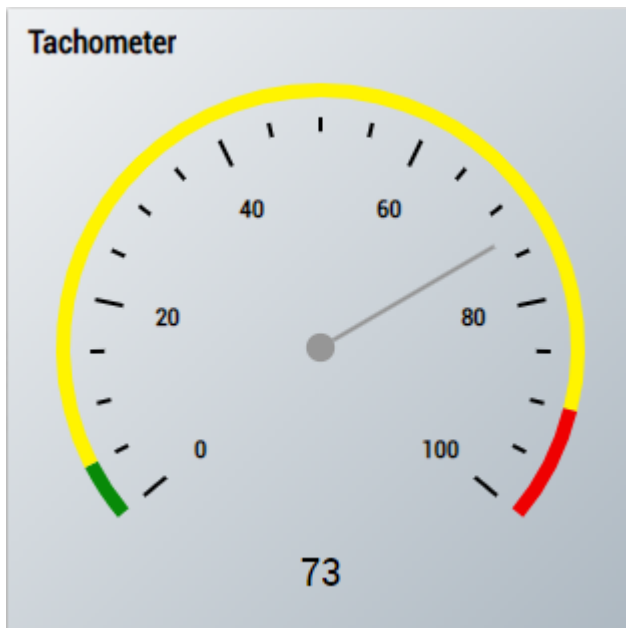
Output Values

- **Acceleration:** Current acceleration of the XTS mover.

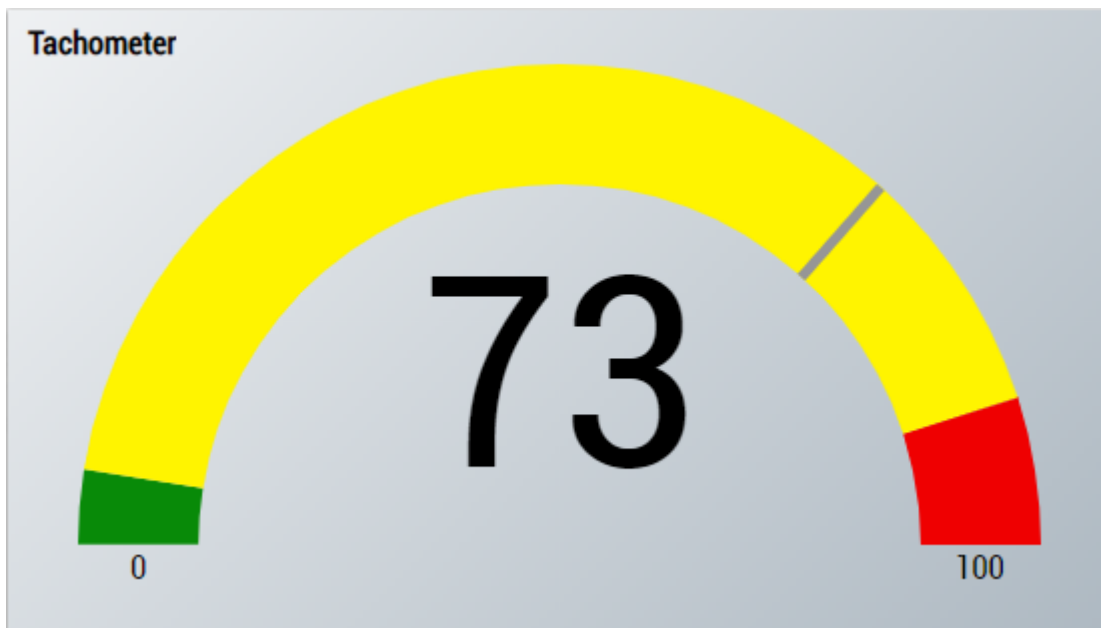
Standard HMI Controls

For the XTS Acceleration Analysis 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Tachometer control visualizes the output value Acceleration.



2. The Radial Gauge control visualizes the output value Acceleration.



3. The SingleValue control visualizes the output value Acceleration.



4. The Table Control or Multivalue Control visualizes all output values: Acceleration.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021 16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the XTS Acceleration Analysis 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.9.1.2 XTS Distance Integrator 1Ch

XTS Distance Integrator 1Ch					
	Position [mm]	<Empty>	-	XTS Length [mm]	4000
				Distance [m]	Empty
				Distance Posit...	Empty
				Distance Neg...	Empty

The *XTS Distance Integrator 1Ch* calculates the distance covered by a XTS mover. The algorithm provides the total distance, the positive distance and the negative distance. For this purpose, the length of the XTS in millimeters must be declared and as input signal the mover position is required.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **XTS Length [mm]:** Length of the given XTS system in millimeters.

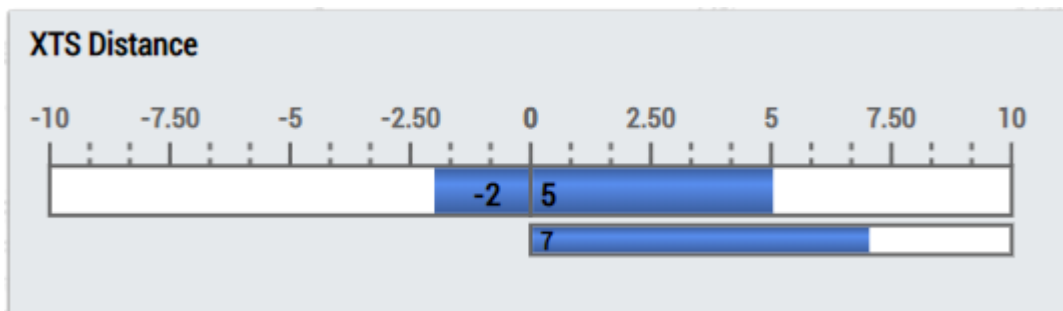
Output Values

- **Distance:** Total distance the XTS mover has covered.
- **Distance Positive:** Positive distance the XTS mover has covered (direction: forward).
- **Distance Negative:** Negative distance the XTS mover has covered (direction: backward).

Standard HMI Controls

For the XTS Distance Integrator 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The XTSDistance control visualizes the output values Distance, Distance Positive and Distance Negative.



2. The SingleValue control visualizes the output value Distance.



3. The Table Control or Multivalue Control visualizes all output values: Output values Distance, Distance Positive, Distance Negative.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the XTS Distance Integrator 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.9.1.3 XTS Velocity Analysis 1Ch

XTS Velocity Analysis 1Ch				
	Position [mm]	<Empty>	XTS Length [mm]	4000
			Velocity [m/s]	Empty

The *XTS Velocity Analysis 1Ch* calculates the current velocity of a XTS mover. For this purpose, the length of the XTS in millimeters must be declared and as input signal the mover position is required.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **XTS Length [mm]:** Length of the given XTS system in millimeters.

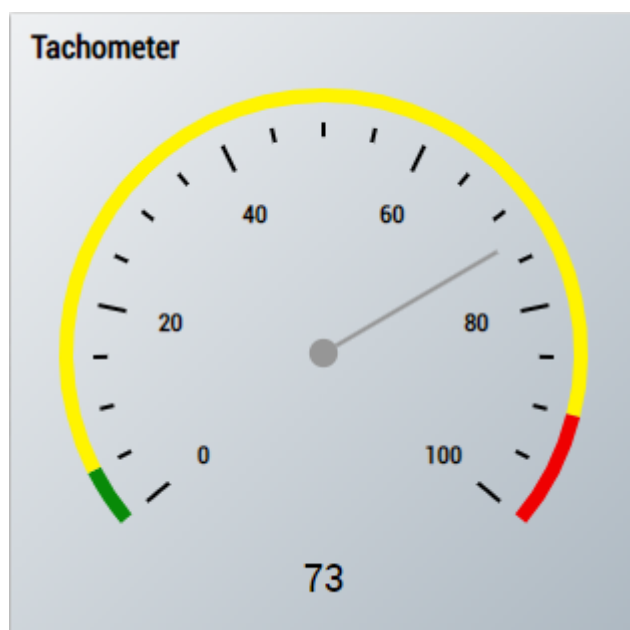
Output Values

- **Velocity:** Current velocity of the XTS mover.

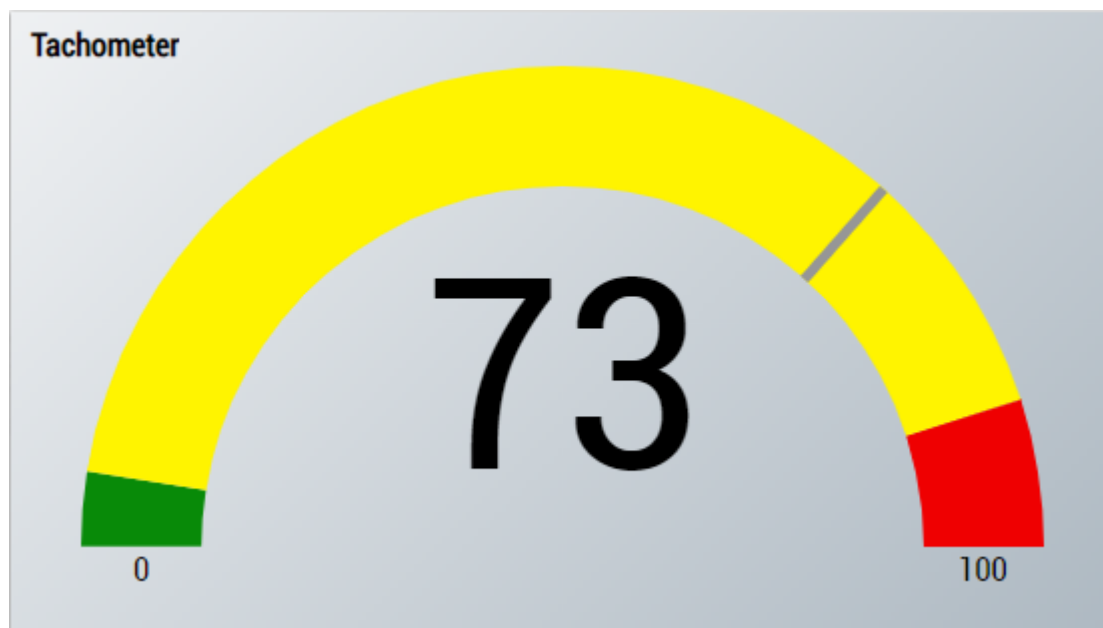
Standard HMI Controls

For the XTS Velocity Analysis 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Tachometer control visualizes the output value Velocity.



2. The Radial Gauge control visualizes the output value Velocity.



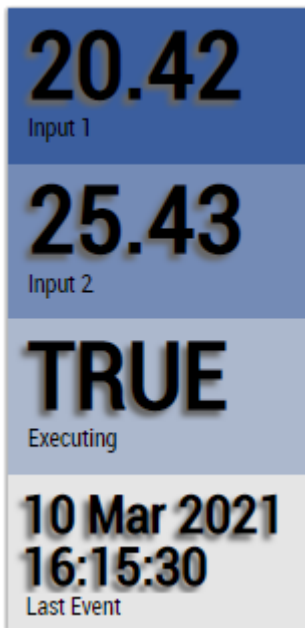
3. The SingleValue control visualizes the output value Velocity.



4. The Table Control or Multivalue Control visualizes all output values: Velocity.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

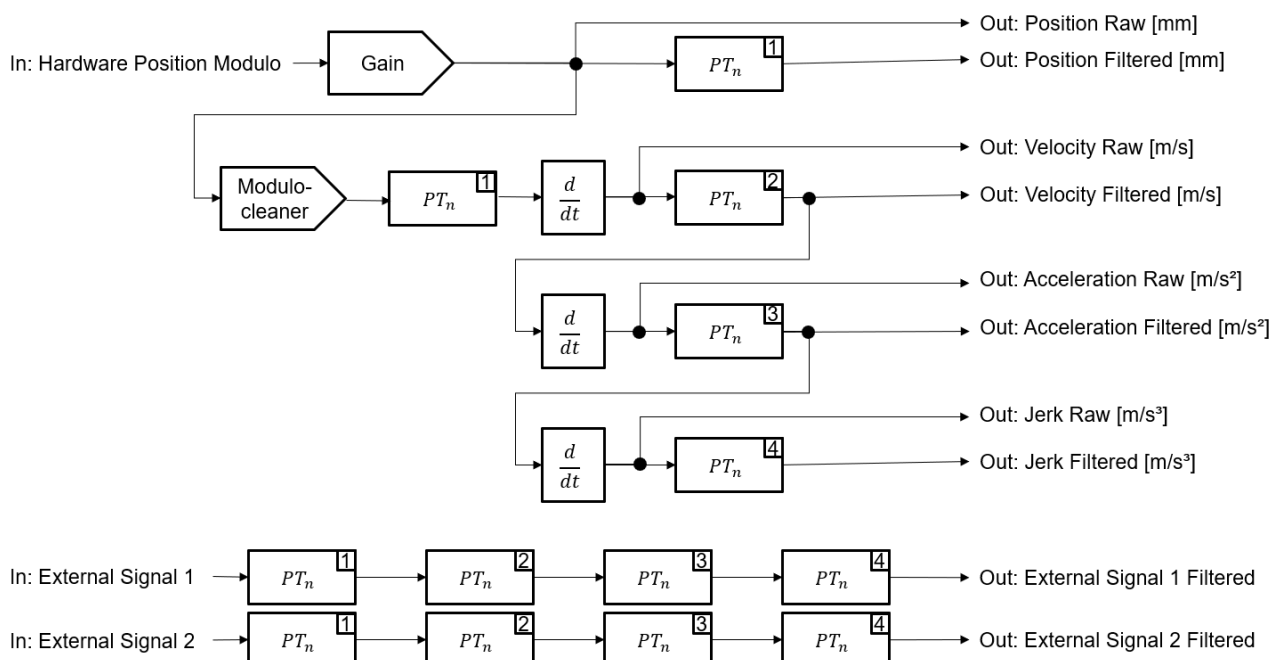
DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the XTS Velocity Analysis 1Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.9.1.4 XTS Movement Analysis

The *XTS Movement Analysis* calculates the position, speed, acceleration and jerk of an XTS mover. Various filter stages are chained together to smooth the signal. In order to be able to analyze additional, associated signals, the module enables the filtering of two further signals with the same filter stages.



Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **XTS Length [mm]:** length of the given XTS system in millimeters.
- **Hardware Pos Factor:** Factor for scaling the input signal in mm.
- **Sample Rate:** Sample rate (samples per second) of the input signal.
- **Filter Time Constant (pos):** Filter order and filter time constant of the PTn filter for calculating the position.
- **Filter Time Constant (vel):** Filter order and filter time constant of the PTn filter for calculating the speed.
- **Filter Time Constant (acc):** Filter order and filter time constant of the PTn filter for calculating the acceleration.
- **Filter Time Constant (jerk):** Filter order and filter time constant of the PTn filter for calculating the jerk.
- **Use Filter Ext 1:** Choice of whether the filter level (position, speed, acceleration, jerk) should be applied to the external input signal 1.
- **Use Filter Ext 2:** Choice of whether the filter level (position, speed, acceleration, jerk) should be applied to the external input signal 2.

Output values

- **Position Raw:** Position in mm.
- **Position Filtered:** Filtered position in mm.
- **Velocity Raw:** Speed in m/s.
- **Velocity Filtered:** Filtered velocity in m/s.
- **Acceleration Raw:** Acceleration in m/s².
- **Acceleration Filtered:** Filtered acceleration in m/s².
- **Jerk Raw:** Jerk in m/s³.
- **Jerk Filtered:** Filtered jerk in m/s³.
- **External Signal 1 Filtered:** External signal 1 to which the same filter stages can be applied as to the position signal.
- **External Signal 2 Filtered:** External signal 2 to which the same filter stages can be applied as to the position signal.

The following human-machine interface controls are available for the algorithm to generate an Analytics Dashboard:

The Table Control or List Control visualizes all output values.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be linked using the [Mapping Wizard](#) [► 382].

5.2.1.9.2 Wind Turbine

The algorithms of the category *Analytics – WT* provide special functionalities for the wind technology industry. For example analysis of mean wind speed, turbulence and turbulence intensity.

5.2.1.9.2.1 WT Turbulence 1Ch

WT Turbulence 1Ch				
	Input	<Empty>	Num Cycles	1
				Mean Empty
				Turbulence Empty
				Turbulence In... Empty

The *WT Turbulence 1Ch* calculates the mean of the wind velocity, the turbulence, and the turbulence intensity according to the standard *EN 61400-1*. As input signal, the wind velocity is required. The output values are updated in a cycle of 10 minutes.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Num Cycles 10Min:** Indicates the number of cycles that fit in the time interval for the calculation, according to *EN 61400-1* this is a ten minutes interval.

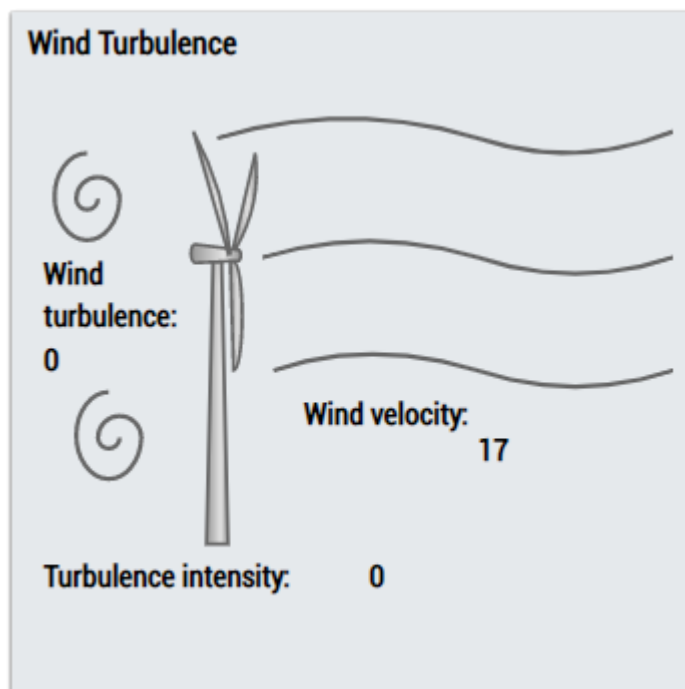
Output Values

- **Mean:** Mean of the wind velocity.
- **Turbulence:** Turbulence of the wind. According to *EN*-standard this is the standard deviation of the wind velocity over a time interval of 10 minutes.
- **Turbulence Intensity:** Intensity of the wind turbulence.

Standard HMI Controls

For the *WT Turbulence 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The *WindTurbulence* control visualizes the output values Mean, Turbulence and Turbulence Intensity.



2. The *Table Control* or *Multivalue Control* visualizes all output values: Mean, Turbulence, Turbulence Intensity.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the WT Turbulence 1Ch algorithm using the [Mapping Wizard](#) [► 382].

- **X Shock | Count:** Number of shocks in the X direction.
- **X Shock | Last Shock:** Timestamp of the last detected shock in the X direction.
- **Y Shock | Shock Detected:** **TRUE**, if a shock is detected in the Y direction. The threshold value of the RMS acceleration is monitored for this purpose.
- **Y Shock | Count:** Number of shocks in the Y direction.
- **Y Shock | Last Shock:** Timestamp of the last detected shock in the Y direction.
- **Total Shocks:** Number of shocks in the X and Y direction.
- **Sections Humidity:** Temporal distribution of humidity within the encoder. (Section Timer)
- **Sections Temperature:** Temporal distribution of the temperature within the encoder. (Section Timer)
- **X Rms Acceleration:** Distribution of the acceleration in the X direction. (Histogram)
- **Y Rms Acceleration:** Distribution of the acceleration in the Y direction. (Histogram)

Standard HMI Controls

For the algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The specific AM8xxx Control visualizes the output values and displays them graphically.

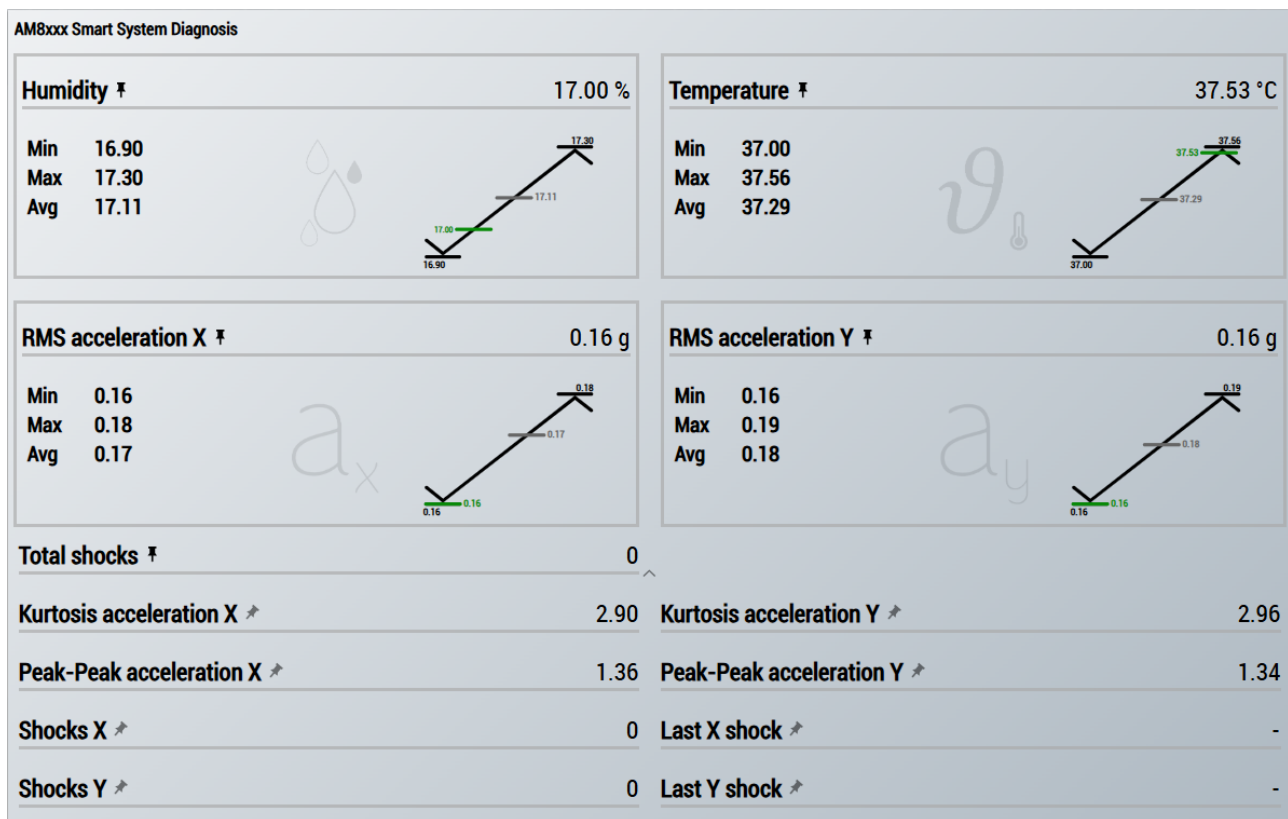


Fig. 1:

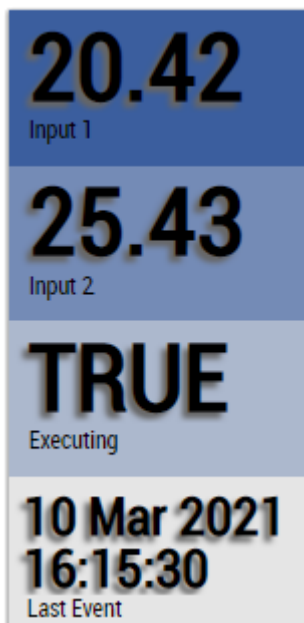
2. The Table Control or List Control visualizes all output values

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

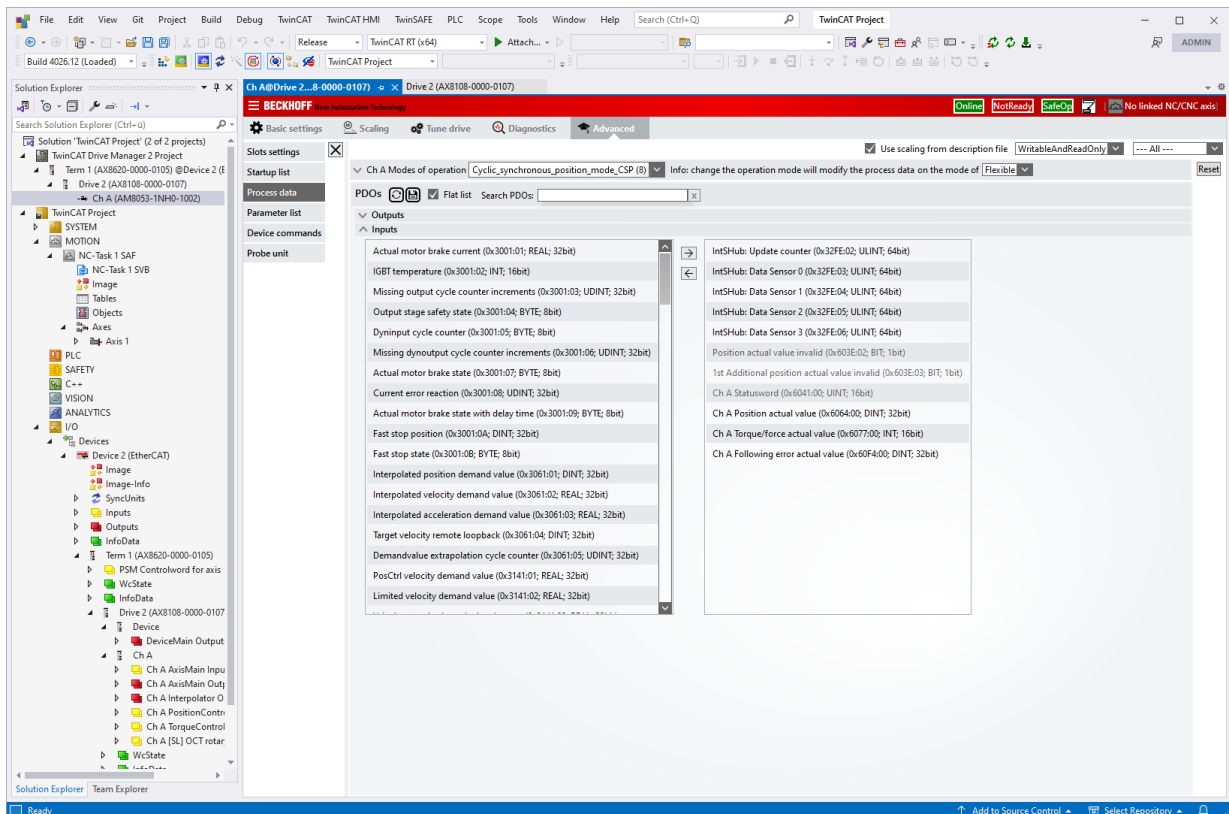


Preparation

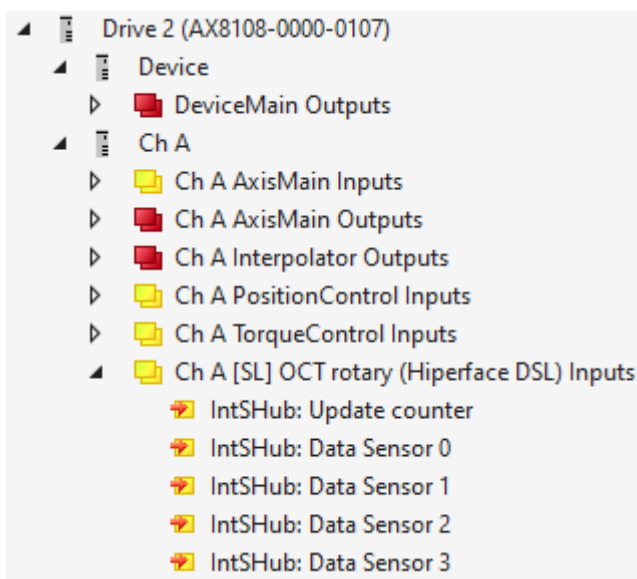
In order for the sensor values to be available for analysis, they must be added to the process image of the drive. This process is described below.

1. Create a TwinCAT project.
2. Scan the I/Os of the TwinCAT project.

3. Add a Drive Manager project to the solution.
4. Select the corresponding channel of the drive to which the motor with the B/SSD-capable encoder is connected.
5. Select the **Advanced** tab.
6. Further process data can be added in the **Process data** subitem.
7. Add the following process data:
 - IntSHub: Update counter
 - IntSHub: Data Sensor 0
 - IntSHub: Data Sensor 1
 - IntSHub: Data Sensor 2
 - IntSHub: Data Sensor 3



⇒ Once the process data has been added, it is displayed in the I/O node and is available for analysis.



5.2.1.9.3.2 AM8xxx Smart System Diagnosis Value Extractor

AM8xxx Smart System Diagnosis Value Extractor							  		
	Update Counter	<Empty>		-			Humidity...	Empty	
	Data Sensor 0	<Empty>		-			Temperat...	Empty	
	Data Sensor 1	<Empty>		-			X RMS Ac...	Empty	
	Data Sensor 2	<Empty>		-			X Kurtosis...	Empty	
	Data Sensor 3	<Empty>		-			X Peak-Pe...	Empty	
							Y RMS Ac...	Empty	
							Y Kurtosis...	Empty	
							Y Peak-Pe...	Empty	

The *AM8xxxSSD_ValueExtractor* enables the analysis of a motor equipped with the Smart System Diagnosis functionality. The built-in encoder records measured values from various internally installed sensors. These include the temperature and humidity in the encoder, as well as the acceleration in the X and Y directions. These measured values are provided as outputs.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Humidity:** Humidity inside the encoder in %.
- **Temperature:** Temperature inside the encoder in °C.
- **X RMS Acceleration:** RMS of the acceleration in the X direction in g.
- **X Kurtosis Acceleration:** Kurtosis of the acceleration in the X direction.
- **X Peak-Peak Acceleration:** Peak-peak value of the acceleration in the X direction in g.
- **Y RMS Acceleration:** RMS of the acceleration in the Y direction in g.
- **Y Kurtosis Acceleration:** Kurtosis of the acceleration in the Y direction.
- **Y Peak-Peak Acceleration:** Peak-peak value of the acceleration in the Y direction in g.

The following human-machine interface controls are available for the algorithm to generate an Analytics Dashboard:

The Table Control or List Control visualizes all output values.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be linked using the [Mapping Wizard](#) [► 382].

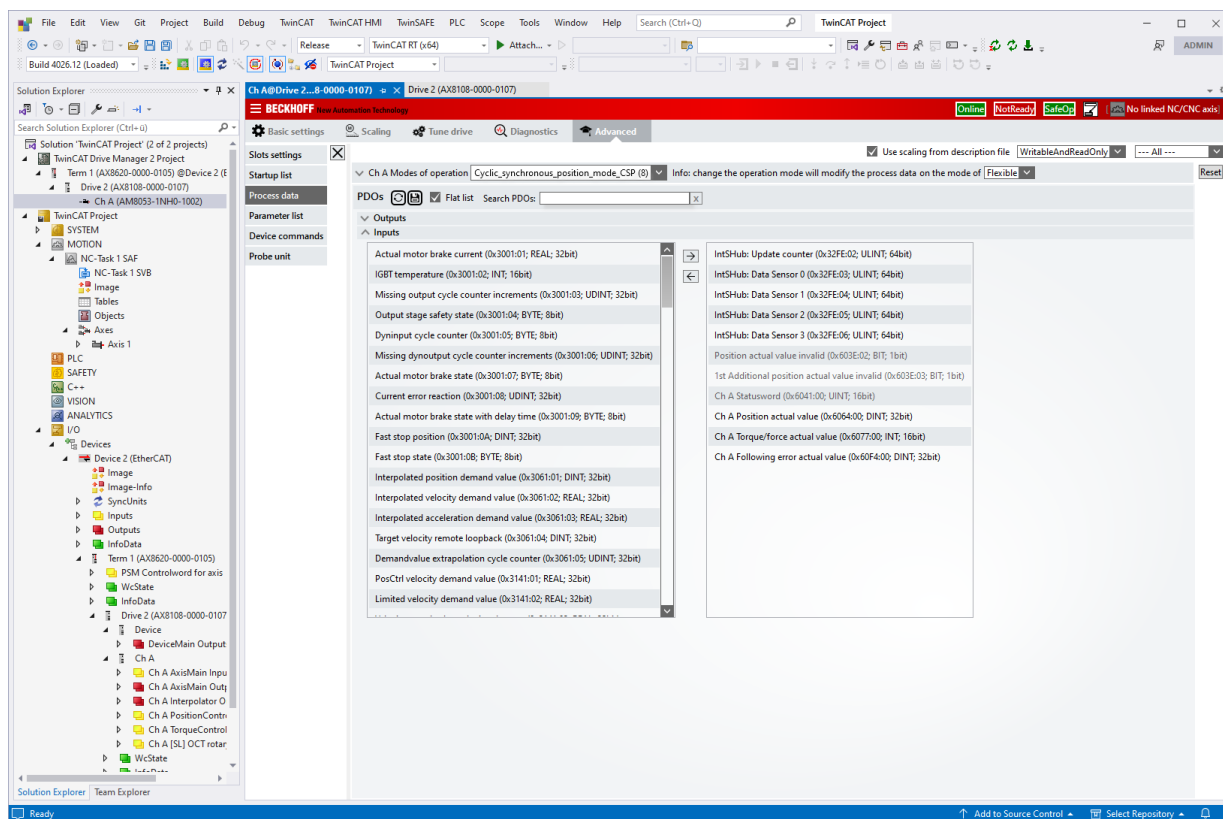
Preparation

In order for the sensor values to be available for analysis, they must be added to the process image of the drive. This process is described below.

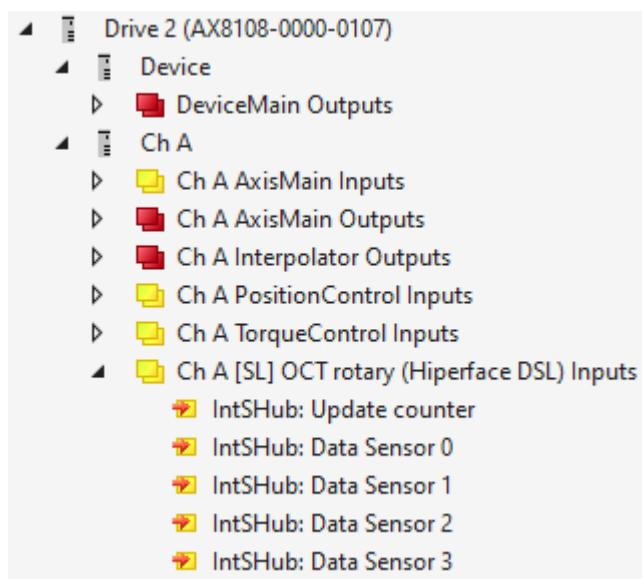
1. Create a TwinCAT project.
2. Scan the I/Os of the TwinCAT project.
3. Add a Drive Manager project to the solution.
4. Select the corresponding channel of the drive to which the motor with the B/SSD-capable encoder is connected.
5. Select the **Advanced** tab.
6. Further process data can be added in the **Process data** subitem.
7. Add the following process data:
 - IntSHub: Update counter
 - IntSHub: Data Sensor 0
 - IntSHub: Data Sensor 1

IntSHub: Data Sensor 2

IntSHub: Data Sensor 3



⇒ Once the process data has been added, it is displayed in the I/O node and is available for analysis.



5.2.1.10 Analytics - Statistics

The algorithms of the category *Statistics* offer functions for data analysis based on statistical methods. This includes, for example, the calculation of signal correlations and regression analyses.

5.2.1.10.1 Array Statistics

Array Statistics	
Array In: <Empty>	Use Bessel Correction: ☒
Threshold Reversal: 0.5	Min: Empty
Threshold Delta: 0.5	Idx Min: Empty
Lower Bound: 1	Max: Empty
	Idx Max: Empty
	Max Delta: Empty
	Idx Max D...: Empty
	Count Pe...: Empty
	Count Vall...: Empty
	Sum: Empty
	Mean: Empty
	Standard...: Empty

The *Array Statistics* algorithm calculates various statistical quantities based on the input array.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Use Bessel Correction:** if the checkbox is activated, Bessel correction will be applied. In order to obtain an expectation-true result for random samples, this parameter must be activated. The parameter is only relevant for the calculation of the standard deviation.

The empirical standard deviation, without Bessel's correction

$$s' = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

The empirical standard deviation, with Bessel's correction

$$s = \sqrt{\frac{1}{N-1} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

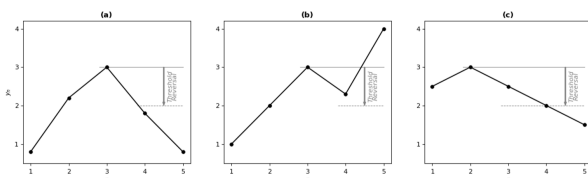
- **Threshold Reversal:** threshold for identifying reversals. Reversals are only detected if their difference from the next reversal exceeds the value of *Threshold Reversal*.

Below are three examples of peak identification using the parameter *Threshold Reversal*.

(a) The value y_3 is identified as a peak immediately after processing the value y_4 because the difference between y_3 and y_4 is greater than *Threshold Reversal*.

(b) The value y_3 is not identified as a peak because the difference between y_3 and y_4 is smaller than *Threshold Reversal* and the curve starts rising again after y_4 .

(c) The value y_2 is identified as a peak after processing the value y_5 because the difference between y_2 and y_5 exceeds *Threshold Reversal*. The value y_2 cannot be identified as a peak beforehand because the difference between y_2 and y_3 (y_4) is less than/equal to *Threshold Reversal* and it is not known whether the values will continue to decrease.



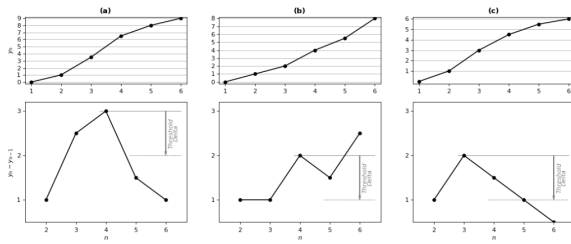
- **Threshold Delta:** threshold for identifying Delta maxima. Maxima of the absolute difference of two successive values (delta) are detected only if the difference between successive deltas exceeds *Threshold Delta*.

Below are three examples of identifying the Delta maxima with the parameter *Threshold Delta*. The upper diagrams show the original input signals, the lower ones the corresponding delta.

(a) The value y_4 is identified as a maximum after processing the value y_5 because the difference between the two deltas exceeds *Threshold Delta*.

(b) No maximum is identified because the difference between the deltas is less than *Threshold Delta*.

(c) The value y_3 is identified as a maximum after processing the value y_6 .



Regardless of *Threshold Delta*, at least one maximum of the Delta between two reversals is detected.

- **Lower Bound:** Lower bound of the array

Output values

- **Min:** smallest value in the input array.
- **Idx Min:** array index of *Min*. Indexing starts at 1.
- **Max:** largest value in the input array.
- **Idx Max:** array index of *Max*. The indexing starts at 1.
- **Max Delta:** maximum of the absolute difference between two consecutive values in the input array.
- **Idx Max Delta:** array index of *Max Delta*. The indexing starts at 1.
- **Count Peaks:** total number of peaks identified.
- **Count Valleys:** total number of valleys identified.
- **Sum:** sum over the entire input array.
- **Mean:** mean value over the entire input array.
- **Standard Deviation:** standard deviation over the entire input array.

Standard HMI Controls

For the Array Statistics algorithm, the following HMI controls are available for generating an Analytics Dashboard:

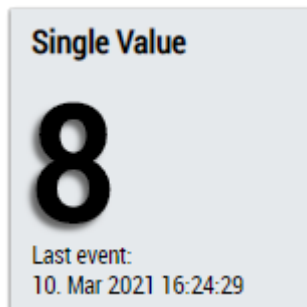
1. The SingleValue control visualizes the Sum output.



2. The SingleValue control visualizes the Mean output.



3. The SingleValue control visualizes the Standard Deviation output.



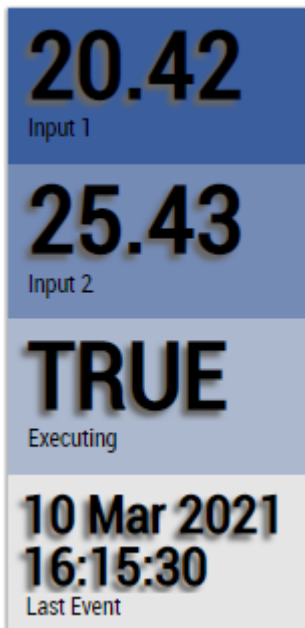
4. The Table Control or Multivalue Control visualizes the output values: Min, Max, Max Delta, CountPeaks, Count Valleys, Sum, Mean, Standard Deviation.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

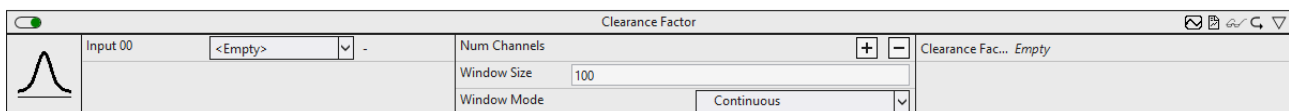
DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Standard Deviation algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.10.2 Clearance Factor



The algorithm *Clearance Factor* calculates the signal feature of the same name from the input values. The output value is calculated from the ratio of the peak value of the input signal to the squared mean of the square roots of the absolute input signal.

$$\text{Clearance Factor} = \frac{\max(|x|)}{\left(\frac{1}{N} \sum_{n=1}^N \sqrt{|x[n]|}\right)^2}$$

The number of input data to be included in the calculation and the type of calculation can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** Configuration of the number of independent input and output channels.
- **Window Mode:** Used window mode. Influences the number of input data that are included in the calculation as well as the time of the calculations.
Continuous: All input values since the start of the algorithm are included in the calculation. The calculation is cyclic.
Fix Window: Only the last *N* input values are included in the calculation. The calculation is performed every *N* calls.
Sliding Window: Only the last *N* input values are included in the calculation. The calculation is performed cyclically.
- **Window Size:** Depending on the window mode used, you can configure the number of values *N* that will be included in the calculation. In the window mode *Continuous* this parameter is ignored.

Output values

- **Clearance Factor 00..n:** Clearance factor of the input channels.

Standard HMI Controls

For the *Clearance Factor* algorithm the following HMI controls are available for generating an Analytics Dashboard:

- 1. The Table Control or Multivalue Control visualizes all output values.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped using the [Mapping Wizard](#) [► 382].

5.2.1.10.3 Correlation Function

Correlation Function			
Channel Ref	<Empty>	-	Num Channels
Channel 00	<Empty>	-	Minimum Lag
			Maximum Lag
			Step Size
			Window Size
			Correlation Mode
			Window Mode
			Output 00
			Minimum Co... Empty
			Maximum Co... Empty
			Minimizing L... Empty
			Maximizing L... Empty

The *Correlation Function* function block calculates the discrete correlation function between a reference signal (Channel Ref) and one or more other signals (Channel 00, ..., Channel 0n). The correlation coefficients are calculated for time shifts of m cycles between the two signals, with the maximum and minimum values for m being limited by the parameters *Minimum Lag* (negative integer) and *Maximum Lag* (positive integer).

The *Step Size* parameter determines the number of cycles by which the signals are moved to calculate two consecutive correlation coefficients. I.e. m is always a multiple of *StepSize*. Accordingly, for *Minimum Lag* and *Maximum Lag* only multiples of *StepSize* are allowed. If the *StepSize* is set to one and *Minimum Lag* is set to -6 and *Maximum Lag* is set to +4 -for example-, correlation coefficients are calculated for shifts of -6, -5, -4, -3, -2, -1, 0, +1, +2, +3 and +4 cycles. If the *StepSize* is set to two, coefficients are calculated for shifts by -6, -4, -2, 0, +2 and +4 cycles.

The coefficients can be calculated over different timeframes, which are set by the *Window Mode* parameter. In *Continuous* mode, all values since the beginning of the analysis are included in the calculations. In *SlidingWindow* mode, the calculation runs continuously for the last number of cycles set by the *Window Size*. In *FixWindow* mode, the calculation is also done for the number of cycles set by the *Window Size*. However, the calculation restarts after each *WindowSize* cycle and the output values are updated only when the last cycle of a window is run through.

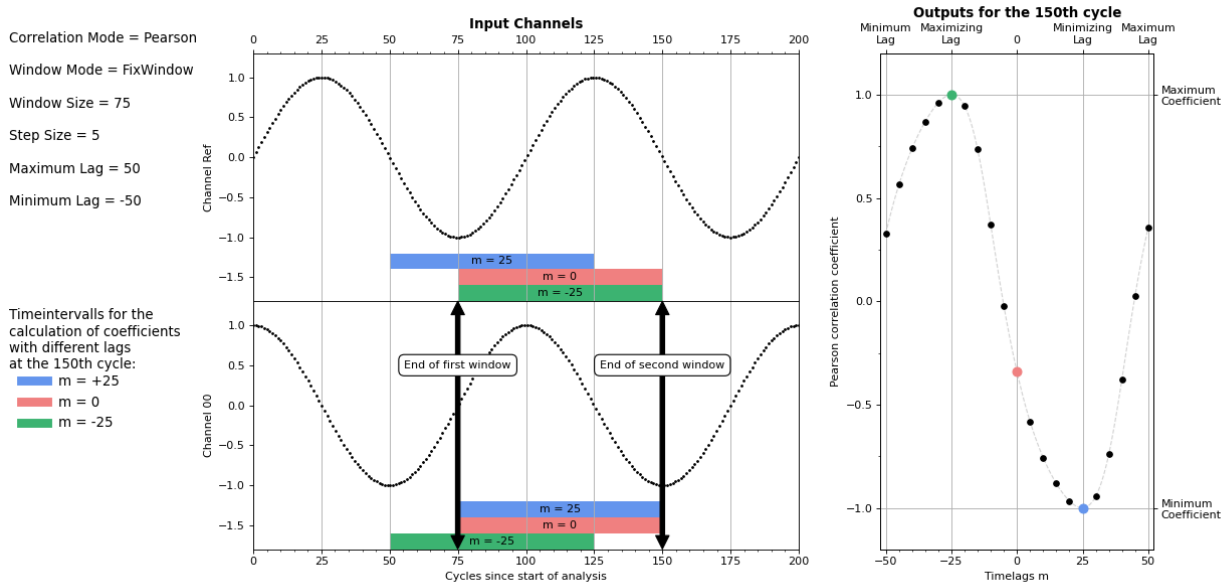
If m is not zero, the window for the corresponding signal shifts by m cycles. The number of values included in the calculation of the coefficients is the same for all values of m . The only exceptions to this are the results from the first cycles after the start of the analysis. If the number of elapsed cycles is less than $|m|$, correspondingly fewer values are included in the calculation.

For positive values of m , the values of the reference signal (Channel Ref) are stored in a ring memory, so that the respective value of the reference signal received before m cycles can be compared with the current values from Channel 00 to Channel 0n. This corresponds to a shift of the reference signal into the past. For negative values of m , the reference signal would accordingly have to be moved into the future. Since this is obviously not possible, the second signal (Channel 00, .. Channel0n) is saved and moved backward instead.

The correlation coefficients can be calculated according to different calculation rules. This is determined by the *Correlation Mode*. In the *Base* and *Normed* modes, the coefficients are calculated analogously to the definition from signal processing, which calculates the correlation over the convolution. In *Normed* mode, the coefficients are also divided by the number of summands. *Covariance* and *CovarianceBessel* calculate the covariance without and with Bessel correction. *Pearson* mode uses the definition of the Pearson correlation coefficient commonly used in statistics. The exact calculation rules are mathematically listed in the configuration options. Here, x_n denotes the value of the reference signal and y_n denotes the value of the second signal (in each case Channel00, ..., Channel0n) at the timestamp t_n (corresponding to the n th cycle since the start of the analysis or since reset, except for the cycles in which *Enable Execution* = FALSE). The value of N depends on the *WindowMode* you select. For *SlidingWindow* mode and *FixWindow* mode, N is equal to the *WindowSize*, provided a corresponding number of cycles have already elapsed since the start of the analysis, so that x_{n-N-m} (or y_{n-N+m}) has been recorded, otherwise N will be reduced to $n-m+1$ or $n+m+1$ respectively. Note that in *FixWindow* mode the output values are only updated every *WindowSize* cycle. In *Continuous* mode, $N = n+1$ always applies.

In the illustration, the output values of the function block for two signals (Channel Ref and Channel 00) for a given cycle ($n = 150$) are shown as an example of a configuration (*Correlation Mode* = *Pearson*, *Window Mode* = *FixWindow*, *Window Size* = 75, *Step Size* = 5, *Maximum Lag* = 50, *Minimum Lag* = -50). In the two left plots, the input signals Channel Ref and Channel 00 are shown over time. The right plot shows the discrete correlation function (the Pearson correlation coefficients in relation to m). Coefficients are shown for the shifts $m = -50, -45, -40, -35, -30, -25, -20, -15, -10, -5, 0, +5, +10, +15, +20, +25, +30, +35, +40, +45, +50$. These are output as an array in the function block. In addition to the two parameters *Minimum Lag* and *Maximum Lag*, the output values *Minimizing Lag* and *Maximizing Lag* are marked on the abscissa. The corresponding coefficients *Minimum Coefficient* and *Maximum Coefficient*, which also represent outputs of

the function block, are marked on the ordinate. For the shifts $m = -25$, $m = 0$ and $m = +25$ in the plots of the input channels (left), the time ranges included in the calculation of the respective coefficient are highlighted in color. In the right plot, the corresponding points are colored.



Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** The number of channels that are correlated with the reference signal. This can be set via Add/Remove Channel
- **Maximum Lag:** Specifies the maximum number of cycles by which the two signals are shifted to calculate the correlation to each other. This is a positive integer.
- **Minimum Lag:** Specifies the minimum number of cycles by which the two signals are shifted to calculate the correlation to each other. This is a negative integer.
- **Step Size:** Specifies by how many cycles the signals are shifted to calculate two consecutive correlation coefficients.
- **Window Size:** For the SlidingWindow and FixWindow window modes, specifies the number of cycles over which the coefficients are calculated. The windowing has no effect for the *Continuous* Window Mode and cannot be set. In *SlidingWindow* mode, Window Size values for all channels are buffered in addition to Maximum Lag values from the Reference Channel and Minimum Lag values for Channel 00 to Channel 0n. The size of the router memory should therefore be taken into account when setting these parameters.
- **Correlation Mode:** The correlation coefficients are calculated according to one of the following definitions:
- *Base:*

$$C_{xy}[m, t_n] = \begin{cases} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i}, & m \geq 0 \\ \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m}, & m < 0 \end{cases}$$

- *Normed:*

$$\hat{c}_{xy}[m, t_n] = \begin{cases} \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i}, & m \geq 0 \\ \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m}, & m < 0 \end{cases}$$

- *Covariance*:

$$cov_{xy}[m, t_n] = \begin{cases} \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i}, & m \geq 0 \\ \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i+m}, & m < 0 \end{cases}$$

- *CovarianceBessel*:

$$\widetilde{cov}_{xy}[m, t_n] = \frac{N}{N-1} cov_{xy}[m, t_n]$$

- *Pearson*:

$$\rho_{xy}[m, t_n] = \frac{cov(x, y)[m, t_n]}{\sigma_x[m, t_n] \sigma_y[m, t_n]}, \text{ where } \sigma_x^2[m, t_n] = cov_{xx}[m, t_n]$$

- **Window Mode**: Specifies the type of window used to calculate the coefficients:

Continuous: All values since the start of the analysis are included in the analysis with equal weighting.

SlidingWindow: The calculation is done via a window of the size Window Size. The current values are always included in the analysis and outputs are updated with each cycle.

FixWindow: The outputs are updated every Window Size cycles and calculated via a window with the length Window Size.

Output values

- **Output00 .. Output0n**: Shows for each input (Channel00, ..., Channel0n) the coefficients for the shifts $m = \text{Minimum Lag} + \text{Step Size}$, ..., $m = -\text{Step Size}$, $m = 0$, $m = +\text{StepSize}$, ..., $m = \text{Maximum Lag}$ as an array.
- **Minimizing Lag00..Minimizing Lag0n**: For each channel, specifies the shift for which the coefficient becomes minimum.
- **Minimum Coefficient00..Minimum Coefficient0n**: Specifies the minimum coefficient for each channel.
- **Maximizing Lag00..Maximizing Lag0n**: for each channel, specifies the shift for which the coefficient becomes maximum.
- **Maximum Coefficient 00..00.. Maximum Coefficient0n**: Specifies the maximum coefficient for each channel.

Standard HMI Controls

For the Correlation Function algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control or Multivalue Control visualizes all output values: minimum coefficient, maximum coefficient, minimum lag, maximum lag and the coefficient array.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42 Input 1
25.43 Input 2
TRUE Executing
10 Mar 2021 16:15:30 Last Event

Alternatively, customer-specific HMI controls can be mapped in the Correlation Function algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.10.4 Correlation Function Reference

The *Correlation Function Reference* function block calculates the discrete correlation function between a recorded signal (referred to below as reference signal), which is read from a tcab file, and one or more input signals (Channel 00, ..., Channel 0n).

The correlation coefficients are calculated for time shifts of m cycles between the two signals, with the maximum and minimum values for m being limited by the parameters *Minimum Lag* (negative integer) and *Maximum Lag* (positive integer).

The *Step Size* parameter determines the number of cycles by which the signals are moved to calculate two consecutive correlation coefficients. I.e. m is always a multiple of the *Step Size*. Accordingly, only multiples of the *Step Size* are also allowed for *Minimum Lag* and *Maximum Lag*. If the *Step Size* is set to one, *Minimum Lag* to -6 and *Maximum Lag* to +4, for example, coefficients are calculated for shifts by -6, -5, -4, -3, -2, -1, 0, +1, +2, +3, and +4 cycles. If the *Step Size* is set to two, coefficients are calculated for shifts by -6, -4, -2, 0, +2, and +4 cycles.

From the start of the analysis, a value is processed from the read-in signal per cycle. When the end of the file is reached, the process starts again with the first value of the file. The read-in signal is therefore assumed to be periodic. If you only want to correlate certain periods of the input signal with the reference signal, you can control this via the *Enable Execution* and *Reset* inputs as well as via the *New Result* output.

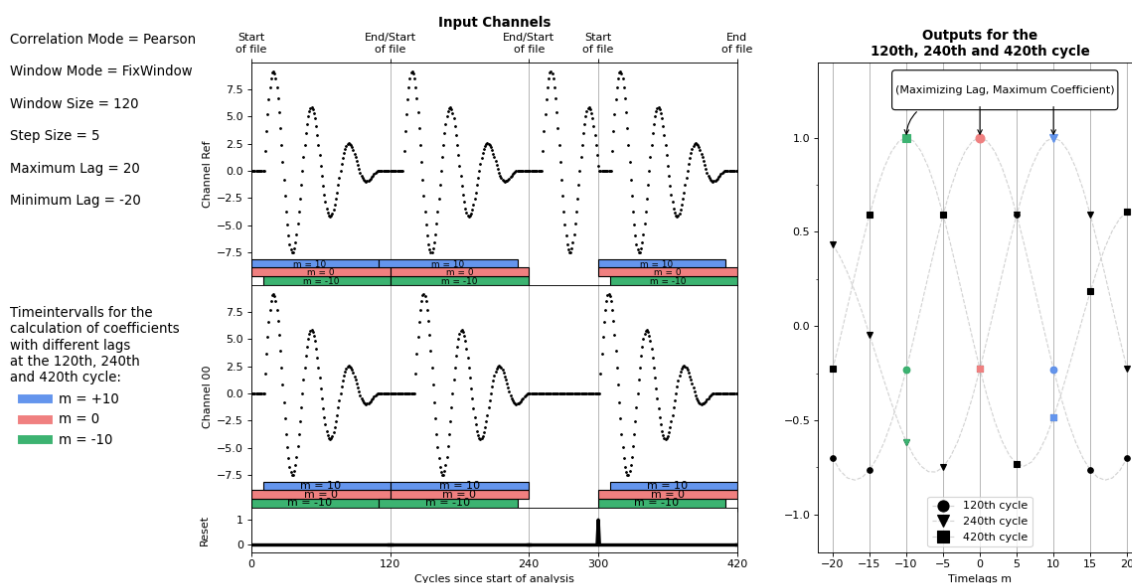
The correlation coefficients can be calculated over different timeframes, which are set by the *Window Mode* parameter. In *Continuous* mode, all values since the beginning of the analysis are included in the calculations. In *SlidingWindow* mode, the calculation runs continuously for the last number of cycles set by the *Window Size*. In *FixWindow* mode, the calculation is done over the length of the reference signal and the output values are always updated at the end of the recorded sequence.

If m is not zero, the window for the corresponding signal shifts by m cycles. The number of values included in the calculation of the coefficients is the same for all values of m . The only exceptions to this are the results from the first cycles after the start of the analysis. If the number of elapsed cycles is less than $|m|$, correspondingly fewer values are included in the calculation.

For positive values of m , the values of the reference signal (Channel Ref) are stored in a ring memory, so that the respective value of the reference signal received before m cycles can be compared with the current values from Channel 00 to Channel 0n. This corresponds to a shift of the reference signal into the past. For negative values of m , the reference signal would accordingly have to be moved into the future. Since this is obviously not possible, the second signal (Channel 00, .. Channel 0n) is saved and moved backward instead.

The correlation coefficients can be calculated according to different calculation rules. This is determined by the *Correlation Mode*. In the *Base* and *Normed* modes, the coefficients are calculated analogously to the definition from signal processing, which calculates the correlation over the convolution. In *Normed* mode, the coefficients are also divided by the number of summands. *Covariance* and *CovarianceBessel* calculate the covariance without and with Bessel correction. *Pearson* mode uses the definition of the Pearson correlation coefficient commonly used in statistics. The exact calculation rules are mathematically listed in the configuration options. Here, x_n denotes the value of the reference signal and y_n denotes the value of the input signal (in each case Channel 00, ..., Channel 0n) at the timestamp t_n (corresponding to the n^{th} cycle since the start of the analysis or since reset, except for the cycles in which *Enable Execution* = FALSE). The value of N depends on the *WindowMode* you select. For *SlidingWindow* mode and *FixWindow* mode, N is equal to the *WindowSize*, provided a corresponding number of cycles have already elapsed since the start of the analysis, so that x_{n-N-m} (or y_{n-N+m}) has been recorded, otherwise N will be reduced to $n-m+1$ or $n+m+1$ respectively. In *FixWindow* mode, the window size is not to be set manually, but corresponds to the length of the read-in signal section (reference signal) and the output values are updated at the end of the signal section. In *Continuous* mode, $N = n+1$ always applies.

The illustration shows the different input and output values of the function block for a configuration (*Correlation Mode = Pearson*, *Window Mode = FixWindow*, *Window Size = 120* (= number of values in the file), *Step Size = 5*, *Maximum Lag = 20*, *Minimum Lag = -20*) and an input channel. On the left side, the input signals Channel 00 and *Reset* are shown in the two lower plots, above which the read-in sequence is shown on the same timeline, according to its processing. The signal starts to be read in at the beginning of the analysis. If the last value from the file is processed in the 120th (or 240th) cycle, the process starts again with the first one. In the 300th cycle, a *reset* is performed and the process starts again with the first value of the file. In addition, all values are deleted from the internal memories and the calculation of the coefficients begins again. For example, it was detected here that Channel 00 did not contain valid values in the previous cycles and the vibration has now been re-energized. In this area the analysis could also be interrupted by *Enable Execution = FALSE*. Since *WindowMode = FixWindow*, the outputs are updated only in the 120th, 240th and 420th cycle. The corresponding coefficients (from Output00) are shown in the right plot. From the value pairs (*Maximizing Lag*, *Maximum Coefficient*) you can read how much the input signal is shifted with respect to the reference signal. For example, in the 420th cycle (*Maximizing Lag*, *Maximum Coefficient*) = (-10.1). This means that if the *Reset* had taken place 10 cycles earlier, the reference signal and Channel 00 would have matched each other exactly in this window.



Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** The number of channels that are correlated with the reference signal. This can be set via Add/Remove Channel
- **File Path:** Path to the data file.
- **Maximum Lag:** Specifies the maximum number of cycles by which the two signals are shifted to calculate the correlation to each other. This is a positive integer.
- **Minimum Lag:** Specifies the minimum number of cycles by which the two signals are shifted to calculate the correlation to each other. This is a negative integer.
- **Step Size:** Specifies by how many cycles the signals are shifted to calculate two consecutive correlation coefficients.
- **Window Size:** For the *SlidingWindow* and *FixWindow* window modes, specifies the number of cycles over which the coefficients are calculated. The windowing has no effect for the *Continuous* Window Mode and cannot be set. In *SlidingWindow* mode, Window Size values for all channels are buffered in addition to Maximum Lag values from the Reference Channel and Minimum Lag values for Channel 00 to Channel 0n. The size of the router memory should therefore be taken into account when setting these parameters.
- **Correlation Mode:** The correlation coefficients are calculated according to one of the following definitions:

- *Base:*

$$C_{xy}[m, t_n] = \begin{cases} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i}, & m \geq 0 \\ \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m}, & m < 0 \end{cases}$$

- *Normed:*

$$\hat{C}_{xy}[m, t_n] = \begin{cases} \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i}, & m \geq 0 \\ \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m}, & m < 0 \end{cases}$$

- *Covariance:*

$$cov_{xy}[m, t_n] = \begin{cases} \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i}, & m \geq 0 \\ \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i+m}, & m < 0 \end{cases}$$

- *CovarianceBessel:*

$$\widetilde{cov}_{xy}[m, t_n] = \frac{N}{N-1} cov_{xy}[m, t_n]$$

- *Pearson:*

$$\rho_{xy}[m, t_n] = \frac{cov(x, y)[m, t_n]}{\sigma_x[m, t_n] \sigma_y[m, t_n]}, \text{ where } \sigma_x^2[m, t_n] = cov_{xx}[m, t_n]$$

- **Window Mode:** Specifies the type of window used to calculate the coefficients:

Continuous: All values since the start of the analysis are included in the analysis with equal weighting.

SlidingWindow: The calculation is done via a window of the size Window Size. The current values are always included in the analysis and outputs are updated with each cycle.

FixWindow: The coefficients are calculated over the entire length of the read-in signal section and the outputs are updated after processing the last value.

Output values

- **Output00 .. Output0n:** Shows for each input (Channel00, ..., Channel0n) the coefficients for the shifts $m = \text{Minimum Lag}$, $m = \text{Minimum Lag} + \text{Step Size}$, ..., $m = -\text{Step Size}$, $m = 0$, $m = +\text{Step Size}$, ..., $m = \text{Maximum Lag}$ as an array.
- **Minimizing Lag00..Minimizing Lag0n:** For each channel, specifies the shift for which the coefficient becomes minimum.

- **Minimum Coefficient00..Minimum Coefficient0n:** Specifies the minimum coefficient for each channel.
- **Maximizing Lag00..Maximizing Lag0n:** for each channel, specifies the shift for which the coefficient becomes maximum.
- **Maximum Coefficient 00..00.. Maximum Coefficient0n:** Specifies the maximum coefficient for each channel.

Standard HMI Controls

For the Correlation Function Reference algorithm, the following HMI controls are available for generating an Analytics Dashboard:

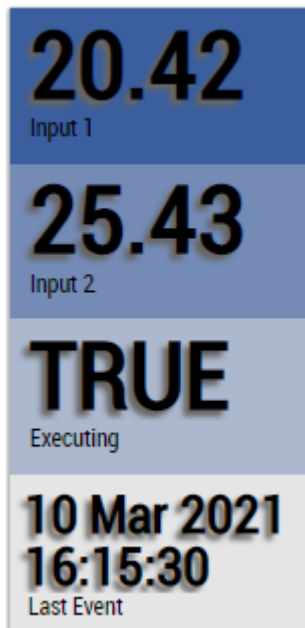
1. The Table Control or Multivalue Control visualizes all output values: read value, minimum coefficient, maximum coefficient, minimum lag, maximum lag and the coefficient array.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

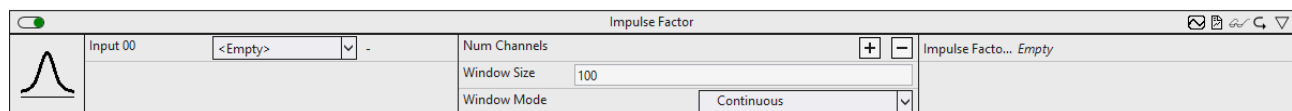
DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Correlation Function Reference algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.10.5 Impulse Factor



The algorithm *Impulse Factor* calculates the signal feature of the same name from the input values. The output value is calculated from the ratio of the peak value of the input signal to the mean value of the input signal.

$$\text{Impulse Factor} = \frac{\max(|x|)}{\frac{1}{N} \sum_{n=1}^N |x[n]|}$$

The number of input data to be included in the calculation and the type of calculation can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** Configuration of the number of independent input and output channels.
- **Window Mode:** Used window mode. Influences the number of input data that are included in the calculation as well as the time of the calculations.
Continuous: All input values since the start of the algorithm are included in the calculation. The calculation is cyclic.
Fix Window: Only the last *N* input values are included in the calculation. The calculation is performed every *N* calls.
Sliding Window: Only the last *N* input values are included in the calculation. The calculation is performed cyclically.
- **Window Size:** Depending on the window mode used, you can configure the number of values *N* that will be included in the calculation. In the window mode *Continuous* this parameter is ignored.

Output values

- **Impulse Factor 00..n:** Impulse factor of the input channels.

Standard HMI Controls

For the *Impulse Factor* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control and Multivalue Control visualize all output values.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped using the [Mapping Wizard](#) [► 382].

5.2.1.10.6 Linear Regression Fitting

Linear Regression Fitting						
Dependent	<Empty>	-	Num Channels	+ -	Result	Empty
Fit	<Empty>	-	Step Size	0.001	MSE	Empty
Input 01	<Empty>	-	Involve Existing File	☒	Output 00	Empty
			File Path	Project\Teach\LinearRegression.json	Output 01	Empty

The Linear Regression Fitting function block approximates one variable (the Dependent input) by linear combination of several other variables (Input 01 ... Input 0n). This is done by the incremental stochastic gradient method. At the end of the analysis, the calculated coefficients are written to a file.

The linear combination is given by the following equation:

$$y = \beta_0 + \sum_{i=1}^n \beta_i \times \text{Input } 0i$$

In each cycle, the values for β_0 to β_n are recalculated using the following rule:

$$\beta_i = \begin{cases} \beta_i - \gamma \times \text{Input } 0i \times (y - \text{Dependent}), & i = 1, 2, \dots, n \\ \beta_i - \gamma \times (y - \text{Dependent}), & i = 0 \end{cases}$$

This corresponds to the minimization of the squared deviation of the calculated values y (output by the

function block as result) from the corresponding input value Dependent. The parameter γ corresponds to the step size and specifies how strongly the parameters are adjusted. The larger the value, the faster the coefficients approach a local optimum. However, if the value is too large, the algorithm may not converge.

Typically, the Linear Regression Fitting function block is first used to fit the weights for the regression of a target variable. Then, using the Linear Regression Inference function block and the fitted weights, the target variable can be predicted based on the input variables.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** The number of channels that are correlated with the reference signal. This can be set via Add/Remove Channel
- **File Path:** Specifies the file path for the file in which the coefficients are saved at the end of the approximation process. File type: *json* or *tas*.
- **Involve Existing File:** If TRUE, the values for the coefficients are read from the file at the start of the analysis and then adjusted further. If FALSE, all coefficients are set to zero at the start.
- **Step Size:** Specifies how much the coefficients are adjusted after each new calculation.
- **Bias (optional):** If FALSE, the Bias Output 00 is set to zero and is not approximated further.
- **Mini Batch Size (optional):** Specifies over how many cycles the MSE is to be calculated before the coefficients are adjusted based on it.

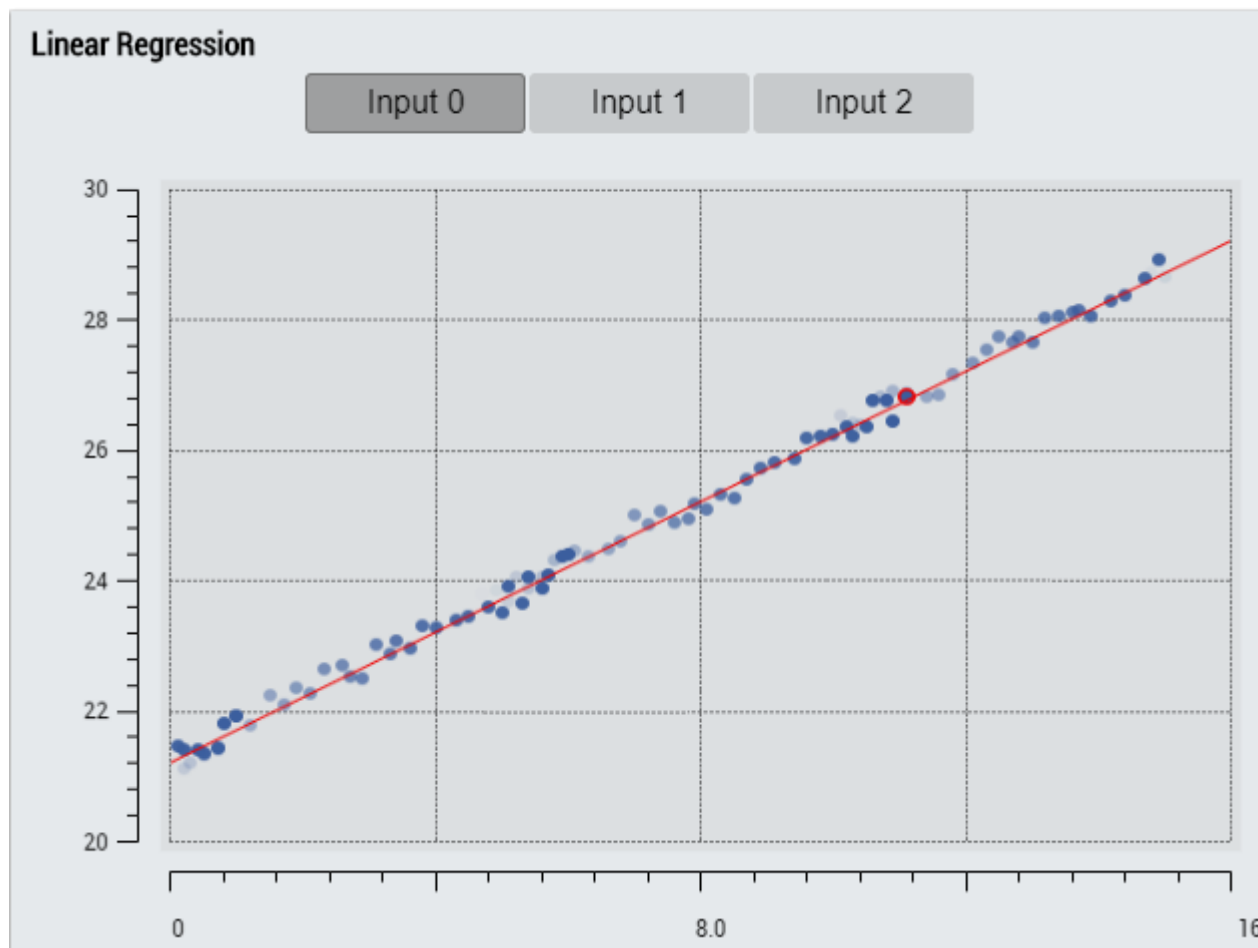
Output values

- **MSE:** Specifies the MSE (mean squared error) between the calculated *Result* value and the *Dependent* input value.
- **Result:** Outputs the approximated value for the inputs of the current cycle with the coefficients updated from them.
- **Output00 .. Output0n:** Shows the calculated coefficients.

Standard HMI Controls

For the Linear Regression Fitting algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Linear Regression Control visualizes the inputs and the calculated regression line. The buttons can be used to select the input channel to be displayed on the x axis. The y axis shows the target values of the regression. A new point is outlined in red, old points gradually fade.

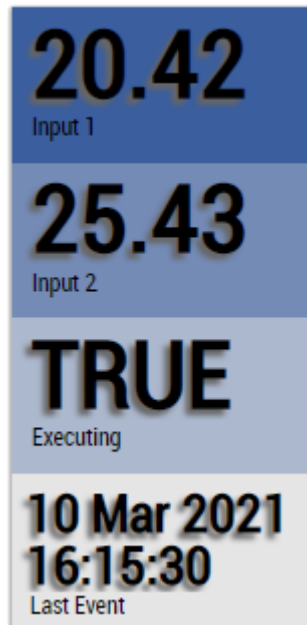


2. The Table Control or Multivalue Control visualizes all output values: MSE, result, output coefficients.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Linear Regression Curve Fitting algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.10.7 Linear Regression Inference

Linear Regression Inference			
Input 01	<Empty>	Num Channels	+ -
		Weights 00	0
		Weights 01	0
		Result	Empty

The Linear Regression Inference function block calculates the linear combination of the inputs (Input 01 .. Input 0n) with the coefficients (Weights 00.. Weights 0n).

$$\text{Result} = \text{Weights } 00 + \sum_{i=1}^n \text{Weights } 0i \times \text{Input } 0i$$

The parameters Weights 00 to Weights 0n can either be set manually or automatically via a file generated by the Linear Regression Fitting function block by dragging / dropping onto the parameter field. Typically, the Linear Regression Fitting function block is first used to fit the weights for the regression of a target variable. Then, using the Linear Regression Inference function block and the fitted weights, the target variable can be predicted based on the input variables.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** The number of channels that are correlated with the reference signal. This can be set via Add/Remove Channel

- **Weights00 .. Weights 0n:** Specifies the coefficients that determine the linear combination to be calculated.
- **File Path (optional):** File path to a file with previously determined parameters. The file is created by the *Linear Regression Fitting* module. File type: *json* or *tas*.

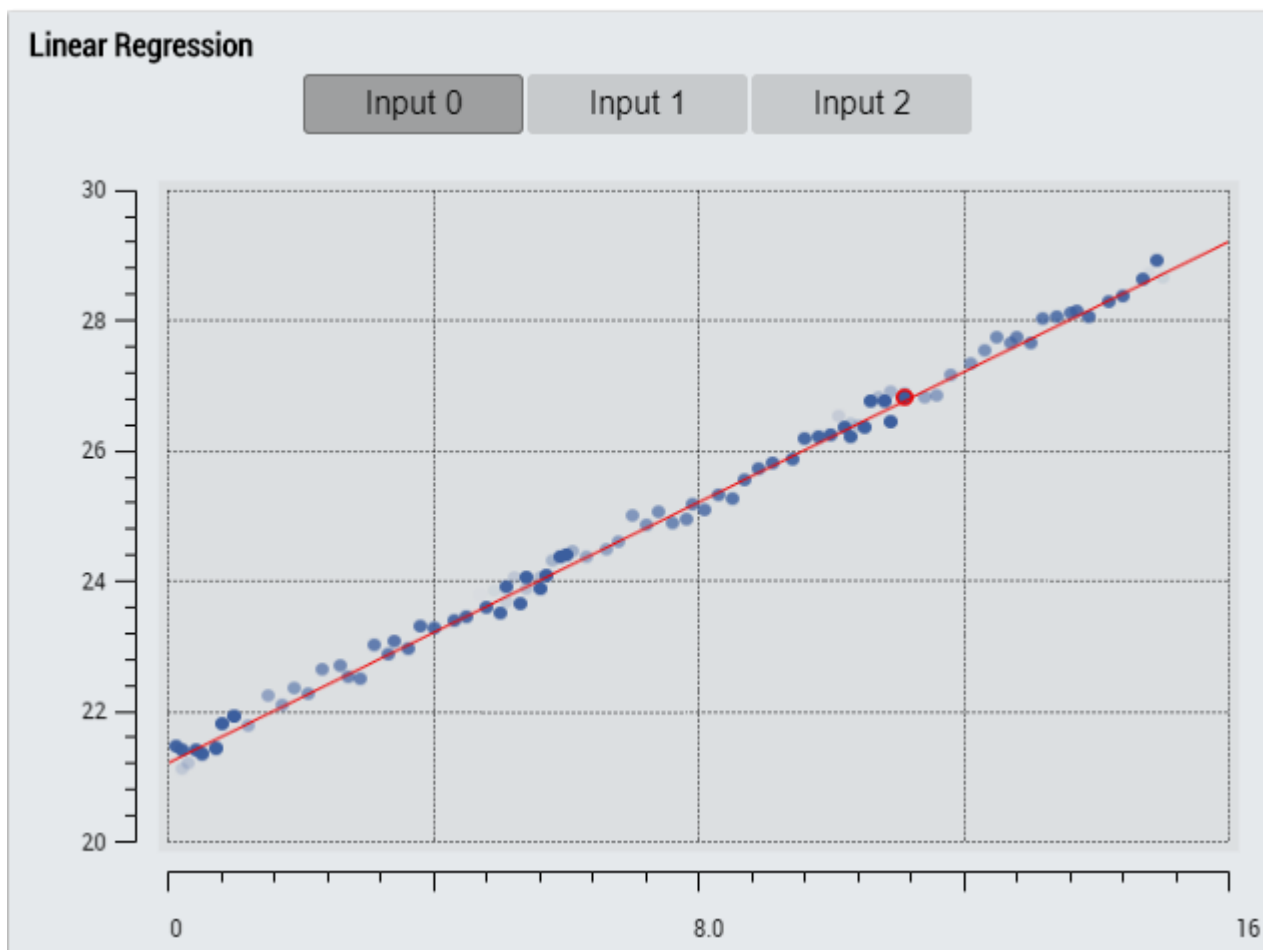
Output values

- **Result:** Specifies the value calculated from the linear combination.

Standard HMI Controls

For the Linear Regression Inference algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Linear Regression Control visualizes the inputs and the calculated regression line. The buttons can be used to select the input channel to be displayed on the x axis. The y axis shows the target values of the regression. A new point is outlined in red, old points gradually fade.



2. The Table Control or Multivalue Control visualizes all output values: MSE, result, output coefficients.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021 16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Linear Regression Inference algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.10.8 Shape Factor



Input 00

<Empty>

-

Num Channels

+

-

Window Size

100

Window Mode

Continuous

▼

Shape Factor 00 Empty

The algorithm *Shape Factor* calculates the signal feature of the same name from the input values. The output value is calculated from the ratio of the root mean square (RMS) of the input signal to the mean of the absolute values of the input signal.

$$\text{Shape Factor} = \frac{x_{\text{rms}}}{\frac{1}{N} \sum_{n=1}^N |x[n]|}$$

The number of input data to be included in the calculation and the type of calculation can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** Configuration of the number of independent input and output channels.
- **Window Mode:** Used window mode. Influences the number of input data that are included in the calculation as well as the time of the calculations.
Continuous: All input values since the start of the algorithm are included in the calculation. The calculation is cyclic.
Fix Window: Only the last *N* input values are included in the calculation. The calculation is performed every *N* calls.
Sliding Window: Only the last *N* input values are included in the calculation. The calculation is performed cyclically.
- **Window Size:** Depending on the window mode used, you can configure the number of values *N* that will be included in the calculation. In the window mode *Continuous* this parameter is ignored.

Output values

- **Shape Factor 00..n:** Shape factor of the input channels.

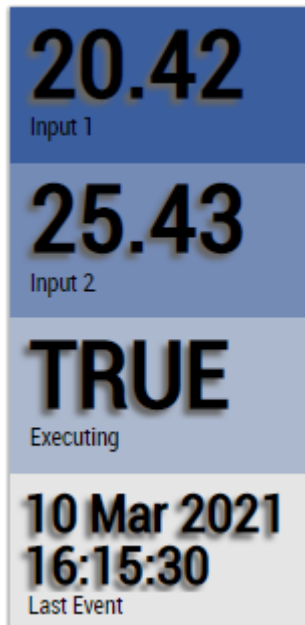
Standard HMI Controls

For the *Shape Factor* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control and Multivalue Control visualize all output values.

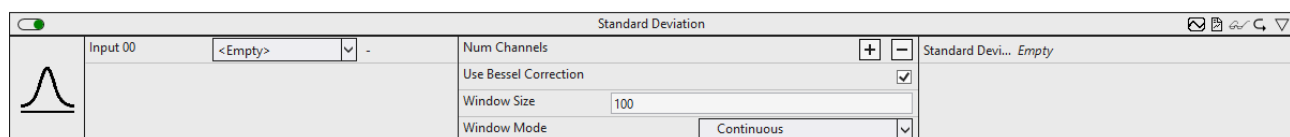
Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped using the [Mapping Wizard](#) [► 382].

5.2.1.10.9 Standard Deviation



The *Standard Deviation* algorithm calculates the empirical standard deviation for a configurable number of input channels. The number of input data to be included in the calculation and the type of calculation can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Channels:** Configuration of the number of independent input and output channels.
- **Use Bessel Correction:** If the checkbox is checked, Bessel's correction is applied. This parameter must be enabled in order to obtain an expected result for random samples.

The empirical standard deviation, without Bessel's correction

$$s' = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

The empirical standard deviation, with Bessel's correction

$$s = \sqrt{\frac{1}{N-1} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

- **Window Mode:** Window mode used. Influences the amount of input data included in the calculation and the timing of the calculations.
Continuous: All input values since the start of the algorithm are included in the calculation. The calculation is performed cyclically.
Fix Window: Only the last N input values are included in the calculation. The calculation is performed every N calls.
Sliding Window: Only the last N input values are included in the calculation. The calculation is performed cyclically.
- **Window Size:** The number of values N that are included in the calculation can be configured depending on the window mode used. With the window mode *Continuous* this parameter is ignored.

Output values

- **Standard Deviation 00..n:** Empirical standard deviation of the input channels.

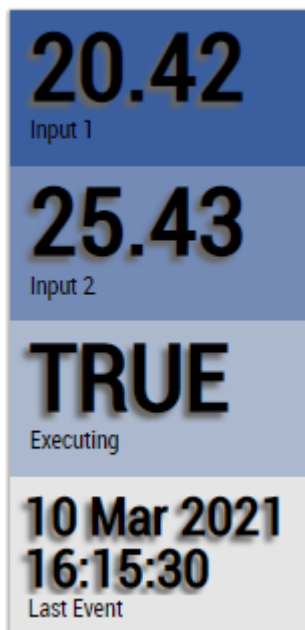
Standard HMI Controls

For the Standard Deviation algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Table Control or Multivalue Control visualizes all output values: Standard Deviation Results.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the Standard Deviation algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.11 Analytics - Training Base

The algorithms of the category *Analytics – Training Base* provide functionalities for teaching periodic signals and write this data to a file. So that it is possible to compare it later on to another input signal, to analyze differences from the optimal behavior.

5.2.1.11.1 Time Based Teach Path 1Ch

Time Based Teach Path 1Ch			
	Input	<Empty>	-
	Start Period	<Empty>	-
	Stop Period	<Empty>	-
	Teach mode	Maximum	
	Number Of Teaches	0	
		Involve Existing File	☒
	File Path	C:\Analytics Project\Teach	
		Executing Tea...	Empty
		Teaching Done	Empty
		Written Values	Empty
		Values in File	Empty
		Current Teach...	Empty

Time Based Teach Path 1Ch periodically writes the input data to a file according to the configured number of teach operations. This means that the values are not written sequentially for each period, but the values of a new period are compared with the existing values. The period can be defined by the input values *Start Period* and *Stop Period* (boolean signals are required). According to the teach mode, each value is overwritten or retained, so that the result is a taught input signal that can later be used as a reference signal for the [Time Based Envelope 1Ch](#) [► 115] algorithm, for example.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

It is recommended that you do not use *Time Based Teach Path 1Ch* at the same time as [Time Based Envelope 1Ch](#) [► 115] due to competing file access. Instead, the reference signal should be learned first and only then should an evaluation with [Time Based Envelope 1Ch](#) [► 115] be performed.

Configuration Options

- **Teach mode:** Mode for teaching. Defines according to which criteria the values will be compared (*Minimum*, *Maximum* or *Mean*). In case of *Mean* a weighted average is calculated, in order to ensure that the values of a later period do not have an increasing weight regarding the total result.
- **Number of Teaches:** Amount of cycles the teach process should be stopped after automatically. If set to 0 the teaching is processed continuously.

- **Involve Existing File:** If the checkbox is checked and a file with data already exists, the values of the existing file will be included in the teach process. Otherwise the existing file will be ignored and overwritten.
- **File Path:** Path to the data file.
- **Negate Start Period:** If the checkbox is checked the Boolean input signal of the *Start Period* is negated.
- **Negate Stop Period:** If the checkbox is checked the Boolean input signal of the *Stop Period* is negated.

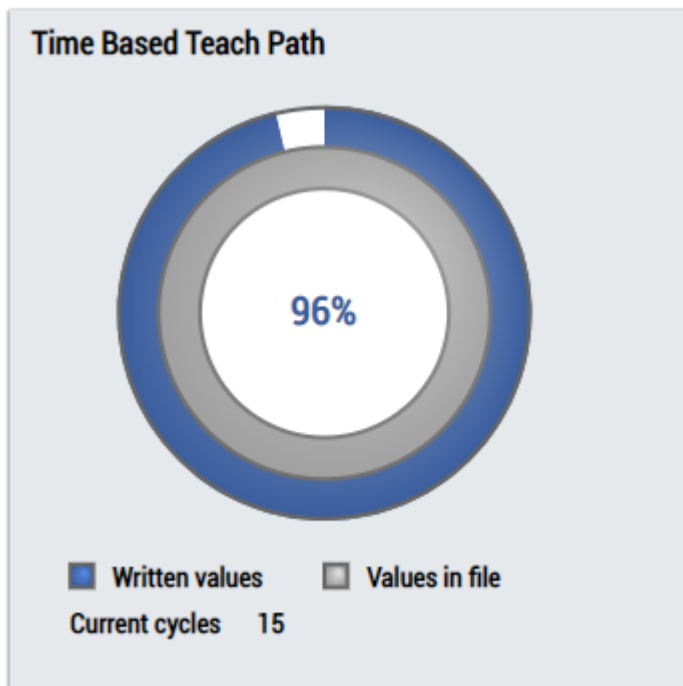
Output Values

- **Executing Teach:** Shows if the teaching is active (time range between start and stop flag).
- **Written Values:** Shows the total amount of written values during the teach process. Not to be confused with the amount of values in File, which are overwritten each teach cycle.
- **Values in File:** Shows the amount of values which are written currently into the file (after one cycle the value will be constant).
- **Current Teach Cycles:** Shows the amount of teach cycles within the file.

Standard HMI Controls

For the Time Based Teach Path 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The TimeBasedTeachPath Control visualizes the output values Written Values, Values in File and Current Teach Cycles.



2. The Table Control or Multivalue Control visualizes all output values: Written Values, Values in File, Current Teach Cycles.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the Time Based Teach Path 1Ch algorithm using the [Mapping Wizard](#) [► 382].

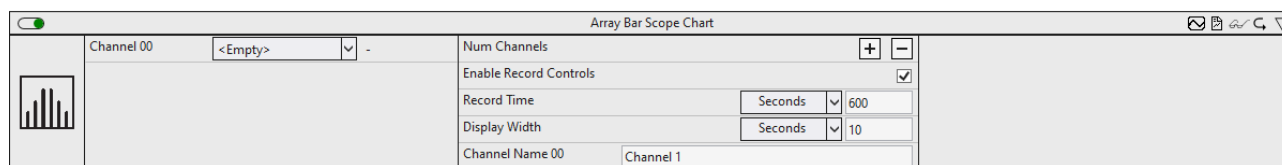
5.2.1.12 **Analytics - Visualization Only**

The algorithms in the *Visualization Only* category represent auxiliary function blocks that provide functions for subsequent visualization in the context of the HMI dashboard.

5.2.1.12.1 Array Bar Scope Chart



The generation of the HMI Scope Control for the HMI One Click Dashboard is only possible with an HMI version > 1.12.752.0.



The *Array Bar Scope Chart* function block is used to display the inputs in the HMI Dashboard. The inputs are displayed in an Array Bar chart in the HMI Scope Control.

Since the function of this function block is irrelevant for the analysis itself, it belongs to the Visualization Only category.

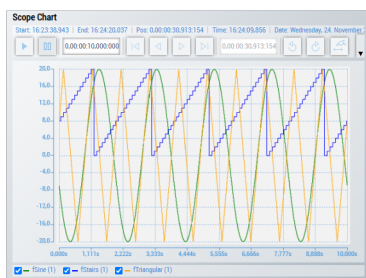
Configuration options

- **Num Channels:** the number of inputs to be displayed in the HMI Scope Control.
- **Channel Name 00/XX:** the display names of the inputs. These appear as legends in the HMI Scope Control.
- **Enable Record Controls:** if the checkbox is activated, a recording can be started and stopped from the HMI.
- **Record Time:** the recording time of a record. After reaching the time, the scope chart runs in a ring buffer.
- **Display Width:** the selected display width of the image. This property can also be modified in the HMI Scope Control.

Standard HMI Controls

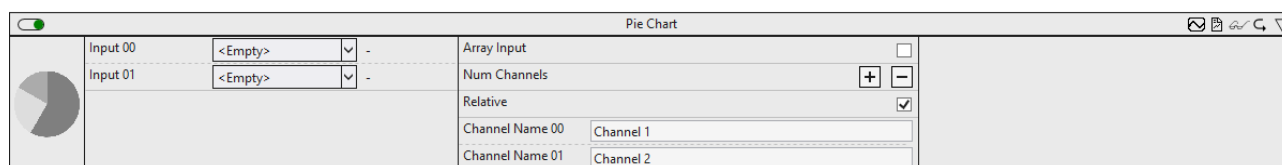
For the algorithm *Array Bar Scope Chart* the following HMI Control is available for generating an Analytics Dashboard.

The HMI Scope Control for displaying a Scope Chart.



No other HMI controls can be added to the function block, as it is intended exclusively for displaying the HMI Scope Control.

5.2.1.12.2 Pie Chart



The *Pie Chart* function block can be used to create a pie chart for later use in the HMI Dashboard. The number of pie pieces can be parameterized via the parameter *Num Channels*.

Since the function of this function block is irrelevant for the analysis itself, it belongs to the Visualization Only category.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

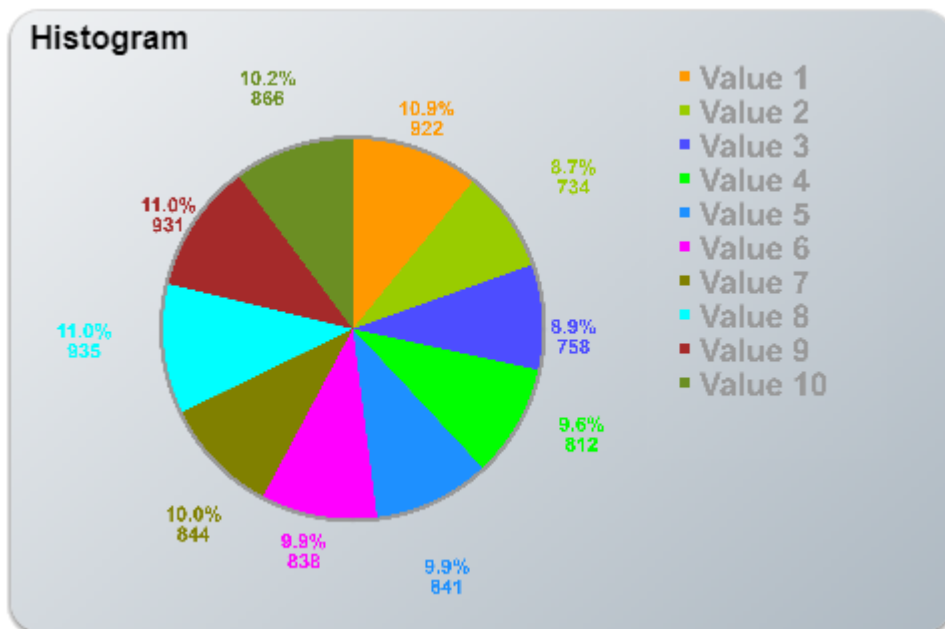
Configuration options

- **Array Input:** Selection of whether a single array should be used as an input or several independent inputs. Is set automatically as soon as an array is dragged onto the first input. As long as the array is still linked, the parameter cannot be reset.
- **Num Channels:** Specifies the number of configurable channels. This corresponds to the number of pie pieces in the pie chart.
- **Channel Name 00:** Specifies the name of the first channel. The name assigned here will later be used for this channel in the HMI Dashboard.
- **Channel Name 01:** Specifies the name of the second channel. The name assigned here will later be used for this channel in the HMI Dashboard.
- **Channel Name xx:** Analog for all other channels.

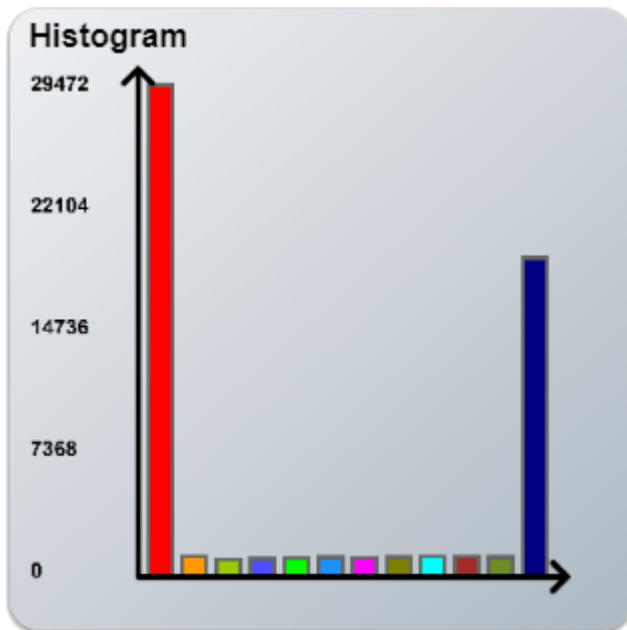
Standard HMI Controls

For the *Pie Chart* algorithm the following HMI controls are available for generating an Analytics Dashboard.

Pie Chart to display the data in a pie chart.



Histogram to display the data in a histogram.



Alternatively, customer-specific HMI controls can be mapped in the *Pie Chart* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.12.3 Sankey diagram

Sankey Diagram					
Input Inflow 00	<Empty>	-	Num Channels Inflow	+	-
Input Outflow 00	<Empty>	-	Num Channels Outflow	+	-
Input Outflow 01	<Empty>	-	Channel Name Rest	Rest	
			Channel Name Inflow 00	Inflow 1	
			Channel Name Outflow 00	Outflow 1	
			Channel Name Outflow 01	Outflow 2	
			Rest [%]	Empty	
			Output Inflow [%]		
			Output Outflow [%]		

The *Sankey Diagram* function block can be used to create a Sankey diagram for later use in the HMI Dashboard. A distinction is made between inflows and outflows. The number of inflows and outflows can be individually parameterized via the parameters *Num Channels Inflow* and *Num Channels Outflow* respectively.

Since the function of this function block is irrelevant for the analysis itself, it belongs to the Visualization Only category.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

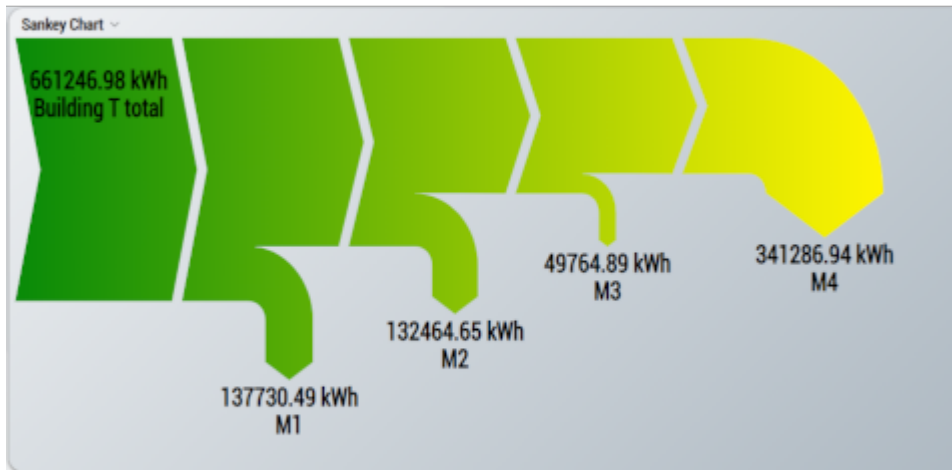
Configuration options

- **Num Channels Inflow:** specifies the number of channels of inflows in the Sankey diagram.
- **Num Channels Outflow:** specifies the number of channels of outflows in the Sankey diagram.
- **Channel Name Rest:** specifies the name of the channel with which a possible rest value is to be designated. The name assigned here will be used later for this outflow in the HMI Dashboard.
- **Channel Name Inflow 00:** specifies the name of the first inflow channel. The name assigned here will be used later for this inflow in the HMI Dashboard.
- **Channel Name Inflow xx:** analog for all other inflow channels.
- **Channel Name Outflow 00:** specifies the name of the first outflow channel. The name assigned here will be used later for this outflow in the HMI Dashboard.
- **Channel Name Outflow 01:** specifies the name of the second outflow channel. The name assigned here will be used later for this outflow in the HMI Dashboard.
- **Channel Name Outflow xx:** analog for all other channels.

Standard HMI Controls

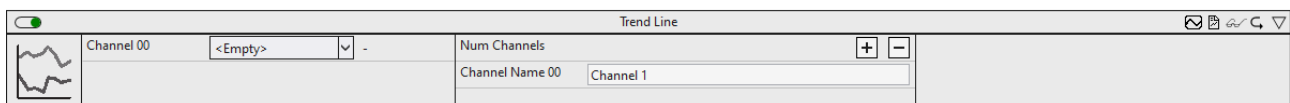
For the *Sankey Diagram* algorithm the following HMI control is available for generating an Analytics Dashboard.

Sankey chart to display the data.



Alternatively, customer-specific HMI controls can be mapped in the *Sankey Diagram* algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.12.4 Trend Line



The *Trend Line* function block provides for the forwarding of trends of the input signals for later use in the HMI dashboard. Any channels can be selected for forwarding; the number of channels can be controlled via NumChannels. One value per minute is then stored for each channel. This means that the function block is not suitable for signal forwarding, but rather for displaying trends for signals that change slowly.

Since the function of this function block is irrelevant for the analysis itself, it belongs to the Visualization Only category.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

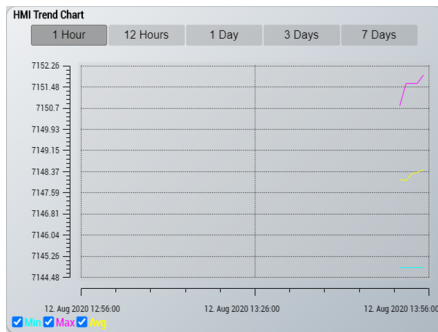
- **Num Channels:** Indicates the number of configurable channels for forwarding the input signals.
- **Channel 00:** Indicates the name of the *first* channel to be forwarded. The name assigned here will later be used for this channel in the HMI Dashboard.

- **Channel 01:** Indicates the name of the *second* channel to be forwarded. The name assigned here will later be used for this channel in the HMI Dashboard.
- **Channel 02:** Indicates the name of the *third* channel to be forwarded. The name assigned here will later be used for this channel in the HMI Dashboard.
- **Channel Name xx:** Analog for all other channels.

Standard HMI Controls

For the algorithm *Trend Line* the following HMI Control is available for generating an Analytics Dashboard.

Trend Line to display the data over time.

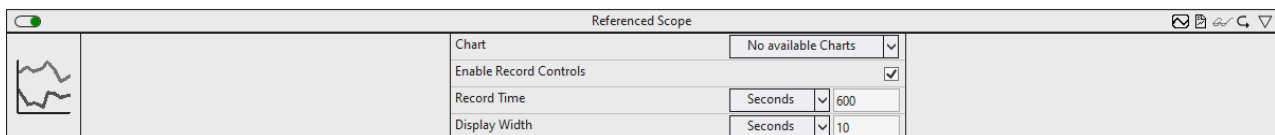


No other HMI controls can be added to the function block, as it is intended exclusively for displaying the Trend Line Control.

5.2.1.12.5 Referenced Scope



The generation of the HMI Scope Control for the HMI One Click Dashboard is only possible with an HMI version > 1.12.752.0.



The *Referenced Scope* function block is used to display a Scope Chart from the Referenced Scope in the HMI Dashboard

Since the function of this function block is irrelevant for the analysis itself, it belongs to the Visualization Only category.

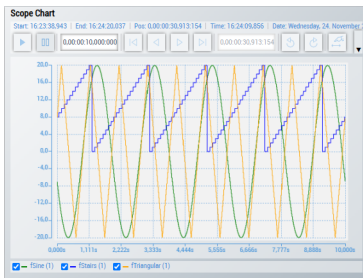
Configuration options

- **Chart:** the chart from the referenced scope to be displayed in the HMI. So far the following charts are supported: YT Chart, XY Chart, Array Bar Chart and Single Bar Chart.
- **Enable Record Controls:** if the checkbox is activated, a recording can be started and stopped from the HMI.
- **Record Time:** the recording time of a record. After reaching the time, the scope chart runs in a ring buffer.
- **Display Width:** the selected display width of the image. This property can also be modified in the HMI Scope Control.

Standard HMI Controls

For the algorithm *Referenced Scope* the following HMI Control is available for generating an Analytics Dashboard.

The HMI Scope Control for displaying a Scope Chart.



No other HMI controls can be added to the function block, as it is intended exclusively for displaying the HMI Scope Control.

5.2.1.12.6 XY Scope Chart



The generation of the HMI Scope Control for the HMI One Click Dashboard is only possible with an HMI version > 1.12.752.0.

XY Scope Chart			
Channel X 00	<Empty>	-	Num Channels
Channel Y 00	<Empty>	-	Enable Record Controls
		Record Time	Seconds 600
		Display Width	Seconds 10
		Channel Name 00	Channel 1

The *XY Scope Chart* function block is used to display the inputs in the HMI Dashboard. The inputs are displayed in an XY chart in the HMI Scope Control.

Since the function of this function block is irrelevant for the analysis itself, it belongs to the Visualization Only category.

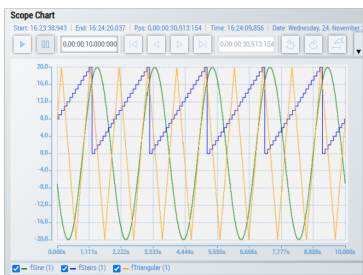
Configuration options

- **Num Channels:** the number of channels or input pairs to be displayed in the HMI Scope Control.
- **Channel Name 00/XX:** the display names of the channels. These appear as legends in the HMI Scope Control.
- **Enable Record Controls:** if the checkbox is activated, a recording can be started and stopped from the HMI.
- **Record Time:** the recording time of a record. After reaching the time, the scope chart runs in a ring buffer.
- **Display Width:** the selected display width of the image. This property can also be modified in the HMI Scope Control.

Standard HMI Controls

For the algorithm *XY Scope Chart* the following HMI Control is available for generating an Analytics Dashboard.

The HMI Scope Control for displaying a Scope Chart.

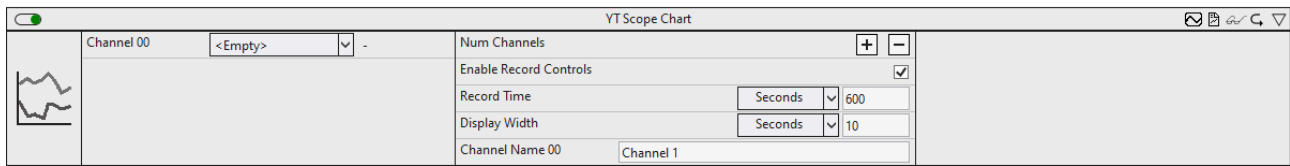


No other HMI controls can be added to the function block, as it is intended exclusively for displaying the HMI Scope Control.

5.2.1.12.7 YT Scope Chart



The generation of the HMI Scope Control for the HMI One Click Dashboard is only possible with an HMI version > 1.12.752.0.



The *YT Scope Chart* function block is used to display the inputs in the HMI Dashboard. The inputs are displayed in a YT chart in the HMI Scope Control.

Since the function of this function block is irrelevant for the analysis itself, it belongs to the Visualization Only category.

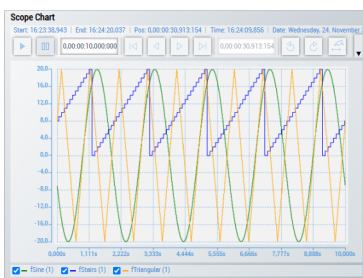
Configuration options

- **Num Channels:** the number of inputs to be displayed in the HMI Scope Control.
- **Channel Name 00/XX:** the display names of the inputs. These appear as legends in the HMI Scope Control.
- **Enable Record Controls:** if the checkbox is activated, a recording can be started and stopped from the HMI.
- **Record Time:** the recording time of a record. After reaching the time, the scope chart runs in a ring buffer.
- **Display Width:** the selected display width of the image. This property can also be modified in the HMI Scope Control.

Standard HMI Controls

For the algorithm *YT Scope Chart* the following HMI Control is available for generating an Analytics Dashboard.

The HMI Scope Control for displaying a Scope Chart.



No other HMI controls can be added to the function block, as it is intended exclusively for displaying the HMI Scope Control.

5.2.1.13 Analytics - XY Path Analysis

The algorithms of the category *XY Path Analysis* provide functions for the position evaluation of XY channels. For example, it is possible to analyze whether the position determined by the input channels is within certain bounds or shapes and how often boundary crossings occur.

5.2.1.13.1 XY Gate Monitor 2Ch

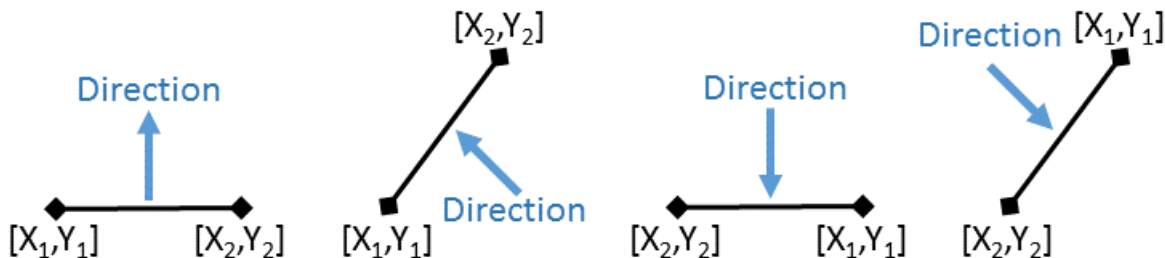
XY Gate Monitor 2Ch					
XY	Input ChX	<Empty>	-	Gate 1 X	1
	Input ChY	<Empty>	-	Gate 1 Y	1
				Gate 2 X	2
				Gate 2 Y	2
				Gate Intersect...	Empty
				Outlier Interse...	Empty
				Position Inter...	Empty
				Position Inter...	Empty
				Count Gate In...	Empty
				Count Outlier...	Empty
				Last Intersecti...	Empty

The *XY Gate Monitor 2Ch* counts the amount of intersections of an XY input with a specified gate or its projection (straight line between the gate points) depending on the configured Gate Mode. The analysis period can be started with the inputs *Start* and *Stop*. The algorithm is direction sensitive, which means that just intersection in the right direction are counted. The direction interpretation depends on the order of the gate points (X_1/Y_1) and (X_2/Y_2). The possible intersection directions are visualized below.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Directions of the intersection points:

The blue arrow represents the signal direction and the black lines illustrate the gate with its gate points (X_1/Y_1) and (X_2/Y_2). The direction of the intersection points is counted when the signal rotates counterclockwise around the first gate point (X_1/Y_1).



Configuration Options

- **Gate Mode:** Mode of the Gate Monitor:

Intersect Gate: Determines if the XY signal intersects the gate in the configured direction. If there is an intersection during the analysis period, it will be classified as *OK*, otherwise *NOK*.

Not Intersect Gate: Monitors if the XY signal does not intersect with the gate in the configured direction during the analysis period. Then it will be classified as *OK*, otherwise *NOK*.

Intersect Projection: Determines if the XY signal intersects the projection of the gate in the configured direction. If there is an intersection during the analysis period, it will be classified as *OK*, otherwise *NOK*.

Not Intersect Projection: Monitors if the XY signal does not intersect with the projection of the gate in the configured direction during the analysis period. Then it will be classified as *OK*, otherwise *NOK*.

Intersect Gate Or Projection: Determines if the XY signal intersects the gate or its projection in the configured direction. If there is an intersection during the analysis period, it will be classified as *OK*, otherwise *NOK*.

- **Gate 1 X:** X position of the first gate point.
- **Gate 1 Y:** Y position of the first gate point.
- **Gate 2 X:** X position of the second gate point.
- **Gate 2 Y:** Y position of the second gate point.

Output values

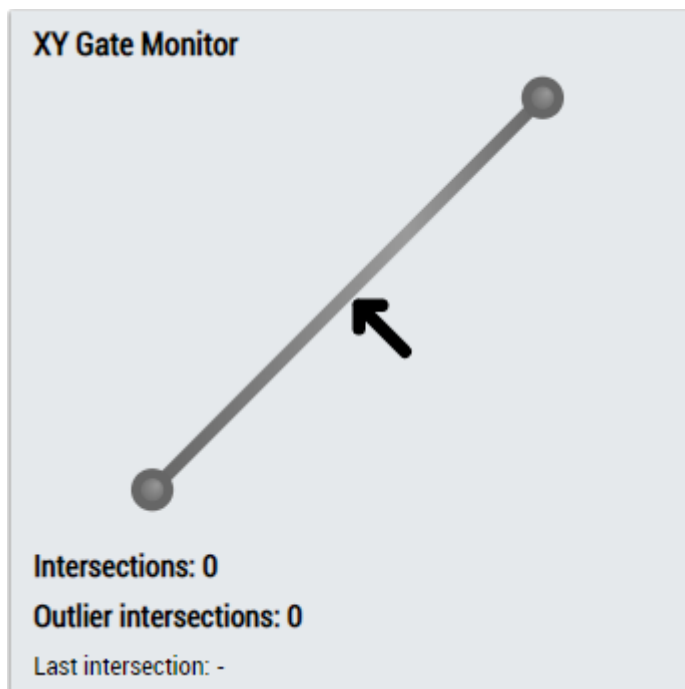
- **Gate Intersection:** Indicates whether there is currently a gate intersection.
- **Outlier Intersection:** Indicates whether there is currently an outlier intersection (gate projection intersection).
- **Position Intersection X:** X-position of the last intersection.

- **Position Intersection Y:** Y-position of the last intersection.
- **Count Gate Intersections:** Indicates the total number of gate intersections.
- **Count Outlier Intersections:** Indicates the total number of outlier intersections.
- **Time Intersection:** Indicates the time of the last intersection event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.
- **New Result:** Indicates whether a new result was calculated or not.
- **Executing:** Indicates whether the algorithm is active or inactive.
- **Classification:** Indicates the classification result. *OK* or *NOK*. The classification depends on the gate mode, as can be seen above.

Standard HMI Controls

For the XY Gate Monitor 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The XYGateMonitor control visualizes the output values Gate Intersection, Outlier Intersection, Count Gate Intersections, Count Outlier Intersections and Last Intersection as well as the direction of the intersection points.



2. The Table Control or Multivalue Control visualizes all output values: Executing, Gate Intersection, Outlier Intersection, Position Intersection X, Position Intersection Y, Count Gate Intersections, Count Outlier Intersections, Last Intersection, Classification.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021 16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the XY Gate Monitor 2Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.13.2 XY Shape Monitor Circle 2Ch

XY Shape Monitor Circle 2Ch						
XY	Input ChX	<Empty>	-	Centre X	0	Within Shape Empty
	Input ChY	<Empty>	-	Centre Y	0	Intersection Empty
				Radius	1	Count Interse... Empty
						Last Intersecti... Empty

The *XY Shape Monitor Circle 2Ch* count the amount of intersections of an XY input with a specified circle shape.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Centre X:** X position of the circle center.
- **Centre Y:** Y position of the circle center.
- **Radius:** Radius of the circle.

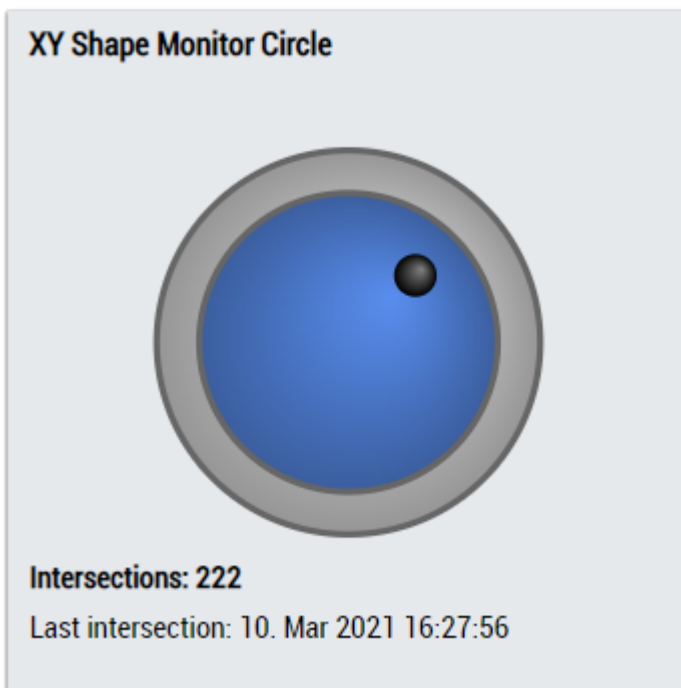
Output values

- **Within Shape:** Indicates whether the input signal is currently within the specified shape.
- **Intersection:** Indicates whether the input signal currently crosses the specified shape.
- **Count Intersections:** Counts the total number of intersections of input signal and shape.
- **Time Intersection:** Indicates the time of the last intersection event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the XY Shape Monitor Circle 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The XYShapeMonitor control visualizes the output values Within Shape, Intersection, Count Intersections and Last Intersection.



2. The SingleValue control visualizes the output values Intersection and Last Intersection.



3. The Table Control or Multivalue Control visualizes all output values: Within Shape, Intersection, Count Intersections, Last Intersection.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

20.42

Input 1

25.43

Input 2

TRUE

Executing

10 Mar 2021
16:15:30

Last Event

Alternatively, customer-specific HMI controls can be mapped in the XY Shape Monitor Circle 2Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.13.3 XY Shape Monitor Rectangle 2Ch

XY Shape Monitor Rectangle 2Ch						
XY 	Input ChX	<Empty>	-	Lower Left Corner X	0	Within Shape Empty
	Input ChY	<Empty>	-	Lower Left Corner Y	0	Intersection Empty
				Length X	2	Count Interse... Empty
				Length Y	2	Last Intersecti... Empty

The *XY Shape Monitor Rectangle 2Ch* count the amount of intersections of an XY input with a specified rectangle shape.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Lower Left Corner X:** X position of the lower left rectangle corner.
- **Lower Left Corner Y:** Y position of the lower left rectangle corner.
- **Length X:** Length of the rectangle in positive X direction.
- **Length Y:** Length of the rectangle in positive Y direction.

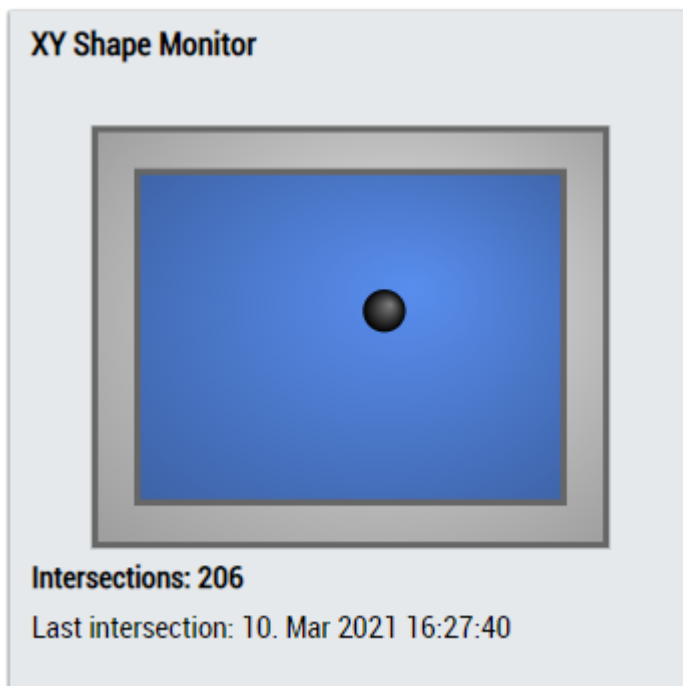
Output values

- **Within Shape:** Indicates whether the input signal is currently within the specified shape.
- **Intersection:** Indicates whether the input signal currently crosses the specified shape.
- **Count Intersections:** Counts the total number of intersections of input signal and shape.
- **Time Intersection:** Indicates the time of the last intersection event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the XY Shape Monitor Rectangle 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The XYShapeMonitor control visualizes the output values Within Shape, Intersection, Count Intersections and Last Intersection.



2. The SingleValue control visualizes the output values Intersection and Last Intersection.

Single Value

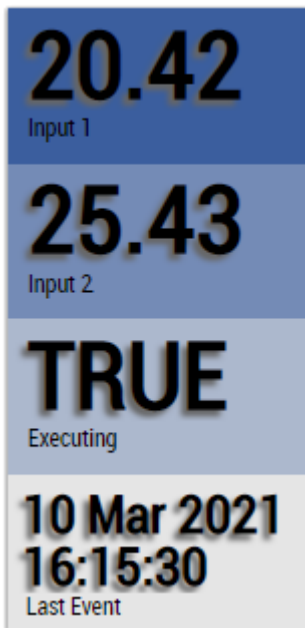
8

Last event:
10. Mar 2021 16:24:29

3. The Table Control or Multivalue Control visualizes all output values: Within Shape, Intersection, Count Intersections, Last Intersection.

Data Table Vertical	
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the XY Shape Monitor Rectangle 2Ch algorithm using the [Mapping Wizard \[► 382\]](#).

5.2.1.13.4 XY Shape Monitor Triangle 2Ch

XY Shape Monitor Triangle 2Ch							
XY	Input ChX	<Empty>	-	Corner 1 X	1	Within Shape	Empty
	Input ChY	<Empty>	-	Corner 1 Y	1	Intersection	Empty
				Corner 2 X	2	Count Interse...	Empty
				Corner 2 Y	2	Last Intersecti...	Empty
				Corner 3 X	3		
				Corner 3 Y	3		

The *XY Shape Monitor Triangle 2Ch* counts the amount of intersections of an XY input with a specified triangle shape.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration Options

- **Corner 1 X:** X position of the first triangle corner.
- **Corner 1 Y:** Y position of the first triangle corner.
- **Corner 2 X:** X position of the second triangle corner.
- **Corner 2 Y:** Y position of the second triangle corner.
- **Corner 3 X:** X position of the third triangle corner.
- **Corner 3 Y:** Y position of the third triangle corner.

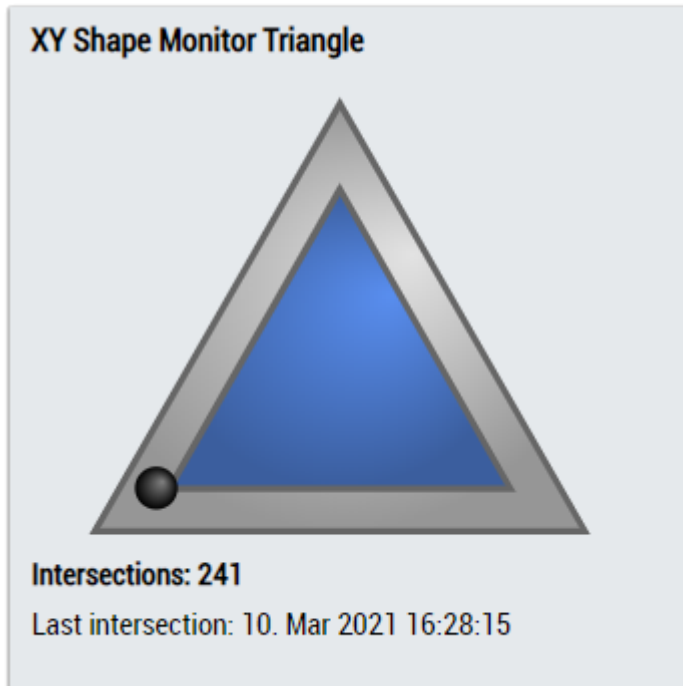
Output values

- **Within Shape:** Indicates whether the input signal is currently within the specified shape.
- **Intersection:** Indicates whether the input signal currently crosses the specified shape.
- **Count Intersections:** Counts the total number of intersections of input signal and shape.
- **Time Intersection:** Indicates the time of the last intersection event → the event can be dragged and dropped into the Scope chart to display it as a trigger-event.

Standard HMI Controls

For the XY Shape Monitor Triangle 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The XYShapeMonitor control visualizes the output values Within Shape, Intersection, Count Intersections and Last Intersection.



2. The SingleValue control visualizes the output values Intersection and Last Intersection.

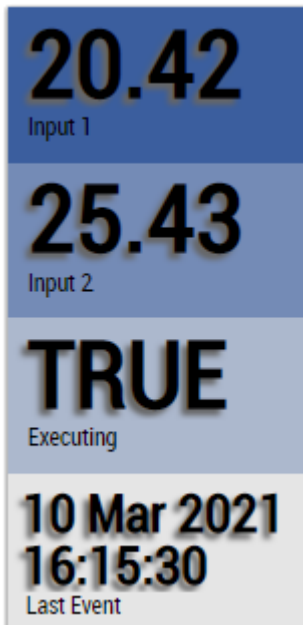


3. The Table Control or Multivalue Control visualizes all output values: Within Shape, Intersection, Count Intersections, Last Intersection.

Data Table Vertical

	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal				
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29



Alternatively, customer-specific HMI controls can be mapped in the XY Shape Monitor Triangle 2Ch algorithm using the [Mapping Wizard](#) [► 382].

5.2.1.14 Analytics - Communication

The algorithms in the *Analytics-Communication* category offer functions for logging channels. This means that analysis results can be communicated from the analysis.

5.2.1.14.1 MQTT logger

MQTT Logger			
Symbols 00	<Empty>	MQTT Target	127.0.0.1:1883
Symbols 01	<Empty>	Topic	My/Test/Topic
		Samples Per Buffer	100
		Compression Width	0
		Num Channels	2
		Use project hierarchy	☐
		New Result	Empty
		Lost Samples	Empty
		Sent Samples	Empty

The MQTT logger sends the data from the input channels in analytics binary format to an MQTT broker. The number of input channels (symbols) is configured via the Num Channels parameter. The data is collected and published in configurable packet sizes via the MQTT protocol to the specified destination.

Configuration options

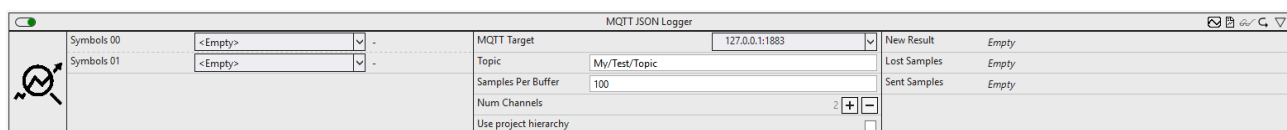
- **MQTT target:** Target address of the MQTT broker, consisting of host and port (e.g. 127.0.0.1:1883). The already configured MQTT targets are reused from the TwinCAT Target Browser TcAnalytics Extension and displayed in the combo box. If an MQTT target from the TwinCAT Target Browser defines a connection with an MQTT broker with certificates, the following CA, Cert and Key File parameters are also set automatically.
- **Topic:** MQTT topic under which the data is published. Two loggers of the same type must not write to the same topic, otherwise they will overwrite each other. It is then not possible to start the analysis and a message appears.

- **Samples Per Buffer:** Number of samples that are collected before a data packet is sent.
- **Compression Width:** Compression width of the data. If the value is 0, no compression takes place.
- **Num Channels:** Number of input channels (symbols) to be sent to the MQTT broker.
- **Use project hierarchy:** Specifies whether the project hierarchy should be used in the MQTT topics.
- **MQTT port** (optional): Port of the MQTT broker (e.g. 1883 for unencrypted, 8883 for TLS-encrypted connections).
- **Host** (optional): Host name or IP address of the MQTT broker.
- **User name** (optional): User name for authentication on the MQTT broker.
- **User password** (optional): Password for authentication on the MQTT broker.
- **Ca File** (optional): Path to the CA certificate file for TLS-encrypted connections.
- **Cert File** (optional): Path to the client certificate file for TLS-encrypted connections.
- **Key File** (optional): Path to the private key file for TLS-encrypted connections.
- **Key password** (optional): Password for the private key file.

Output values

- **New Result:** Shows TRUE if new data was successfully sent, otherwise FALSE.
- **Lost samples:** Number of samples that were discarded due to a buffer overflow.
- **Sent Samples:** Total number of successfully sent samples.
- **Current Timestamp** (optional): Specifies the timestamp of the last transmission.

5.2.1.14.2 MQTT JSON logger



The MQTT JSON Logger sends the data of the input channels in JSON format to an MQTT broker. The number of input channels (symbols) is configured via the Num Channels parameter. The data is collected and published in configurable packet sizes via the MQTT protocol to the specified destination.

Configuration options

- **MQTT target:** Target address of the MQTT broker, consisting of host and port (e.g. 127.0.0.1:1883). The already configured MQTT targets are reused from the TwinCAT Target Browser TcAnalytics Extension and displayed in the combo box. If an MQTT target from the TwinCAT Target Browser defines a connection with an MQTT broker with certificates, the following CA, Cert and Key File parameters are also set automatically.
- **Topic:** MQTT topic under which the data is published. Two loggers of the same type must not write to the same topic, otherwise they will overwrite each other. It is then not possible to start the analysis and a message appears.
- **Samples Per Buffer:** Number of samples that are collected before a data packet is sent.
- **Compression Width:** Compression width of the data. If the value is 0, no compression takes place.
- **Num Channels:** Number of input channels (symbols) to be sent to the MQTT broker.
- **Use project hierarchy:** Specifies whether the project hierarchy should be used in the MQTT topics.
- **MQTT port** (optional): Port of the MQTT broker (e.g. 1883 for unencrypted, 8883 for TLS-encrypted connections).
- **Host** (optional): Host name or IP address of the MQTT broker.
- **User name** (optional): User name for authentication on the MQTT broker.
- **User password** (optional): Password for authentication on the MQTT broker.
- **Ca File** (optional): Path to the CA certificate file for TLS-encrypted connections.
- **Cert File** (optional): Path to the client certificate file for TLS-encrypted connections.
- **Key File** (optional): Path to the private key file for TLS-encrypted connections.

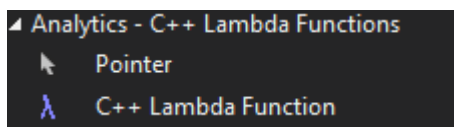
- **Key password** (optional): Password for the private key file.

Output values

- **New Result:** Shows TRUE if new data was successfully sent, otherwise FALSE.
- **Lost samples:** Number of samples that were discarded due to a buffer overflow.
- **Sent Samples:** Total number of successfully sent samples.
- **Current Timestamp** (optional): Specifies the timestamp of the last transmission.

5.2.2 C++ lambda functions

The C++ lambda functions offer the possibility to develop user-specific algorithms and integrate them into the TwinCAT Analytics Workflow. To guide through the development of the algorithm, a lambda template is available in the Visual Studio Toolbox.



The Lambda Template can be used to configure, program and publish the lambda function. After the successful development of the lambda function, the algorithm is located in the toolbox group "Analytics - C++ Lambda Functions". In the further course, all subsequent steps of the Analytics workflow are available. For example, the lambda function can be linked to the HMI controls. After deployment, the lambda functions are available in the TwinCAT Analytics Runtime and are displayed on the generated HMI One Click Dashboard after appropriate configuration.

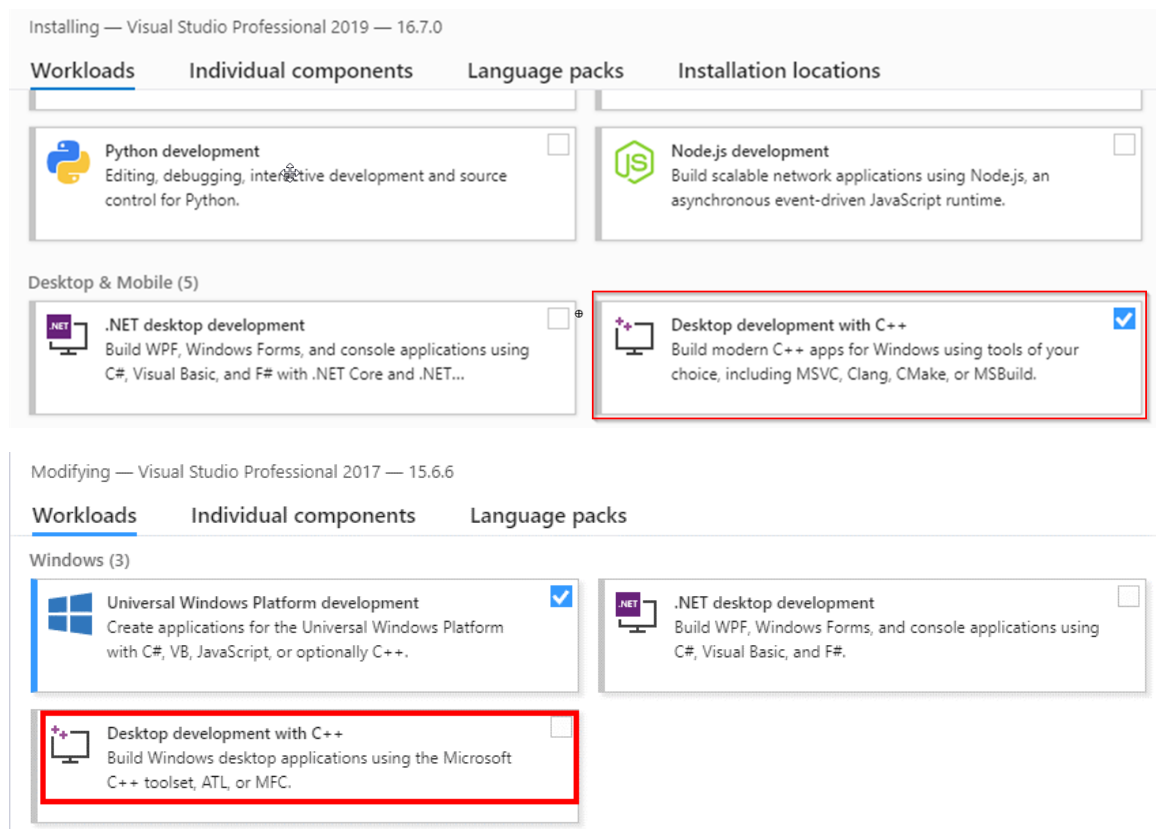
5.2.2.1 Requirements

Overview of minimum requirements

The following minimum requirements must be met for the implementation and debugging of TwinCAT 3 C++ modules.

The following must be installed on the engineering PC:

- Microsoft Visual Studio 2017 or 2019 Professional / Enterprise.
 - When installing Visual Studio, the **Desktop development with C++** option must be additionally selected, as this option is not selected with the automatic installation:



- TwinCAT 3 installation (XAE engineering)
- XAE Shell is sufficient for integrating and using existing binary C++ modules in a TwinCAT 3 PLC environment (Visual Studio is not required).

On the runtime PC:

- IPC or Embedded CX PC with one of the operating systems Windows 10 or TwinCAT/BSD. Both 32 and 64-bit versions are supported.
- Microsoft Visual Studio **does not** have to be installed.
- TwinCAT 3.1 installation (XAR runtime)

5.2.2.2 Preparation - only once

A PC for engineering lambda functions and thus TwinCAT C++ modules must be prepared. You only have to carry out these steps once per system:

- Configure the TwinCAT Basis as well as the configuration and platform toolbar.
- Create a test certificate to sign the modules
- Use the TcSignTool to store the certificate password outside the project
- Create an environment variable with the certificate name

When publishing the C++ Lambda function, an overview of the corresponding signature requirements is output in the log file. The log file can be searched for "Checking Signing Preconditions" and the corresponding error handling can be taken. A successful signing of the drivers looks like this:

```
== Checking Signing Preconditions
==> Succeeded: Certificate folder 'C:\TwinCAT\3.1\CustomConfig\Certificates' is valid.
==> Succeeded: Environment variable 'TWINCATCERTIFICATENAME' is set with value 'TwinCATLfCertNew'.
==> Succeeded: Certificate folder 'C:\TwinCAT\3.1\CustomConfig\Certificates' contains a certificate 'TwinCATLfCertNew.tccert'.
==> Succeeded: Environment variable 'TWINCATTESTCERTIFICATE' is set with value 'MyTestSigningCert'.
==> Succeeded: One or more certificate passwords are stored outside of the project with the TcSignTool.
```

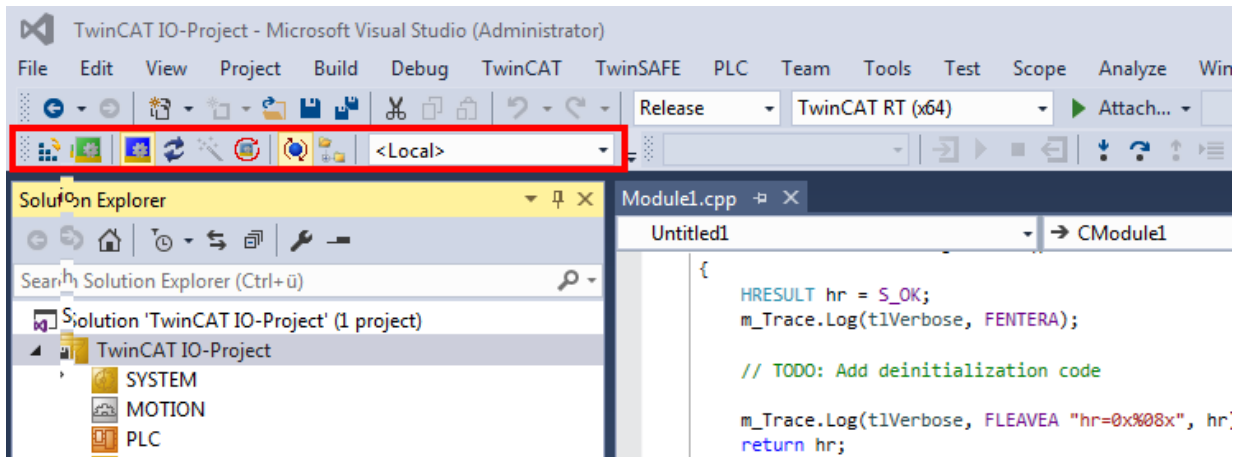
5.2.2.2.1 Visual Studio - TwinCAT XAE Base toolbar

Efficient engineering through TwinCAT XAE base toolbar

TwinCAT 3 integrates its own toolbar in the Visual Studio menu for better efficiency. It assists you in the creation of C++ projects. This toolbar is automatically added to the Visual Studio menu by the TwinCAT 3 setup. If you wish to add it manually, however, do the following:

1. Open the **View** menu and select **Toolbars\TwinCAT XAE Base**

⇒ The selected toolbar appears below the menu.



5.2.2.2.2 Module signing

For Windows and TwinCAT/BSD® runtime systems, TwinCAT C++ modules are signed with a certificate so that they can be executed. Beckhoff RT Linux® systems do not recognize file-based signatures, so this step can be omitted if Beckhoff RT Linux® alone is to be provided.

The signature ensures that only C++ software whose origin can be traced is executed on productive systems.

The C++ modules are loaded by the TwinCAT runtime system and must be signed with a TwinCAT user certificate.

For test purposes, certificates that cannot be verified can be used for signing. However, this is only possible if the operating system is in test mode so that these certificates are not used on productive systems.

i No signing for engineering and Beckhoff RT Linux® target systems.

Only execution on Windows and TwinCAT/BSD® requires certificates – the engineering and Beckhoff RT Linux® target systems do not.

Organizational separation of development and production software

Beckhoff recommends working organizationally with (at least) two certificates.

1. A certificate which is not countersigned, thus the test mode is needed for the development process. This certificate can also be issued individually by each developer. The test systems are then set to test mode.
2. Only the software that has passed the corresponding final tests is signed by a countersigned certificate. This software can thus also be installed on machines and delivered.

Such a separation of development and operation ensures that only tested software runs on productive systems.

5.2.2.2.2.1 TwinCAT

Versioned C++ projects are stored as binary in a TMX file (TwinCAT Module Executable).

For the implementation of TwinCAT 3 C++ modules, this compiled, executable TMX file must be signed with a TwinCAT user certificate if it is to be loaded by the TwinCAT Runtime.

Signing

TwinCAT TMX files can only be loaded after a successful signing.

NOTICE

Signing on 32-bit and 64-bit systems

In contrast to the operating system signing, TwinCAT signing is intended for both 32-bit and 64-bit systems. Thus, the test mode is also assumed for a test signing on 32-bit systems.

For signing a TMX file, a TwinCAT user certificate is required [► 243], which is configured accordingly in the project for signing.

Test signing

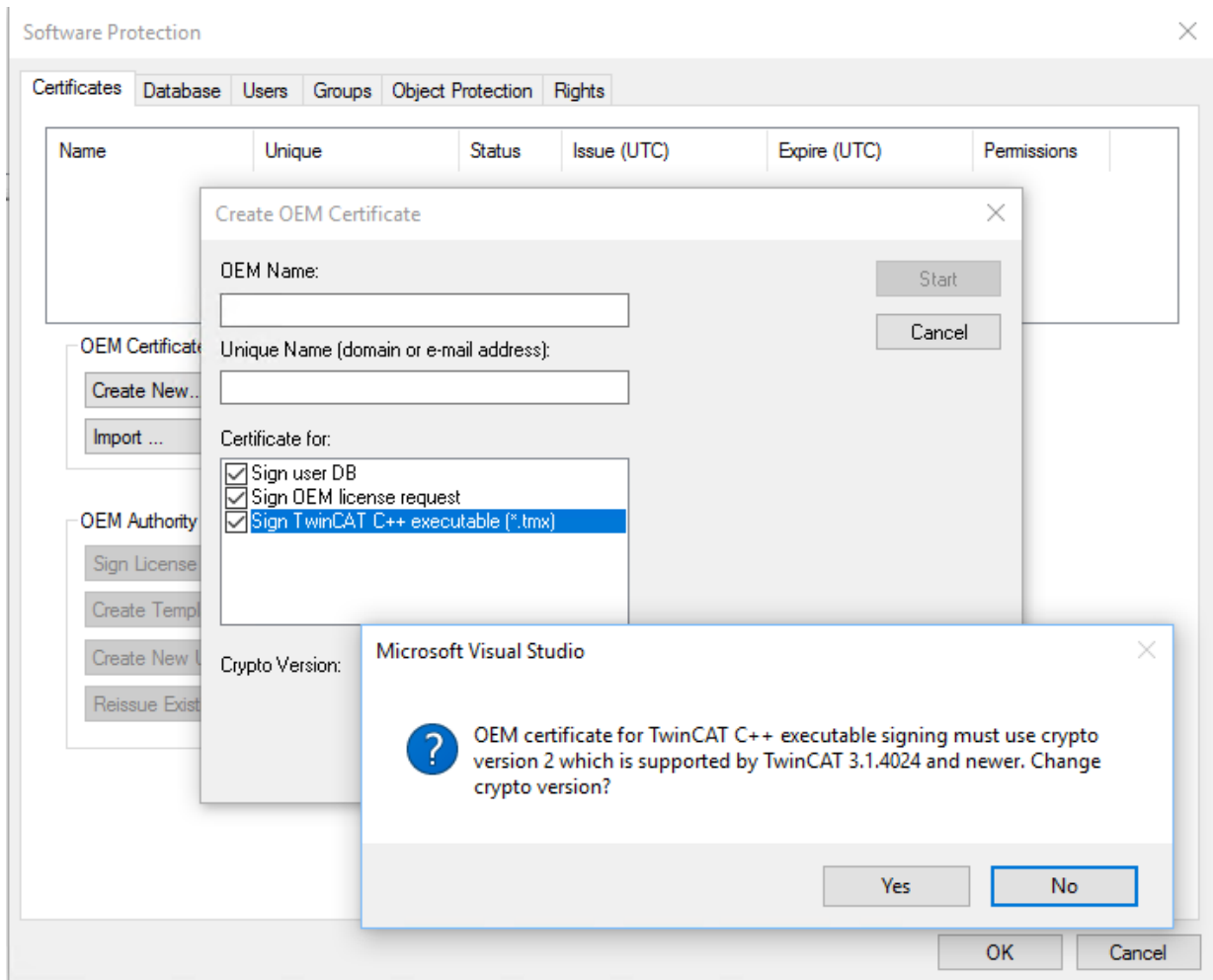
The user certificate can be created locally in TwinCAT. As long as it is not countersigned by Beckhoff, it is necessary to activate the test mode [► 243].

As soon as the TwinCAT user certificate has been countersigned by Beckhoff [► 245], the test mode can be dispensed with accordingly. It can be deactivated in the same way as it is activated.

5.2.2.2.1.1 Test signing

The test signing for TwinCAT can be carried out with the same TwinCAT user certificate as for the actual delivery (see Request TwinCAT 3 user certificate [► 246]).

1. For test operation, e.g., during software development, it is sufficient to create a TwinCAT user certificate [► 247]. Make sure that you select the purpose "Sign TwinCAT C++ executable (*.tmx)". For this the Crypto version 2 is required, a message appears.



On XAR (and XAE, if it is a local test), activate the test mode so that the operating system can accept the self-signed certificates. This can be done on both engineering systems (XAE) and runtime systems (XAR).

For Windows

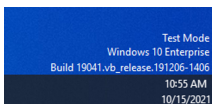
Use the administrator prompt to execute the following:

```
bcdedit /set testsigning yes
```

and reboot the target system.

You may have to switch off "SecureBoot" for this, which can be done in the bios.

If test signing mode is enabled, this is displayed at the bottom right of the desktop. The system now accepts all signed drivers for execution.



For TwinCAT/BSD®

In the file `/usr/local/etc/TwinCAT/3.1/TcRegistry.xml` enter `<Value Name="EnableTestSigning" Type="DW">1</Value>` under Key "System".

```
<Key Name="System">
  <Value Name="RunAsDevice" Type="DW">1</Value>
  <Value Name="RTimeMode" Type="DW">0</Value>
</Key>
```

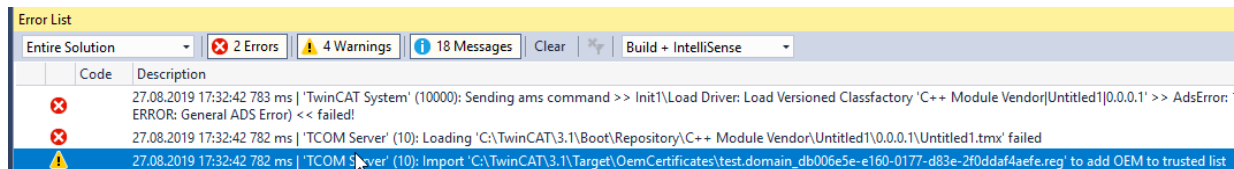
```
<Value Name="AmsNetId" Type="BIN">052445B00101</Value>
<Value Name="LockedMemSize" Type="DW">33554432</Value>
<Value Name="EnableTestSigning" Type="DW">1</Value>
</Key>
```

Then restart the TwinCAT system service:

```
doas service TcSystemService restart
```

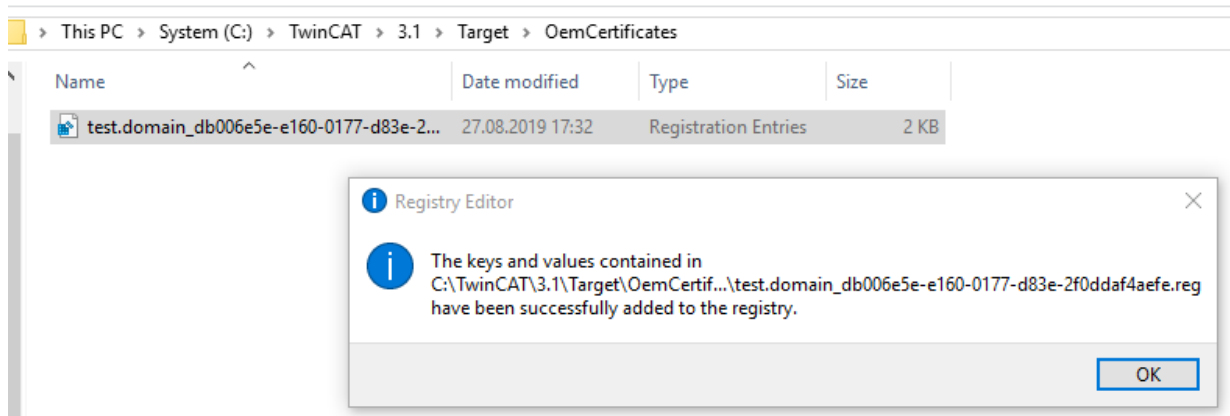
After the respective procedure, the system accepts all signed drivers for execution.

1. During the first activation (Activate Configuration) with a TwinCAT user certificate, the target system detects that the certificate is not trusted and the activation process is aborted:



For Windows:

A local user with administration rights can trust the certificate via the created REG file by simply executing it:



For TwinCAT/BSD®:

If the "Tcimportcert" package is not installed, install it: `pkg install TcImportCert`
Trust the certificate via `doas tcimportcert /usr/local/etc/TwinCAT/3.1/Target/OemCertificates/<CreatedFile>.reg`.

Then restart the TwinCAT System Service or reboot the system:

```
doas service TcSystemService restart
```

⇒ This process only enables C++ modules with a signature from the trusted TwinCAT user certificates to run.

2. Following this process you can use the TwinCAT user certificate for signing with the test mode of the operating system.

This is configured in the project properties.

Use the [TcSignTool](#) [► 256] to avoid storing the password of the TwinCAT user certificate in the project, where it would also end up in version management, for example.

If you want to use the TwinCAT user certificate without TestMode for delivery, you must have the [certificate countersigned by Beckhoff](#) [► 245].

5.2.2.2.2.1.2 User certificates for delivery without test mode

System requirements

- Min. TwinCAT 3.1 Build 4024
- Min. Windows 10 or TwinCAT/BSD® (on the target system)

With TwinCAT Build 4024, Beckhoff offers existing customers the issuing of a "TwinCAT 3 OEM user certificate", which can be used for signing TMX files created with TwinCAT 3 in C++.

- This certificate requires secure validation of the applicant data, since it is used in the Windows environment. TwinCAT 3 user certificates must therefore be officially ordered for validation of the address and contact data, and are only issued to existing Beckhoff customers.
- Order number: **TC0008**
- The issuing of this TwinCAT 3 user certificate is free of charge.
- Directory for saving the certificate: **C:\TwinCAT\3.1\CustomConfig\Certificates**

i The TwinCAT 3 user certificate is not required for using the TwinCAT 3 TMX files

The TwinCAT 3 user certificate is used exclusively for the one-time signing of the TMX files and is not required for the use of the TMX files signed with it.

i On which computers is the TwinCAT 3 user certificate TC0008 required?

The TwinCAT 3 user certificate should be located exclusively on the engineering computer on which the TMX files are signed - i.e. **NOT** on each target system.

Validity of the TwinCAT 3 user certificate

The validity of the TwinCAT 3 user certificate is limited to two years for security reasons.

i What happens if the certificate has expired?

You can no longer sign new TMX files.
However, the use of already signed TMX files is still possible without any restrictions.

You can apply for a renewal of your certificate before the expiry of the two years (and even after that).

To extend a TwinCAT 3 user certificate, the same process applies as for requesting a new certificate. In this case, the certificate must also be ordered (the order numbers for a certificate extension are the same as for a new certificate request).

In contrast to a new certificate, you do not generate a new OEM Certificate Request File but send your existing certificate to the Beckhoff certificate section for renewal. Please notify us in the email that this is a certificate extension and not a new issue. Otherwise, the same criteria apply regarding the content of the email as for the application for a new certificate.

The existing certificate receives a new expiration date, is then re-signed and is valid for another 2 years.

The newly signed certificate is thus fully compatible with the original version.

5.2.2.2.1.2.1 Request TwinCAT 3 user certificate

Overview of the ordering and validation process

An official order is required to request a TwinCAT user certificate.

- Order number: **TC0008** (TwinCAT 3 Certificate Extended Validation)
- The issuing (and renewal) of a TwinCAT 3 user certificate is free of charge.
- Since a TwinCAT 3 user certificate is a digital ID card, verification of the inquirer's contact data is required according to the usual market standards.
- A TwinCAT 3 user certificate is therefore only issued to existing Beckhoff customers.

Overview of the ordering and validation process

i Your email address must be a company email account (freemailers such as GMail or similar are not permitted) and correspond with the company name of the inquirer.

1. Contact your Beckhoff sales contact and announce the request of a TwinCAT 3 OEM certificate. Order "TC0007" or "TC0008".

2. Important: as the inquirer, please provide your contact details as the delivery address (= contact name and email address) and the area of use of the certificate (company name, address).
3. The contact details provided in the order will be verified and you (the inquirer named in the delivery address) will be contacted by Beckhoff Sales.
4. When requesting a new OEM certificate, [Creation of the Certificate Request file for TC0008 \[► 247\]](#).
5. Determine the "File Fingerprint" of the OEM certificate file using TwinCAT Engineering (see [Determining the file fingerprint of the OEM certificate file \[► 250\]](#)). Please inform the Beckhoff sales contact of this File Fingerprint as part of your contact data verification. The transmission of the File Fingerprint must be done by a different communication channel than the one used for sending the OEM certificate request file.
6. Now send the "OEM certificate file" to the Beckhoff sales contact.
7. After signing the certificate file at the Beckhoff headquarters, you will receive it by e-mail from your contact person.

Please note that it may take a few days to validate your contact details and issue the certificate.

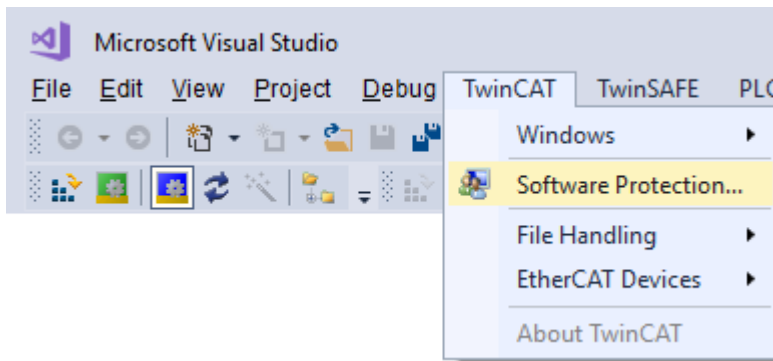
5.2.2.2.1.2.2 Creation of the Certificate Request file for TC0008



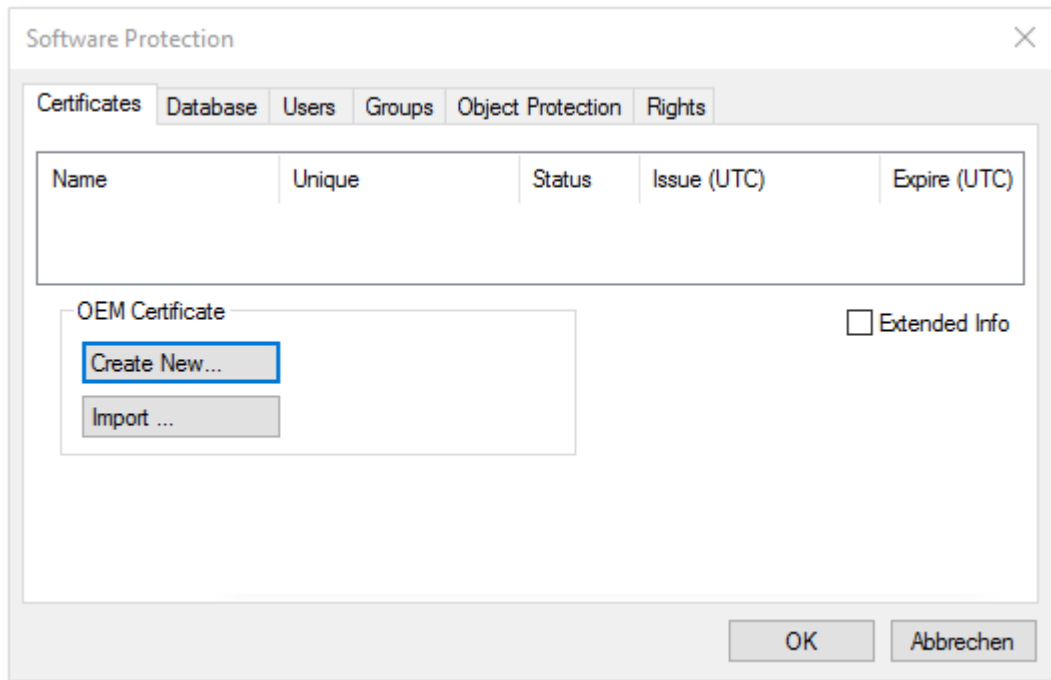
System requirements

- Min. TwinCAT 3.1 Build 4024
- Min. Windows 10 or TwinCAT/BSD® (on the target system)

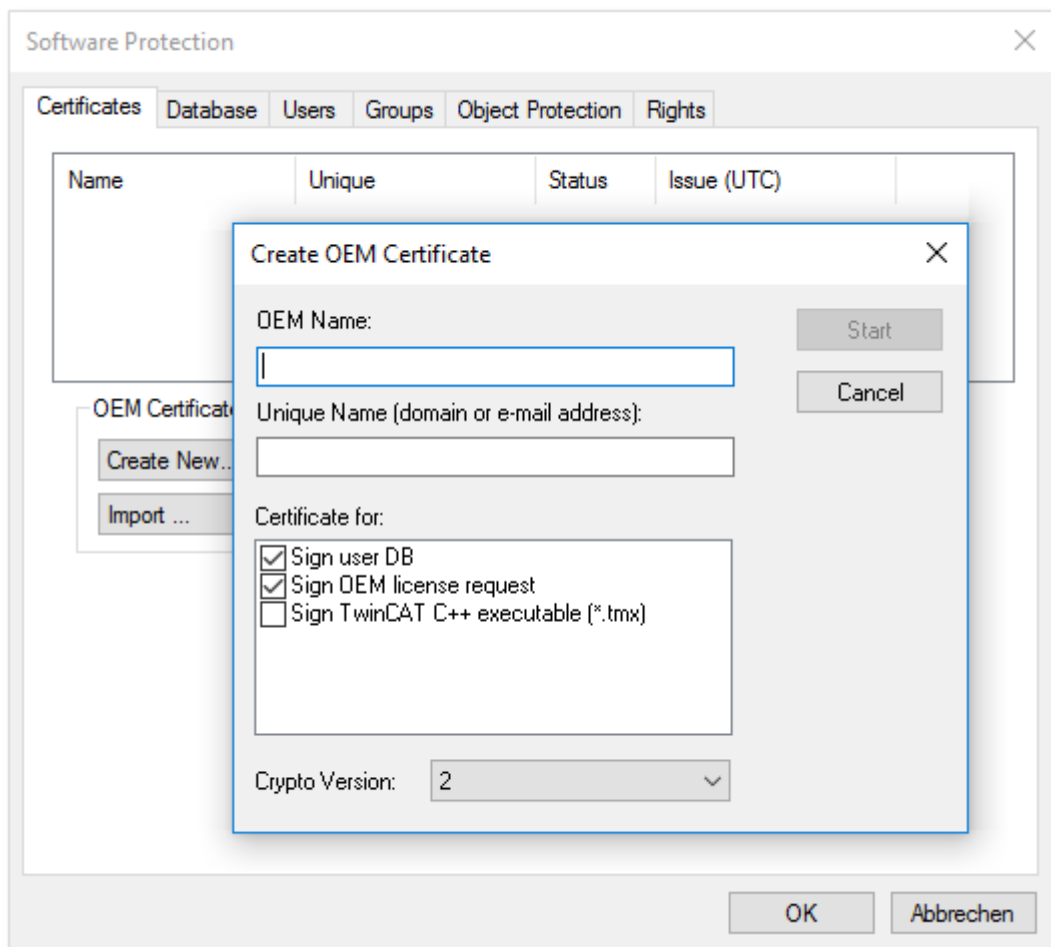
1. Call up the Software Protection configurator. To do this, select the menu item **Software Protection** in the main menu below the item **TwinCAT**:



- In the window that opens, select the **Certificates** tab.
Click **Create New....**:

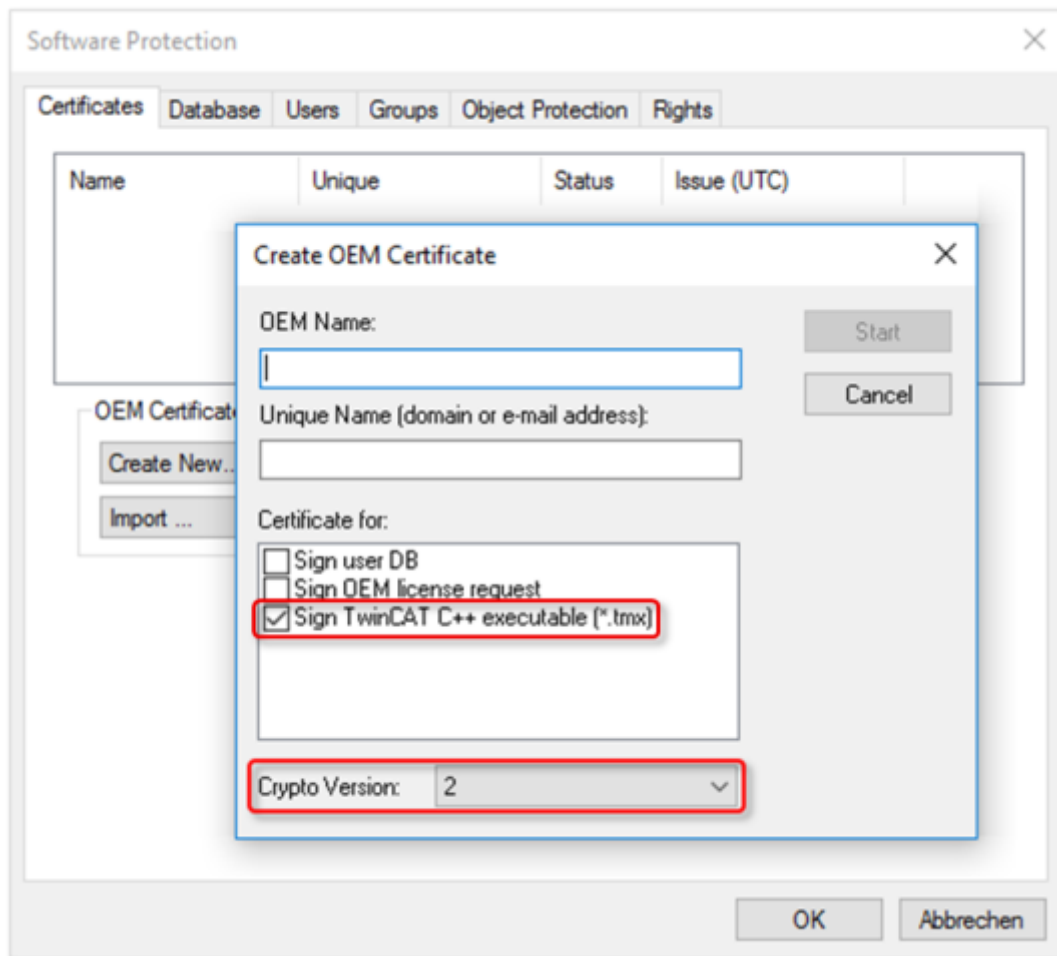


- The **Create OEM Certificate** input window opens:



- Enter the required data:
 - Enter your company name in the **OEM Name** text field. The name must have a clear reference to your company or your business unit.

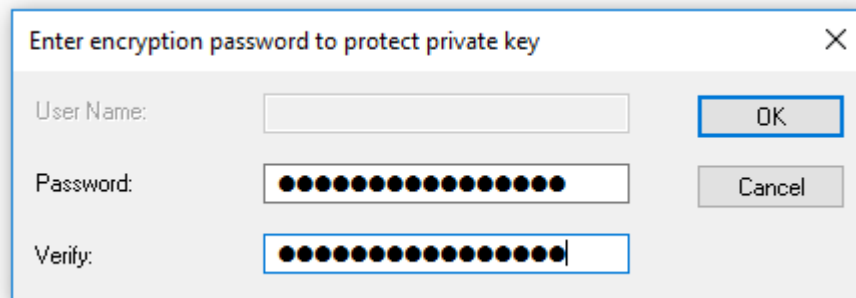
- Enter a **Unique Name**. The "OEM Unique Name" must be a unique name that uniquely identifies the owner of the certificate worldwide, preferably the URL of your company's website or your email address. The email address must be a company email address, i.e. it must be possible to assign it unambiguously to your company.
- Check the checkbox **Sign TwinCAT C++ executables**:



If you only want to sign TwinCAT driver software with this certificate, uncheck the other two checkboxes. (These are only used in the PLC area)

- Make sure that Crypto version 2 (for the encrypted content of the certificate content) is set. (default setting)
- Once you have entered the data, click **Start** and select a directory to save the file.
You can simply accept the suggested directory "c:\twincat\3.1\customconfig\certificates". You need the newly created file in this directory in order to be able to read out the "File Fingerprint" for this file [► 250] in a subsequent step.
 - ⇒ A dialog for selecting a password for the OEM Private Key opens.
 - Issue a password for the OEM Private Key.
Password security - Be sure to use a strong password for your certificate! Take suitable measures to protect your password so that it cannot fall into the wrong hands!
Password cannot be recovered if lost - Beckhoff cannot recover or reset your password. If you forget or lose the password for your certificate, you can no longer use it and have to request a new certificate.

7. Confirm the password by entering it again and close the dialog with **OK**.



⇒ The file is saved.

The "Certificate Request File" generated in this way must now be signed by the Beckhoff certificate section in order to be valid. The procedure is described in chapter [Requesting a certificate](#) [► 246].

5.2.2.2.1.2.3 Determining the file fingerprint of the OEM certificate file

You need this functionality to request a **TwinCAT OEM Certificate Extended Validation (TC0008)**.



System requirements

This functionality requires TwinCAT 3.1 Build 4024 of higher.



The OEM Certificate Request File becomes the TwinCAT OEM certificate once it is signed by Beckhoff. The files do not differ except for this signature. For this reason, the term "TwinCAT OEM certificate file" is used for both file versions in the following sections.

Reading the "file fingerprint" of an OEM certificate file via TwinCAT 3 Engineering

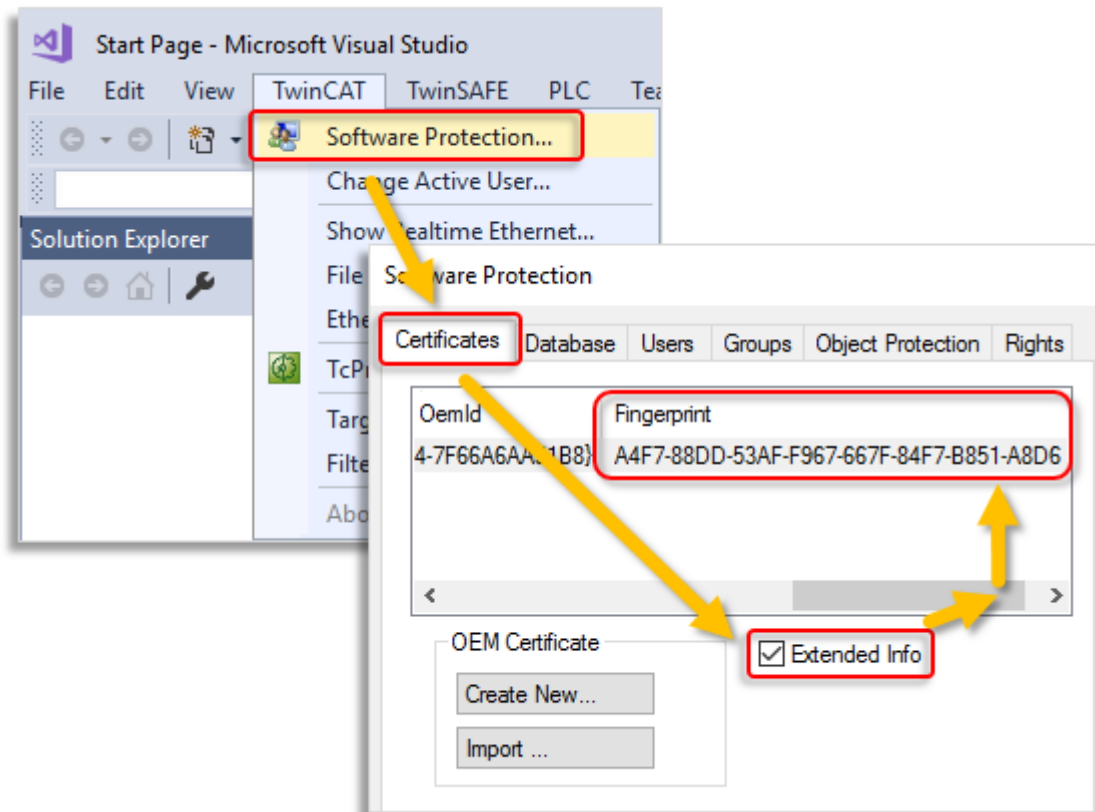
For this function it is necessary that the OEM certificate file is located in this directory: "c:\twincat\3.1\customconfig\certificates".

This directory contains your OEM certificate, if you already have a certificate and want to renew it.

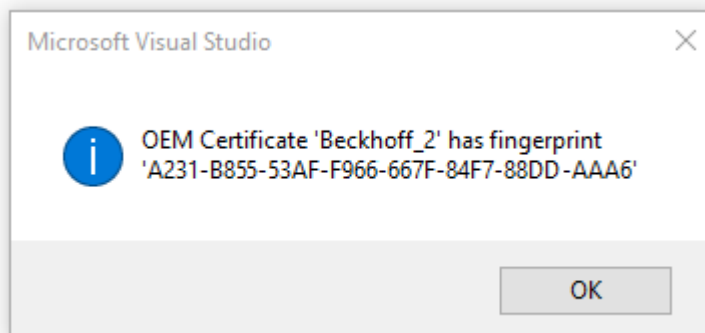
If you did not change the suggested directory when creating the "OEM Certificate Request File", the file is already in this directory.

Procedure:

1. Call up the TwinCAT 3 Software Protection configurator.



2. Select the **Certificates** tab.
3. Check the **Extended Info** checkbox.
4. In the window scroll to the right until you see the **Fingerprint** column. (As an alternative, you can simply double-click the certificate line. The Fingerprint file is then displayed in a pop-up window:



The shortcut [Ctrl] + [C] can be used to copy the fingerprint data from the message window to the Windows clipboard.

5.2.2.2.2.1.2.4 Saving the signed TwinCAT user certificate

Recommended directory for saving the certificate: **C:\TwinCAT\3.1\CustomConfig\Certificates**



System requirements

- Min. TwinCAT 3.1 Build 4024
- Min. Windows 10 or TwinCAT/BSD® (on the target system)

i The TwinCAT 3 user certificate is not required for using the TwinCAT 3 TMX files

The TwinCAT 3 user certificate is used exclusively for the one-time signing of the TMX files and is not required for the use of the TMX files signed with it.

i On which computers is the TwinCAT 3 user certificate TC0008 required?

The TwinCAT 3 user certificate should be located exclusively on the engineering computer on which the TMX files are signed - i.e. **NOT** on each target system.

5.2.2.2.2 Operating system

NOTICE

Migration to TMX with TwinCAT Loader recommended

Since TwinCAT 3.1 4024.0 versioned C++ projects are available, whose binaries can be loaded directly from TwinCAT. Migration is recommended!

For the implementation of TwinCAT 3 C++ modules on x64 platforms, the driver (*.sys file) must be signed with a certificate if it is to be loaded by the operating system.

The signature, which is automatically executed during the TwinCAT 3 build process, is used by 64-bit Windows operating systems for the authentication of the drivers.

A certificate is required to sign a driver. [This Microsoft documentation](#) describes the process and background knowledge for obtaining a test and release certificate that is accepted by 64-bit Windows operating systems.

To use such a certificate in TwinCAT 3, configure the step after compiling your x64 build target as documented in "[Creating a test certificate for test mode \[► 252\]](#)".

Test certificates

For testing purposes, self-signed test certificates can be created and used without technical limitations.

The following tutorials describe how to enable this option.

To create drivers with real certificates for production machines, this option must be disabled.

- [Creating a test certificate for test mode \[► 252\]](#)
- [Delete \(test\) certificates \[► 254\]](#)

Further references:

MSDN, MakeCert test certificates (Windows drivers),

<https://docs.microsoft.com/en-us/windows-hardware/drivers/install/makecert-test-certificate>

5.2.2.2.2.1 Test signing

Overview

Implementing TwinCAT 3 C++ modules for x64 platforms requires signing the driver with a certificate.

This article describes how to create and install a test certificate for testing a C++ driver.

i Note the procedure when creating test certificates

Developers may have a wide range of tools for creating certificates themselves. Please follow this description exactly to activate the test certificate mechanism.

The following commands must be executed from a command line that has been opened in either way:

- **Command prompt from Microsoft Visual Studio 2010 / 2012 with administrator rights.** (Via: **All Programs** -> **Microsoft Visual Studio 2010/2012** -> **Visual Studio Tools** -> Visual Studio Command Prompt, then right-click **Run as administrator**)
- **Developer Command Prompt from Microsoft Visual Studio 2017 / 2019 with administrator rights..** (Via: **All Programs** -> **Visual Studio 2017** -> Visual Studio Command Prompt for VS 2017/2019, then right-click on **Run as administrator**)
- Only if the WINDDK has been installed:
Normal prompt (**Start** -> Command Prompt) with administrator rights, then change to directory %WINDDK7%\bin\x86\, which contains the corresponding tools.

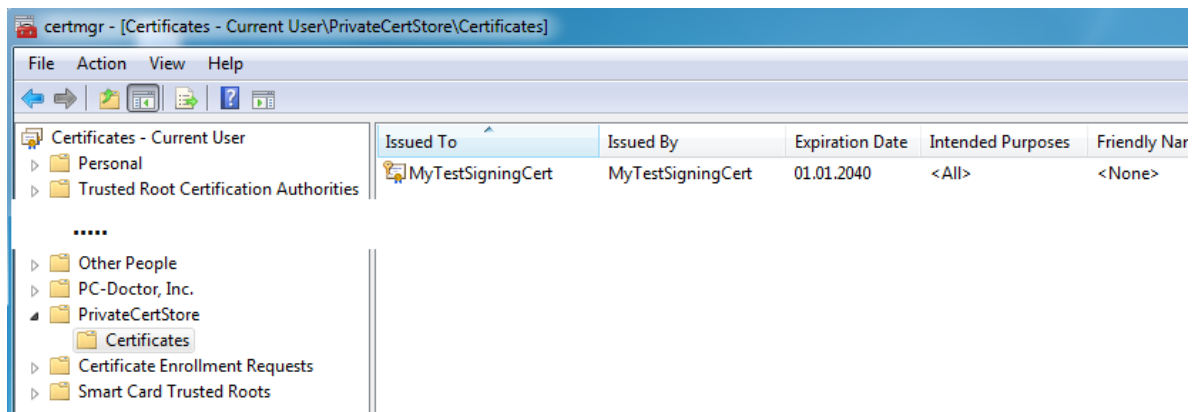
1. On XAE:

In the engineering system, enter the following command in the Microsoft Visual Studio 2010 / 2012 command prompt with administrator rights (see note above):

```
makecert -r -pe -ss PrivateCertStore -n CN=MyTestSigningCert  
MyTestSigningCert.cer
```

(If you do not have access rights to the PrivateCertStore, you can use a different storage location. This must also be used in the PostBuild event [▶ 256].)

- ⇒ This is followed by creation of a self-signed certificate, which is stored in the file "MyTestSigningCert.cer" and in the Windows Certificate Store.
- ⇒ Check the result with mmc (**Use File->Add/Remove Snap-in->Certificates**):



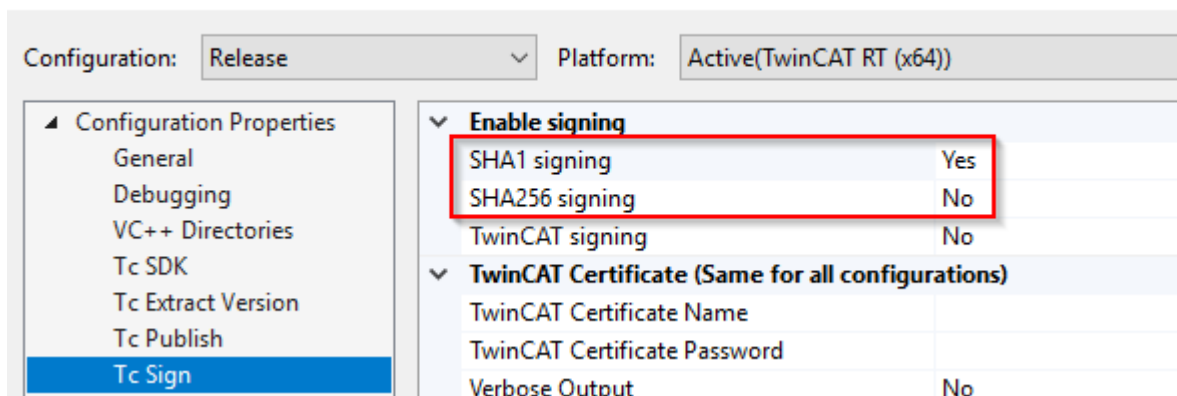
2. On XAE:

configure the certificate so that it is recognized by TwinCAT XAE on the engineering system. Set the environment variable TWINCATTESTCERTIFICATE to "MyTestSigningCert" in the engineering system or edit the post build event of Debug|TwinCAT RT (x64) and Release|TwinCAT RT (x64). The name of the variable is NOT the name of the certificate file, but the CN name (in this case MyTestSigningCert).



From TwinCAT 3.1 4024.0, the configuration of the certificate to be used is carried out under Tc Sign in the project properties. To use signing via the operating system, as described here, please pay attention to the project settings.

Untitled1 Property Pages



On XAR (and XAE, if it is a local test), activate the test mode so that the operating system can accept the self-signed certificates. This can be done on both engineering systems (XAE) and runtime systems (XAR).

For Windows

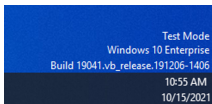
Use the administrator prompt to execute the following:

```
bcdedit /set testsigning yes
```

and reboot the target system.

You may have to switch off "SecureBoot" for this, which can be done in the bios.

If test signing mode is enabled, this is displayed at the bottom right of the desktop. The system now accepts all signed drivers for execution.



After the respective procedure, the system accepts all signed drivers for execution.

1. Test whether a configuration with a TwinCAT module implemented in a TwinCAT C++ driver can be enabled and started on the target system.

⇒ Compilation of the x64 driver generates the following output:

```

Output
Show output from: Build
1>----- Build started: Project: Untitled2, Configuration: Debug TwinCAT RT (x64) -----
1> header file << C:\TwinCAT\3.1\SDK\products\TwinCAT RT (x64)\Debug\Untitled2\Untitled2Version.h >> is up-to-date!
1> TcPch.cpp
1> Module1.cpp
1> Untitled2ClassFactory.cpp
1> Untitled2Driver.cpp
1> Untitled2.vcxproj -> C:\TwinCAT\3.1\SDK\products\TwinCAT RT (x64)\Debug\Untitled2.sys
1> The following certificate was selected:
1>   Issued to: MyTestSigningCert
1>   Issued by: MyTestSigningCert
1>   Expires:   Sun Jan 01 00:59:59 2040
1>   SHA1 hash: E27A66E6A0C7BC0C86DFDD093DDF2486D1EE502E
1> Done Adding Additional Store
1> Successfully signed and timestamped: C:\TwinCAT\3.1\SDK\products\TwinCAT RT (x64)\Debug\Untitled2.sys
1> Number of files successfully Signed: 1
1> Number of warnings: 0
1> Number of errors: 0
1> ===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
  
```

References:

MSDN, MakeCert test certificates (Windows drivers),

<https://docs.microsoft.com/en-us/windows-hardware/drivers/install/makecert-test-certificate>

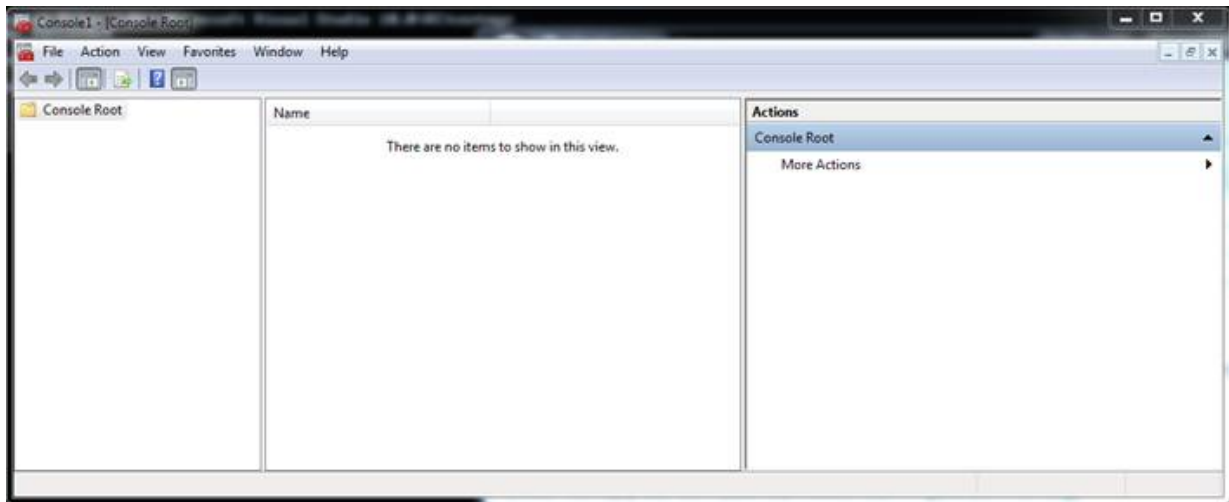
5.2.2.2.2.2 Delete test certificate

This article is about how to delete a test certificate.

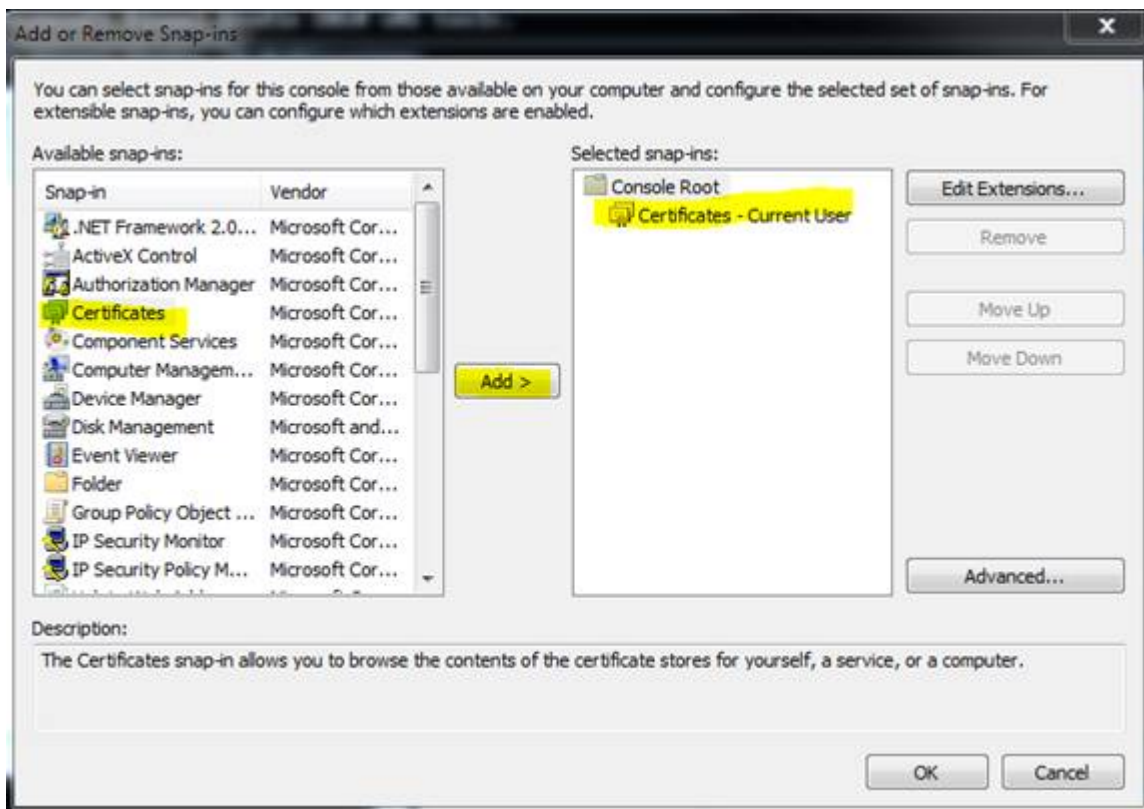
Overview

A certificate can be deleted with the Microsoft Management Console:

1. Start the management console MMC.exe via the Start menu or the user interface.

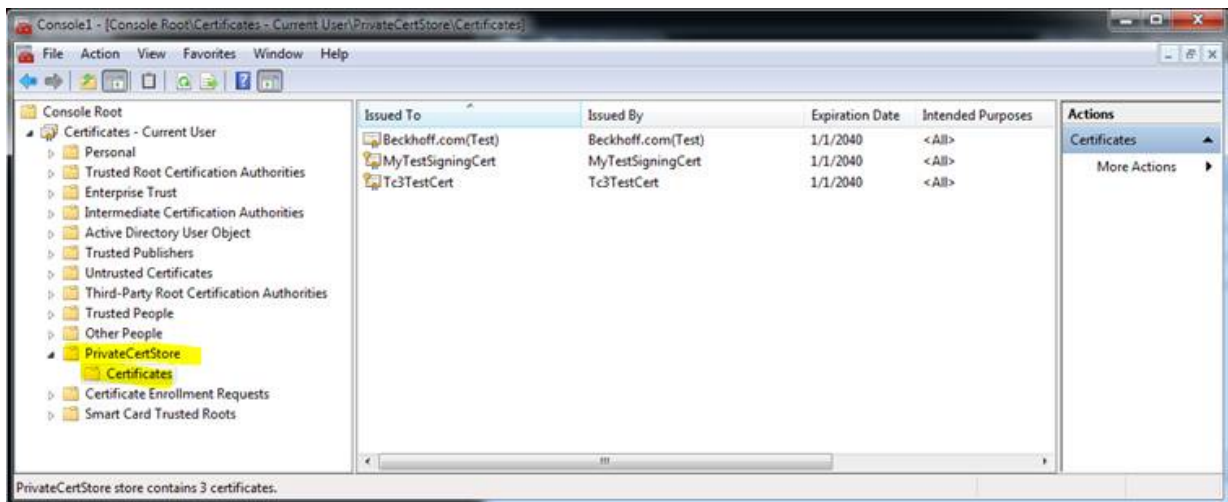


2. Click in the menu on **File -> Add/Remove Snap-in..** and select the certificate snap-in for the current user; conclude with **OK**.



⇒ The certificates are listed in the node under **PrivateCertStore/Certificates**.

3. Select the certificate to be deleted.



5.2.2.2.2.3 Windows driver without test mode

For Windows operating systems the driver has to be signed via "Attestation Signing". This requires an EV certificate.

Microsoft provides relevant instructions: <https://docs.microsoft.com/en-us/windows-hardware/drivers/dashboard/attestation-signing-a-kernel-driver-for-public-release>
 The drivers created in this way are also suitable for devices that have secure boot enabled.

Microsoft announced that the previous procedure using CrossSigning certificates (signing tool with parameter /ac) will be discontinued from July 2021. It can no longer be used (depending on the expiry date of the individual CrossSigning certificate).

Versioned C++ projects, which are loaded via the TwinCAT Loader, have been available for TwinCAT C++ for some time, so that they are not drivers in the sense of the operating system. Beckhoff therefore recommends using versioned C++ projects.

A guide is available in the [How-to section](#) for the migration of TwinCAT C++ drivers to versioned C++ projects.

5.2.2.2.3 TcSignTool - Storage of the certificate password outside the project

The TcSignTool can be used to store a password for a TwinCAT user certificate in the registry. Thus, the password is not needed in the projects, where the passwords would end up unintentionally in version control systems.

The TcSignTool is a command line program located in the path `C:\TwinCAT\3.1\sdk\Bin\` or `C:\Program Files (x86)\Beckhoff\TwinCAT\3.1\SDK\Bin`.

The storage of the password is carried out with the following parameters:

```
tcsigntool grant /f "...CustomConfig\Certificates\MyCertificate.tccert" /p MyPassword
```

The password is deleted with the following parameters:

```
tcsigntool grant /f "...CustomConfig\Certificates\MyCertificate.tccert" /r
```

The unencrypted password is stored under: `HKEY_CURRENT_USER\SOFTWARE\Beckhoff\TcSignTool\`

5.2.2.2.4 Environment variables

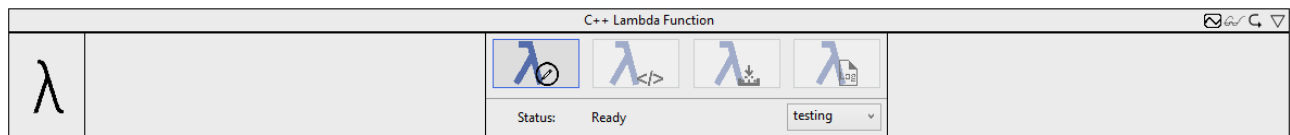
Finally, two environment variables must be created with the certificate name. These are read during the generation of the lambda function and sign the lambda function with the corresponding certificates.

Variable name	Variable value
TWINCATTESTCERTIFICATE	<Operating System Certificate Name> (e.g. MyTestSigningCert)
TWINCATCERTIFICATENAME	<TwinCAT certificate name> (e.g. TwinCATTestCert)



The file extensions (.cert and .tccert) must not be included.

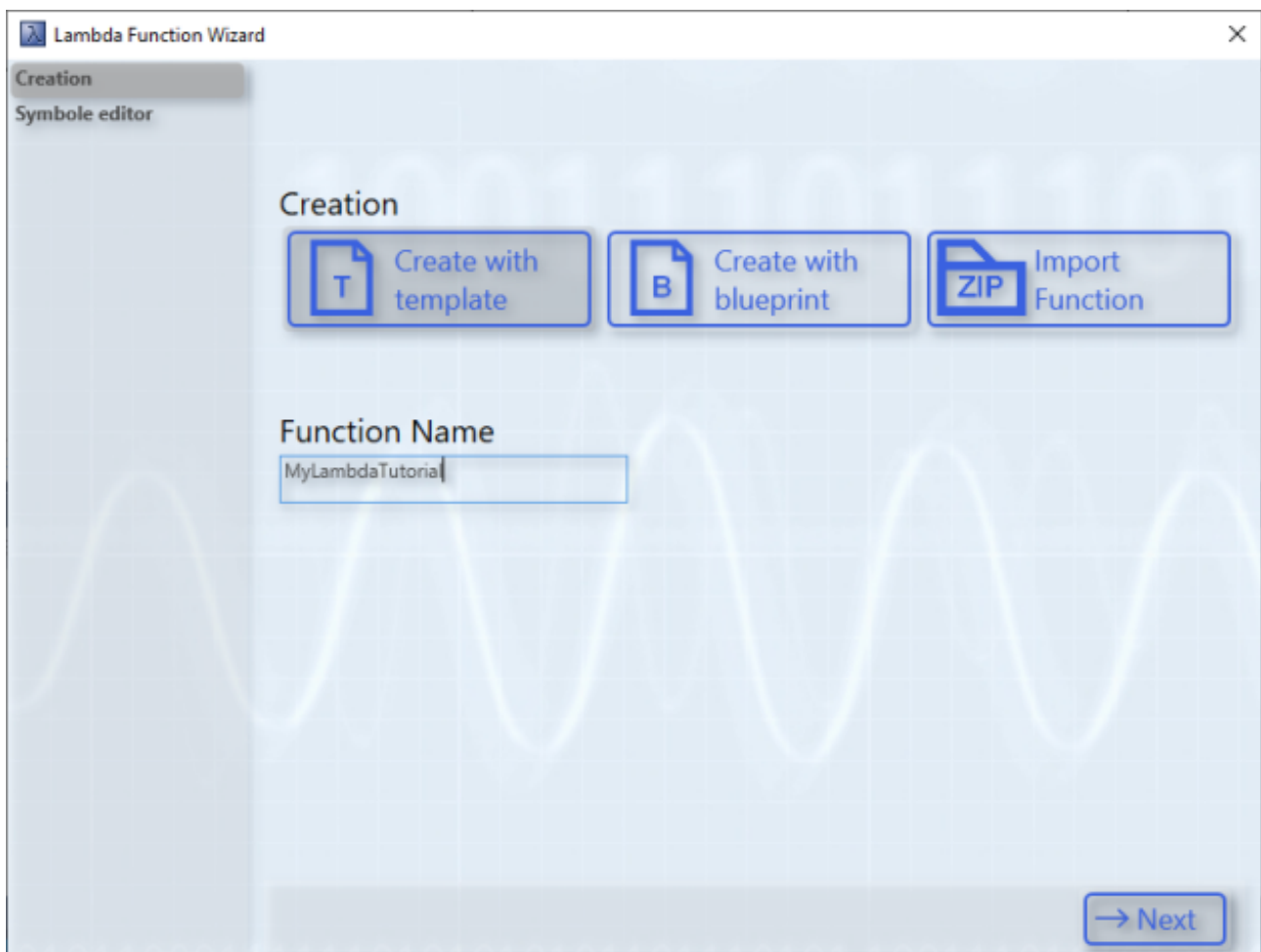
5.2.2.3 Lambda Template



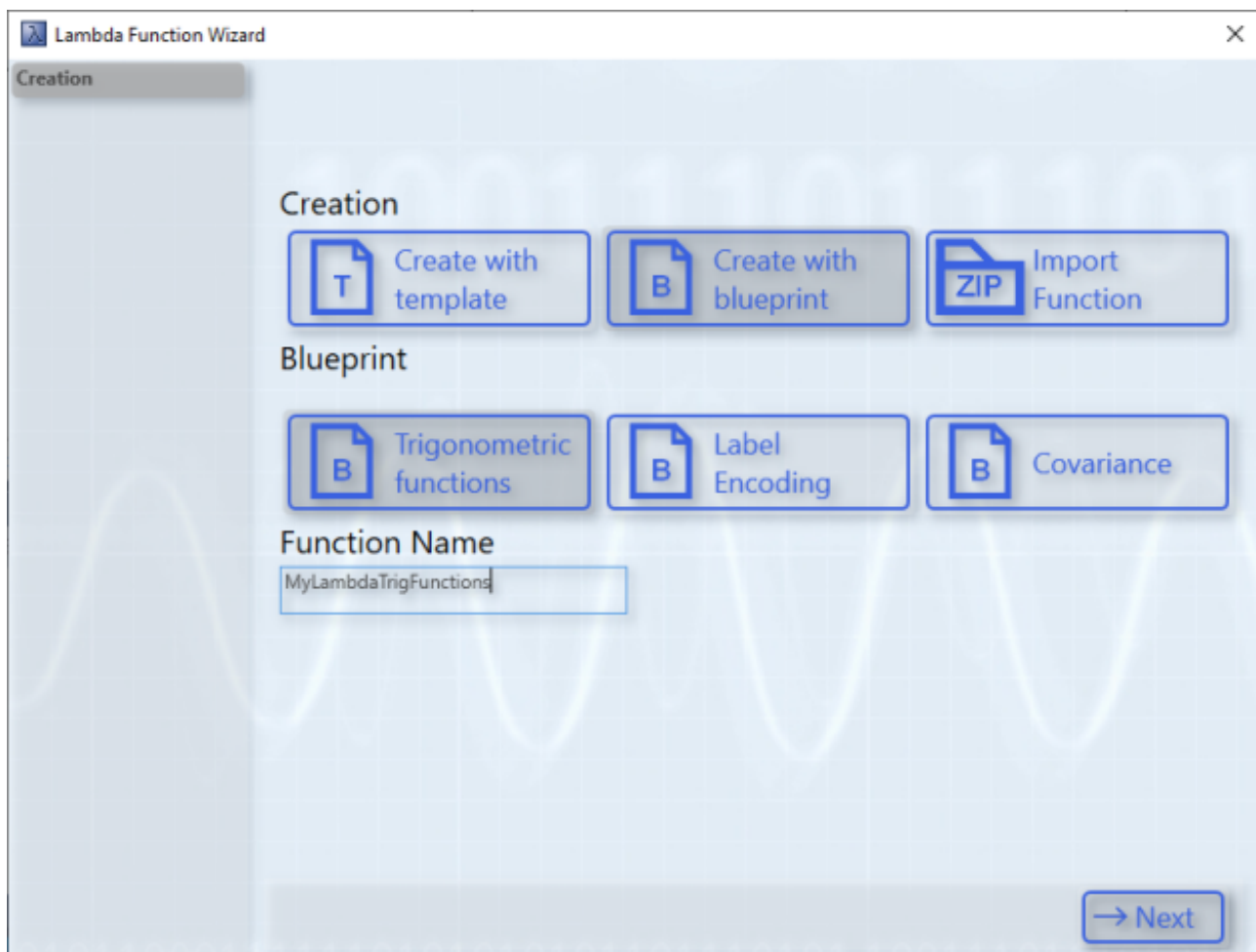
The lambda template guides through the development of a lambda function. Within the parameter area there is a control. This control contains four buttons for configuring, programming and publishing the lambda function. The fourth button is for viewing the log files of the development process. A status bar provides additional information during the process. A drop-down menu at the bottom right offers the possibility to choose between a release and a testing version of the lambda function.

Configuring the lambda function

The first button from the left opens a wizard. With the help of this wizard you can configure the lambda function.



On the first page you can choose between three modes. The first mode is called Template. With this mode new lambda functions can be developed. All files and dependencies are prepared in a template and can be implemented with user-specific logic.



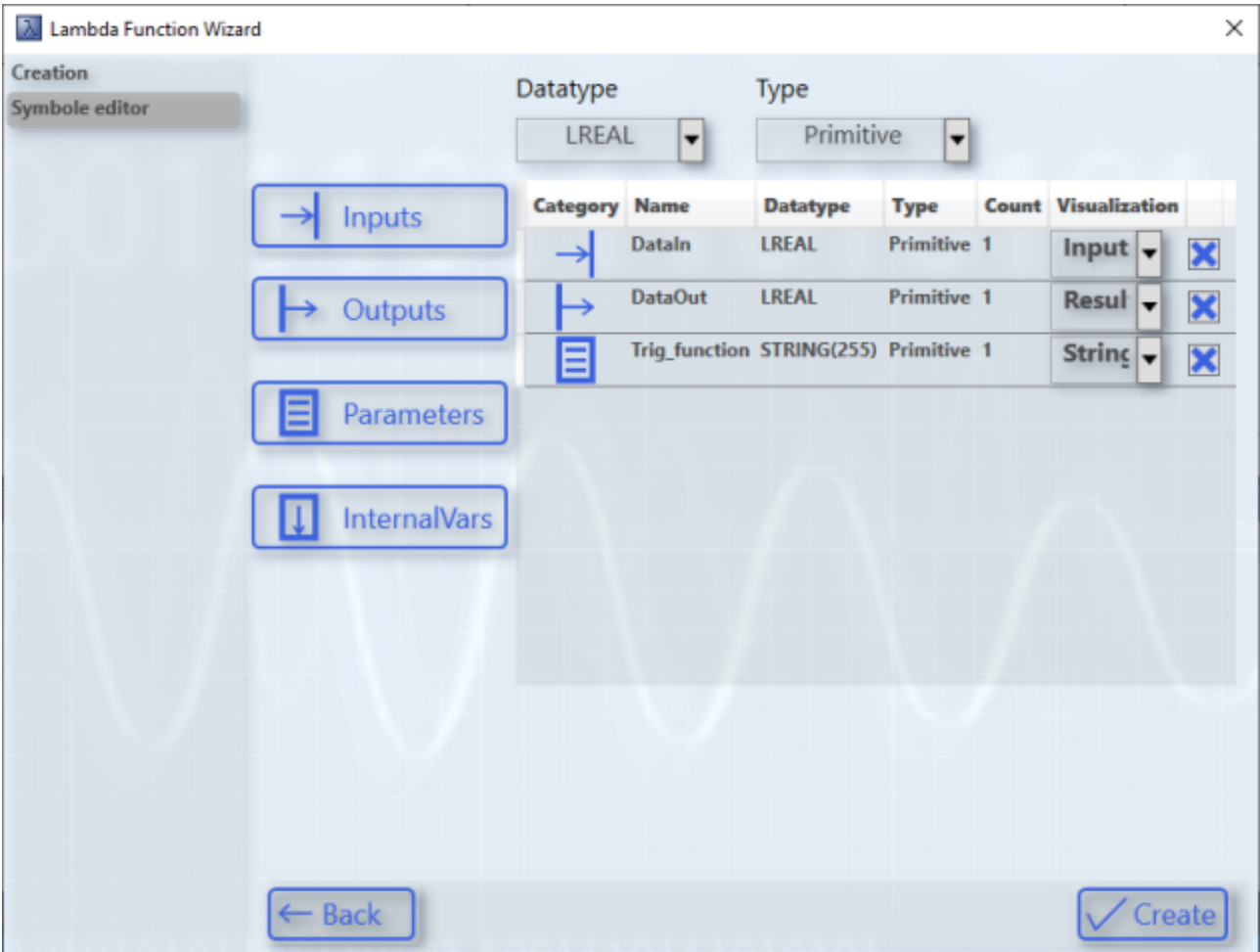
The second mode is called Blueprint. There are three blueprints already implemented in each case. There is a blueprint for calculating trigonometric functions, for label encoding and for calculating covariance. These blueprints serve as an example configuration and implementation of a lambda function. More detailed information is available at the following link (see: [Lambda Blueprints \[► 265\]](#)).

Via the third mode you import exported lambda functions. With a click on the button **Import Function**, an Open File dialog opens. Then you can select the exported lambda function, which should be available as a ZIP archive. As soon as the lambda function is imported, the name can be recognized and the configured symbols are shown on the next page.

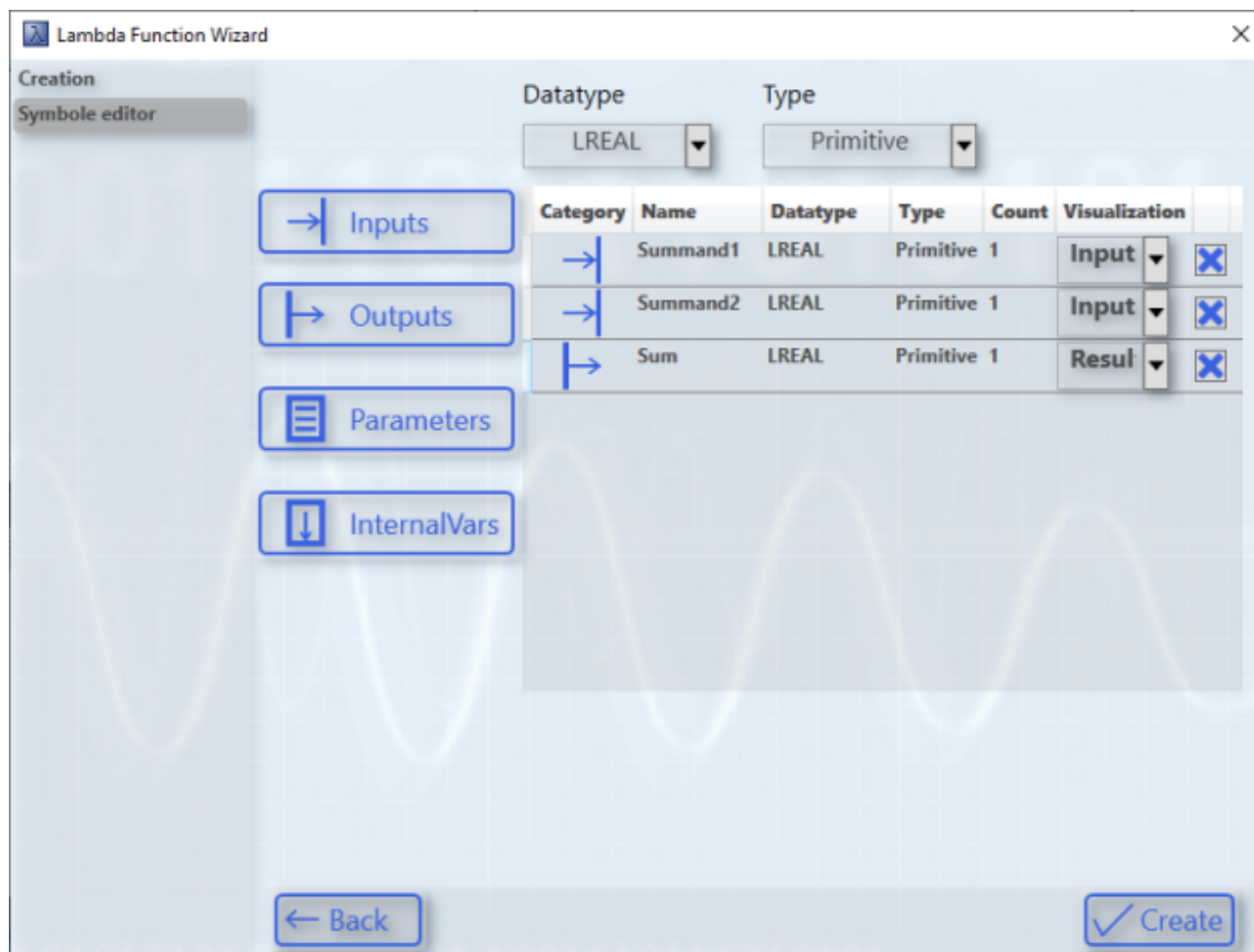
The name of the lambda function can be selected individually. However, the version number in the name of the lambda function should not be adjusted or changed. Otherwise, the proper use of the lambda functions may be impaired. Also the function name should contain at least four characters.

On the second page of the wizard you configure the symbols and variables of the lambda function.

In the Blueprint mode, some symbols are already preconfigured.



In the template mode the list of symbols is empty. As shown in the following figure, you can configure user-specific symbols.



Use the two drop-down boxes at the very top of the page to select the data type and the data structure type, respectively.

Available data types				
LREAL	REAL	BOOL	WORD	INT
LINT	ULINT	LTIME	STRING(255)	

For the data structure types you can choose between primitive and array.

Use the buttons on the right to add the variables with the selected data type and data structure type.

Internal variables are member variables. They are not displayed in the algorithm interface and cannot be linked in Analytics Workbench. Their purpose is to store variable values over cycles.

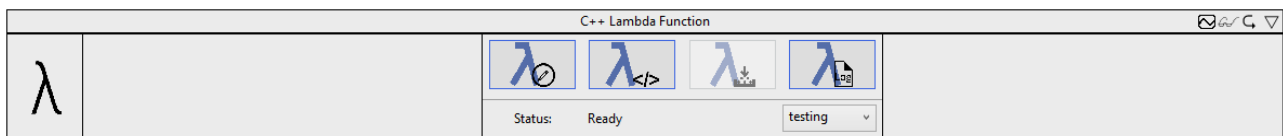
If the data structure type Array is selected for a variable, the size of the array can be specified in the Count column of the variable. To create multi-dimensional arrays, the dimensions must be separated with the letter "x" (example: 2x4).

You can configure the visualization of the symbol in the Workbench using the drop-down menu. Depending on the data type, different visualizations are possible. Undefined visualizations are not possible. No visualization can be selected for the internal variables because they are not displayed in the Workbench. The following visualizations are possible for the respective data types for inputs, outputs and parameters:

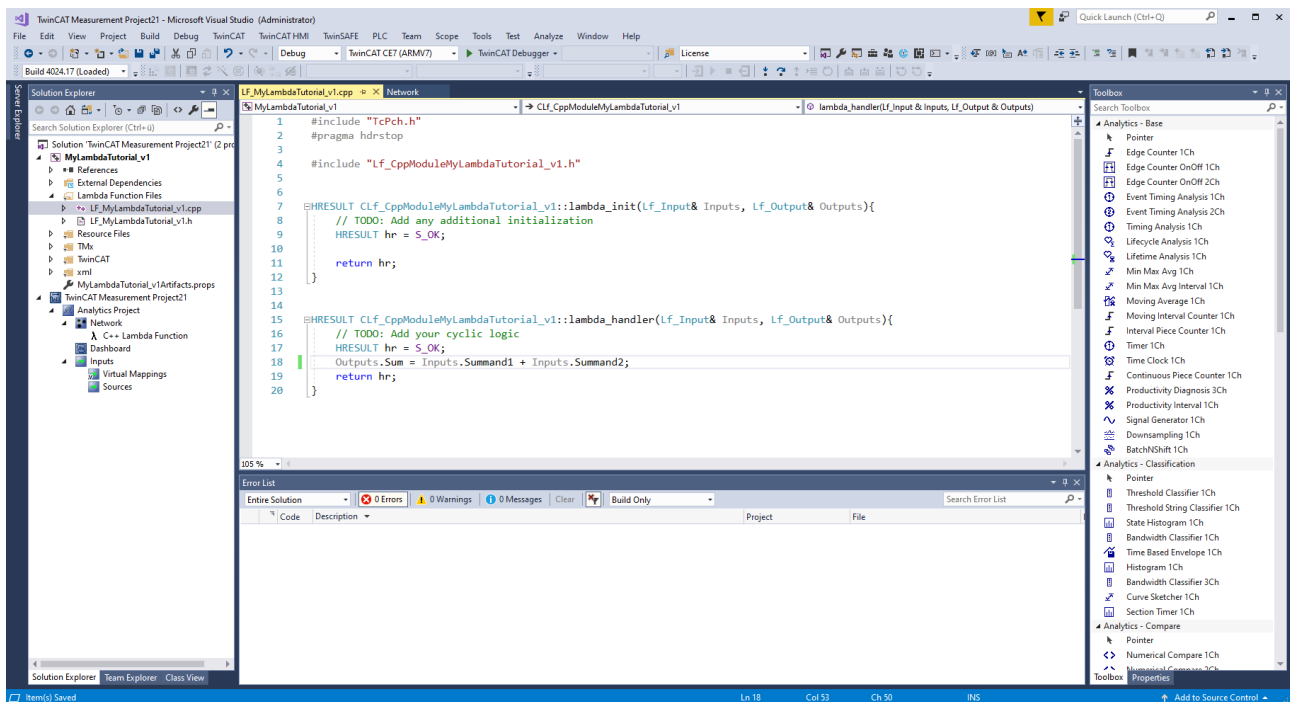
Data type	Visualization	Sample
STRING(255)	String (Input)	
	String (Output)	Output String String Result
	String (Parameter)	String Level 1 String Result
	FileOpen/FileSave (Parameter)	File Path TwinCAT3.1\Boot\Teach.tas ...
ULINT	Timestamp (Output)	Last Event 15.11.2021 11:14:57.048
BOOL	Boolean (Input)	Enable Execution New Result @ String Compare 1Ch False
	Boolean (Output)	New Result False
	Boolean (Parameter)	Use Absolute Values
REAL	Input (Input)	Input Variables.FTriangular @ Virtual Input (2)... -1,4508
	Result (Output)	Avg -0,224
INT	Count (Output)	Count 6
LINT	Timespan (Output)	Elapsed Time 04:35:28:298
	Clock (Parameter)	Time On hh 12 mm 0 ss 0
	Interval (Parameter)	Interval Seconds 30
WORD	DayOfWeekMask (Parameter)	Day Of Week Mask Mon Tue Wed Thu Fri Sat Sun
LREAL	Input (Input)	Input Variables.FTriangular @ Virtual Input (2)... -1,4508
	Result (Output)	Avg -0,224
	Level (Parameter)	Tolerance 1,25

After clicking the Create button, the C++ project is generated in the background and the wizard is closed.

Programming the lambda function



After you click the second button, the C++ project will be added to the Solution Explorer. The project contains all files, dependencies and configuration to develop an executable TcCOM module for the Analytics Workbench as well as for use in a real-time context.



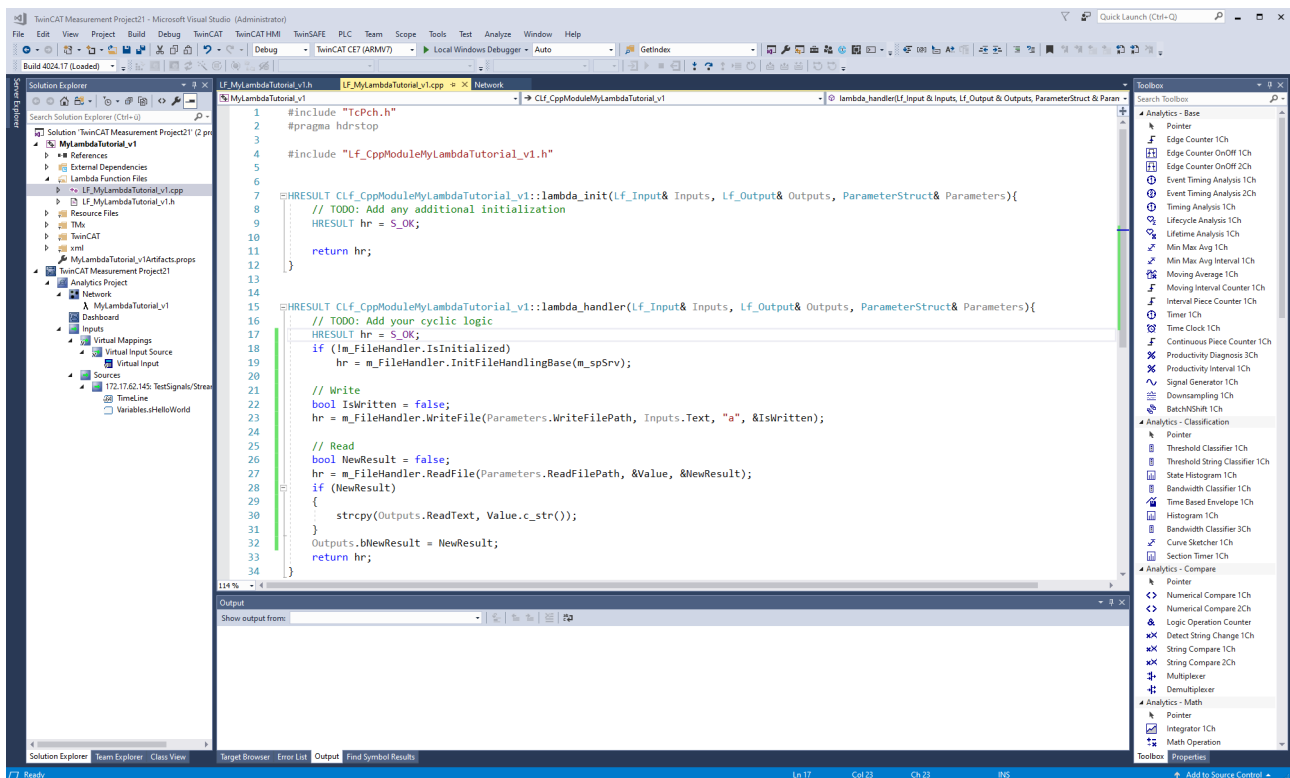
The automatically opened file contains several encapsulated functions. The so-called `lambda_init` method is for the initialization of variables. The `lambda_handler` method is executed cyclically. The `lambda_uninit` method is for deinitializing variables. In both methods the inputs, outputs, parameters and internal variables are available as structures in the method parameters. The following inputs and outputs are available by default in the lambda functions.

Category	Symbol name	Data type	Description	Visibility
Input	tTimestamp	ULINT	The variable contains the TwinCAT timestamp.	Hidden
	bReset	BOOL	The variable resets the configured symbols.	Optional
	bEnableExecution	BOOL	The variable activates the <code>lambda_handler</code> method.	Optional
Output	bNewResult	BOOL	The variable indicates whether the algorithm returns new results.	Optional

Variables created within the methods lose their content after one cycle. Internal variables offer the possibilities to store variable values over cycles. An object named "m_FileHandler" is available in the two Lambda methods. You must initialize this object once, if you have not already done so. Then you can use the two functions "ReadFile" and "WriteFile" of the object. The functions have the following properties:

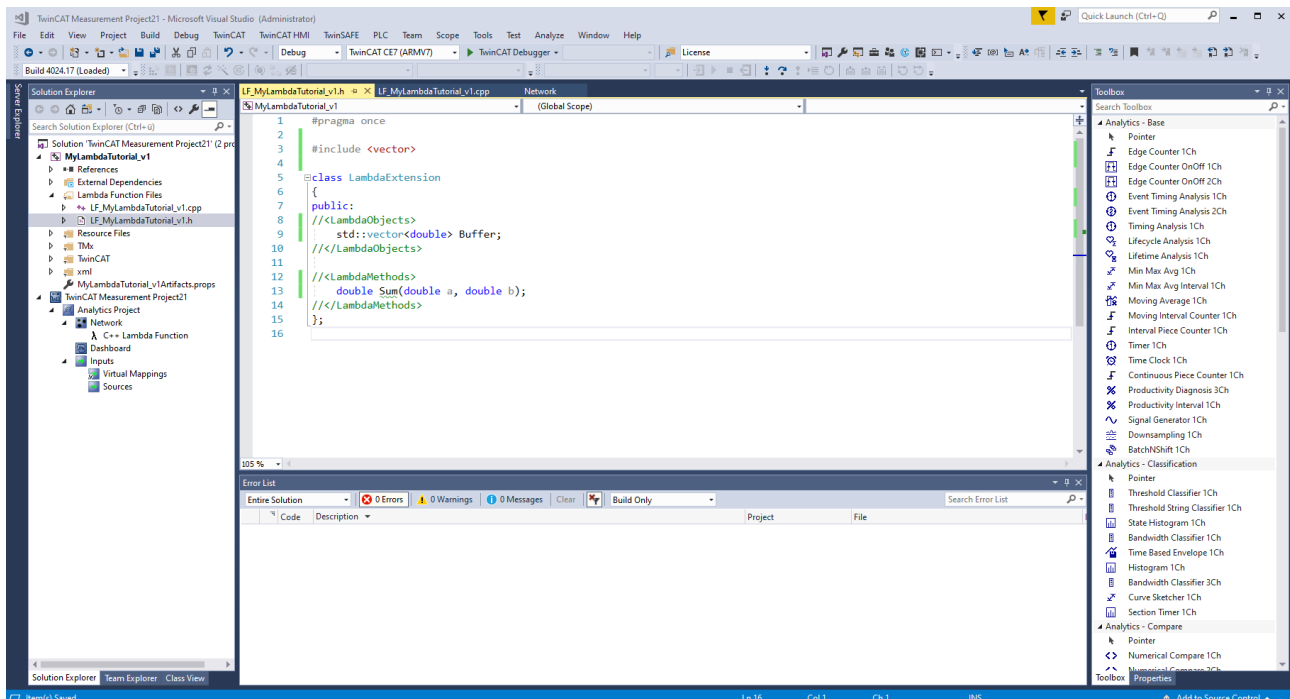
Function name	Parameter				Return value
InitFileHandlingBase	ITComObjectServer* ipSrv				Hresult
ReadFile	PCCH filename	Std::string* value	bool* bNewResult		Hresult
WriteFile	PCCH filename	Std::string value	Std::string mode	bool* blsWritten	Hresult

In TwinCAT Analytics Workbench, file access is synchronous, whereas in Analytics Runtime it is asynchronous. This feature is automatically distinguished in the methods. In case of an asynchronous read access, the method does not return the read value in the same cycle. Only as soon as the parameter `bNewResult` returns the value true, the parameter value contains the read value. With asynchronous write access, the data is stored internally and written at the next opportunity. Once the data is written, the `blsWritten` parameter returns true. You can use the two parameters `bNewResult` and `blsWritten` analogously for a synchronous file access. The following figure shows an example usage of the file access in the lambda functions:

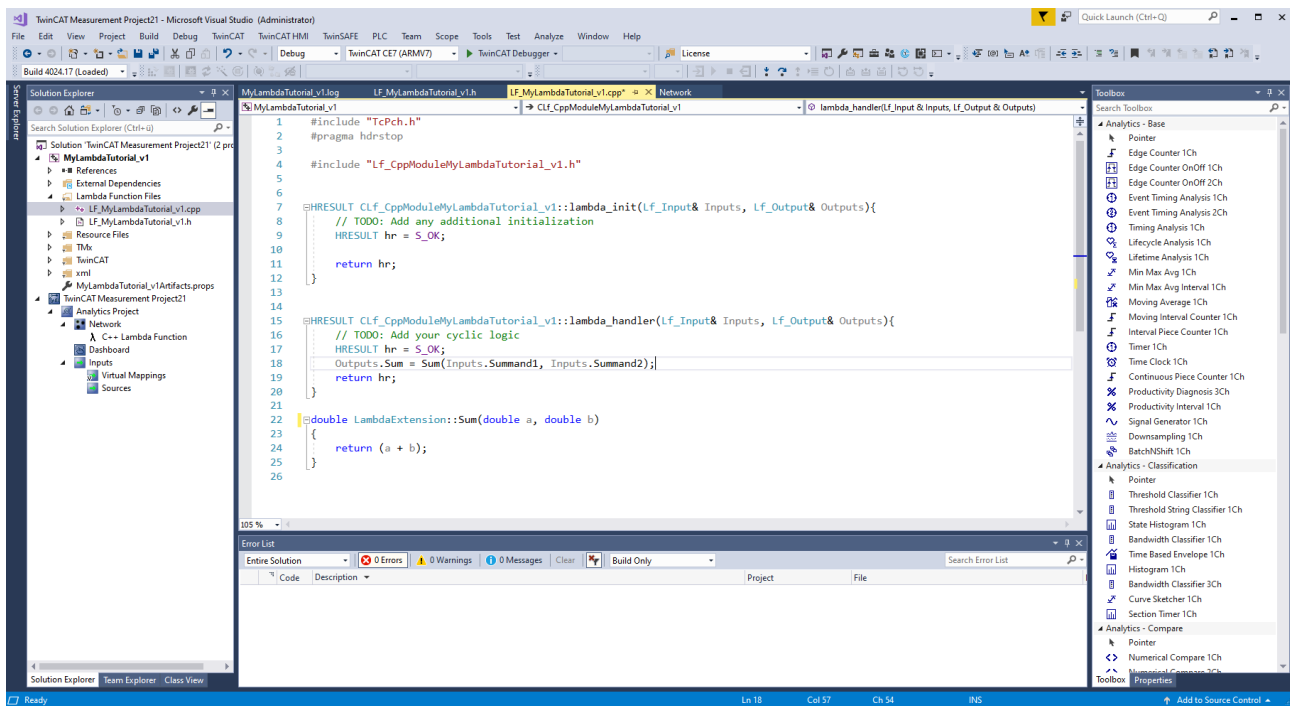


Create the project as usual in Visual Studio to check the syntax.

In the header file in the "Lambda Function Files" folder you can declare user-specific objects and methods.



Above the class "LambdaExtension" libraries can be included. Within the class, objects and methods can be declared and used in the lambda functions. The newly declared methods can be implemented below the two lambda functions.

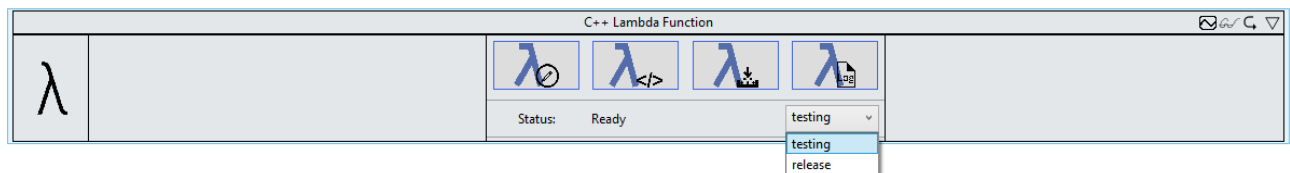


After the logic is integrated, you need to save the project. Then switch back to the lambda template in the Analytics project.

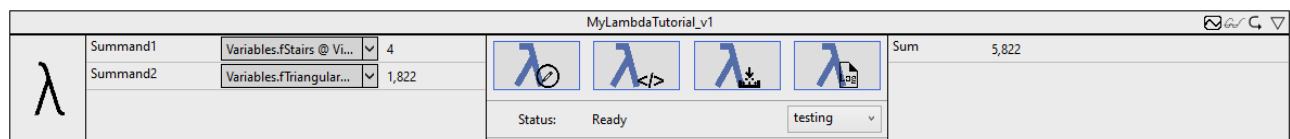
If sufficient know-how in dealing with TcCOM modules is available, the C++ project can be extended individually. If the TcCOM modules are used improperly, errors may occur during subsequent steps of the lambda functions. More information on the development of TcCOM modules can be found at the following link [TwinCAT C++ development](#).

Publishing the lambda function

Before publishing, you can configure the type of lambda function using the drop-down menu at the bottom right.



A lambda function of type "testing" contains the control with the four buttons for developing lambda functions. This type is useful during the development and testing phase.



Once the lambda function is working, select the "release" type. After clicking the third button, a Release Wizard opens.

Lambda Function Release Wizard

✕

Configurations

Release Configuration

Final Function Name:

MyLambdaTutorial_v2

Function Icons:
Icon 256 x 256 px

...

Icon 16 x 16 px

...

Group Name

C++ Lambda Functions

Map HMI Control: ☒

✓ Create

The wizard offers the possibility to assign a final name and select individual icons. One icon with 256 x 256 pixels and one icon with 16 x 16 pixels must be selected. Alternatively, you can continue to use the Lambda icons. Use the drop-down menu to assign the lambda function to a specific group in the toolbox. By default, the "C++ Lambda Functions" group is selected. For example, a final lambda function can be placed in the "Analytics - Base" section of the toolbox. If the "Map HMI Control" checkbox is activated, a wizard for linking HMI controls with the lambda function opens after publishing. In Analytics HMI Control Manager, the points for module selection are preselected directly with the corresponding lambda function (see also [Use customized and own controls \[► 382\]](#)). Use the Create button to generate the lambda function.

MyLambdaTutorial_v2							
	Summand1	<Empty>	▼	-		Sum	Empty
	Summand2	<Empty>	▼	-			

After successful publishing, the lambda template is automatically replaced by the newly generated lambda function.

Viewing the log files

If an error occurs while publishing the lambda function, it is shown in the status display and the log file can be viewed via the fourth button.

5.2.2.4 Lambda Blueprints

Trigonometric functions

TrigonometricFunctions							
	DataIn	<Empty>	▼	-	Trig_function	sin	DataOut Empty

The blueprint **Trigonometric function** performs the calculation of a trigonometric function on an input channel and returns the result to the output channel. The input channel expects the angle in radians. The respective trigonometric function can be specified in the parameter area.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Trig_function**: the short form of the trigonometric function to be calculated, e.g. sin for sine. The choices include: sin, cos, tan, asin, acos and atan. The input value for asin and acos must lie in the interval [-1,1].

Output values

- **DataOut**: outputs the result of the calculation of the trigonometric function.

Label Encoding

LabelEncoding			
λ	Message	<Empty>	-
			EncodedMessage Empty
			CategoryCount Empty

The blueprint **Label Encoding** converts text on an input channel to numeric values. For this purpose, the texts are classified into different categories at the input channel. Equal input texts return the same numeric value.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **EncodedMessage**: this output channel returns the index of the corresponding category.
- **CategoryCount**: this output channel outputs the number of categories.

Covariance

Covariance			
λ	xi	<Empty>	-
	yi	<Empty>	-
	WindowSize	1000	Cov Empty

The blueprint **Covariance** calculates the covariance between two input channels. WindowSize specifies how many values are included in the calculation of the covariance.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

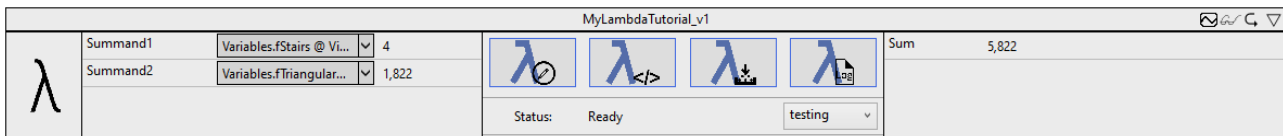
- **WindowSize**: this parameter specifies the number of cycles over which the covariance is calculated. A sliding window is implemented in the blueprint. The current values are included in the analysis and outputs are updated with each cycle. The size of the router memory should be taken into account when setting this parameter.

Output values

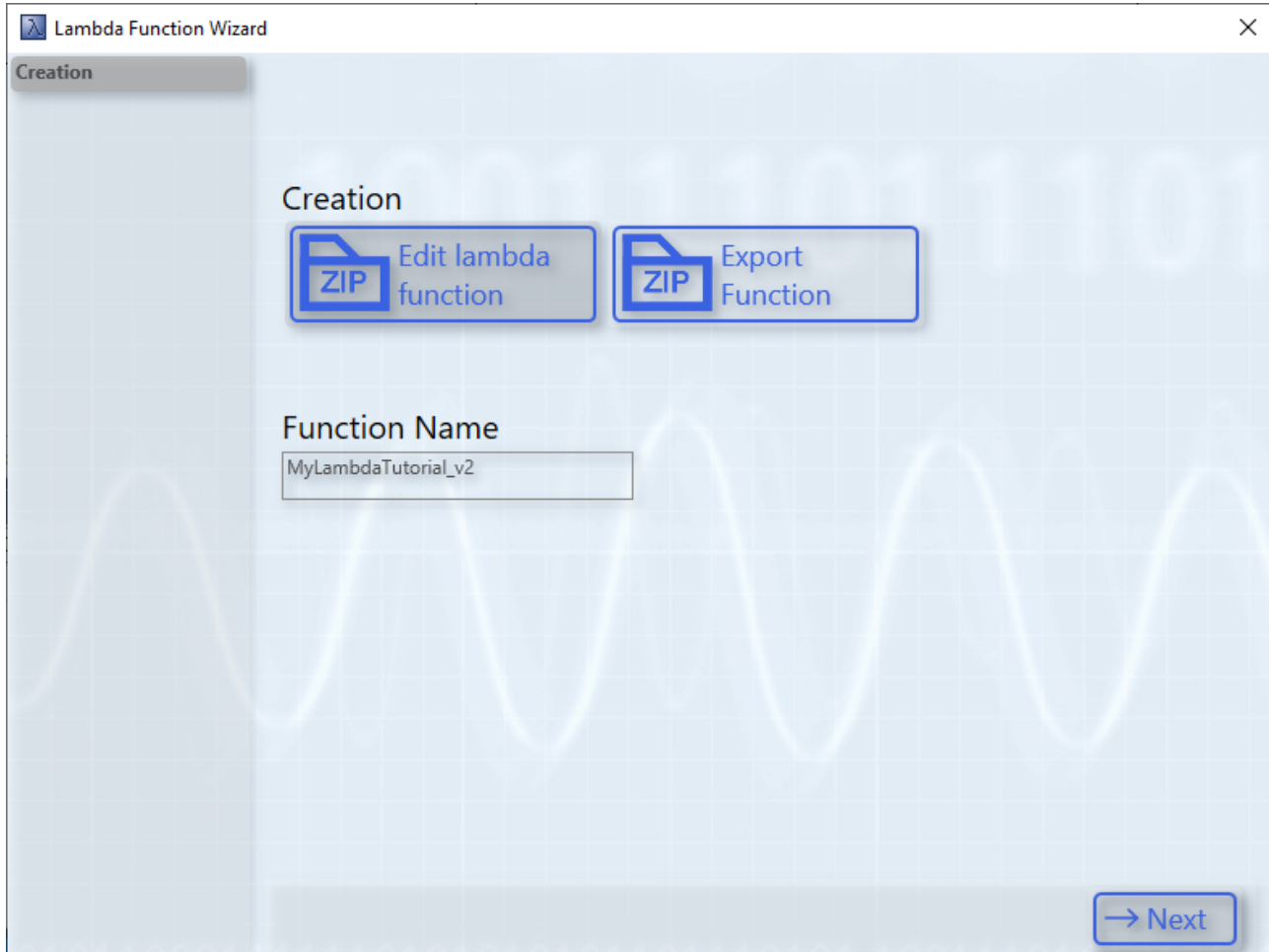
- **Cov**: this output channel indicates the covariance of the two input channels over the configured window size.

5.2.2.5 Lambda Function

In the development and testing phase, lambda functions should be published as testing versions. These contain the configured symbols, as well as the Lambda Control with the four buttons.

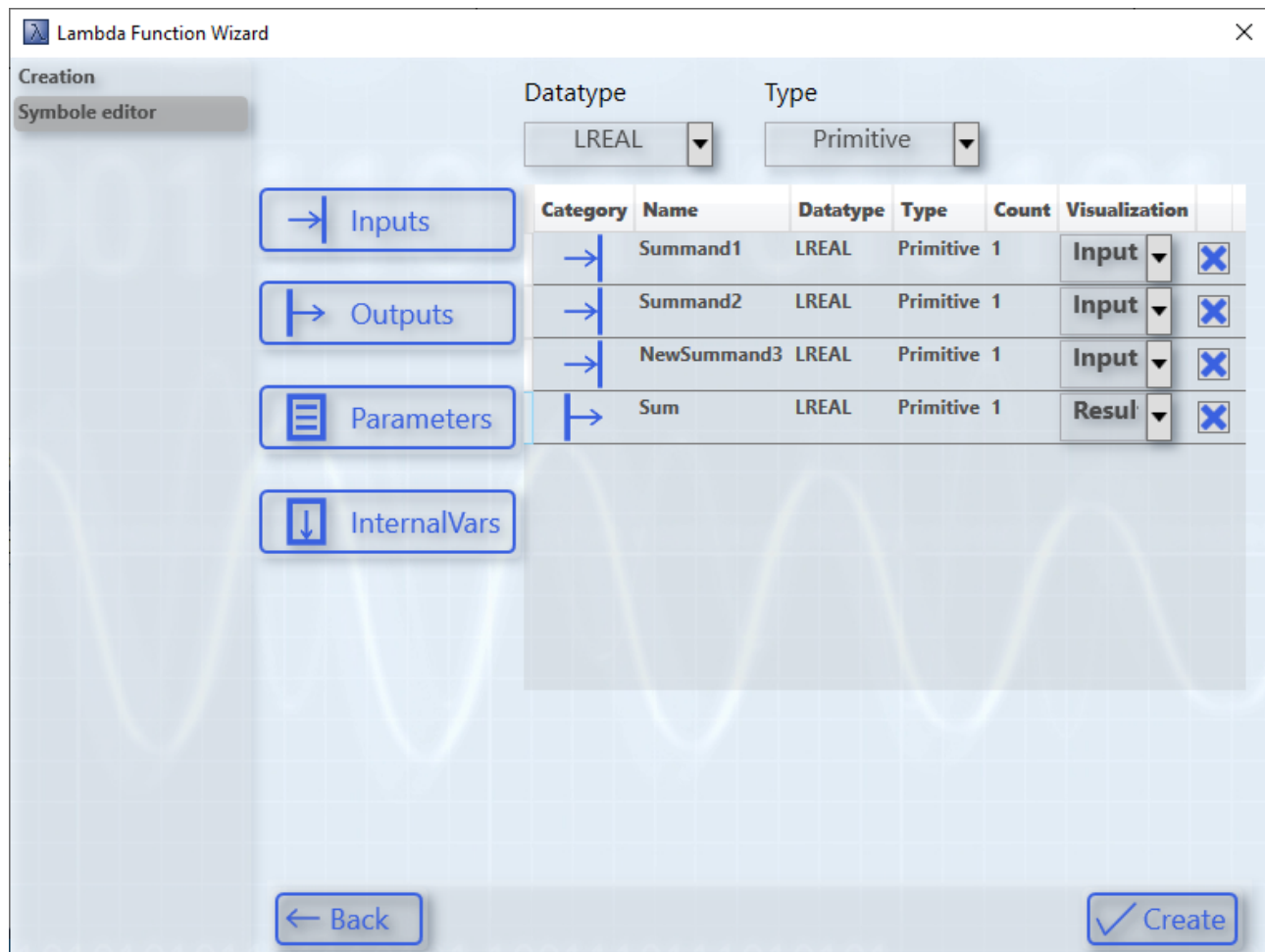


This lambda function can be tested like any other algorithm in Analytics Workbench. If changes are necessary, the Lambda Control guides you through the further development. After clicking the first button, a wizard for configuring the lambda function opens. The wizard for existing lambda functions offers two modes.

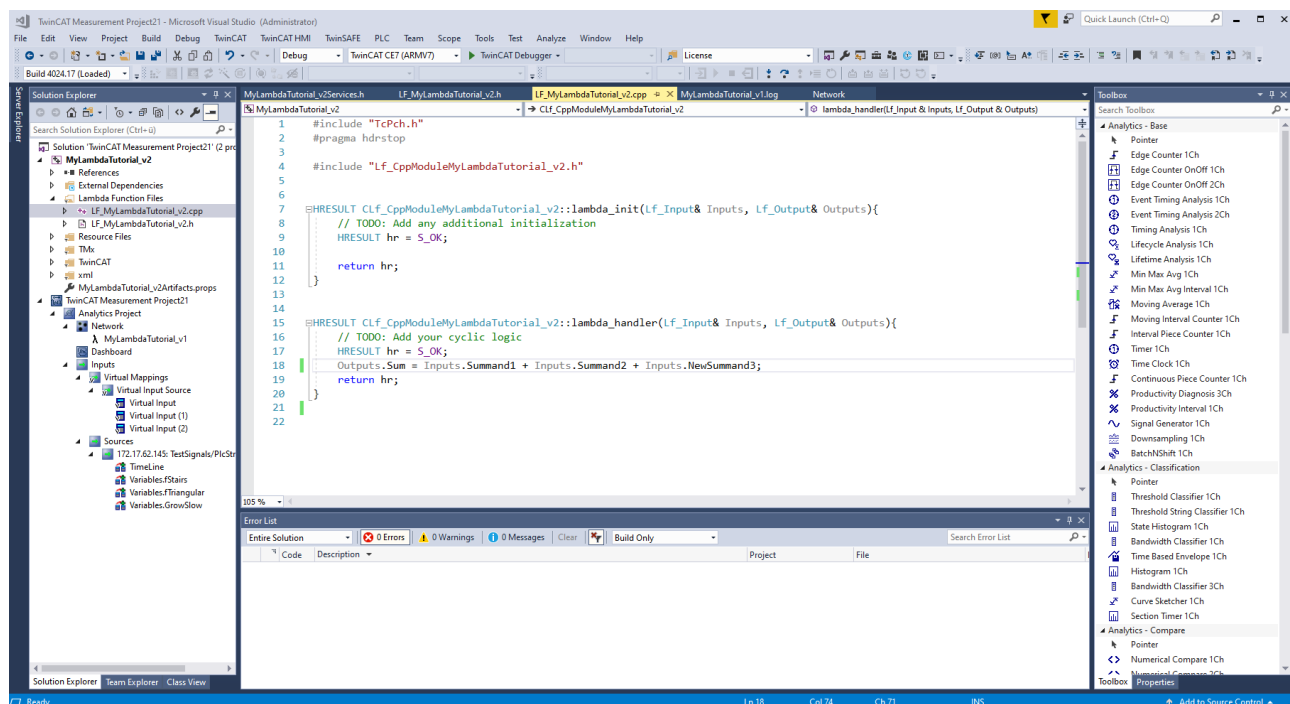


The button **Export Function** offers the possibility to export lambda functions. For this purpose, the C++ project and the drivers of the lambda function are saved in a zip archive. Using the import function in the lambda template, the lambda function can be conveniently imported and integrated on another system. Via the mode **Edit lambda function** the already developed lambda function can be edited. The second button can be used to export the current lambda function. A Save File dialog opens for this purpose. This dialog can be used to specify the file location. Exporting starts after the selection. The exported lambda function can then be distributed to other systems. The other systems can import the lambda function from the lambda template using the wizard.

In the **Edit lambda function** mode, the symbols can be customized on the next page of the wizard.



In this example, another input named "NewSummand3" has been added. These changes are also included in the C++ file in the subsequent step.



The new variable is included in the structure from inputs and can be used in the program code. The program code from the previous version was taken over. If symbols have been removed across a version, the corresponding changes must be made in the program code.

The further steps in the development of the lambda function are analogous to the procedure for the lambda template.

Debugging Lambda Functions

To debug lambda functions, the tools of TwinCAT C++ have to be used. For this purpose a new TwinCAT project must be created and the C++ project of the lambda function must be inserted under the C++ node. All TwinCAT C++ debugging options are available (see Debugging). The Debug tab can be used to attach to the TcScopeSever process. Afterwards the analysis is to be started in the Workbench and the set breakpoints are reached in the C++ project.

Tidying up the toolbox

After some lambda functions have been developed and tested, the toolbox may be very full. To clean up the toolbox, the corresponding lambda functions must be deleted from the Modules folder. The folder can be found under the path *<TwinCAT3Dir>/CustomConfig/Modules*.

5.2.2.6 Further steps for the lambda functions

Runtime deployment

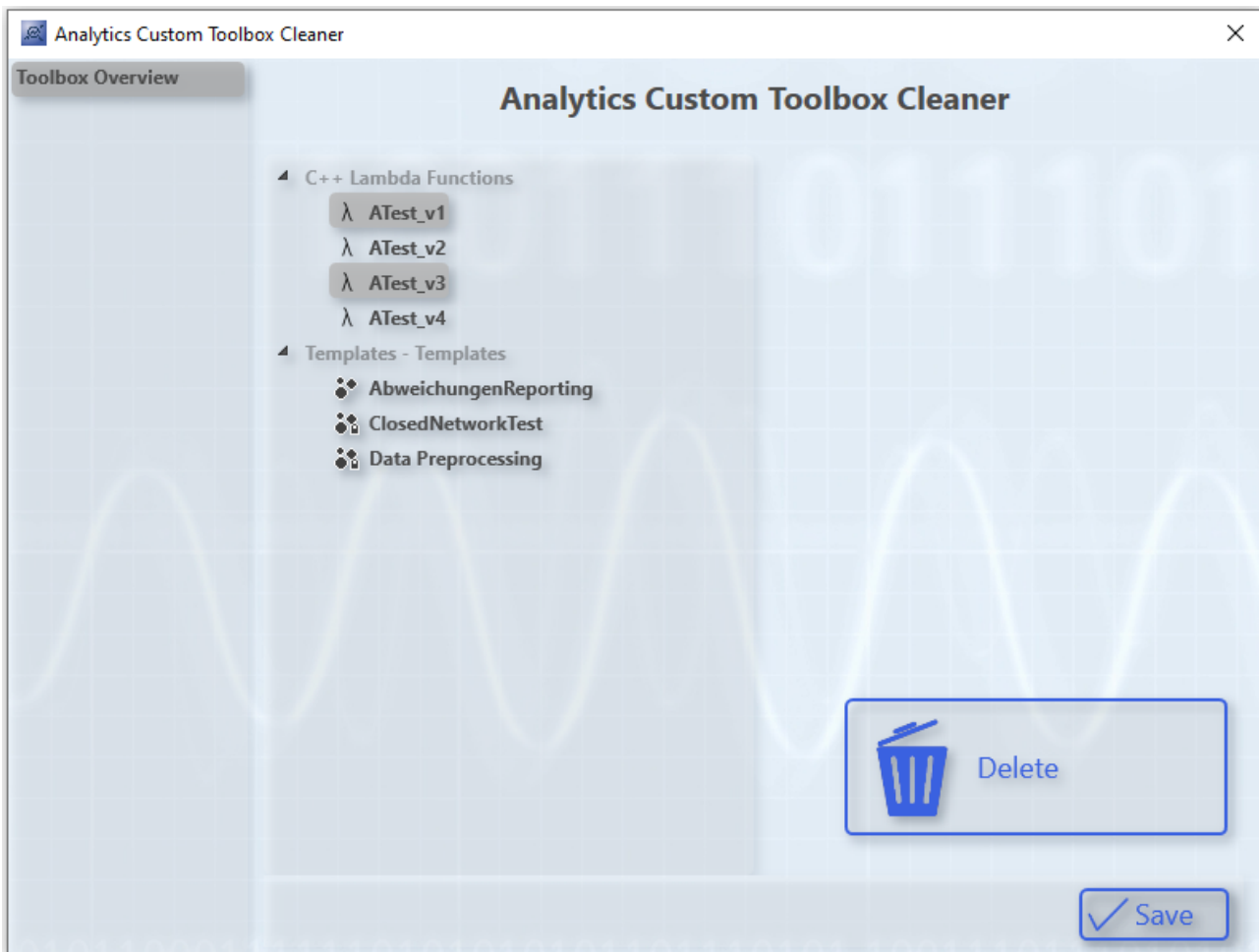
Lambda functions are also suitable for runtime deployment. The developed lambda function is created as a TcCOM module in the TwinCAT project for this purpose. A PLC function block handles the TcCOM module. The PLC function block calls the corresponding interface methods of the lambda function for this purpose. The interface methods of the lambda function, as well as the associated PLC function block are generated automatically and do not require any explicit configuration. Thus, the user-specific lambda functions can be downloaded into a TwinCAT Analytics Runtime and an individual 24/7 data analysis can be realized. More information can be found at the following link (see [Runtime deployment](#) [► 303]).

HMI One Click Dashboard

To display the results of the lambda functions, the integration into the HMI One Click Dashboard is ideal. A new control can be linked via the Dashboard node of an Analytics project. In the Analytics Dashboard Wizard, an HMI Control and the corresponding lambda function can be selected and linked. The Analytics Dashboard Wizard is described in more detail at the following link (see [Manage dashboard structure and content in Analytics project](#) [► 372]).

5.2.2.7 Analytics Custom Toolbox Cleaner

A wide range of toolbox elements can be created using the network templates and the Lambda functions. To conveniently remove these elements, Analytics Custom Toolbox Cleaner can be used. The following figure shows the wizard:

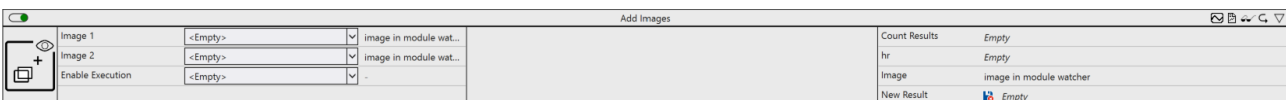


The individual toolbox elements can be selected via the tree structure. By holding down CTRL several elements can be selected in the building structure. The Delete button removes the elements from the building structure. The changes are confirmed by clicking on the Save button. If the wizard is closed otherwise, the changes will not be applied.

5.2.3 Vision

The vision algorithms of the TwinCAT Analytics Engineering Tools originate from the PLC libraries of the TF7xxx product family.

5.2.3.1 Add Images



Item-by-item addition of two images with saturation arithmetic.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.2 Clahe

Clahe						
Image	<Empty>	image in module wat...	Clip Limit	0.5	Count Results	Empty
Enable Execution	<Empty>	-	Tiles X	8	hr	Empty
			Tiles Y	6	Image	image in module watcher
					New Result	Empty

Applies a Contrast Limited Adaptive Histogram Equalization. A local histogram uniform distribution is applied to individual tiles.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Clip Limit:** Threshold value for the contrast limit. Clipping is disabled with ≤ 0 .
- **Tiles X:** Number of tiles in the X direction.
- **Tiles Y:** Number of tiles in Y direction.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.3 Closest Points BF

Closest Points BF				
Image	<Empty>	image in module wat...	Count Results	Empty
Point Set 1	<Empty>	-	hr	Empty
Point Set 2	<Empty>	-	Minimum Distance	Empty
Enable Execution	<Empty>	-	Point 1	0 0
			Point 2	0 0
			Image	image in module watcher
			New Result	Empty

Determines the smallest distance between two 2D point sets of the same type using the brute force method.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Minimum Distance:** Minimum distance between the points of two point sets.
- **Point 1:** Point of the first set of points with which the minimum distance was determined.
- **Point 2:** Point of the second point set with which the minimum distance was determined.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.4 Color Map

Color Map						
Image	<Empty>	image in module wat...	Color Map	AUTUMN	Count Results	Empty
Enable Execution	<Empty>	-			hr	Empty
					Image	image in module watcher
					New Result	Empty

Applies a color scheme to a grayscale image. If the image is not a grayscale image, it is automatically converted.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

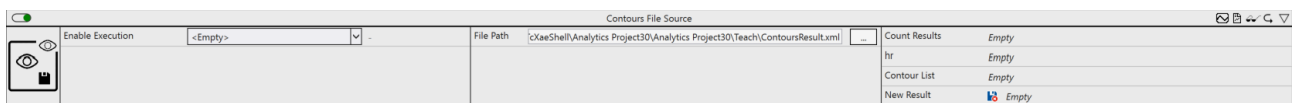
Configuration options

- **Color Map:** Choice of color scheme to be applied to the image.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.5 Contours File Source



Loads contours from the file system. These can be saved using the [Contours File Store](#) [► 272] module, for example.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

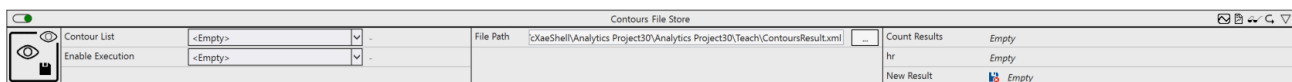
Configuration options

- **File Path:** File path of an XML file in which contours are saved.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Contour List:** Container in which the loaded contour is made available.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.6 Contours File Store



Saves contours to the file system.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

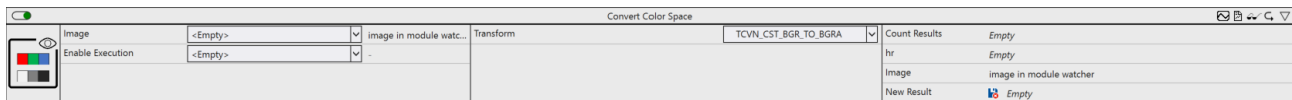
Configuration options

- **File Path:** File path of an XML file in which contours are to be saved.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.7 Convert Color Space



Converts an image from one color space to another.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

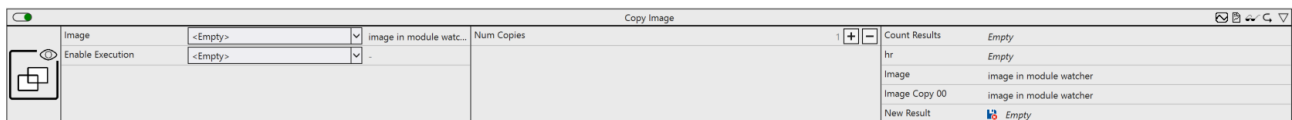
Configuration options

- **Transform:** Choice of transformation to be performed.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.8 Copy Image



Creates one or more deep copies of an image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

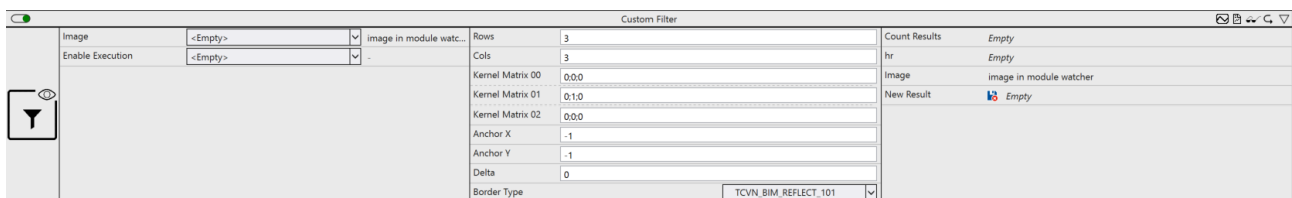
Configuration options

- **Num Copies:** Specifies how many independent copies are to be created.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image Copy 00..n:** Copy of the input image.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.9 Custom filter



Applies a user-defined filter to the image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Rows:** Rows of the filter core matrix.
- **Cols:** Columns of the filter core matrix.

- **Kernel Matrix:** Filter kernel matrix
- **Anchor X:** (Optional) The anchor point defines which point of the filter kernel is moved over each pixel of the input image.
- **Anchor Y:** (Optional) The anchor point defines which point of the filter kernel is moved over each pixel of the input image.
- **Delta:** (Optional) The constant delta is added to the result of the filter operation.
- **Border Type:** (Optional) Treatment of the image border.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.10 Detect Blobs

Recognizes blob contours. To do this, a threshold value is first applied to the input image. A contour search is then carried out using blob detection.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Filter By Area:** Activates filtering by area. This must lie between *Min Area* and *Max Area*. Recommended for noise suppression.
- **Filter By Circularity:** Activates filtering by circularity. This must lie between *Min Circularity* and *Max Circularity*. It is defined as follows: $4\pi \cdot \text{area} / \text{perimeter}^2$.
- **Min Area:** (Visible when the *By Area filter* is activated) Minimum area of a blob in pixels.
- **Max Area:** (Visible when the *By Area filter* is activated) Maximum area of a blob in pixels.
- **Min Circularity:** (Visible when the *By Circularity filter* is activated) Minimum circularity (1: ideal circle, < 1: less circular, 0: not circular).
- **Max Circularity:** (Visible when the *By Circularity filter* is activated) Maximum circularity (1: ideal circle, < 1: less circular, 0: not circular).
- **Threshold:** Threshold value for the binary image.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Number Of Objects:** Displays the number of objects found.
- **Image:** Initial image of the algorithm.
- **Contour List:** Container in which the contours found are made available.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Image Threshold:** (Optional) Image to which the set threshold is applied.
- **Image Preview:** (Optional) Image in which the objects found are marked.

5.2.3.11 Detect Codes Neural Network

Detect Codes Neural Network			
Image	<Empty>	image in module watc...	Num Codes
Enable Execution	<Empty>	-	Model
			Code Type
			Code Region Ratio
			Dot Code Module Width
			Count Results
			hr
			Last Event
			Codes Detected
			Image
			Code Type 00
			Top Left Corner 00
			New Result
			Image Preview

Recognizes different code types such as barcodes, QR codes and data matrix codes in the image provided using a trained neural network. The recognized codes are sorted from high to low according to their model confidence. If a code is recognized, the area is enlarged according to the ratio of the code region. Depending on the code type detected, the appropriate reading algorithm is then applied.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Codes:** Maximum number of codes to be found.
- **Model:** Model of the neural network for code recognition. Different numbers of models are available. These include specialized models for 1D codes, 2D codes or all code types.
- **Code Type:** Fixes the code type(s) to be recognized.
- **Code Region Ratio:** (Optional) Enlarges or reduces the code region found in proportion to its current size.
- **Dot Code Module Width:** (Optional) Average dot diameter of the code in the image in pixels. The value must be at least 3, preferably 5 to 8. Only required for dot codes.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Last Event:** Timestamp of the last result.
- **Codes Detected:** Number of codes found.
- **Image:** Initial image of the algorithm.
- **Code Type 00..n:** Type of the code found
- **Decoded Data 00..n:** Decoded data of the code found.
- **Top Left Corner 00..n:** Top left corner of the code found.
- **Bottom right corner 00..n:** Bottom right corner of the code found.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Image Preview:** (Optional) Image in which the objects found are marked.

5.2.3.12 Draw Arrow

Draw Arrow			
Image	<Empty>	image in module watc...	Color
Start Point	<Empty>	-	Thickness
End Point	<Empty>	-	Double Head
Enable Execution	<Empty>	-	Line Type
			Shift
			Tip Length
			Count Results
			hr
			Image
			New Result

Draw a line with an arrowhead in the picture.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Color:** Color of the arrow.
- **Thickness:** Line width.

- **Double Head:** Draws an arrow on each side of the line.
- **Line Type:** (Optional) Line type.
- **Shift:** (Optional) Decimal bits of the coordinates (bit shift).
- **Tip Length:** (Optional) Length of the arrowhead as a proportion of the line length.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.13 Draw Circle

Draw Circle						
Image	<Empty>	image in module watc...	Radius	0	Count Results	Empty
Center Point	<Empty>	-	Color	Green	hr	Empty
Enable Execution	<Empty>	-	Thickness	3	Image	image in module watcher
			Line type	TCVN_LT_8_CONNECTED	New Result	Empty

Draw a circle in the picture.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Radius:** Radius of the circle in pixels.
- **Color:** Color of the circle.
- **Thickness:** Line width. If the value is negative, the circle is filled.
- **Line Type:** (Optional) Line type.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.14 Draw Contours

Draw Contours						
Image	<Empty>	image in module watc...	Contour Index	0	Count Results	Empty
Contours	<Empty>	-	Color	Green	hr	Empty
Enable Execution	<Empty>	-	Thickness	3	Image	image in module watcher
			Line type	TCVN_LT_8_CONNECTED	New Result	Empty

Draws a single set of points or several sets of points in the image, which are interpreted as contours.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Contour Index:** Index of the contour if a container with several contours is provided at the input.
- **Color:** Color of the contour.
- **Thickness:** Line width. If the value is negative, the form is filled.
- **Line Type:** (Optional) Line type.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.

- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.15 Draw Line

Draw Line						
Image	<Empty>	image in module watc...	Color	Green	Count Results	Empty
Start Point	<Empty>	-	Thickness	3	hr	Empty
End Point	<Empty>	-	Line type	TCVN_LT_8_CONNECTED	Image	image in module watcher
Enable Execution	<Empty>	-	Shift	0	New Result	Empty

Draw a line in the picture.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Color:** Color of the line.
- **Thickness:** Line width.
- **Line Type:** (Optional) Line type.
- **Shift:** (Optional) Decimal bits of the coordinates (bit shift).

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.16 Draw Point

Draw Point						
Image	<Empty>	image in module watc...	Shape	TCVN_DS_CIRCLE	Count Results	Empty
Point	<Empty>	-	Color	Green	hr	Empty
Enable Execution	<Empty>	-	Size	3	Image	image in module watcher
			Thickness	3	New Result	Empty
			Line type	TCVN_LT_8_CONNECTED		

Draws a point in the picture.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Shape:** Shape of the point.
- **Color:** Color of the point.
- **Size:** (Optional) Size of the shape.
- **Thickness:** (Optional) Line width. If the value is negative, the form is filled.
- **Line Type:** (Optional) Line type.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.17 Draw Rectangle

Draw Rectangle						
Image	<Empty>	image in module watc...	Color	Green	Count Results	Empty
Top Left Corner	<Empty>	-	Thickness	3	hr	Empty
Bottom Right Corner	<Empty>	-			Image	image in module watcher
Enable Execution	<Empty>	-			New Result	Empty

Draws a rectangle in the picture.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Color:** Color of the rectangle.
- **Thickness:** Line width. If the value is negative, the rectangle is filled.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.18 Extract ROI From Contour

Extract ROI From Contour					
Image	<Empty>	image in module watc...	Auto Revert ROI	☒	Count Results
Contour List	<Empty>	-			hr
Enable Execution	<Empty>	-			Image
					New Result

Extracts a region of interest (ROI) from a contour.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Auto Revert ROI:** Automatically resets the ROI at the end of the network.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.19 Gaussian filter

Gaussian Filter					
Image	<Empty>	image in module watc...	Filter Width	3	Count Results
Enable Execution	<Empty>	-	Filter Height	3	hr
			Sigma X	1	Image
			Sigma Y	0	New Result
			Border Type	TCVN_BIM_REFLECT_101	

Applies a Gaussian filter to smooth the image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

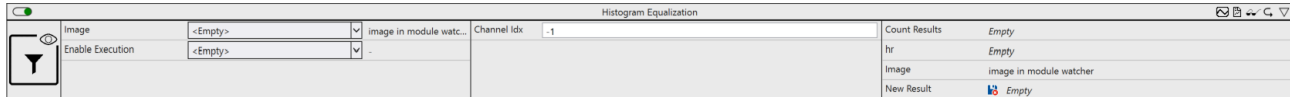
Configuration options

- **Filter Width:** Filter width in pixels (1, 3, 5, 7, ...).
- **Filter Height:** Filter height in pixels (1, 3, 5, 7, ...).
- **Sigma X:** (Optional) Standard deviation of the Gaussian kernel in the X direction (≥ 0 , automatically selected at 0).
- **Sigma Y:** (Optional) Standard deviation of the Gaussian kernel in the Y direction (≥ 0 , automatically selected at 0).
- **Border Type:** (Optional) Treatment of the image border.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.20 Histogram Equalization



Equalizes the histogram of a grayscale or color image, thereby normalizing the brightness and improving the contrast. The channel index to be equalized must be specified. The value -1 expects an RGB image, converts it to YCbCr, equalizes the Y channel and then converts the image back to RGB. If no RGB image is provided, automatic conversion takes place internally.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

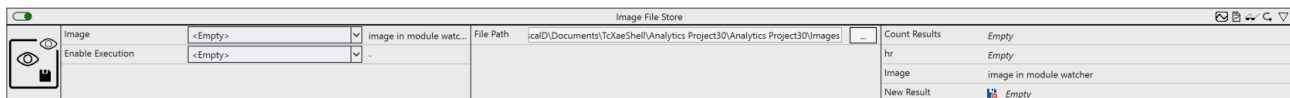
Configuration options

Channel Idx: (Optional) Index of the image channel to be equalized. -1 expects an RGB image, converts it to YCbCr, equalizes the Y channel and then converts back to RGB.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.21 Image File Store



Writes an image to the file system.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

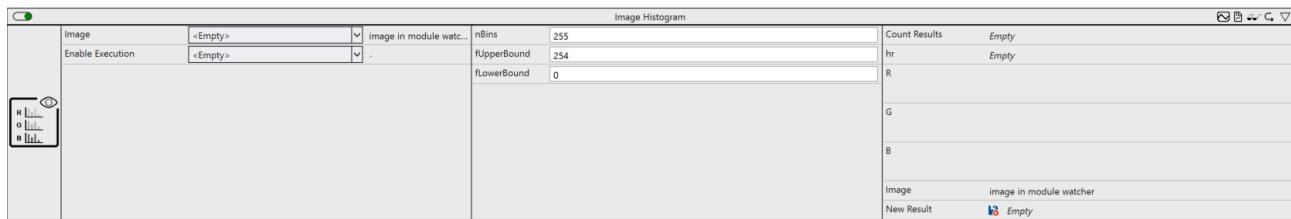
Configuration options

- **File Path:** File path for saving the images.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.22 Image Histogram



Calculates the histogram of an image. An RGB image is expected at the input. If no RGB image is provided, automatic conversion takes place internally.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Bins:** Indicates the size of the buffer and thus the number of values that are stored. The number of output channels equals the buffer size.
- **Upper Bound:** Upper (exclusive) limit of the last histogram bin $nBins-1$.
- **Lower Bound:** Lower (including) limit of the 0th histogram bin.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **R:** Histogram of the red channel
- **G:** Histogram of the green channel
- **B:** Histogram of the blue channel
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.23 Image Info



Reads out image information.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image Size:** Number of pixels of the input image.
- **Width:** Width of the input image in pixels.
- **Height:** Height of the input image in pixels.
- **X Padding:** X-padding of the input image.
- **Y Padding:** Y-padding of the input image.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.24 Image Statistics



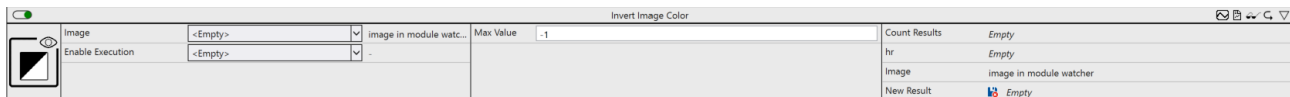
Outputs image statistics such as minimum value, maximum value, mean value, and standard deviation of the pixel values. An RGB image is expected at the input for this. If no RGB image is provided, automatic conversion takes place internally.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Average:** Average values of the individual RGB channels
- **Standard deviation:** Standard deviation of the individual RGB channels
- **Min:** Average values of the individual RGB channels
- **Max:** Average values of the individual RGB channels
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.25 Invert Image Color



Inverts the color of an image. In the case of signed integers or negative floating point values, only the sign is reversed. If the image contains only positive floating point values, each pixel value is subtracted from the maximum available pixel value or from 1.0, whichever is greater.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

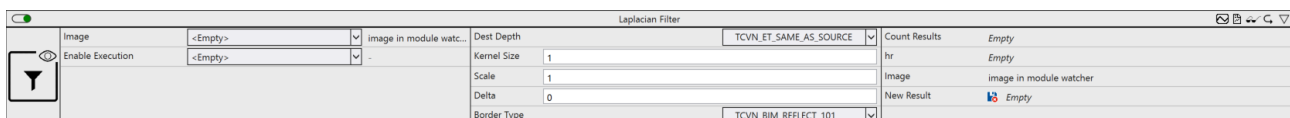
Configuration options

- **Max Value:** Maximum pixel value if, for example, a 16 bit image contains 12 bit values. -1 uses the default values.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.26 Laplacian filter



Applies a Laplace filter to an image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Dest Depth:** Depth of the original image.
- **Kernel Size:** (Optional) Aperture size for calculating the second derivative filter (1, 3, 5, ..., 31).
- **Scale:** (Optional) Scaling factor for the calculated derivative values.
- **Delta:** (Optional) Delta value that is added to the results before saving.
- **Border Type:** (Optional) Treatment of the image border.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.27 Locate Edge

Localizes the points of the first occurring edge within a specified search window with subpixel accuracy.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Start Point:** Position from which the search process is started in the direction of *the End Point*.
- **End Point:** Position at which the search process ends.
- **Edge Direction:** Definition of the edge direction to be searched for.
- **Min Strength:** Definition of the minimum strength or intensity difference of the edge being searched for.
- **Search Lines:** Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line defined by *Start Point* and *End Point*.
- **Threshold:** Threshold value for the binary image.
- **Search Line Dist:** (Optional) Distance between the search lines in pixels (> 0).
- **Max Thickness:** (Optional) Definition of the maximum thickness of the edge searched for. This means that *Min Strength* must be achieved within *Max Thickness* pixels.
- **Algorithm:** (Optional) Choice of edge detection algorithm.
- **Subpixels Iterations:** (Optional) Fixes the number of subpixels. For *INTERPOLATION*, 10 to 20 are usually sufficient. Alternatively, the value specifies the maximum number of iterations for optimizing the parameters; for *APPROX_ERF* and *APPROX_GAUSSIAN*, 50 to 100 is usually sufficient.
- **Approx Precision:** (Optional) Fixes the approximation precision for *APPROX_ERF* and *APPROX_GAUSSIAN*. 0.001 is usually sufficient; the value is not used for *INTERPOLATION*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.

- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Image Preview:** (Optional) Image in which the edge found is drawn.
- **Image Threshold:** (Optional) Image to which the set threshold is applied.
- **Edge Points:** Container in which the points of the found edge are made available.

5.2.3.28 Match Contours 1vsN

Match Contours 1vsN						
Image	<Empty>	image in module watc...	Num Contours	1	Count Results	Empty
Contour List	<Empty>	-	File Path	Analytics Project30\Analytics Project30\TeachShapeCollection.xml	hr	Empty
Enable Execution	<Empty>	-	Max Dissimilarity	0.02	File Read	Empty
			Contour Name 00	XML is not valid	Result	Empty
					Image	image in module watcher
					Match Indexes	0
					Dissimilarities	0
					New Result	Empty

Compares contours with contours previously saved in a file.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Num Contours:** Number of contours with which the input contour is to be compared.
- **File Path:** File path of an XML file in which contours were saved.
- **Max Dissimilarity:** Maximum distance between the original contour and its approximation.
- **Contour Name 00..n:** Alias names of the contours.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **File Read:** TRUE if the file was read successfully.
- **Result:** Output of the best matching contour name if the maximum dissimilarity is not exceeded.
- **Image:** Initial image of the algorithm.
- **Match Indexes:** Listing of the indexes of the contours. Sorted by equality.
- **Dissimilarities:** Listing of the inequality of the contours. Sorted by equality.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.29 Measure Angle Between Edges

Measure Angle Between Edges						
Image	<Empty>	image in module watc...	Inner Point	0.0	Count Results	Empty
Enable Execution	<Empty>	-	Outer Point 1	0.0	hr	Empty
			Outer Point 2	0.0	Angle	Empty
			Edge Direction	TCVN_ED_DARK_TO_LIGHT	Image	image in module watcher
			Min Strength	1	New Result	Empty
			Search Lines	3	Image Preview	image in module watcher
			Invert Search Direction	☐	Image Threshold	image in module watcher
			Threshold	110	Edge Points 1	Empty
			Search Line Dist	1	Edge Points 2	Empty
			Max Thickness	10		
			Subpixels Iterations	10		
			Angle In Degrees	☐		
			Approx Precision	0.001		
			Algorithm	TCVN_EDA_INTERPOLATION		

Measures the angle between two edges.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Inner Point:** Position within the angle to be measured from which the edge search is started in the direction of *Outer Point 1* and *Outer Point 2*.

- **Outer Point 1:** Position at which the search process for edge 1 ends.
- **Outer Point 2:** Position at which the search process for edge 2 ends.
- **Edge Direction:** Definition of the edge direction to be searched for from *Inner Point* to *Outer Point 1* and *Outer Point 2*.
- **Min Strength:** Definition of the minimum strength or intensity difference of the edge being searched for.
- **Search Lines:** Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line defined by *Inner Point* and *Outer Point 1* and *Outer Point 2*.
- **Invert Search Direction:** If *TRUE*, the search starts from each *outer point* in the direction of the *inner point*.
- **Threshold:** Threshold value for the binary image.
- **Search Line Dist:** (Optional) Distance between the search lines in pixels (> 0).
- **Max Thickness:** (Optional) Definition of the maximum thickness of the edge searched for. This means that *Min Strength* must be achieved within *Max Thickness* pixels.
- **Algorithm:** (Optional) Choice of edge detection algorithm.
- **Subpixels Iterations:** (Optional) Fixes the number of subpixels. For *INTERPOLATION*, 10 to 20 are usually sufficient. Alternatively, the value specifies the maximum number of iterations for optimizing the parameters; for *APPROX_ERF* and *APPROX_GAUSSIAN*, 50 to 100 is usually sufficient.
- **Approx Precision:** (Optional) Fixes the approximation precision for *APPROX_ERF* and *APPROX_GAUSSIAN*. 0.001 is usually sufficient; the value is not used for *INTERPOLATION*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Angle:** Angle between the edges found in degrees.
- **Image:** Initial image of the algorithm.
- **Image Preview:** (Optional) Image in which the edges found are shown.
- **Image Threshold:** (Optional) Image to which the set threshold is applied.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Edge Points 1:** (Optional) Outputs the points found at the edge between *Inner Point* and *Outer Point 1*.
- **Edge Point 2:** (Optional) Outputs the points found at the edge between *Inner Point* and *Outer Point 2*.

5.2.3.30 Measure Edge Distance

Measure Edge Distance	
Image	<Empty>
Enable Execution	<Empty>
Start Point	0.0
End Point	0.0
Edge Direction	TCVN_ED_DARK_TO_LIGHT
Min Strength	1
Search Lines	3
Invert Search Direction	☐
Threshold	110
Search Line Dist	1
Max Thickness	10
Search Gap	0
Subpixels Iterations	10
Approx Precision	0.001
Algorithm	TCVN_EDA_INTERPOLATION
Count Results	Empty
hr	Empty
Avg Distance	Empty
Image	image in module watcher
New Result	Empty
Image Preview	image in module watcher
Image Threshold	image in module watcher
Edge Points 1	Empty
Edge Points 2	Empty
Distances	Empty

Exhibits the distance between two parallel edges.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

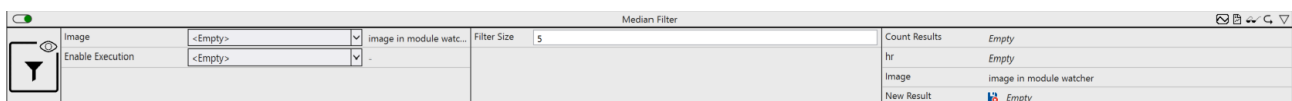
- **Start Point:** Position from which the search process is started in the direction of *the End Point*.
- **End Point:** Position at which the search process ends.
- **Edge Direction:** Definition of the edge direction to be searched for from *Start Point* to *End Point*.

- **Min Strength:** Definition of the minimum strength or intensity difference of the edge being searched for.
- **Search Lines:** Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line defined by *Start Point* and *End Point*.
- **Invert Search Direction:** If *TRUE*, the search starts from the midpoint between *Start Point* and *End Point* in both directions.
- **Threshold:** Threshold value for the binary image.
- **Search Line Dist:** (Optional) Distance between the search lines in pixels (> 0).
- **Max Thickness:** (Optional) Definition of the maximum thickness of the edge searched for. This means that *Min Strength* must be achieved within *Max Thickness* pixels.
- **Search Gap:** (Optional) Width of a gap (≥ 0 , centered between *Start Point* and *End Point*), which is ignored during the edge search and can reduce the execution time. The two edges you are looking for must be on different sides of the gap
- **Algorithm:** (Optional) Choice of edge detection algorithm.
- **Subpixels Iterations:** (Optional) Fixes the number of subpixels. For *INTERPOLATION*, 10 to 20 are usually sufficient. Alternatively, the value specifies the maximum number of iterations for optimizing the parameters; for *APPROX_ERF* and *APPROX_GAUSSIAN*, 50 to 100 is usually sufficient.
- **Approx Precision:** (Optional) Fixes the approximation precision for *APPROX_ERF* and *APPROX_GAUSSIAN*. 0.001 is usually sufficient; the value is not used for *INTERPOLATION*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Avg Distance:** Average distance between the two edges found.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Image Preview:** (Optional) Image in which the set edges are drawn.
- **Image Threshold:** (Optional) Image to which the found threshold is applied.
- **Edge Points 1:** (Optional) Displays the points found on the edge near the start point.
- **Edge Point 2:** (Optional) Outputs the points found on the edge near the end point.
- **Distances:** (Optional) Displays the distances between the points found.

5.2.3.31 Median filter



Applies a median filter to an image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Filter Size:** Size of the filter in width and height (3, 5, 7, ...).

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.32 Morphological Operator

Morphological Operator					
Image	<Empty>	image in module watc...	SE Shape	TCVN_SES_RECTANGLE	Count Results
Enable Execution	<Empty>	-	SE Width	15	hr
			SE Height	15	Image
			Operator	TCVN_MO_OPENING	New Result
					Empty

Creates a tree item and applies it to the input image using a morphological operator.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **SE Shape:** Shape of the tree item.
- **SE Width:** Width of the tree item.
- **SE Height:** Height of the tree item.
- **Morphological operator.**

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.33 Normalize Image

Normalize Image					
Image	<Empty>	image in module watc...	Alpha	0	Count Results
Enable Execution	<Empty>	-	Beta	255	hr
			Norm Type	TCVN_NT_MINMAX	Image
			Dest Type	TCVN_ET_SAME_AS_SOURCE	New Result
					Empty

Normalizes an image with regard to its value range, for example by spreading the pixel values [50..150] to the full range [0..255].

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Alpha:** (Optional) Lower range limit for range normalization or target value for normalization.
- **Beta:** (Optional) Upper range limit when normalizing the value range.
- **Norm Type:** (Optional) Normalization type, only INF, L1, L2 or MINMAX.
- **Dest Type:** (Optional) Color depth of the target image. Usually SAME_AS_SOURCE.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.34 OCR

OCR					
Image	<Empty>	image in module watc...	Model	TCVN_OMT_NUMBERS	Count Results
Enable Execution	<Empty>	-	Pattern		hr
					Code
					Last Event
					Image
					New Result
					Empty

Recognizes and identifies characters in a binary image. White characters on a black background are supported.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Model:** Choice of OCR model.
- **Pattern:** Character template with the format in which characters are displayed.
 “!” ignores characters, but not spaces.
 “. ” permits any characters, also without spaces.
 “_ ” stands for a space.
 “d” expects a digit from 0 to 9.
 “#” expects one of the special characters (. / - : =).
 “u” expects a capital letter from A to Z.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Code:** Identified characters as a string.
- **Last Event:** Timestamp of the last result.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.35 Put Label

Write text on a uniform background in the picture.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **X:** X-coordinate of the bottom left corner. If Position has an input, this value is used as an additional offset.
- **Y:** Y-coordinate of the bottom left corner. If Position has an input, this value is used as an additional offset.
- **Font Type:** Font type.
- **Font Size:** Font size.
- **Font Color:** Font color.
- **Background Color:** Background color.
- **Thickness:** Line width.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.36 Put Text

Put Text				
Image	<Empty>	image in module watc...	X	0
Text	<Empty>	-	Y	33
Enable Execution	<Empty>	-	Font Type	TCVN_FT_HERSHEY_SIMPLEX
Position	<Empty>	-	Font Scale	1.5
			Font Color	Black
			Thickness	1

Count Results	Empty
hr	Empty
Image	image in module watcher
New Result	Empty

Write text in the picture.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **X:** X-coordinate of the bottom left corner. If Position has an input, this value is used as an additional offset.
- **Y:** Y-coordinate of the bottom left corner. If Position has an input, this value is used as an additional offset.
- **Font Type:** Font type.
- **Font Size:** Font size.
- **Font Color:** Font color.
- **Thickness:** Line width.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.37 Pyramid Down

Pyramid Down			
Image	<Empty>	image in module watc...	
Enable Execution	<Empty>	-	

Count Results	Empty
hr	Empty
Image	image in module watcher
New Result	Empty

Scales an image down to half its width and height.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.38 Pyramid Up

Pyramid Up			
Image	<Empty>	image in module watc...	
Enable Execution	<Empty>	-	

Count Results	Empty
hr	Empty
Image	image in module watcher
New Result	Empty

Scales an image up to twice its width and height.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.

- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.39 Read Barcode

Read Barcode			
Image	<Empty>	Barcode Type	TCVN_BT_EAN13
Enable Execution	<Empty>	Search Direction	TCVN_BSD_ANY
		Count Results	Empty
		hr	Empty
		Code	Empty
		Last Event	Empty
		Image	image in module watcher
		New Result	Empty
		Image Preview	image in module watcher
		Contours	Empty

Recognizes and interprets a 1D barcode in the image provided.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Barcode type:** Barcode types to be searched for.
- **Search Direction:** Search direction for barcodes.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Code:** Identified characters as a string.
- **Last Event:** Timestamp of the last result.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Image Preview:** (Optional) Image in which the code found is displayed.
- **Contours:** (Optional) Contours of the code found.

5.2.3.40 Read Data Matrix Code

Read Data Matrix Code			
Image	<Empty>	Search Strategy	TCVN_CSS_DEFAULT
Enable Execution	<Empty>		
		Count Results	Empty
		hr	Empty
		Code	Empty
		Last Event	Empty
		Image	image in module watcher
		New Result	Empty
		Image Preview	image in module watcher
		Contours	Empty

Recognizes and interprets a Data Matrix code (ECC200) in the image provided.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Search Strategy:** (Optional) Search strategy used.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Code:** Identified characters as a string.
- **Last Event:** Timestamp of the last result.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

- **Image Preview:** (Optional) Image in which the code found is displayed.
- **Contours:** (Optional) Contours of the code found.

5.2.3.41 Read Pharma Code

Read Pharma Code			
Image	<Empty>	image in module watc...	Min Bar Number 4
Enable Execution	<Empty>	-	
Count Results Empty hr Empty Code Empty Last Event Empty Image image in module watcher New Result Empty Image Preview image in module watcher Contours Empty			

Recognizes and interprets a pharma code in the image provided.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Min Bar Number:** (Optional) Minimum number of dark bars that codes must have.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Code:** Identified characters as a string.
- **Last Event:** Timestamp of the last result.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Image Preview:** (Optional) Image in which the code found is displayed.
- **Contours:** (Optional) Contours of the code found.

5.2.3.42 Read QR Code

Read QR Code			
Image	<Empty>	image in module watc...	Search Strategy TCVN_CSS_DEFAULT
Enable Execution	<Empty>	-	
Count Results Empty hr Empty Code Empty Last Event Empty Image image in module watcher New Result Empty Image Preview image in module watcher Contours Empty			

Recognizes and interprets a QR code in the image provided.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Search Strategy:** (Optional) Search strategy used.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Code:** Identified characters as a string.
- **Last Event:** Timestamp of the last result.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.
- **Image Preview:** (Optional) Image in which the code found is displayed.
- **Contours:** (Optional) Contours of the code found.

5.2.3.43 Reference Color Similarity

Reference Color Similarity			
Image	<Empty>	image in module watc...	Reference Color
Enable Execution	<Empty>	-	Variance
			Luminance Weight
			Count Results
			hr
			Image
			New Result

Calculates the similarity of each pixel to a reference color and returns the result as a grayscale image. Objects in the reference colors are then detected on the basis of these grayscale images.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Reference Color:** Todo
- **Variance:** (Optional) Approved color variance (0.1 to 0.3 is often a good starting value).
- **Luminance Weight:** (Optional) Weights the influence of luminance ([0..1], for example 0 for higher robustness against uneven illumination. However, weighting may be required to distinguish some colors.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.44 Reset ROI

Reset ROI			
Image	<Empty>	image in module watc...	Count Results
Enable Execution	<Empty>	-	hr
			Image
			New Result

Resets the region of interest (ROI) of an image. After this operation, the ROI covers the entire image again.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.45 Rotate Image

Rotate Image			
Image	<Empty>	image in module watc...	Angle in deg
Enable Execution	<Empty>	-	Adjust dest size
			Interpolation Type
			Border Interpolation
			Border Value
			Count Results
			hr
			Image
			New Result

Rotates an image clockwise by the specified angle.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Angle in deg:** (Optional) Angle of rotation in degrees (positive: counterclockwise, negative: clockwise).
- **Adjust dest size:** (Optional) If TRUE, the size of the output image is adjusted so that the fully rotated input image is included. Otherwise, the output image is set to the size of the input image.
- **Interpolation Type:** (Optional) Interpolation method.

- **Border Interpolation:** (Optional) Interpolation method at the border (ISOLATED is not supported).
- **Border Value:** (Optional) Border value if CONSTANT is used.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.46 Set ROI

Set ROI			
Image	<Empty>	image in module watc...	X
Enable Execution	<Empty>	-	Y
			Width
			Height
			Auto Revert ROI
			Count Results
			hr
			Image
			New Result

Sets a region of interest (ROI) within an image. This can optionally be reset.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **X:** X-coordinate of the top left corner.
- **Y:** Y-coordinate of the top left corner.
- **Width:** Width of the ROI.
- **Height:** Height of the ROI.
- **Auto Revert ROI:** Automatically resets the ROI at the end of the network.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.47 Subtract Images

Subtract Images			
Image 1	<Empty>	image in module watc...	
Image 2	<Empty>	image in module watc...	
Enable Execution	<Empty>	-	
			Count Results
			hr
			Image
			New Result

Item-by-item subtraction of two images with saturation arithmetic.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.48 Threshold

Threshold			
Image	<Empty>	image in module watc...	Threshold
Enable Execution	<Empty>	-	Max Value
			Threshold Type
			TCVN TT_BINARY
			Count Results
			hr
			Pixels above Thres...
			Image
			New Result

Applies a binary threshold operation and counts pixels.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Threshold:** Fixed threshold for conversion to a binary image.
- **Max Value:** Maximum value.
- **Threshold Type:** Type of threshold value.

Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Pixels above Threshold:** Number of pixels with value greater than *Threshold*
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.2.3.49 Variance filter

Variance Filter			
Image	<Empty>	image in module watc...	Kernel Size
Enable Execution	<Empty>	-	Scale
			Border Type
			TCVN BIM_REFLECT_101
			Count Results
			hr
			Image
			New Result

Calculates the local variance of an image.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Kernel Size:** Size of the filter kernel.
- **Scale:** Scaling factor for the calculated variance values.
- **Border Type:** Treatment of the image border.

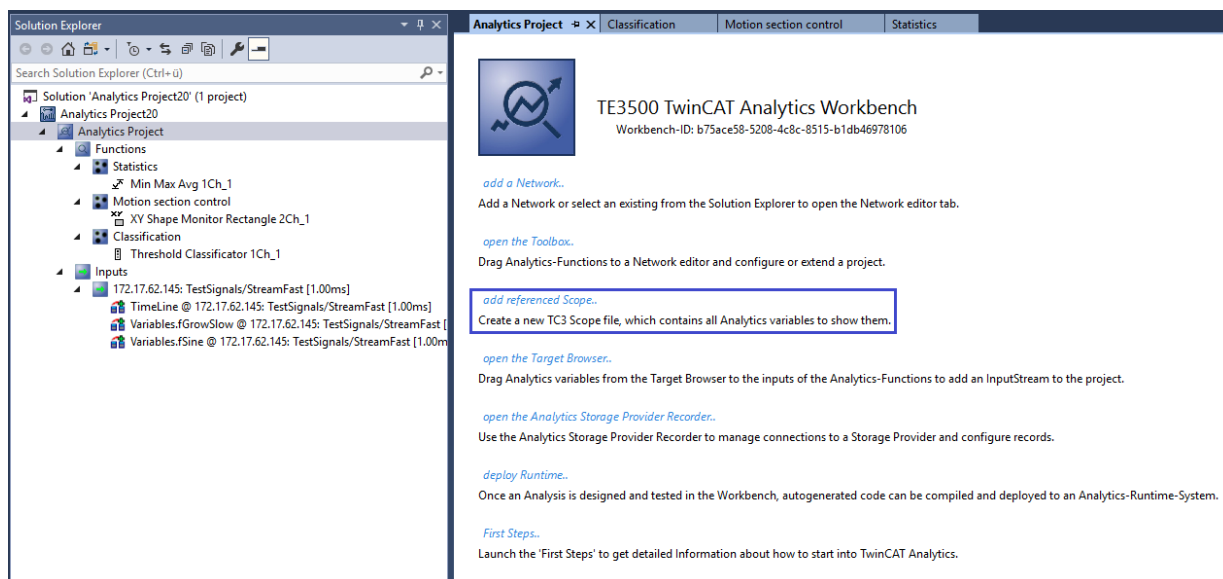
Output values

- **Count Results:** Is incremented when a new result is provided.
- **hr:** HRESULT error code that outputs the result of the processing.
- **Image:** Initial image of the algorithm.
- **New Result:** (Optional) TRUE if a new result is provided.

5.3 Interaction with Scope

The TwinCAT Analytics engineering tools offer easy interaction between the Analytics Configurator and Scope View. They allow you to highlight significant values in the data stream and examine them with other process data within the exact cycle. In addition, it is possible to display the algorithm's result data, such as an average or maximum value, in the Scope View.

- ✓ After configuring the analysis, switch to the **Analytics Project** start page.

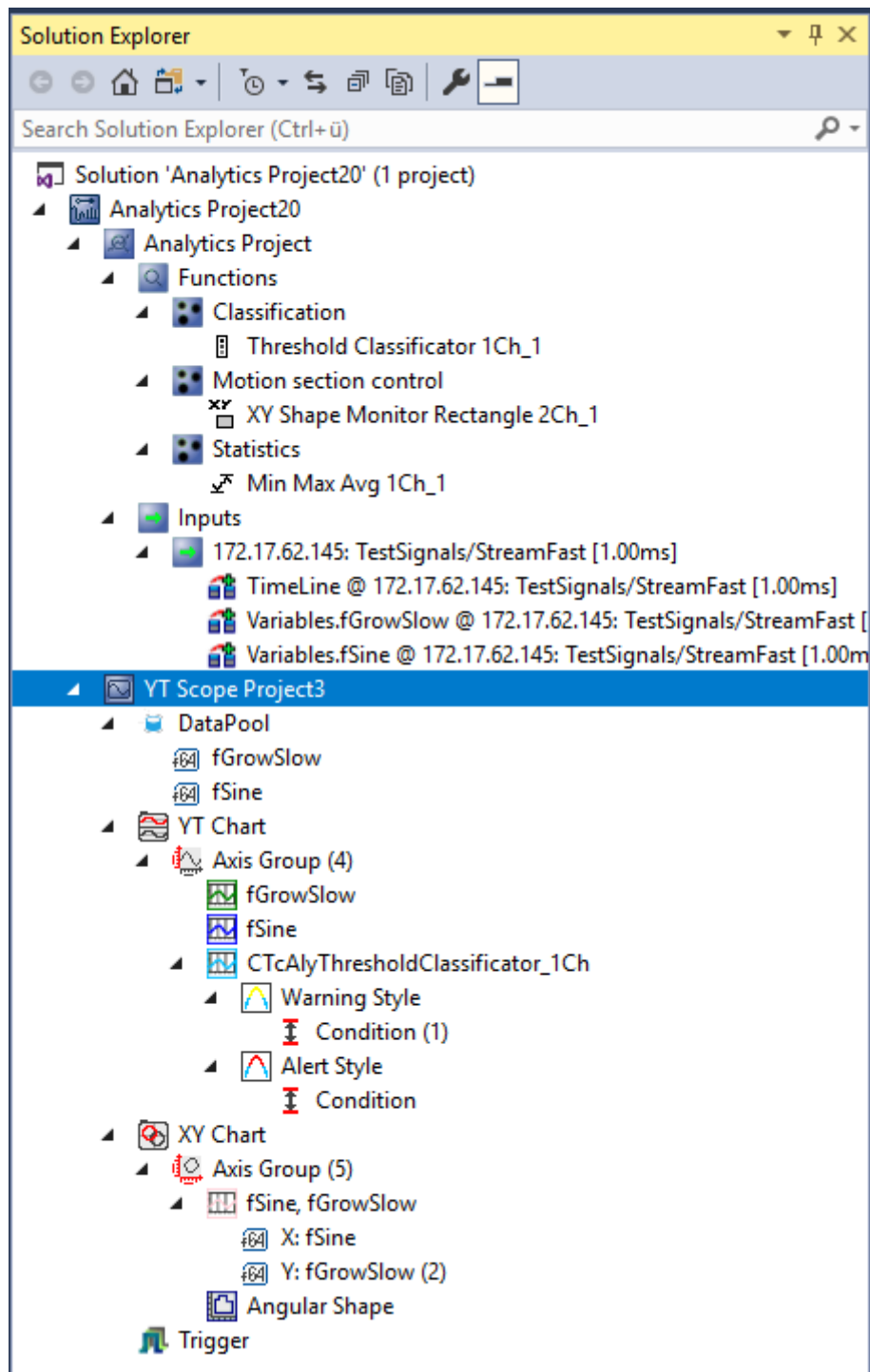
1. Click **add referenced Scope...**


The screenshot displays the TwinCAT Analytics Workbench interface. On the left, the **Solution Explorer** shows a project structure under 'Analytics Project20'. The 'Analytics Project' folder is expanded, revealing subfolders: 'Functions' (containing 'Statistics' with 'Min Max Avg 1Ch_1', 'Motion section control', and 'XY Shape Monitor Rectangle 2Ch_1'), 'Classification' (containing 'Threshold Classifier 1Ch_1'), and 'Inputs' (containing '172.17.62.145: TestSignals/StreamFast [1.00ms]', 'TimeLine @ 172.17.62.145: TestSignals/StreamFast [1.00ms]', 'Variables.fGrowSlow @ 172.17.62.145: TestSignals/StreamFast [1.00ms]', and 'Variables.fSine @ 172.17.62.145: TestSignals/StreamFast [1.00ms]').

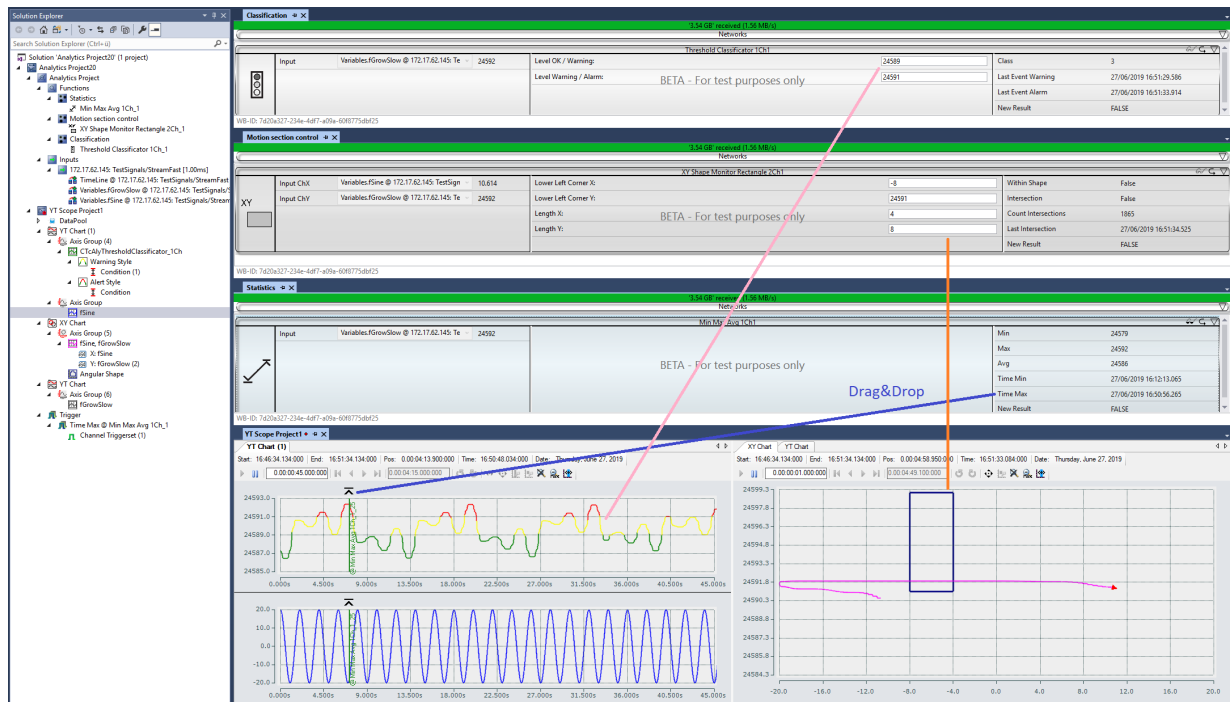
The main workspace on the right is titled 'TE3500 TwinCAT Analytics Workbench' with a Workbench-ID of 'b75ace58-5208-4c8c-8515-b1db46978106'. It features a toolbar with icons for 'add a Network...', 'open the Toolbox...', 'add referenced Scope...', 'open the Target Browser...', 'open the Analytics Storage Provider Recorder...', 'deploy Runtime...', and 'First Steps...'. The 'add referenced Scope...' option is highlighted with a red box. Below the toolbar, instructions are provided for each function:

- add a Network..**: Add a Network or select an existing from the Solution Explorer to open the Network editor tab.
- open the Toolbox..**: Drag Analytics-Functions to a Network editor and configure or extend a project.
- add referenced Scope..**: Create a new TC3 Scope file, which contains all Analytics variables to show them.
- open the Target Browser..**: Drag Analytics variables from the Target Browser to the inputs of the Analytics-Functions to add an InputStream to the project.
- open the Analytics Storage Provider Recorder..**: Use the Analytics Storage Provider Recorder to manage connections to a Storage Provider and configure records.
- deploy Runtime..**: Once an Analysis is designed and tested in the Workbench, autogenerated code can be compiled and deployed to an Analytics-Runtime-System.
- First Steps..**: Launch the 'First Steps' to get detailed Information about how to start into TwinCAT Analytics.

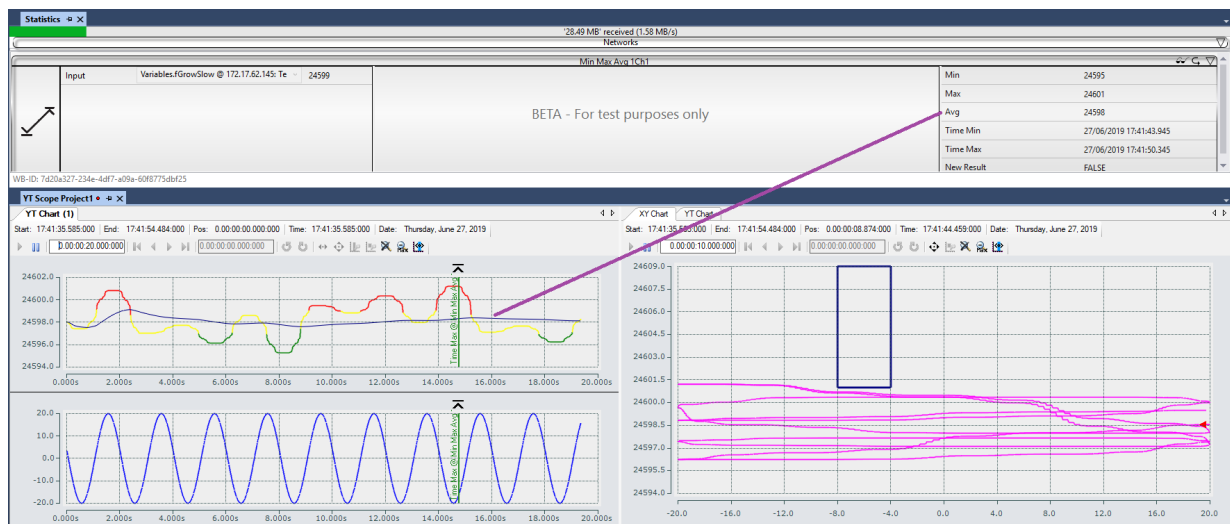
⇒ An appropriate Scope configuration is automatically added to the project.



2. There are numerous possibilities to visualize the data from the various algorithms. Any timestamps of an algorithm output can be dragged and dropped into the graphs. The Scope marks the position of the event with a colored marker (see blue line in the image below).



3. For each type of Shape algorithm, the SW creates an XY diagram in the Scope configuration including a shape with the given definition (see orange line in the image above). For the Threshold classification algorithm, a dynamic color change from channel color to yellow (warning) and to red (alarm) (pink line) also occurs automatically in the Scope View.
4. In addition, you can drag and drop the algorithm's result values into a graph before or during recording to add a new channel showing, for example, the average, minimum, or maximum value.



5. Click the appropriate trigger group in the Scope View tree to control the number of events/markers displayed.

6. In the Properties window you can set a number for Visible Trigger Release Capacity.

The screenshot displays the Beckhoff TwinCAT software interface. The Solution Explorer on the left shows a project structure for 'YT Scope Project1'. The Properties window on the right is open for the 'Last Intersection @ XY Shape Monitor Rectangle 2Ch_1' object. The 'Trigger Group' section is expanded, showing various properties. The 'Visible Trigger Release Capacity' property is highlighted, and its dropdown menu is open, showing options: 'Show All', 'Hide all', 'Show Last', 'Show Last 2', 'Show Last 5' (selected), 'Show Last 10', 'Show Last 20', and 'Show Last 50'.

Solution Explorer:

- YT Scope Project1
 - DataPool
 - YT Chart (1)
 - Axis Group (4)
 - CTcAlyThresholdClassifier_1Ch
 - Warning Style
 - Condition (1)
 - Alert Style
 - Condition
 - Avg @ Min Max Avg 1Ch_1 (1)
 - Axis Group
 - fSine
 - XY Chart
 - Axis Group (5)
 - fSine, fGrowSlow
 - X: fSine
 - Y: fGrowSlow (2)
 - Max @ Min Max Avg 1Ch_1, Avg @ Min Max Avg 1Ch_1
 - Y: Avg @ Min Max Avg 1Ch_1 (2)
 - Y: Max @ Min Max Avg 1Ch_1 (1)
 - Angular Shape
 - YT Chart
 - Axis Group (6)
 - fGrowSlow
 - Trigger
 - Last Intersection @ XY Shape Monitor Rectangle 2Ch_1
 - Channel Triggerset (2)
 - Time Max @ Min Max Avg 1Ch_1
 - Channel Triggerset (1)

Properties Window:

Last Intersection @ XY Shape Monitor Rectangle 2Ch_1 TwinCAT.Measurement.B

Common

Name	Last Intersection @ XY Shape Monitor Rec
------	--

Trigger Action

Trigger Action	Set Mark
----------------	----------

Trigger Group

Color	27, 183, 0
Enabled	True
Trigger Category	CustomPicture
Trigger Category Image	.\Images\5efb9f84-42f0-4d33-9f37-690b...
Trigger Image Size	16
Trigger Release Capacity	200
Visible	True
Visible Trigger Release Capacity	Show Last 5

Visible Trigger Release Capacity Options:

- Show All
- Hide all
- Show Last
- Show Last 2
- Show Last 5
- Show Last 10
- Show Last 20
- Show Last 50

⇒ You can choose between:

- All
- Hide All
- Show Last
- Show Last 2
- Show Last 5
- Show Last 10
- Show Last 20
- Show Last 50

Scope configuration stored in network template

A created Scope configuration can be saved together with the associated network in a [network template](#) [► 298], in order to automatically obtain the same Scope configuration when a network is used again.



Time relationship between Analytics configuration and Scope View

Note that the recording time in Scope View may differ from that in Analytics. Especially if the ring buffer is selected in the Scope View, it could be possible that some significant values of your analysis are in a past Scope recording!

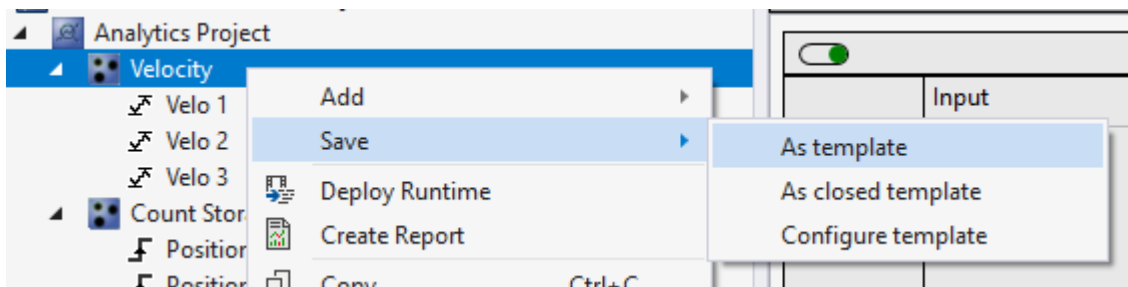
5.3.1 Scope configuration in the network template

In addition to the configuration of a network, the configuration of the connected Scope project can also be saved in a network template. However, only those parts of the Scope can be saved that have a connection to the selected network.

Depending on whether the Scope configuration to be saved is still to be adjusted manually, the template can be saved in two ways.

Save without manual configuration

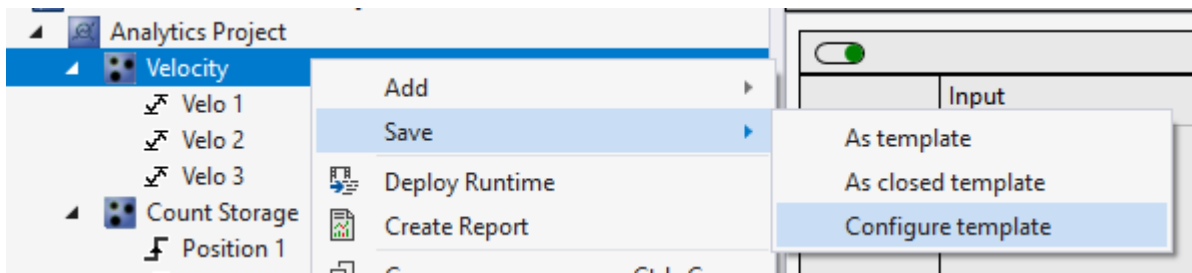
If you want to save the Scope configuration in the template without any manual adjustment, select the context menu items **Save-As template** and **Save-As closed template**.



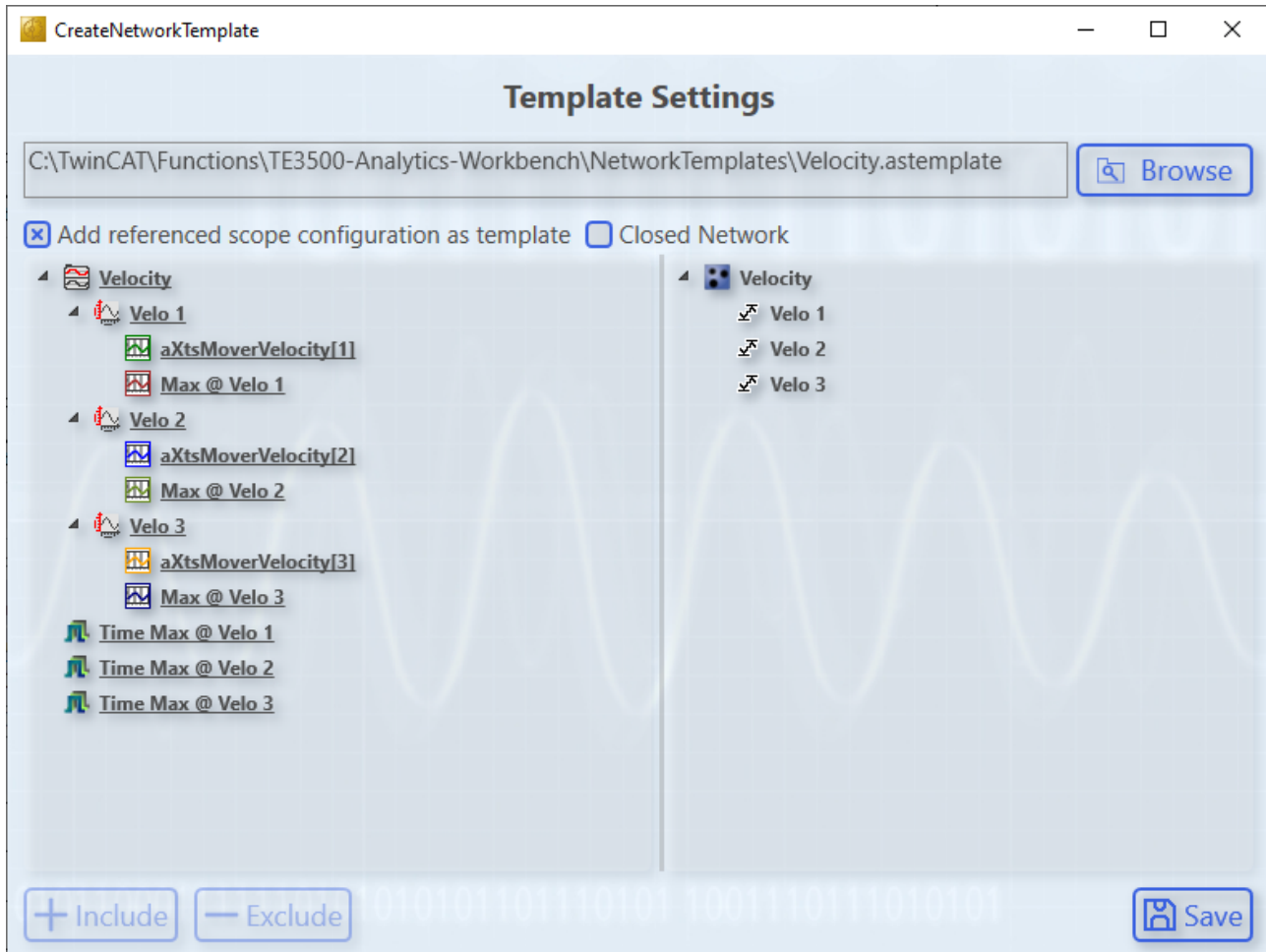
In the following dialog, specify the location and name of the template. At this point you can use the file type **Save as type** to specify whether the Scope configuration should also be saved (**Scope Network Template**) or whether only the network (**Network Template**) should be saved.

Save with manual configuration

If the Scope configuration is to be checked and, if necessary, adjusted before saving, select the lowest entry in the context menu **Configure template**.



To control and adjust the Scope configuration, a new window opens, showing all Scope components on the left and all Analytics components on the right:

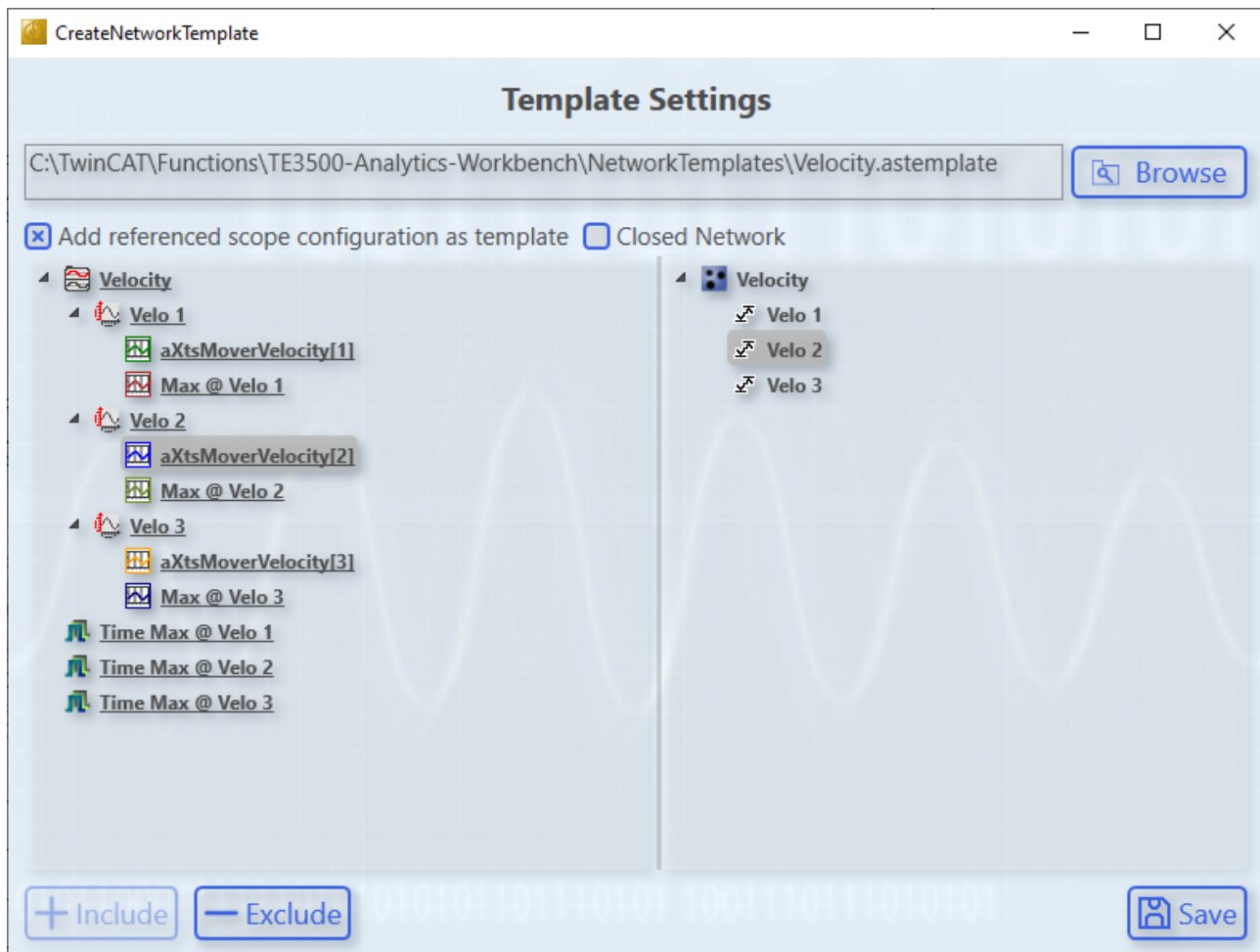


In the upper line you can set the location and name of the template. Directly below you can make general settings that affect the template.

- **Add referenced scope configuration as template**
Activate this setting to save the scope configuration in the template.
- **Closed Network**
This setting determines in which format the template [► 13] should be saved. This setting is only visible if at least one network input and one network output are configured in the network.

In the two tree views below, the configurations are displayed in order to still edit the Scope configuration if necessary. Here you can exclude Scope components from the template but also add them again. All components which are added to the template are underlined in the view.

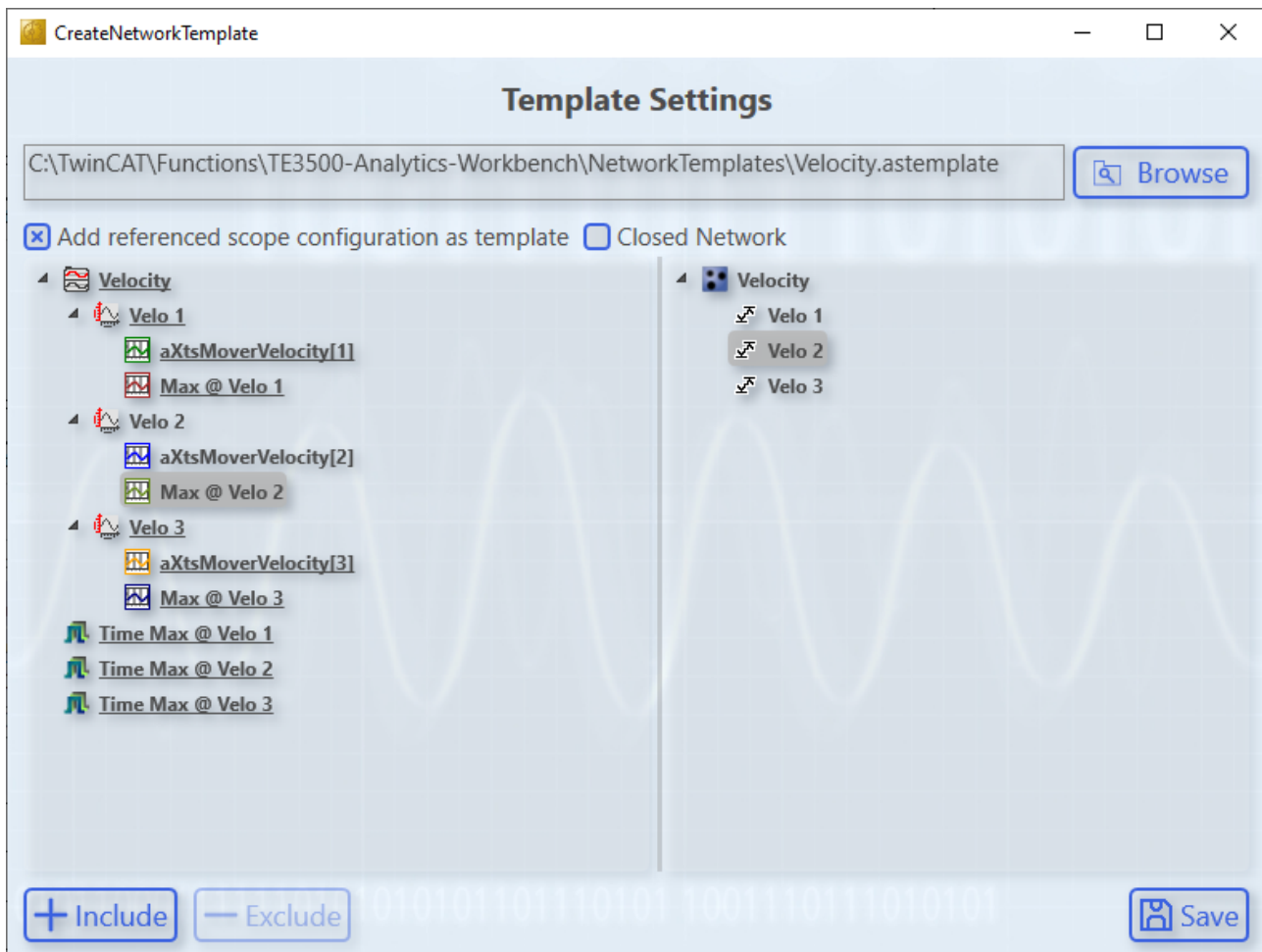
When elements are selected in the Scope tree, all Analytics components that have a direct dependency on the Scope component are also selected.



In the example above, selecting **aXtsMoverVelocity[2]** automatically selects the module **Velo 2** because the channel indicates the module's input variable.

If a Scope component is selected that is still underlined and thus stored in the template, you can use the **Exclude** button to remove it from the template.

If you remove an element from the template, all elements that lie below it are also automatically removed. In addition, the element above is taken out if every element below it is also taken out.



Identical to removing elements, you can select Scope components to add them back to the template using the button **Include**.

It should be noted that all overlying elements are also automatically added back to the template, even if they were previously deselected.

Adding Scope configurations from the template

To use a template with the Scope configuration, you need to add the template to the Analytics project. If there is already a referenced Scope project for the project, the template configuration is added directly there. If no referenced Scope project exists yet, the template configuration will be added when the Scope project is created later.

Before the configuration is added to the project, a query opens where you can also cancel the automatic addition.

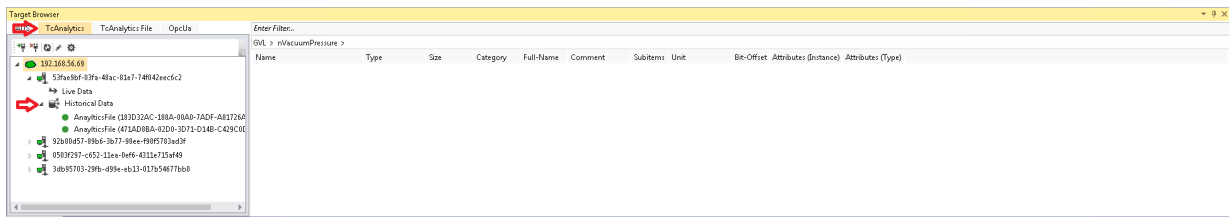
5.4 Working with Historical Data

Historical Data can be analysed with the Analytics Workbench or the Analytics Service Tool. To see your recorded data, you need the TwinCAT Target Browser.

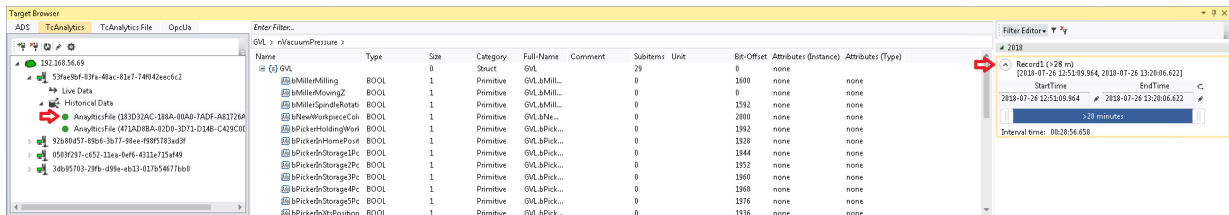
Selection of data from the TwinCAT Target Browser

The historical data can be pulled directly from the Target Browser to an input of an analysis algorithm.

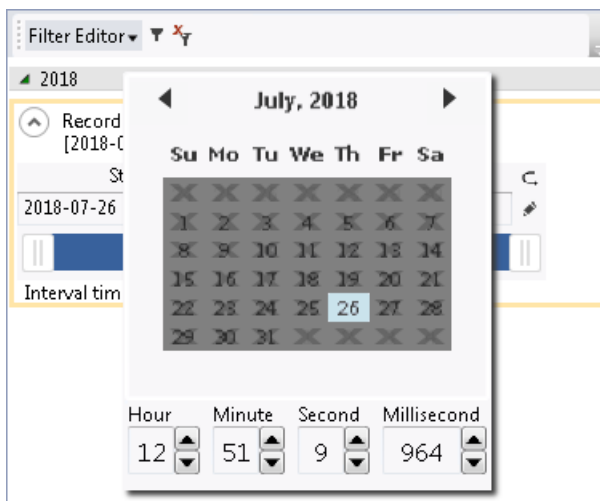
1. First, you need to click **TcAnalytics** in the left corner of Target Browser. There you can see your configured broker, which lists live and historical data from your various devices. This should look like the following figure.



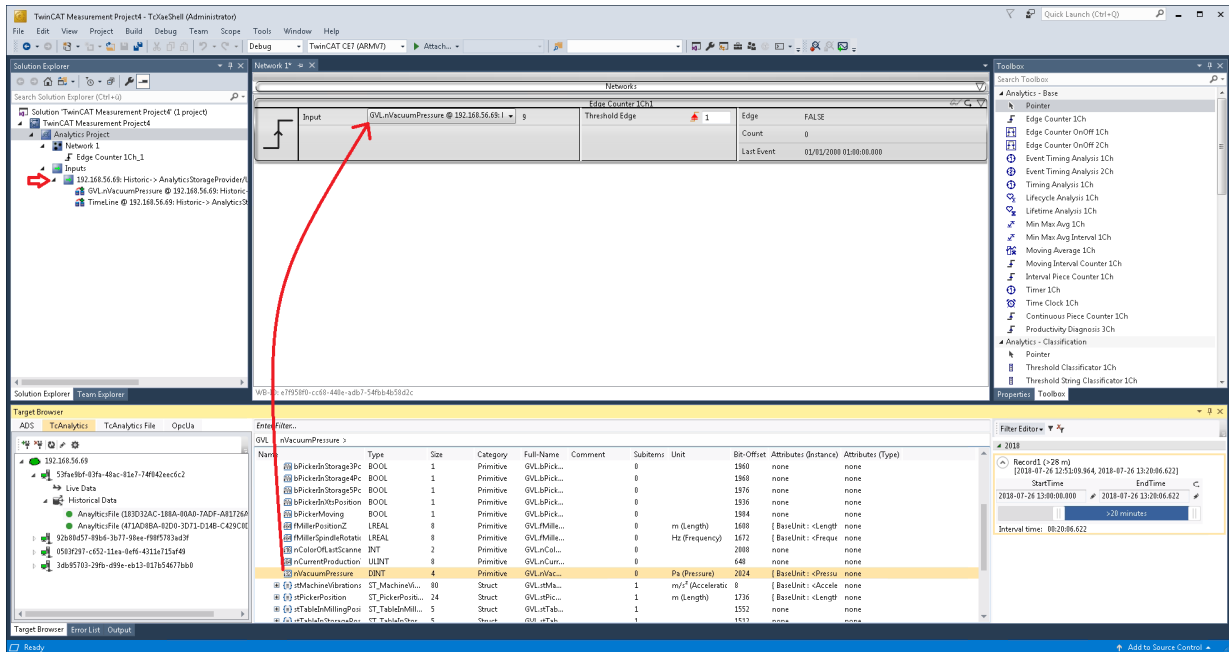
2. Go to the historical stream you created and select the recording to be analyzed. All your records are listed in the **Record** window on the right. The last recording is selected by default.



3. When you record live, the time range of the recording is updated every few seconds. The entire time range of a recording is used by default. You can also edit the start and end time to analyze your desired data area. This can be done with a slider, text fields or in a graphical calendar view. If you click on the symbol to the right of the text fields, the calendar view will be displayed.



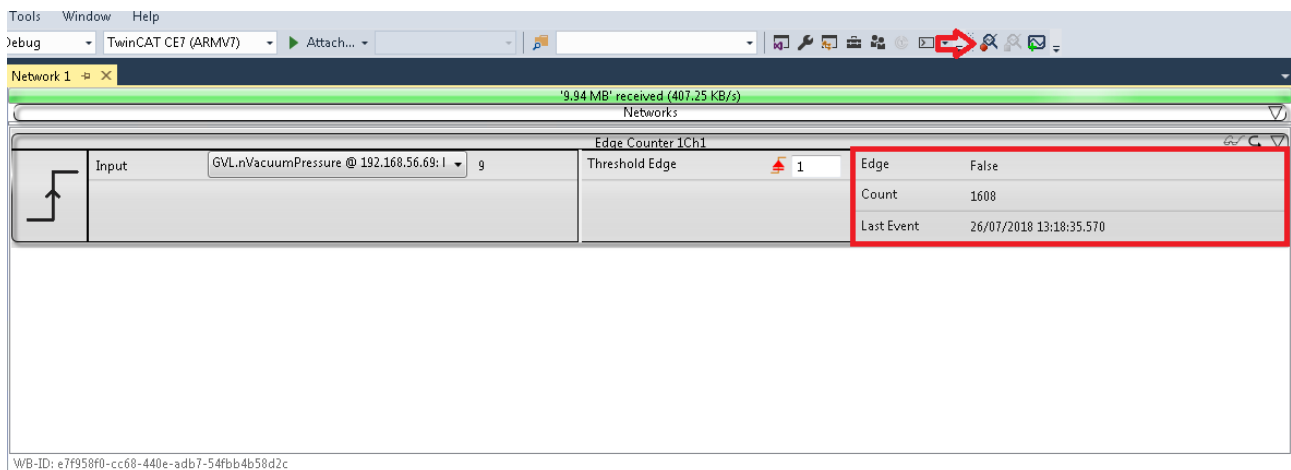
- After these steps, you can drag and drop a symbol to an input of an algorithm just as you do with the symbols of the live data.



- ⇒ A new input source for your historical stream is then generated and can be displayed in the Solution Explorer of your Visual Studio. First, the dragged symbol and a timestamp of the current device time are listed under this stream. Also new drawn symbols of this stream are listed there.

Analyse your historical data in the Analytics Configurator

To analyse your historical data press on the Start Analytics button. In contrast to analysing live data, a green progress bar appears. The speed of your analysis depends on your record length, the amount and size of your symbols as well as on your broadband speed to the broker. The analysis stops automatically when the progress bar ends. The results will remain visible.



5.5 Runtime deployment

PLC code can be generated with all modules and parameters configured in the TwinCAT Analytics Workbench Configurator. This code can be downloaded into a TwinCAT Analytics Runtime to realize 24/7 data analysis.

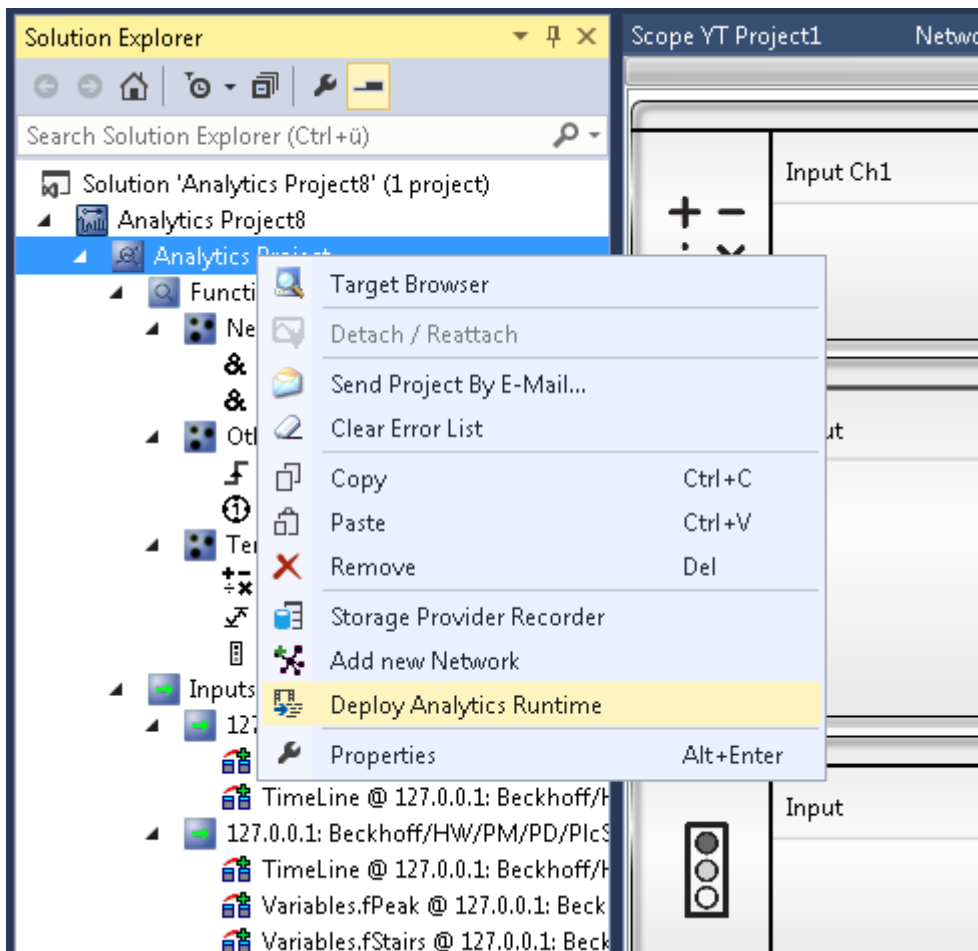
NOTICE**Compatibility of automatically generated PLC code**

The automatically generated PLC code is based on the TwinCAT Analytics Library. Interfaces of the base models of the library are compatible with earlier versions after release of the library. The automatically generated code itself is only a sample code! This code generation may change from version to version. As far as possible, this is solved by the code generation version.

NOTICE**TwinCAT Version (4026.20 or later) recommended**

It is recommended to work with a current TwinCAT version. Especially when using the **Usermode Runtime**, both the Usermode Runtime and TwinCAT should be at least **version 4026.20**.

- ✓ After configuration, click on the **Deploy Analytics Runtime** command in the context menu.



- ✓ The Deploy Wizard starts and it is possible to set up the entire required configuration step by step for use.

1. On the first tab **Solution**, select whether a new Solution is to be created (**Create new Solution**), whether the analysis PLC code is to be integrated or added to an existing Solution as a new project, new PLC or to an existing PLC (**Add to existing Solution**), or whether the new project is to be added to an existing solution using the TwinCAT Project Compare Tool (**Merge to existing Project**).

The screenshot shows the 'Deploy Analytics Runtime' dialog box with the 'Solution' tab selected. The 'Codegeneration' dropdown is set to 'Latest Version ("Tc3_Analytics >= V3.1.0.4")'. The 'Create new Solution' radio button is selected. The 'Solution Path' is 'C:\Users\Beckhoff\Documents', 'Solution Name' is 'AnalyticsSolution', 'Project Name' is 'AnalyticsProject', and 'PLC Name' is 'AnalyticsPLC'. The 'Add to existing Solution' and 'Merge to existing Project' options are not selected. At the bottom, there are 'Back', 'Cancel', 'Next', and 'Deploy' buttons.

2. Select several data source configurations on the second tab **Input Source**. This is necessary, for example, if both live data and historical data are to be analyzed.

The screenshot shows the 'Deploy Analytics Runtime' dialog box with the 'Input Source' tab selected. The 'Codegeneration' dropdown is set to 'Latest Version ("Tc3_Analytics >= V3.1.0.4")'. The 'Sources' tree on the left shows a hierarchy: 'Sources' > 'Building A' > 'messagebroker2.beckhoff-cloud.com: EnergyData/Buildin' > 'Default'. The 'Active Configuration (C1)' section shows 'Display Name: Live Data' and two dropdowns for 'Building A' and 'Building B', both set to '[Default] messagebroker2.beckhoff-cloud.com:'. The 'Configuration (C2)' section shows 'Display Name: Historical Data' and two dropdowns for 'Building A' and 'Building B', both set to '[Default] messagebroker2.beckhoff-cloud.com:'. At the bottom, there are 'Back', 'Cancel', 'Next', and 'Deploy' buttons.

3. On the third tab **TwinCAT PLC Target** PLC-specific parameters such as target system, task cycle time or task assignment to corresponding CPU cores can be specified.

Deploy Analytics Runtime

Codegeneration: Latest Version ("Tc3_Analytics" >= V3.1.0.4)

Solution | Input Source | **TwinCAT PLC Target** | Results | HMI Dashboard | IDE | Summary

Target

Target System: <Local> (172.17.60.160.1.1) ☐ Create Boot-Project

ADS Port: 851 ☐ Enable UTF8 Encoding ☒ Activate PLC Runtime

Realtime

Available CPU cores: Shared: 12 Isolated: 0 Router Memory Configured Size [MB]: 256

Tasks

Name	Cycle time (ms)	CPU Core
AnalyticsTask	10	Core1 (Shared)
AnalyticsTask_C1	10	Core1 (Shared)
AnalyticsTask_C2	10	Core1 (Shared)

☒ Overwrite Task Config Maximal Stack Size [KB]: 512 KB

Back Cancel Next Deploy

4. If you have set the **PLC Result** property of some functions in the configurator, the **Result** tab is enabled. Specify where the results are streamed or saved.

Deploy Analytics Runtime

Codegeneration: Latest Version ("Tc3_Analytics" >= V3.1.0.4)

Solution | Input Source | TwinCAT PLC Target | **Results** | HMI Dashboard | IDE | Summary

Result Type

☒ Create no Results (NO MODULES SELECTED FOR RESULTSTREAM)

☐ Map Results to ADS Struct

☐ Stream Results to MQTT Broker

Topic: Analytics/Analysis/ResultStream

MQTT Connection Settings

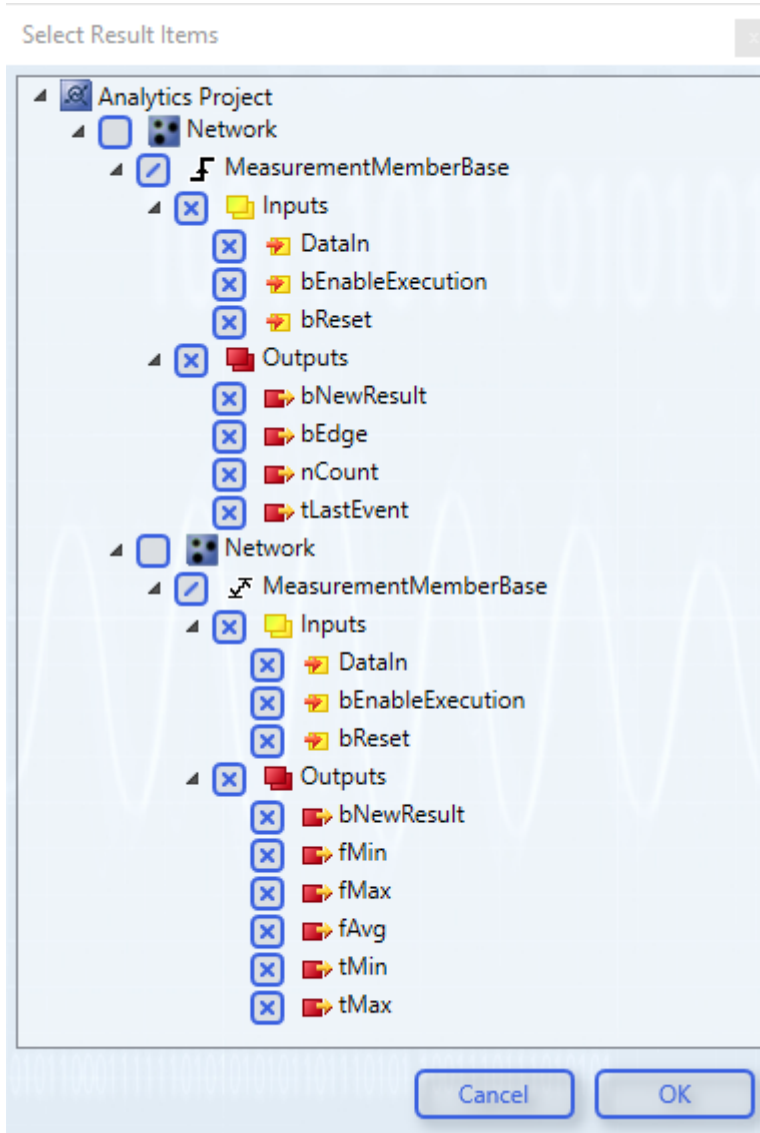
☐ Write Results to Analytics File

File Path:

Max File Size: 256 Sample buffer count

Back Cancel Next Deploy

5. By clicking **Select Result Items** it is possible to select only the desired values.



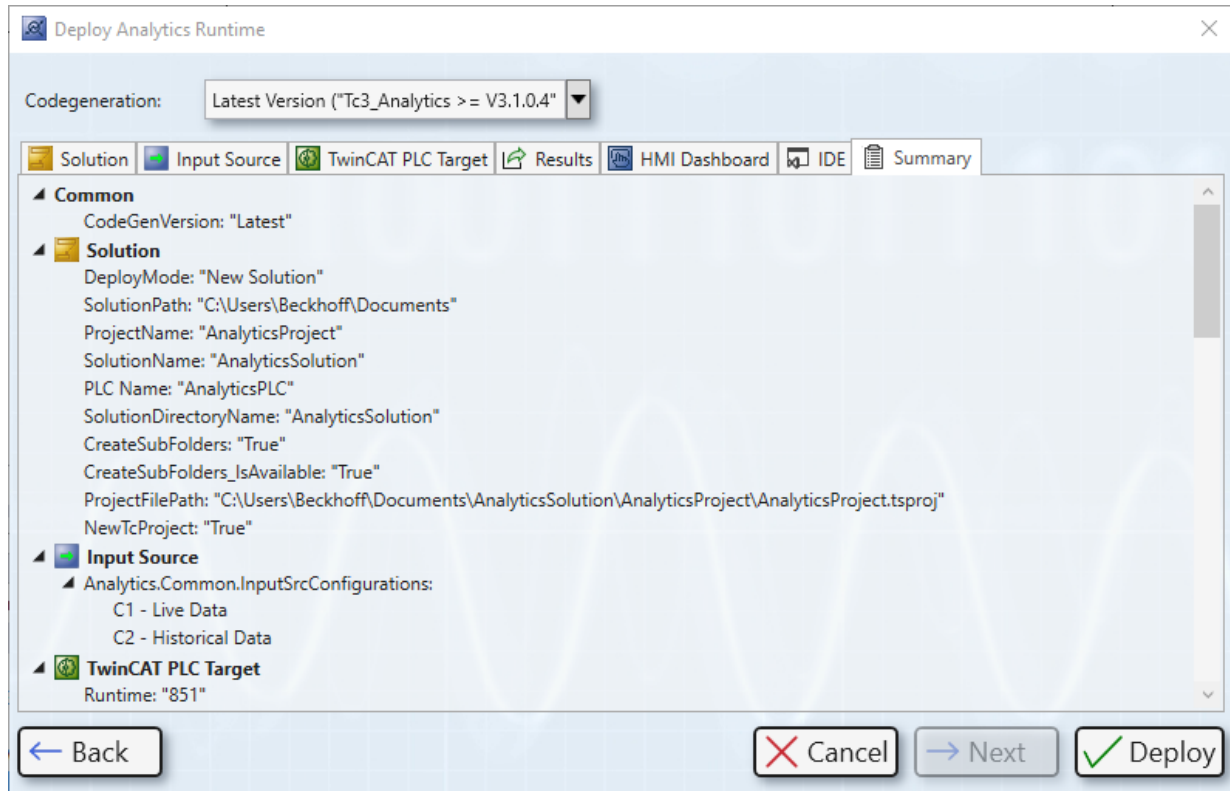
6. If you have assigned HMI Controls to one or more functions, the **HMI Dashboard** tab is enabled. Here you can make various settings to generate a customized dashboard that suits your needs.

The screenshot shows the 'Deploy Analytics Runtime' dialog box with the 'HMI Dashboard' tab selected. The 'Codegeneration' dropdown is set to 'Latest Version ("Tc3_Analytics" >= V3.1.0.4)'. The 'HMI generated Settings' section includes a checked 'Generate HMI Dashboard' checkbox and an unchecked 'Create only HMI Project (no PLC)' checkbox. The 'HMI Project Name' is 'AnalyticsHMIProject' and the 'Dashboard Version' is 'Latest (Version 2.0)'. The 'Dashboard Options' section shows 'Dashboard Title' as 'Analytics Dashboard', 'Desktop Width' as 1920, 'Desktop Height' as 864, and both 'Create Start-Page' and 'Show current time' are checked. The 'Dashboard Styles' section shows 'Dashboard Layout' as 'Align Center' and 'Dashboard Sorting' as 'Control Type'. At the bottom are 'Back', 'Cancel', 'Next', and 'Deploy' buttons.

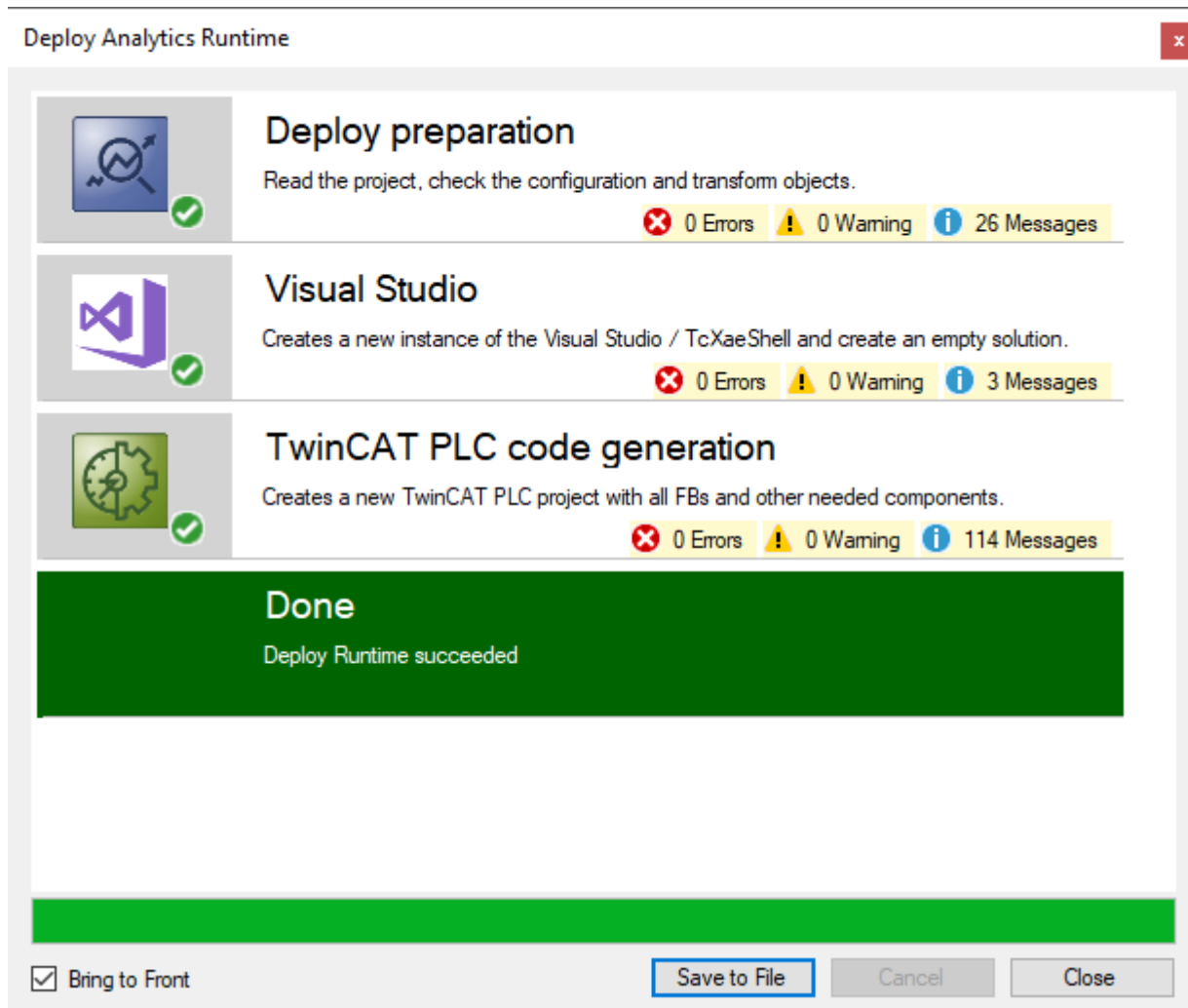
7. On the **IDE** tab, select which Visual Studio version or TwinCAT XAE Shell is to be used for the generation if several are installed.

The screenshot shows the 'Deploy Analytics Runtime' dialog box with the 'IDE' tab selected. The 'Codegeneration' dropdown is set to 'Latest Version ("Tc3_Analytics" >= V3.1.0.4)'. The 'Common' section shows 'CodeGenVersion: "Latest"'. The 'Solution' section shows 'DeployMode: "New Solution"', 'SolutionPath: "C:\Users\Beckhoff\Documents"', 'ProjectName: "AnalyticsProject"', 'SolutionName: "AnalyticsSolution"', 'PLC Name: "AnalyticsPLC"', 'SolutionDirectoryName: "AnalyticsSolution"', 'CreateSubFolders: "True"', 'CreateSubFolders_IsAvailable: "True"', 'ProjectFilePath: "C:\Users\Beckhoff\Documents\AnalyticsSolution\AnalyticsProject\AnalyticsProject.tsproj"', and 'NewTcProject: "True"'. The 'Input Source' section shows 'Analytics.Common.InputSrcConfigurations: C1 - Live Data, C2 - Historical Data'. The 'TwinCAT PLC Target' section shows 'Runtime: "851"'. At the bottom are 'Back', 'Cancel', 'Next', and 'Deploy' buttons.

8. The **Summary** tab shows you all the settings you have made for the generation. Click on **Deploy** to start the generation process.



- ⇒ The overview window shows each step during the generation process, clearly arranged and divided into categories.



5.5.1 Algorithm properties

Each algorithm of the Analytics Configurator is providing some properties. The sections of HMI and PLC are necessary for the automatic code generation.

Properties ▼ 🔍 ✕

Edge Counter 1Ch_1 TwinCAT.Mechanics.AnalyticsFunctionNodeProper ▼

🔍 📄 🔍 🔍

[-] **Appearance**

Color	 Black
Comment	
SortPriority	100

[-] **Function**

ClassId	02040101-0000-0000-f000-000000000064
ClassName	
FactoryName	TcAnalyticsKernel
Group	TcAnalyticsKernel Base
Title	Edge Counter 1Ch_1

[-] **HMI**

Generate GVL	False
GlobalVariableType	InOutVariables

[-] **Network**

NetworkID	1
-----------	---

[-] **PLC**

Persistent Results	False
Stream Results	False

Appearance

Toolbox Properties

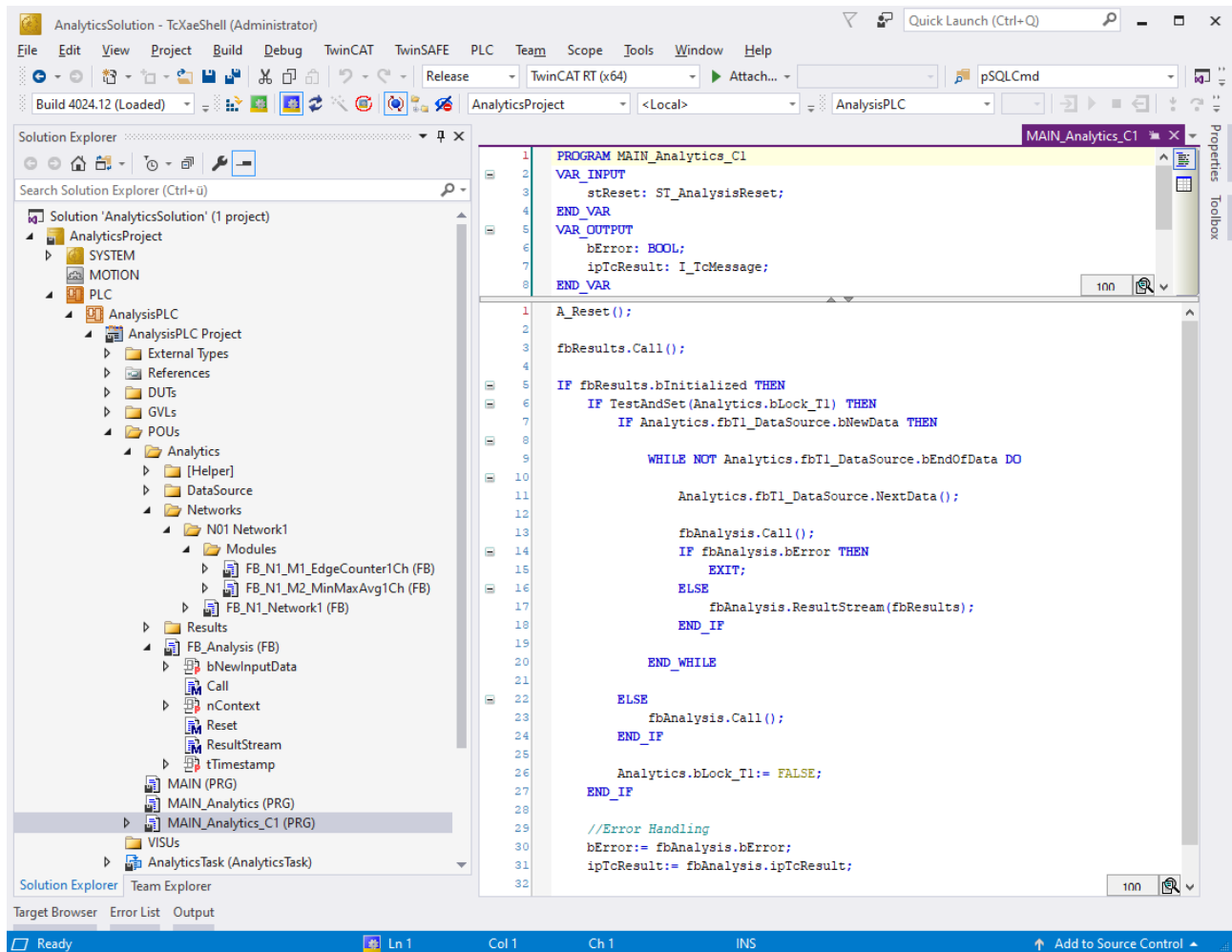
HMI

- **Generate GVL:** Enable the generation of an Global Variable List with a collection of variables and corresponding data type mapping for TwinCAT HMI
- **GlobalVariableType:** Choose the type with InOutVariables just for inputs and outputs of the algorithm or KeyValuePairs for general mapping to STRING for tables
-

PLC

- **Persistent Results:** Enable this flag to store results of algorithm persistent to target system of the Analytics Runtime
- **Stream Results:** Enable this flag to add the In- and Outputs of the algorithm to a result stream which will be generated by the code generation

5.5.2 PLC Code



NOTICE

Compatibility of automatically generated PLC code

The automatically generated PLC code is based on the TwinCAT Analytics Library. Interfaces of the base models of the library are compatible with earlier versions after enabling the library. The automatically generated code itself is only a sample code! This code generation may change from version to version.

Code generation version compatibility

	Version 4.0 (obsolete)	Version 4.1 (obsolete)	Version 5.0 (obsolete)	Version 5.1	Version 6.0	Version 7.0
Analytics algorithms	(X)	(X)	(X)	X	X	X
Filter algorithms	(X)	(X)	(X)	X	X	X
HMI support	(X)	(X)	(X)	X	X	X
HMI with advanced input stream handling	(X)	(X)	(X)	X	X	X
Support of array inputs	(X)	(X)	(X)	X	X	X
Support of oversampling inputs	(X)	(X)	(X)	X	X	X
Network templates	(X)	(X)	(X)	X	X	X
"Closed" network templates	(X)	(X)	(X)	X	X	X
Network with inputs, outputs and parameters	(X)	(X)	(X)	X	X	X
"Condition Monitoring" network templates	(X)	(X)	(X)	X	X	X
Lambda algorithms	(X)	(X)	(X)	X	X	X
Writing parameters via HMI	-	(X)	(X)	X	X	X
"Power Monitoring" algorithms	(X)	(X)	(X)	X	X	X
Scope Support	-	-	(X)	X	X	X
AnalyticsFile Inputsource Support	-	-	-	-	X	X
Vision Support	-	-	-	-	-	X

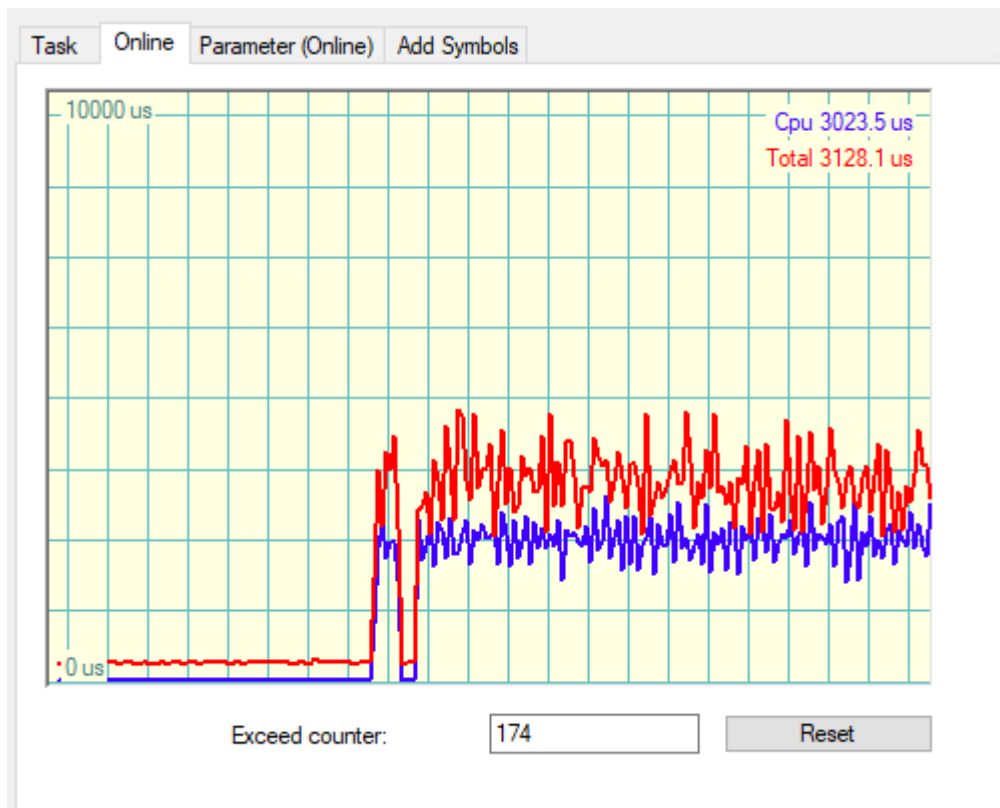
NOTICE

Possible manual PLC code optimizations

Every automatically generated PLC project has specific properties and requirements for smooth and error-free execution. Some of these properties cannot be predicted at the time of generation. To optimize the execution, various places are listed under [Manual optimizations](#) [► 314] where the generated code can be manually adapted.

5.5.2.1 Manual optimizations

Cycle overruns in the processing of historical data



This can be optimized using the following parameters in Analytics.GVL

```
{attribute 'qualified_only'}
VAR_GLOBAL
    //DataSources
    //Live:
    fbT1_InputSource: FB_T1_InputSource;
    //Historical:
    fbT2_C2_InputSourceHist: FB_T2_C2_InputSourceHistorical;
    //Empty:

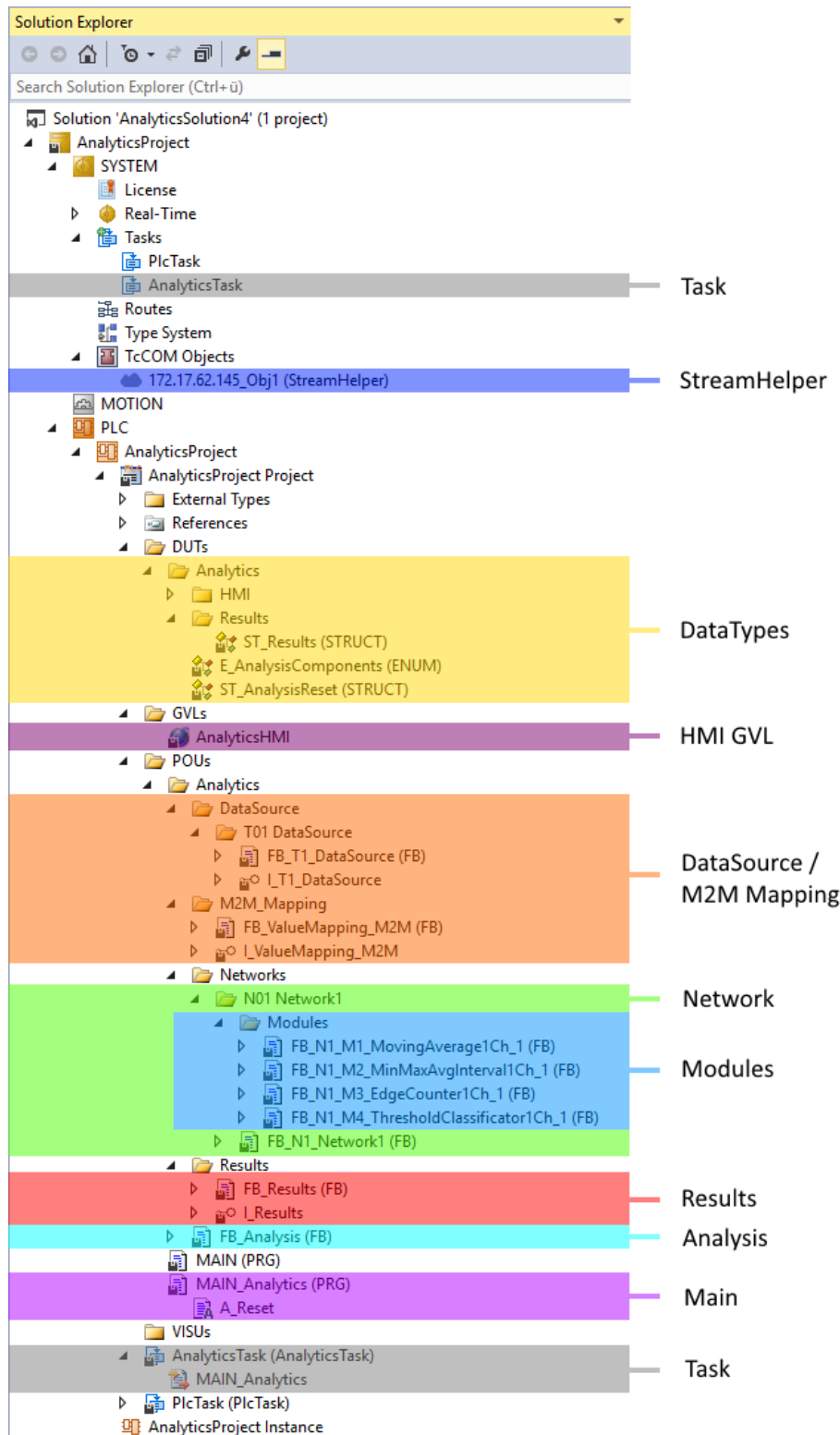
    fbT1_IecCriticalSection: FB_IecCriticalSection;

    //Historical DataSource parameter
    nHistMaxInputBuffer: UDINT := 100;
    nHistMaxSampleCount: UDINT := 400;

    stReset: ST_AnalysisReset;
END_VAR
```

- The parameter `nHistMaxInputBuffer` specifies the maximum number of samples to be processed per cycle. The higher this value, the greater the risk of cycle overruns. If this value is too small, the StreamHelper buffer may overflow, resulting in data loss.
- The parameter `nHistMaxSampleCount` specifies how many samples a message from the ASP may contain. The higher the value, the faster the data of a historical recording is transmitted. However, this can also cause the StreamHelper buffer to overflow more quickly

5.5.2.2 Code version 2.1 (obsolete)



Task:

A separate task is created for the analytics analysis.

StreamHelper:

If one or more data sources are of type MQTT binary stream, the code generation creates an instance of a StreamHelper object to process the incoming binary stream patterns.

DataTypes:

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

HMI GVL:

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables.

DataSource/M2M Mapping:

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs. The FB ValueMapping_M2M manages the value mapping between the modules (M2M - Module to Module) from the module INPUTs to the module OUTPUTs.

Network:

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Results:

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Analysis:

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN:

The FB Analysis is called in the program MAIN_Analytics. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

5.5.2.2.1 FB_DataSource

The DataSource FB manage the receiving of the input values of the different sources. In the OUTPUT declaration you can find all configured inputs.

Syntax

Definition:

```
FUNCTION_BLOCK FB_T[n]_DataSource IMPLEMENTS I_T[n]_DataSource
VAR
END_VAR
```

 Methods

Name	Definition location	Description
Call [► 317]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [► 317]	Local	Method to get the data of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.2.1.1 GetData**Syntax**

```
METHOD GetData : BOOL
VAR_INPUT
    nElement : UDINT;
END_VAR
```

 Inputs

Name	Type	Description
nElement	UDINT	Element ID to get the specific sample

 Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected

5.5.2.2.1.2 Call**Syntax**

```
METHOD Call : BOOL
```

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.2.2 FB_Network

All modules are sorted in a specific network to get a better overview and structure of the configured analysis.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_INPUT
    [module FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
```

```

    ipTcResult: I_TcMessage;
END_VAR

```

Inputs

Name	Type	Description
Module FBs		FBs of the configured modules

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE as soon as an error situation occurs.
ipTcResult	I_TcMessage	Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.

Methods

Name	Definition location	Description
Call [► 322]	Local	Method for background communication. The method must be called cyclically.
Reset [► 322]	Local	Reset the Network with all sub modules
ValueMapping [► 318]	Local	Map the input values to the different module inputs
SetHMIValues [► 319]	Local	Optional: Map in- outputs of the modules to the global HMI variable

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.2.1 ValueMapping

Syntax

```

METHOD ValueMapping : BOOL
VAR_INPUT
    ipDataSource : I_T[n]_DataSource;
END_VAR

```

Inputs

Name	Type	Description
ipDataSource	I_T[n]_DataSource	Data for the analysis from the specific data source

Return value

Name	Type	Description
ValueMapping	BOOL	Always returns FALSE.

5.5.2.2.2.2 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to global HMI struct

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE if done

5.5.2.2.2.3 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE if done

5.5.2.2.2.4 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipDataSource: I_T[n]_DataSource;
    [ipValueMapping_M2M: I_ValueMapping_M2M;]
END_VAR
```

Inputs

Name	Type	Description
ipDataSource	I_T[n]_DataSource	Data for the analysis.
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Needed for mapping values between modules

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.2.3 FB_Module

The module FBs contains all inputs and outputs of the configured modules from the Workbench Configurator. It is also possible to reconfigure the modules at runtime. You only have to change the parameter and then start the reconfigure process with a rising edge at the bReconfigure INPUT.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE as soon as an error situation occurs.
ipTcResult	I_TcMessage	Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.
Module outputs		Outputs of the selected module

Methods

Name	Definition location	Description
Call  320	Local	Method for background communication. The method must be called cyclically.
Reset  321	Local	Reset the module
SetHMI  321	Local	Optional: Sets the in- outputs to the global HMI structs

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.2.3.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.2.3.2 Reset**Syntax**

```
METHOD Reset : BOOL
VAR
END_VAR
```

 Return value

Name	Type	Description
Reset	BOOL	Is TRUE if done

5.5.2.2.3.3 SetHMI**Syntax**

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

 Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to global HMI struct

 Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE if done

5.5.2.2.4 FB_Analysis

In the analysis FB the whole analytics routine is defined. All configured networks with their modules and error handling is created.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    [network FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

 Inputs

Name	Type	Description
Network FBs		FBs of the configured networks

 Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE as soon as an error situation occurs.
ipTcResult	I_TcMessage	Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.

 Methods

Name	Definition location	Description
Call [▶ 322]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [▶ 322]	Local	Reset the whole analysis
ResultStream [▶ 323]	Local	Optional: If a result stream has to be created

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.2.4.1 Call**Syntax**

```
METHOD Call : BOOL
VAR_INPUT
    ipDataSource: I_T[n]_DataSource;
END_VAR
```

 Inputs

Name	Type	Description
ipDataSource	I_T[n]_DataSource	Data for the analysis.

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.2.4.2 Reset**Syntax**

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

 Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Struct to define witch module or network should be reset.

 Return value

Name	Type	Description
Reset	BOOL	Is TRUE if done

5.5.2.2.4.3 ResultStream

Syntax

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

 Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the Result FB

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.2.5 FB_Results

If results of the analysis has to be stored or streamed, the result FB managed this and streamed the selected variables to the message broker or store the data into the analytics binary file.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    stResults: ST_Results;
END_VAR
```

 Outputs

Name	Type	Description
stResults	ST_Results	Result struct which contains all items of the result stream

Methods

Name	Definition location	Description
Call [▶ 324]	Local	Method for background communication. The method must be called cyclically.
AddResult [▶ 324]	Local	Add a sample to the result stream
SendResults [▶ 324]	Local	Sends all buffered samples of the result stream
Release [▶ 325]	Local	Close stream or file of the result stream

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.2.5.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.2.5.2 AddResult

Syntax

```
METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample structure

Return value

Name	Type	Description
AddResult	BOOL	Always returns FALSE.

5.5.2.2.5.3 SendResults

Syntax

```
METHOD SendResults : BOOL
VAR
END_VAR
```

 Return value

Name	Type	Description
SendResults	BOOL	Always returns FALSE.

5.5.2.2.5.4 Release

Syntax

```
METHOD Release : BOOL  
VAR  
END_VAR
```

 Return value

Name	Type	Description
Release	BOOL	Always returns FALSE.

5.5.2.2.6 MAIN_Analytics

In the MAIN_Analytics program the analysis FB is called. The program is assign to the separated Task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First you have to choose the component you would like to reset. Then a rising edge at the bReset INPUT starts the reset process.

Inside of the A_Reset Action all reset calls are defined.

Syntax

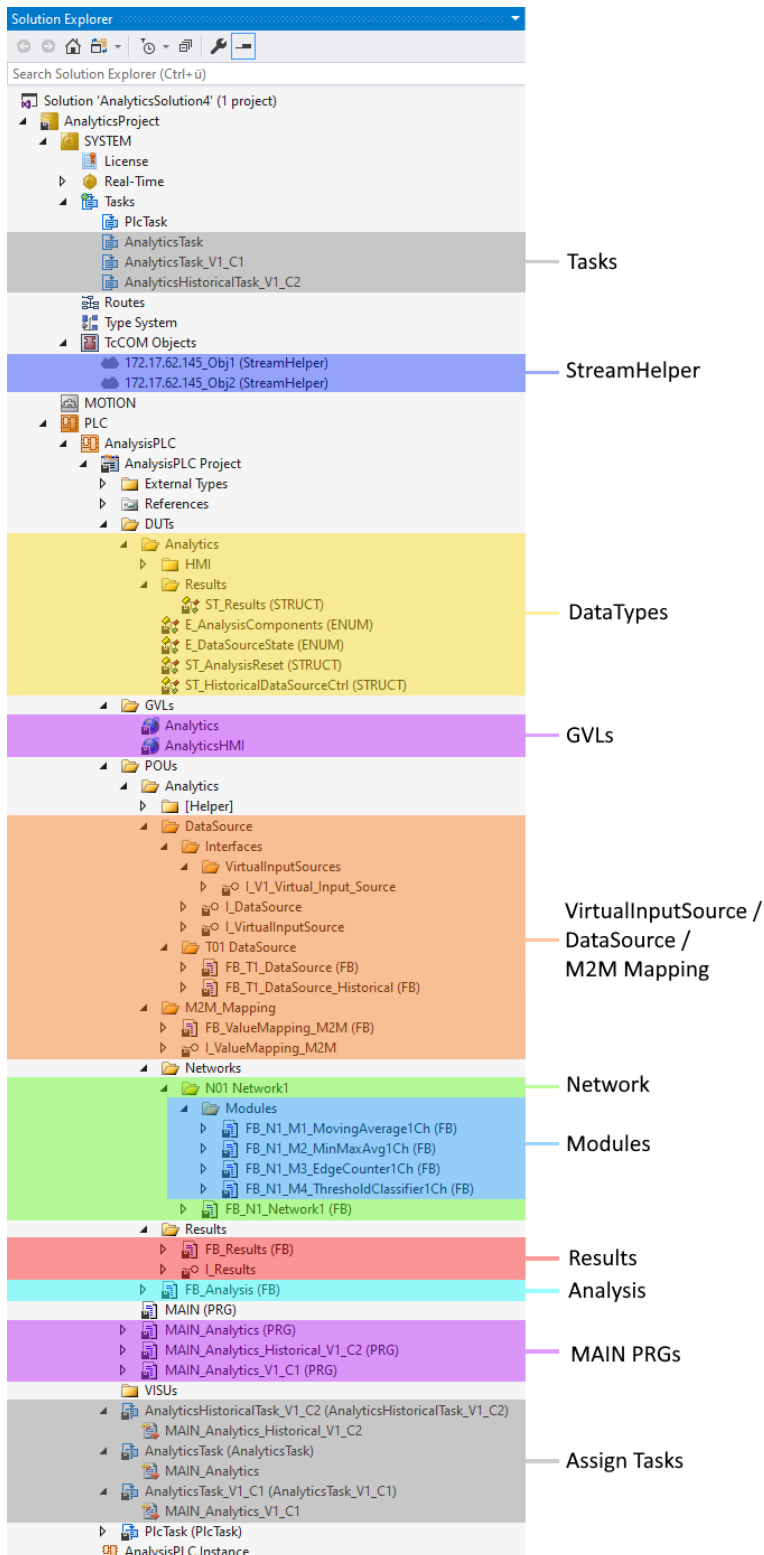
Definition:

```
PROGRAM MAIN_Analytics  
VAR_INPUT  
    stReset: ST_AnalysisReset;  
END_VAR  
VAR_OUTPUT  
    bError: BOOL;  
    ipTcResult: I_TcMessage;  
END_VAR
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.3 Code version 3.0 (obsolete)



Tasks:

A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

StreamHelper:

For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.

DataTypes:

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

GVLs:

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

VirtualInputSource / DataSource / M2M Mapping:

The Virtual Input Source interfaces abstract the Data Source symbols from the analysis. The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs. The FB ValueMapping_M2M manages the value mapping between the modules (M2M - Module to Module) from the module INPUTs to the module OUTPUTs.

Network:

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Results:

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

Analysis:

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN PRGs:

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

The FB Analysis is called in the programs MAIN_Analytics_Vx_Cx. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

5.5.2.3.1 FB_DataSource

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Syntax

Definition:

```
FUNCTION BLOCK FB_T[n]_DataSource IMPLEMENTS I_DataSource, I_V[n]_Virtual_Input_Source
VAR_OUTPUT
eDataState: E_DataSourceState;
END_VAR
```

Methods

Name	Definition location	Description
Call [▶ 328]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [▶ 328]	Local	Method to retrieve the data of the specified element.
GetDataOversampling [▶ 329]	Local	Method to retrieve the oversampling data of the specified element.
NewDataAvailable [▶ 329]	Local	Method to check if new data is available.
HistoricalCtrl [▶ 329]	Local	Method for retrieving historical data.
UpdateRecordList [▶ 330]	Local	Method for updating the historical record list.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.3.1.1 GetData

Syntax

```
METHOD GetData : BOOL
VAR_INPUT
nElement : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific example.

Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected.

5.5.2.3.1.2 Call

Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.3.1.3 GetDataOversampling

Syntax

```
METHOD GetDataOversampling : BOOL
VAR_INPUT
  nElement : UDINT;
  nSample : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample element.
nSample	UDINT	Sample ID to obtain the specific sample.

Return value

Name	Type	Description
GetDataOversampling	BOOL	Is TRUE if a new element is selected.

5.5.2.3.1.4 NewDataAvailable

Syntax

```
METHOD NewDataAvailable : BOOL
VAR_INPUT
  nLastDataHandle : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nLastDataHandle	ULINT	Handle from the last fetched data packet.

Return value

Name	Type	Description
NewDataAvailable	BOOL	Is TRUE if new data is available.

5.5.2.3.1.5 HistoricalCtrl

Syntax

```
METHOD HistoricalCtrl : BOOL
VAR_INPUT
  stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
  stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
  stRecordInfo : REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
```

Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
stRecordInfo	ST_HMI_DataSourceHistRecordInfo	

 Return value

Name	Type	Description
HistoricalCtrl	BOOL	Always returns FALSE.

5.5.2.3.1.6 UpdateRecordList**Syntax**

```

METHOD UpdateRecordList : BOOL
VAR_INPUT
    stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
    stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
    sStreamSystemID : GUID;
END_VAR

```

 Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
sStreamSystemID	GUID	

 Return value

Name	Type	Description
UpdateRecordList	BOOL	Always returns FALSE.

5.5.2.3.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Syntax

Definition:

```

FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_INPUT
    [module FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR

```

 Inputs

Name	Type	Description
Module FBs		Function blocks of the configured modules.

 Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Methods

Name	Definition location	Description
Call [▶ 335]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 335]	Local	Resetting the network with all submodules.
ValueMapping [▶ 331]	Local	Assignment of the input values to the various module inputs.
SetHMIValues [▶ 331]	Local	Optional: Mapping of the inputs/outputs of the modules to the global HMI variable.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.3.2.1 ValueMapping

Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for analysis from the specific data source.

Return value

Name	Type	Description
ValueMapping	BOOL	Always returns FALSE.

5.5.2.3.2.2 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure.

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed.

5.5.2.3.2.3 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.3.2.4 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M: I_ValueMapping_M2M;]
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_T[n]_Virtual_Input_Source	Data for the analysis.
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules.

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.3.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module.

 Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.
Module outputs		Outputs of the selected module.

 Methods

Name	Definition location	Description
Call [▶ 333]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 333]	Local	Resetting the module.
SetHMI [▶ 334]	Local	Optional: Sets the inputs/outputs to the global HMI structures.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.3.3.1 Call**Syntax**

```

METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR

```

 Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules.

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.3.3.2 Reset**Syntax**

```

METHOD Reset : BOOL
VAR
END_VAR

```

 Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.3.3 SetHMI**Syntax**

```

METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR

```

 Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure.

 Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE when completed.

5.5.2.3.4 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

Syntax

Definition:

```

FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    [network FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR

```

 Inputs

Name	Type	Description
Network FBs		Function blocks of the configured networks.

 Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

 Methods

Name	Definition location	Description
Call [▶ 335]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [▶ 335]	Local	Resets the entire analysis.
ResultStream [▶ 336]	Local	Optional: If a result stream needs to be created.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.3.4.1 Call**Syntax**

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

 Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.3.4.2 Reset**Syntax**

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

 Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Structure to define which module or network is to be reset.

 Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.3.4.3 ResultStream

Syntax

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the FB Results.

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.3.5 FB_Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    stResults: ST_Results;
END_VAR
```

Outputs

Name	Type	Description
stResults	ST_Results	Result structure that contains all elements of the result stream.

Methods

Name	Definition location	Description
Call [► 337]	Local	Method for background communication. The method must be called cyclically.
AddResult [► 337]	Local	Add a sample to the result stream
SendResults [► 337]	Local	Sends all buffered samples of the result stream
Release [► 337]	Local	Close stream or file of the result stream

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.3.5.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.3.5.2 AddResult

Syntax

```
METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample structure

Return value

Name	Type	Description
AddResult	BOOL	Always returns FALSE.

5.5.2.3.5.3 SendResults

Syntax

```
METHOD SendResults : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
SendResults	BOOL	Always returns FALSE.

5.5.2.3.5.4 Release

Syntax

```
METHOD Release : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Release	BOOL	Always returns FALSE.

5.5.2.3.6 MAIN_Analytics

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI Dashboard. The program is assigned to a separate task.

Syntax

Definition:

```
PROGRAM MAIN_Analytics
VAR_INPUT
    sCurrentStreamSystemID: GUID;
    stHistStreamInfo: REFERENCE TO ST_HMI_DataSourceHist;
    stHistRecordInfo: REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.3.7 MAIN_Analytics_V[n]_C[n]

The FB Analysis is called in the program MAIN_Analytics_V[n]_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Mapping of the HMI values takes place in the action A_MapToHMI.

Syntax

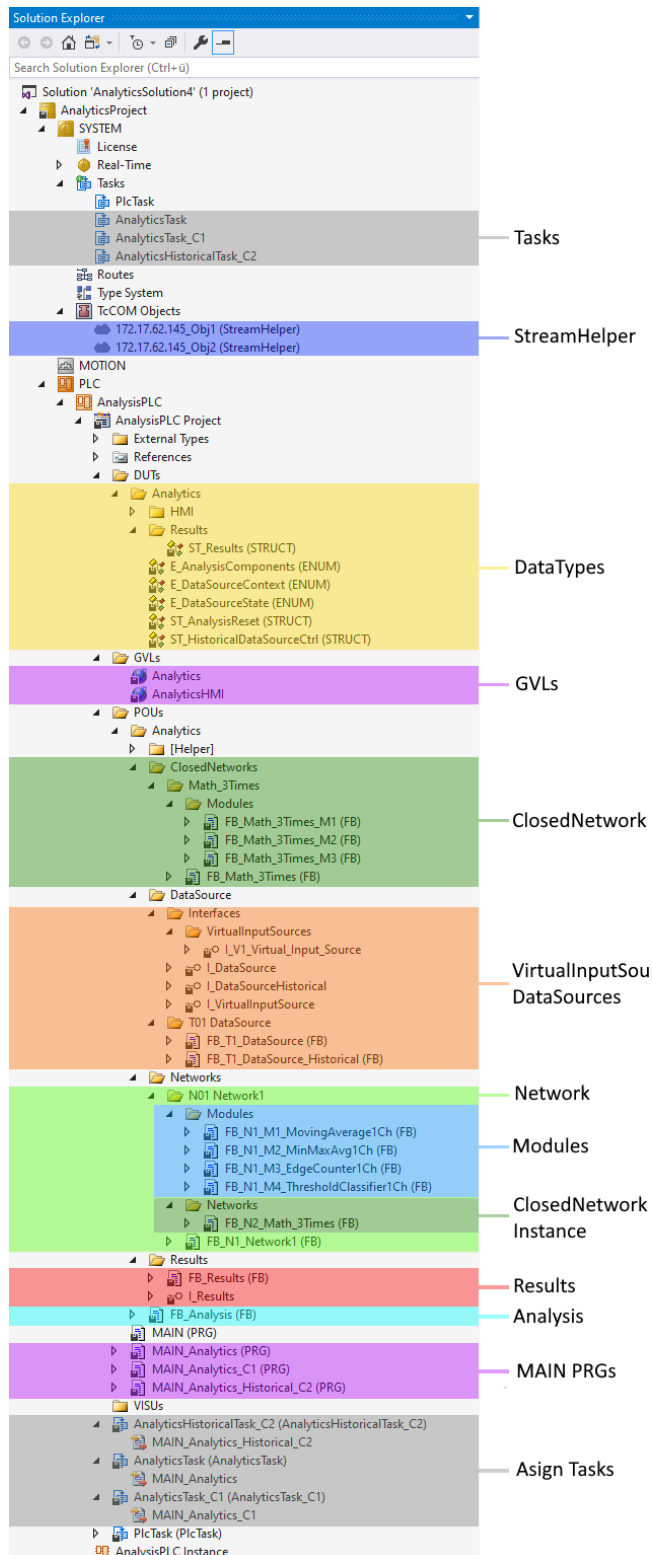
Definition:

```
PROGRAM MAIN_Analytics_V[n]_C[n]
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    nLastDataHandle: ULINT;
END_VAR
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.4 Code version 4.x (obsolete)



Tasks:

A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

StreamHelper:

For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.

DataTypes:

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

GVLs:

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

ClosedNetwork:

The ClosedNetwork FBs are generated once with all subnetworks and modules. They can be instantiated multiple times in the analysis. In this way, the generated code can be reduced and simplified.

VirtualInputSource / DataSource:

The VirtualInputSource interfaces abstract the DataSource symbols from the analysis. The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Network:

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

ClosedNetwork Instance:

In this FB the corresponding ClosedNetwork is instantiated for the analysis. Internally used modules are no longer generated as module FBs in this case.

Results:

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

Analysis:

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN PRGs:

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

The FB Analysis is called in the programs MAIN_Analytics_Vx_Cx. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

5.5.2.4.1 FB_DataSource

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Syntax

Definition:

```
FUNCTION_BLOCK FB_T[n]_DataSource IMPLEMENTS I_DataSource, I_V[n]_Virtual_Input_Source
VAR_OUTPUT
eDataState: E_DataSourceState;
END_VAR
```

Methods

Name	Definition location	Description
Call [▶ 342]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [▶ 341]	Local	Method to retrieve the data of the specified element.
GetDataOversampling [▶ 342]	Local	Method to retrieve the oversampling data of the specified element
NewDataAvailable [▶ 342]	Local	Method to check if new data is available.
HistoricalCtrl [▶ 342]	Local	Method for retrieving historical data
UpdateRecordList [▶ 343]	Local	Method for updating the historical record list.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.4.1.1 GetData

Syntax

```
METHOD GetData : BOOL
VAR_INPUT
nElement : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample

Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected

5.5.2.4.1.2 Call

Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.4.1.3 GetDataOversampling

Syntax

```
METHOD GetDataOversampling : BOOL
VAR_INPUT
nElement : UDINT;
nSample: : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample element
nSample	UDINT	Sample ID to obtain the specific sample

Return value

Name	Type	Description
GetDataOversampling	BOOL	Is TRUE if a new element is selected

5.5.2.4.1.4 NewDataAvailable

Syntax

```
METHOD NewDataAvailable : BOOL
VAR_INPUT
nLastDataHandle : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nLastDataHandle	ULINT	Handle of the last fetched data packet

Return value

Name	Type	Description
NewDataAvailable	BOOL	Is TRUE if new data is available

5.5.2.4.1.5 HistoricalCtrl

Syntax

```
METHOD HistoricalCtrl : BOOL
VAR_INPUT
stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
```

```
    stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;  
    stRecordInfo : REFERENCE TO ST_HMI_DataSourceHistRecordInfo;  
END_VAR
```

Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
stRecordInfo	ST_HMI_DataSourceHistRecordInfo	

Return value

Name	Type	Description
HistoricalCtrl	BOOL	Always returns FALSE.

5.5.2.4.1.6 UpdateRecordList

Syntax

```
METHOD UpdateRecordList : BOOL  
VAR_INPUT  
    stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;  
    stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;  
    sStreamSystemID : GUID;  
END_VAR
```

Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
sStreamSystemID	GUID	

Return value

Name	Type	Description
UpdateRecordList	BOOL	Always returns FALSE.

5.5.2.4.1.7 NextData

Syntax

```
METHOD NextData : BOOL  
VAR_INPUT  
END_VAR
```

5.5.2.4.1.8 NextDataOversample

Syntax

```
METHOD GetDataOversampling : BOOL  
VAR_INPUT  
    nMaxOversampling : UDINT;  
END_VAR
```


 Inputs

Name	Type	Description
nMaxOversampling	UDINT	Specifies the maximum oversampling factor.

 Return value

Name	Type	Description
NextDataOversample	BOOL	Is TRUE if a new element is selected

5.5.2.4.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [module FBs]
END_VAR
```

 Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

 Methods

Name	Definition location	Description
Call [► 335]	Local	Method for background communication. The method must be called cyclically.
Reset [► 335]	Local	Resetting the network with all submodules.
ValueMapping [► 345]	Local	Assignment of the input values to the various module inputs.
SetHMIValues [► 345]	Local	Optional: Mapping of the inputs/outputs of the modules to the global HMI variable.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.4.2.1 ValueMapping

Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    pAnalysis : POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB

Return value

Name	Type	Description
ValueMapping	BOOL	Always returns FALSE.

5.5.2.4.2.2 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed

5.5.2.4.2.3 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pAnalysis: POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB.

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.4.2.4 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.4.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module.

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.
Module outputs		Outputs of the selected module.

Methods

Name	Definition location	Description
Call  347	Local	Method for background communication. The method must be called cyclically.
Reset  347	Local	Resetting the module.
SetHMI  347	Local	Sets the inputs/outputs to the global HMI structures.
GetHMI  348	Local	Optional: Sets the inputs of the global HMI structures to the inputs of the module

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.4.3.1 Call

Syntax

```

METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR

```

 Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.4.3.2 Reset

Syntax

```

METHOD Reset : BOOL
VAR
END_VAR

```

 Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.4.3.3 SetHMI

Syntax

```

METHOD GetHMI : BOOL
VAR_INPUT
    nContent : DINT
    pContent : PVOID
    bHMIReinit : BOOL
END_VAR

VAR
    pHMI_C[n]_[Content] : POINTER TO ST_HMI_C[n]_[Content];
END_VAR

```

 Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure
bHMIReinit	BOOL	Initialize the HMI content structure

 Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE when completed

5.5.2.4.3.4 GetHMI (4.1)**Syntax**

```

METHOD GetHMI : BOOL
VAR_INPUT
    nContent : DINT
    pContent : PVOID
END_VAR

VAR
    pHMI_C[n]_[Content] : POINTER TO ST_HMI_C[n]_[Content];
END_VAR

```

 Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure

 Return value

Name	Type	Description
GetHMI	BOOL	Is TRUE when completed

5.5.2.4.4 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

Syntax

Definition:

```

FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    ipV[n]_VirtualInputs: I_V[n]_Virtual_Input_Source;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [network FBs]
END_VAR

```

 Inputs

Name	Type	Description
ipV[n]_VirtualInputs	I_V[n]_Virtual_Input_Source	Data for analysis from the specific data source

 Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

 Methods

Name	Definition location	Description
Call [► 349]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [► 349]	Local	Reset the whole analysis
ResultStream [► 350]	Local	Optional: If a result stream has to be created

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.4.4.1 Call**Syntax**

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

 Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.4.4.2 Reset**Syntax**

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

 Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Structure to define which module or network is to be reset.

 Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.4.4.3 ResultStream**Syntax**

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

 Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the FB Results

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.4.5 FB_Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    stResults: ST_Results;
END_VAR
```

 Outputs

Name	Type	Description
stResults	ST_Results	Result structure that contains all elements of the result stream.

Methods

Name	Definition location	Description
Call [▶ 351]	Local	Method for background communication. The method must be called cyclically.
AddResult [▶ 351]	Local	Add a sample to the result stream
SendResults [▶ 351]	Local	Sends all buffered samples of the result stream
Release [▶ 352]	Local	Close stream or file of the result stream

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.4.5.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.4.5.2 AddResult

Syntax

```
METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample structure

Return value

Name	Type	Description
AddResult	BOOL	Always returns FALSE.

5.5.2.4.5.3 SendResults

Syntax

```
METHOD SendResults : BOOL
VAR
END_VAR
```


Return value

Name	Type	Description
SendResults	BOOL	Always returns FALSE.

5.5.2.4.5.4 Release

Syntax

```
METHOD Release : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Release	BOOL	Always returns FALSE.

5.5.2.4.6 MAIN_Analytics

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI Dashboard. The program is assigned to a separate task.

Syntax

Definition:

```
PROGRAM MAIN_Analytics
VAR_INPUT
    sCurrentStreamSystemID: GUID;
    stHistStreamInfo: REFERENCE TO ST_HMI_DataSourceHist;
    stHistRecordInfo: REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.4.7 MAIN_Analytics_C[n]

The FB Analysis is called in the program MAIN_Analytics_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Mapping of the HMI values takes place in the action A_MapToHMI.

Syntax

Definition:

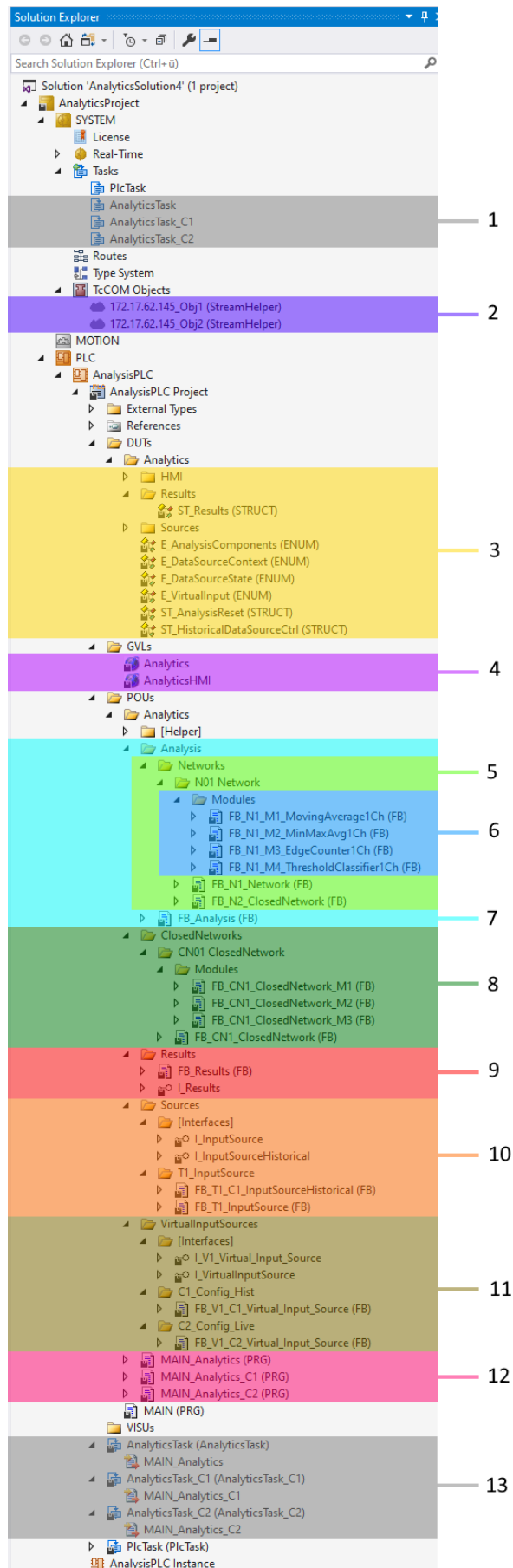
```
PROGRAM MAIN_Analytics_C[n]
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
```

```
bError: BOOL;  
ipTcResult: I_TcMessage;  
nLastDataHandle: ULINT;  
END_VAR
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5 Codeversion 5.x – 7.0



1: Tasks

A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

2: StreamHelper

For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.

3: DataTypes

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

4: GVLs

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

5: Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

6: Modules

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

7: Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

8: ClosedNetwork

The ClosedNetwork FBs are generated once with all subnetworks and modules. They can be instantiated multiple times in the analysis. In this way, the generated code can be reduced and simplified.

9: Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

10: InputSource

The FB InputSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

11: VirtualInputSource

The VirtualInputSource interfaces abstract the InputSource symbols from the analysis.

12: MAIN PRGs

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

In the programs MAIN_Analytics_Vx_Cx the FB-Analysis is called. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

5.5.2.5.1 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    nConfigurationID: INT;
    ipV[n]_VirtualInputs: I_V[n]_Virtual_Input_Source;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [network FBs]
END_VAR
```

Inputs

Name	Type	Description
nConfigurationID	INT	Configuration index
ipV[n]_VirtualInputs	I_V[n]_Virtual_Input_Source	Data for analysis from the specific data source

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Properties

Name	Type	Access	Description
nContext	DWORD	Get	Context index

Methods

Name	Definition location	Description
Call [▶ 357]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [▶ 357]	Local	Resets the entire analysis.
ResultStream [▶ 358]	Local	Optional: If a result stream needs to be created.
SetHMIValues [▶ 358]	Local	Method for filling the HMI structures
GetHMIValues [▶ 358]	Local	Method for setting the parameters from the HMI into the analysis

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5.1.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.5.1.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Structure to define which module or network is to be reset.

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.5.1.3 ResultStream

Syntax

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the FB results

Return value

Name	Type	Description
ResultStream	BOOL	Always returns FALSE.

5.5.2.5.1.4 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed

5.5.2.5.1.5 GetHMIValues

Syntax

```
METHOD GetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed

5.5.2.5.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [module FBs]
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Methods

Name	Definition location	Description
Call [► 335]	Local	Method for background communication. The method must be called cyclically.
Reset [► 335]	Local	Resetting the network with all submodules.
ValueMapping [► 360]	Local	Assignment of the input values to the various module inputs.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5.2.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pAnalysis: POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB.

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.5.2.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.5.2.3 ValueMapping

Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    pAnalysis : POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB

Return value

Name	Type	Description
ValueMapping	BOOL	Always returns FALSE.

5.5.2.5.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_INPUT PERSISTENT
    [module persistent parameter inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module.

 Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.
Module outputs		Outputs of the selected module.

 Properties

Name	Type	Access	Description
nContext	DWORD	Get	Context index

 Methods

Name	Definition location	Description
Call [► 361]	Local	Method for background communication. The method must be called cyclically.
Reset [► 362]	Local	Resetting the module.
SetHMI [► 362]	Local	Sets the inputs/outputs to the global HMI structures.
GetHMI [► 362]	Local	Optional: Sets the inputs of the global HMI structures to the inputs of the module

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5.3.1 Call**Syntax**

```

METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR

```

 Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.5.3.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

5.5.2.5.3.3 SetHMI

Syntax

```
METHOD SetHMI : BOOL
VAR_INPUT
    nContent : DINT
    pContent : PVOID
    bHMIRereinit : BOOL
END_VAR

VAR
    pHMI_C[n][Content] : POINTER TO ST_HMI_C[n][Content];
END_VAR
```

Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure
bHMIRereinit	BOOL	Initialize the HMI content structure

Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE when completed

5.5.2.5.3.4 GetHMI

Syntax

```
METHOD GetHMI : BOOL
VAR_INPUT
    nContent : DINT
    pContent : PVOID
END_VAR

VAR
    pHMI_C[n][Content] : POINTER TO ST_HMI_C[n][Content];
END_VAR
```

Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure

 Return value

Name	Type	Description
GetHMI	BOOL	Is TRUE when completed

5.5.2.5.4 FB_InputSource

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Syntax

Definition:

```
FUNCTION_BLOCK FB_T[n]_InputSource IMPLEMENTS I_InputSource
VAR
END_VAR
```

 Interfaces

Type	Description
I_InputSource	Interface for communication with a data source

 Methods

Name	Definition location	Description
Call [► 363]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [► 364]	Local	Method to retrieve the data of the specified element.
NewDataAvailable [► 364]	Local	Method to check if new data is available.
AddClient [► 365]	Local	Method for adding data Clients
ClientDone [► 365]	Local	Method to signal that the client has received all data.

 Properties

Name	Type	Access	Description
bReadNewData	BOOL	Get	.
eDataState	E_DataSourceState	Get	
nDataHandle	ULINT	Get	
nElements	UDINT	Get	
nMaxOversamplingFactor	UDINT	Get	

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5.4.1 Call**Syntax**

```
METHOD Call : BOOL
```

 Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.5.4.2 GetData**Syntax**

```
METHOD GetData : BOOL
VAR_INPUT
    nElement      : UDINT;
    pInputs       : PVOID;
    nInputsSize   : UDINT
END_VAR
VAR_OUTPUT
    nTimestamp    : ULINT;
    nContext      : DWORD;
END_VAR
```

 Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample
pInputs	PVOID	Pointer to the data structure
nInputsSize	UDINT	Size of the data structure

 Outputs

Name	Type	Description
nTimestamp	ULINT	Timestamp data
nContext	DWORD	Data context

 Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected

5.5.2.5.4.3 NewDataAvailable**Syntax**

```
METHOD NewDataAvailable : BOOL
VAR_INPUT
    nLastDataHandle : ULINT;
END_VAR
```

 Inputs

Name	Type	Description
nLastDataHandle	ULINT	Handle of the last fetched data packet

 Return value

Name	Type	Description
NewDataAvailable	BOOL	Is TRUE if new data is available

5.5.2.5.4.4 AddClient

Syntax

```
METHOD AddClient : BOOL
VAR_OUTPUT
    nClientID : DWORD;
END_VAR
```

Outputs

Name	Type	Description
nClientID	LWORD	Client-ID

5.5.2.5.4.5 ClientDone

Syntax

```
METHOD ClientDone : BOOL
VAR_OUTPUT
    nClientID : DWORD;
END_VAR
```

Outputs

Name	Type	Description
nClientID	LWORD	Client-ID

5.5.2.5.5 FB_VirtualInputSource

The FB VirtualInputSource abstracts the InputSources for the different analysis configurations. The virtual inputs configured in the Analytics Workbench are set here.

Syntax

Definition:

```
FUNCTION_BLOCK FB_V[n]_C[n]_VirtualInputSource IMPLEMENTS I_VirtualInputSource,
I_V[n]_VirtualInputSource
VAR
END_VAR
```

Interfaces

Type	Description
I_VirtualInputSource	Interface for communication with a data source
I_V[n]_VirtualInputSource	Interface that provides all defined virtual inputs

Methods

Name	Definition location	Description
SourceSync [► 366]	Local	Method for synchronizing multiple FB instances
NextData [► 366]	Local	Method to accept the next data set
Done [► 366]	Local	Method of signaling that the entire data packet has been processed.

 Properties

Name	Type	Access	Description
bEndOfData	BOOL	Get	Signals the end of the data packet
dtTimestamp	DCTIMESTRUCT	Get	Timestamp of the currently accepted data set
nDataHandle	ULINT	Get	Data handle
nContext	DWORD	Get	Data context
tTimestamp	ULINT	Get	Timestamp of the currently received data set

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5.5.1 SourceSync**Syntax**

```
METHOD SourceSync : BOOL
```

5.5.2.5.5.2 NextData**Syntax**

```
METHOD NextData : BOOL
VAR_INPUT
    nMaxOversampling : UDINT;
END_VAR
```

 Inputs

Name	Type	Description
nMaxOversampling	UDINT	Specifies the maximum oversampling value

 Return value

Name	Type	Description
NextData	BOOL	Is TRUE if a new element is selected

5.5.2.5.5.3 Done**Syntax**

```
METHOD Done : BOOL
VAR_INPUT
END_VAR
```

5.5.2.5.6 FB_Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Syntax

Definition:

```

FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    nTimestamp: ULINT;
    stResults: ST_Results;
END_VAR

```

Outputs

Name	Type	Description
nTimestamp	ULINT	Associated timestamp of the result structure data
stResults	ST_Results	Result structure that contains all elements of the result stream.

Properties

Name	Type	Access	Description
bInitialized	BOOL	Get	Indicates whether the function block has been properly initialized
nMaxSamples	INT	Get	Maximum number of buffered results
nResultCount	INT	Get	Current number of buffered results

Methods

Name	Definition location	Description
Call [► 367]	Local	Method for background communication. The method must be called cyclically.
AddResult [► 367]	Local	Add a sample to the result stream
SendResults [► 368]	Local	Sends all buffered samples of the result stream
Release [► 368]	Local	Close stream or file of the result stream

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5.6.1 Call

Syntax

```

METHOD Call : BOOL
VAR
END_VAR

```

Return value

Name	Type	Description
Call	BOOL	Always returns FALSE.

5.5.2.5.6.2 AddResult

Syntax

```

METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR

```


 Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample structure

 Return value

Name	Type	Description
AddResult	BOOL	Always returns FALSE.

5.5.2.5.6.3 SendResults**Syntax**

```
METHOD SendResults : BOOL
VAR
END_VAR
```

 Return value

Name	Type	Description
SendResults	BOOL	Always returns FALSE.

5.5.2.5.6.4 Release**Syntax**

```
METHOD Release : BOOL
VAR
END_VAR
```

 Return value

Name	Type	Description
Release	BOOL	Always returns FALSE.

5.5.2.5.7 MAIN_Analytics

In the MAIN_Analytics program, the InputSource FBs are called, the reset function is managed and, if necessary, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

Syntax

Definition:

```
PROGRAM MAIN_Analytics
VAR
END_VAR
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.5.2.5.8 MAIN_Analytics_C[n]

The FB Analysis is called in the program MAIN_Analytics_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

The handling of the "InputSources" is done in the action A_InputSources.

The mapping of the HMI values is done in the action A_MapToHMI.

Syntax

Definition:

```
PROGRAM MAIN_Analytics_C[n]
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    nAnalysisResultsTimestamp: ULINT;
    stAnalysisResults: ST_Results;
END_VAR
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.6 HMI One-Click Dashboard

It is possible to automatically generate an HMI dashboard with HMI Controls for all modules and parameters configured in the TwinCAT Analytics Workbench Configurator. The HMI Dashboard is based on the [TwinCAT HMI](#) and visualizes the PLC data from the [runtime deployment](#) [► 303].

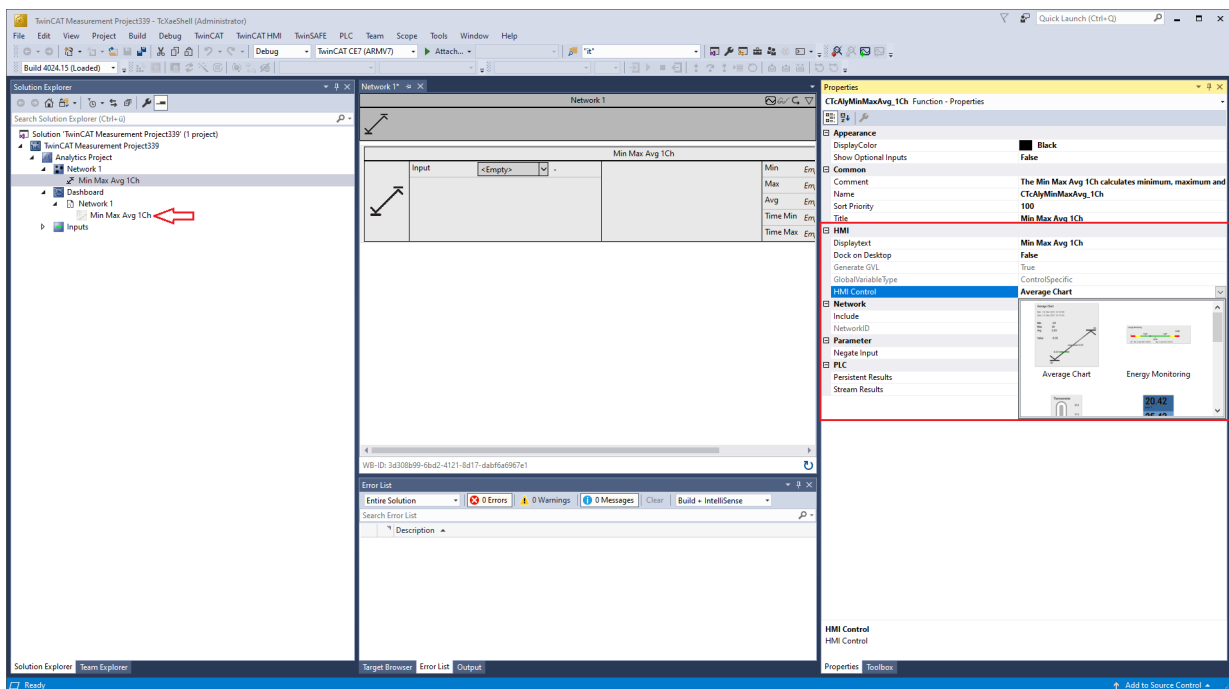


The automatically generated One-Click Dashboard is only available with the new HMI version 1.12. An Analytics Runtime license is required in order to use the Analytics HMI Controls.

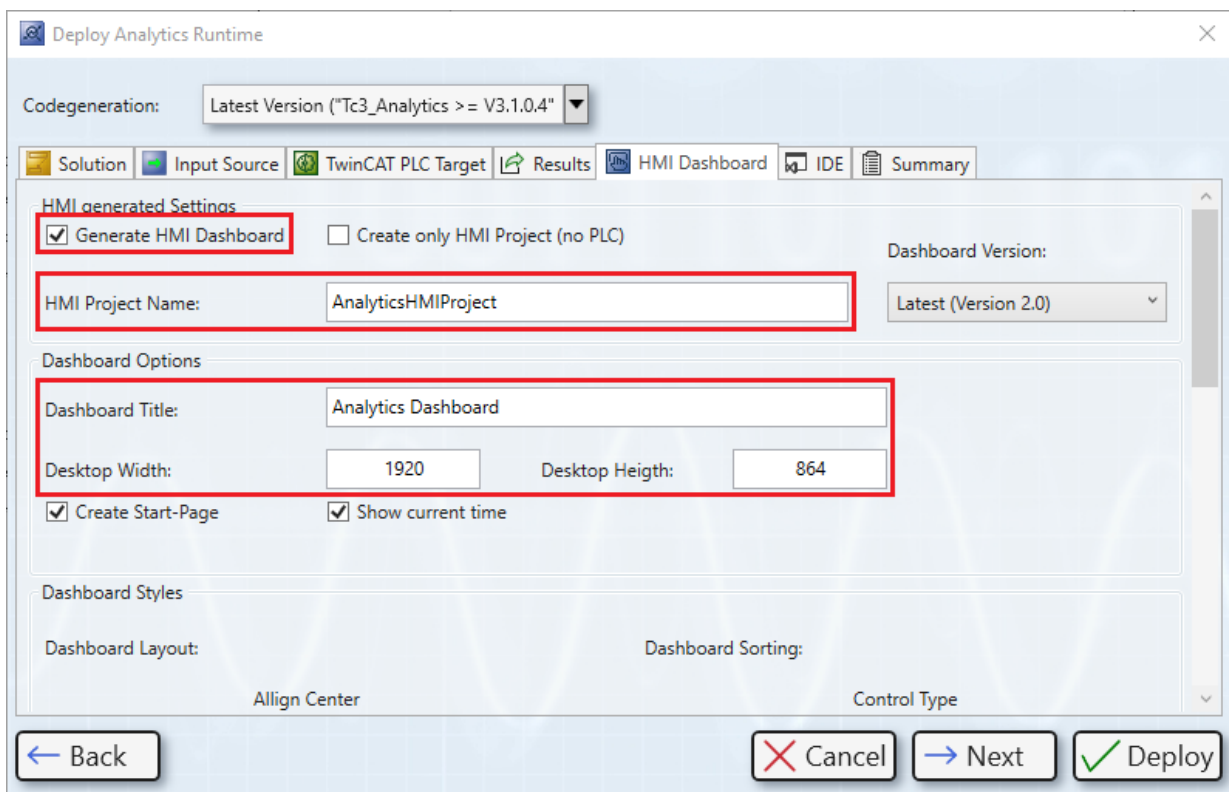


- ✓ After configuring your Analytics Workbench project, an HMI Control can be selected for each algorithm.
- 1. Open the **Properties** window of the module and select an **HMI Control** from the dropdown list. You can change the display text for the title in the HMI dashboard (display text). You can also choose whether the control should be docked to the start page (Dock on Desktop). In the **Solution Explorer** all the controls

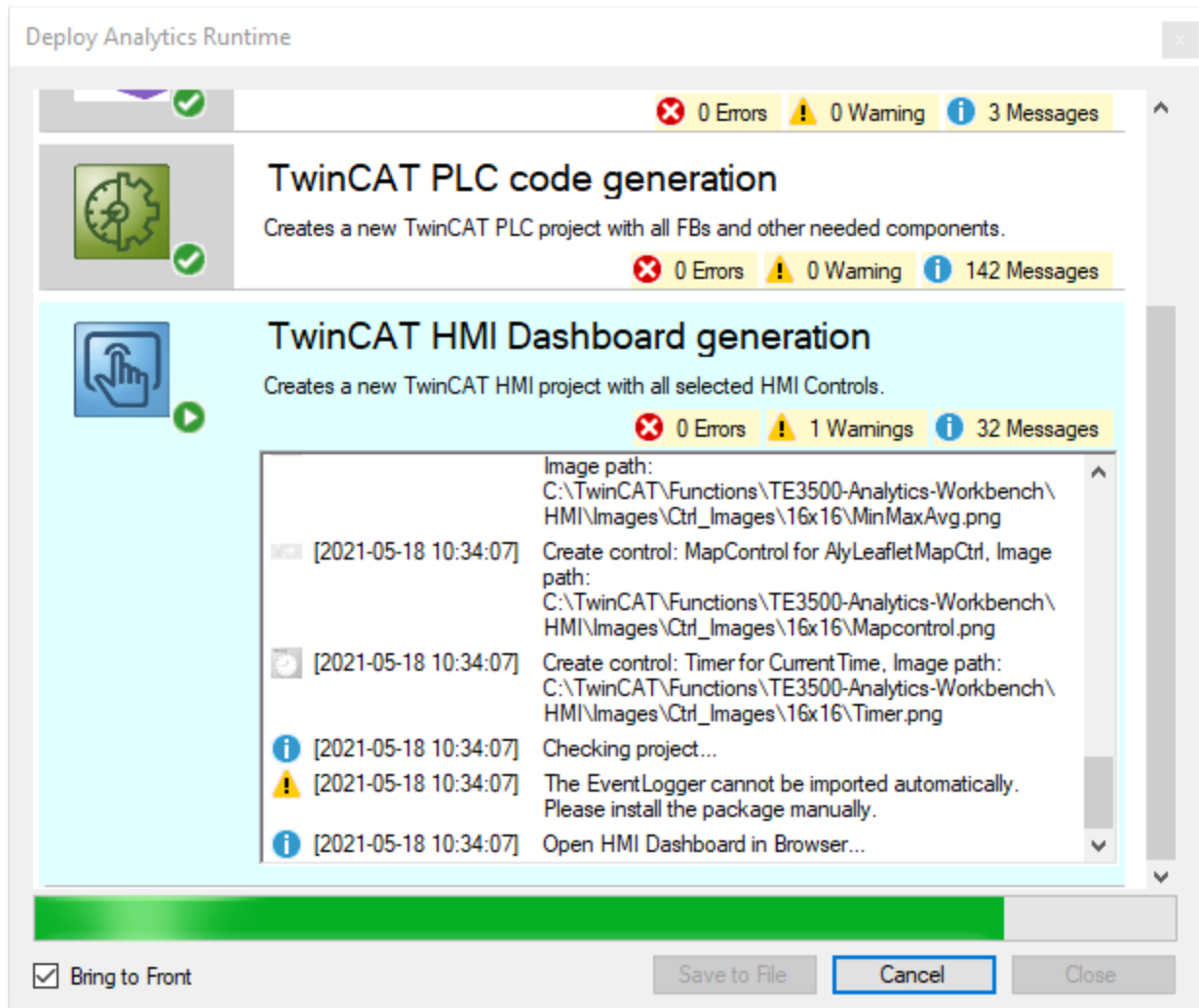
to be generated are stored under the Manage dashboard structure and content in Analytics project [► 372]



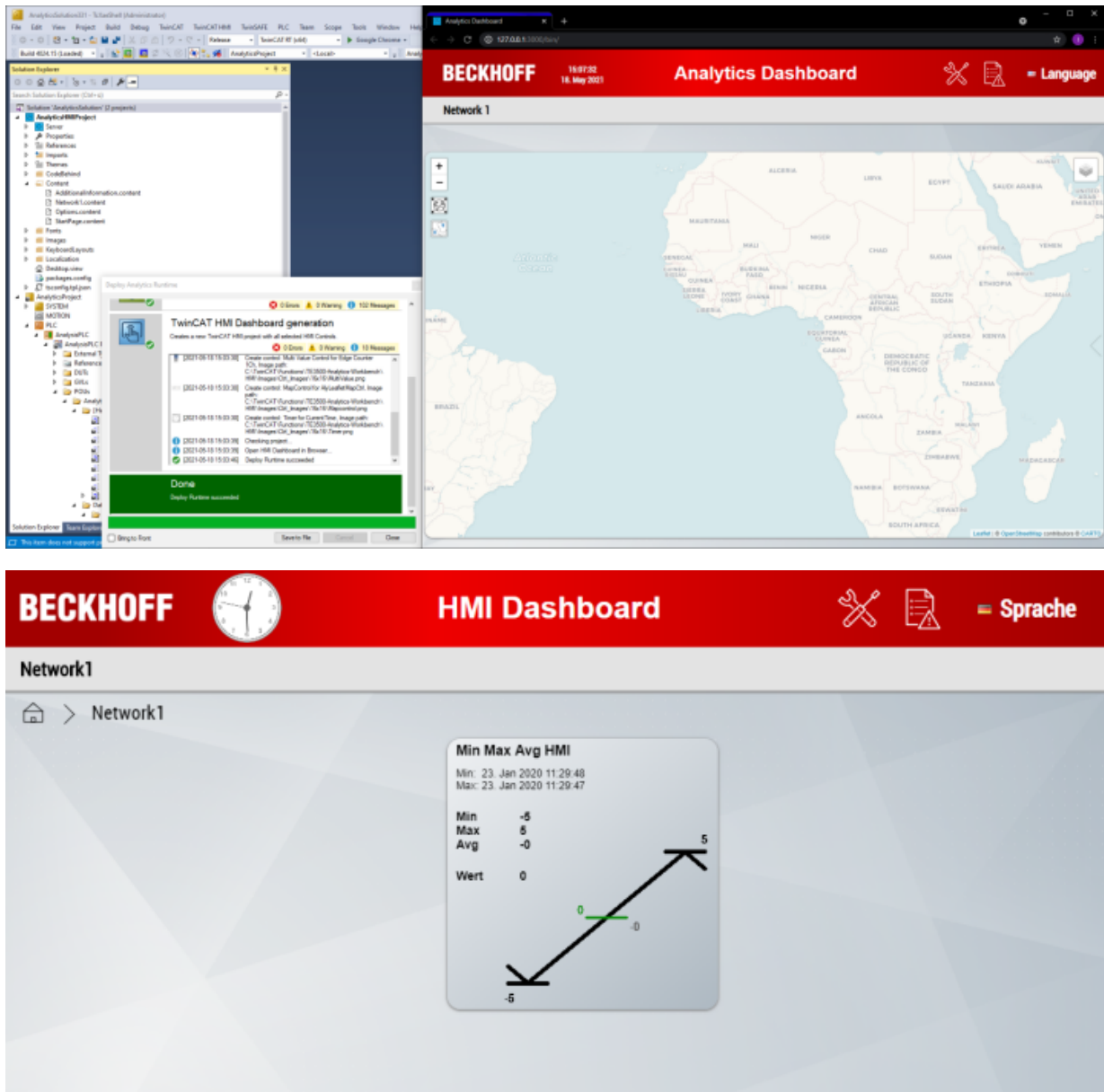
- After completion of the configuration, click the **Deploy Analytics Runtime** command in the context menu. The Deploy Wizard starts and it is possible to set up the entire required configuration step by step for use.
⇒ You can configure your HMI Dashboard on the **HMI Dashboard** tab.
- Activate the **Generate HMI Dashboard** checkbox. It is also possible to create only one HMI project without a PLC. Furthermore, you can also assign an **HMI Project Name** to the dashboard and set a Dashboard Title as well as the Desktop Height and Desktop Width in order to generate a tailor-made dashboard that suits your needs. The remaining configurations are explained in Dashboard Configuration [► 404].



- ⇒ As usual, the last tab shows you all the settings you have made for the generation.
4. Now you can start the generation process by clicking **Deploy**.
- ⇒ The HMI generation begins immediately after runtime deployment (if selected). Each step for generating the HMI dashboard is also displayed in the overview window during the generation process.

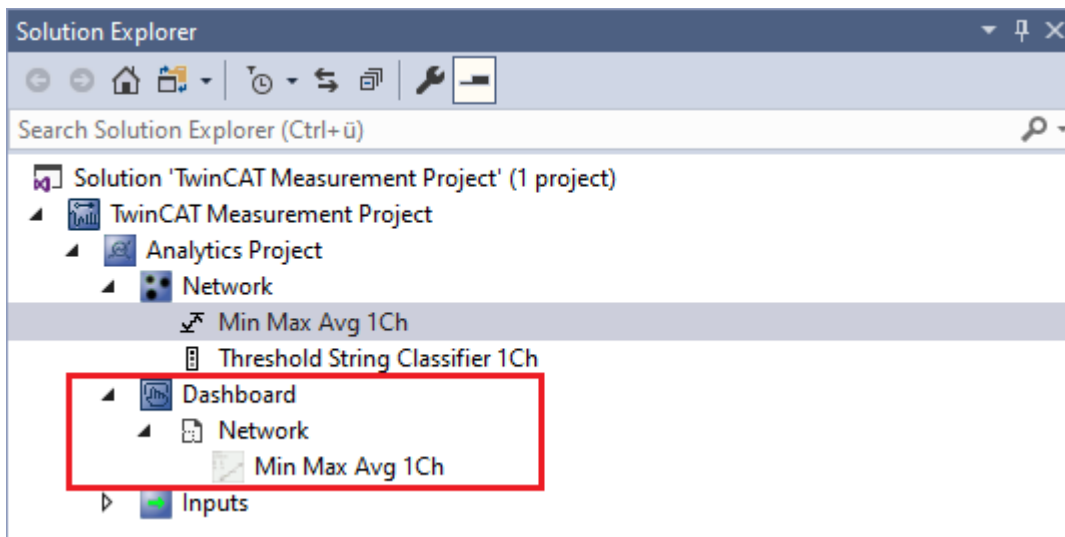


⇒ The dashboard opens automatically in your default browser.

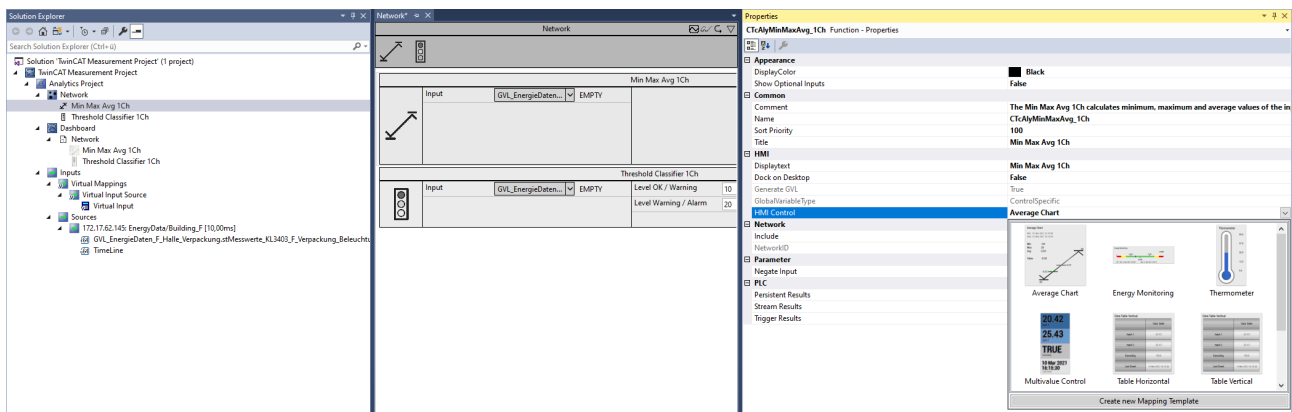


5.6.1 Manage dashboard structure and content in Analytics project

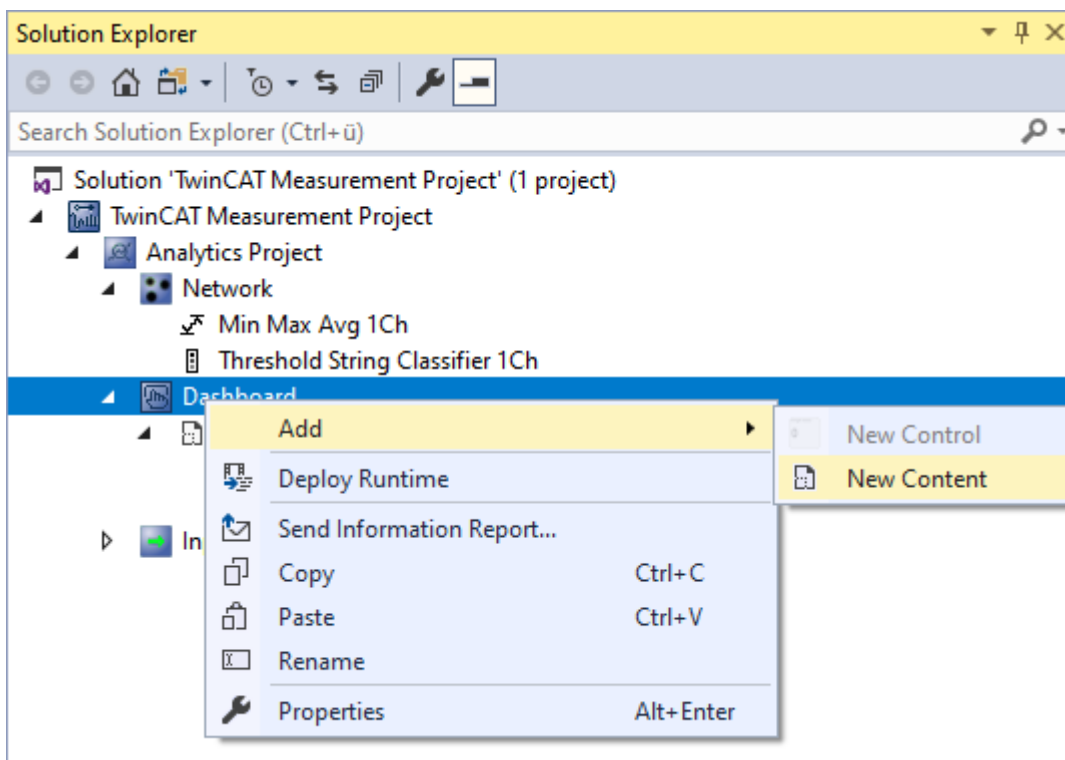
An Analytics project has a configuration of HMI contents (pages) and HMI controls (display elements) that are created during dashboard generation. This configuration can be viewed and changed via the **Dashboard node** in the Solution Explorer. You can rename the contents and controls at any time, move them to other contents via drag & drop, copy them (**Ctrl-C**, **Ctrl-V**) or delete them. A control can also be edited by double-clicking on it.



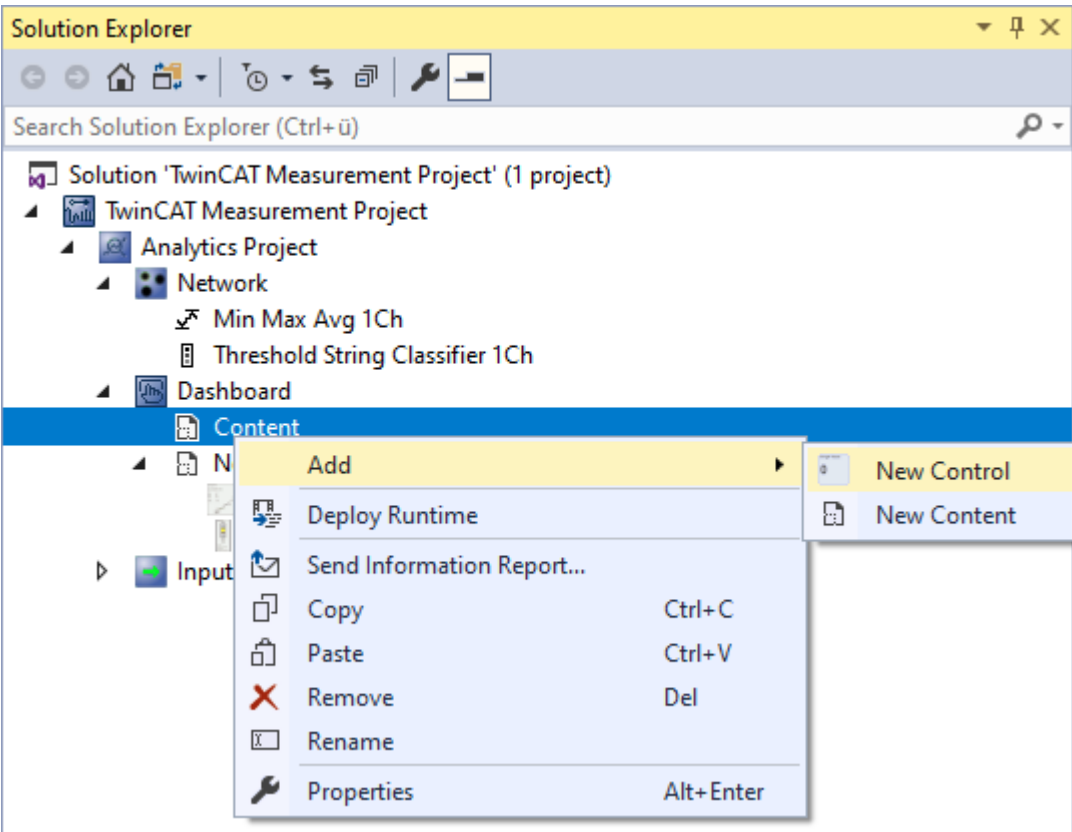
Each Analytics module has existing control mappings, which can be selected via the window **Properties** (you can also create/edit these yourself via "Create new Mapping Template" or via the Use customized and own controls [► 382] (point 2)). Once a control is selected for a module, it is listed under the **Dashboard** node.



You can create new content for the **Dashboard** and **Content** nodes by right-clicking.



Likewise, you can add **new controls** by right-clicking on a **Content** node.



This opens the **Analytics Dashboard Wizard**. This wizard guides you step-by-step through the configuration for adding a control.

i Analytics Dashboard Wizard

Familiarize yourself with the wizard by adding a control (select a control from the Properties window of the module). You can then look up the entire configuration in the wizard by double-clicking on the control. There you can follow the individual configuration steps.

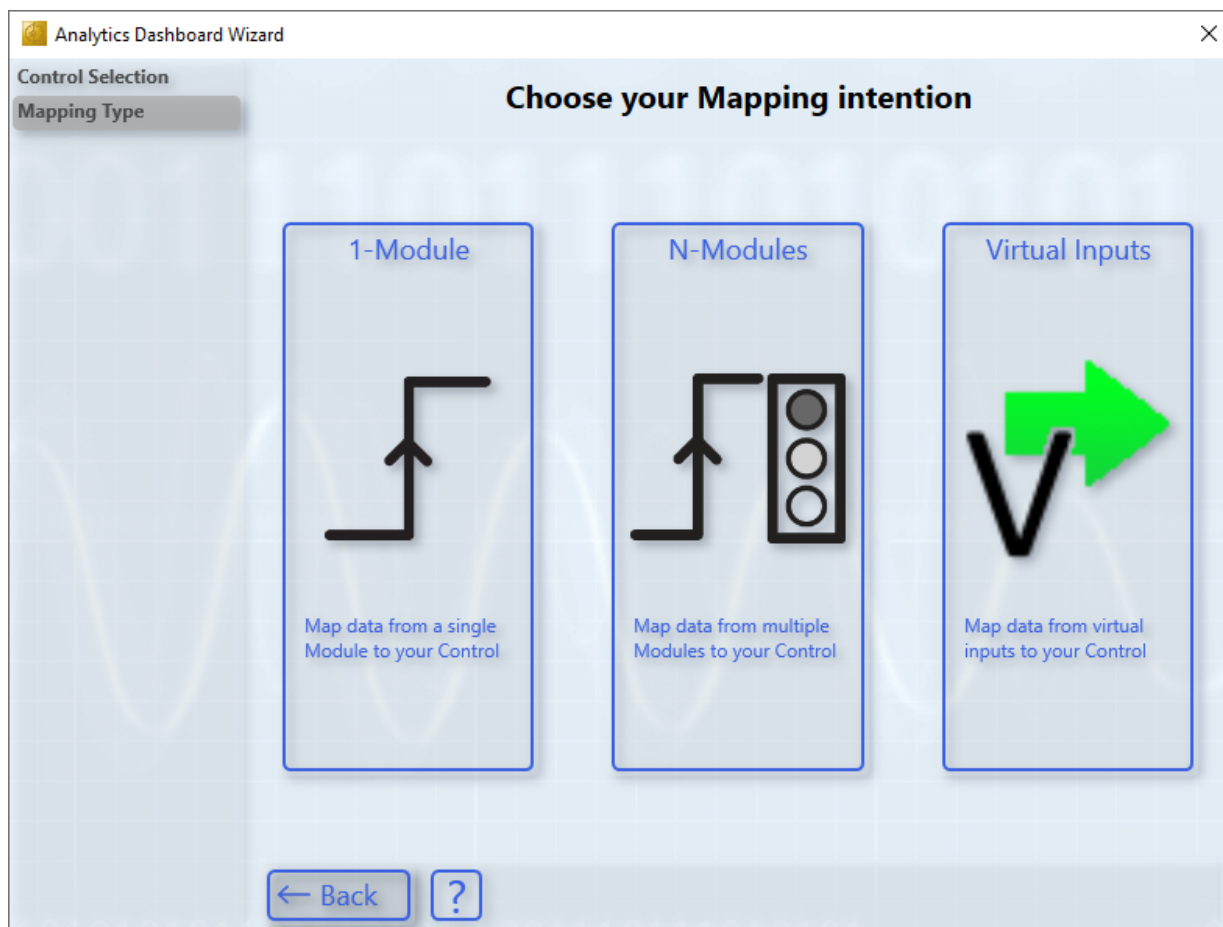
Analytics Dashboard Wizard

Select a control. All available controls are listed on the left. By default, only controls that are not algorithm-specific are listed. All controls can be made available by unchecking the **Show only default Controls** checkbox.

1. Select **Single Value** to display a single value in the dashboard. Click the **Next** button to continue.



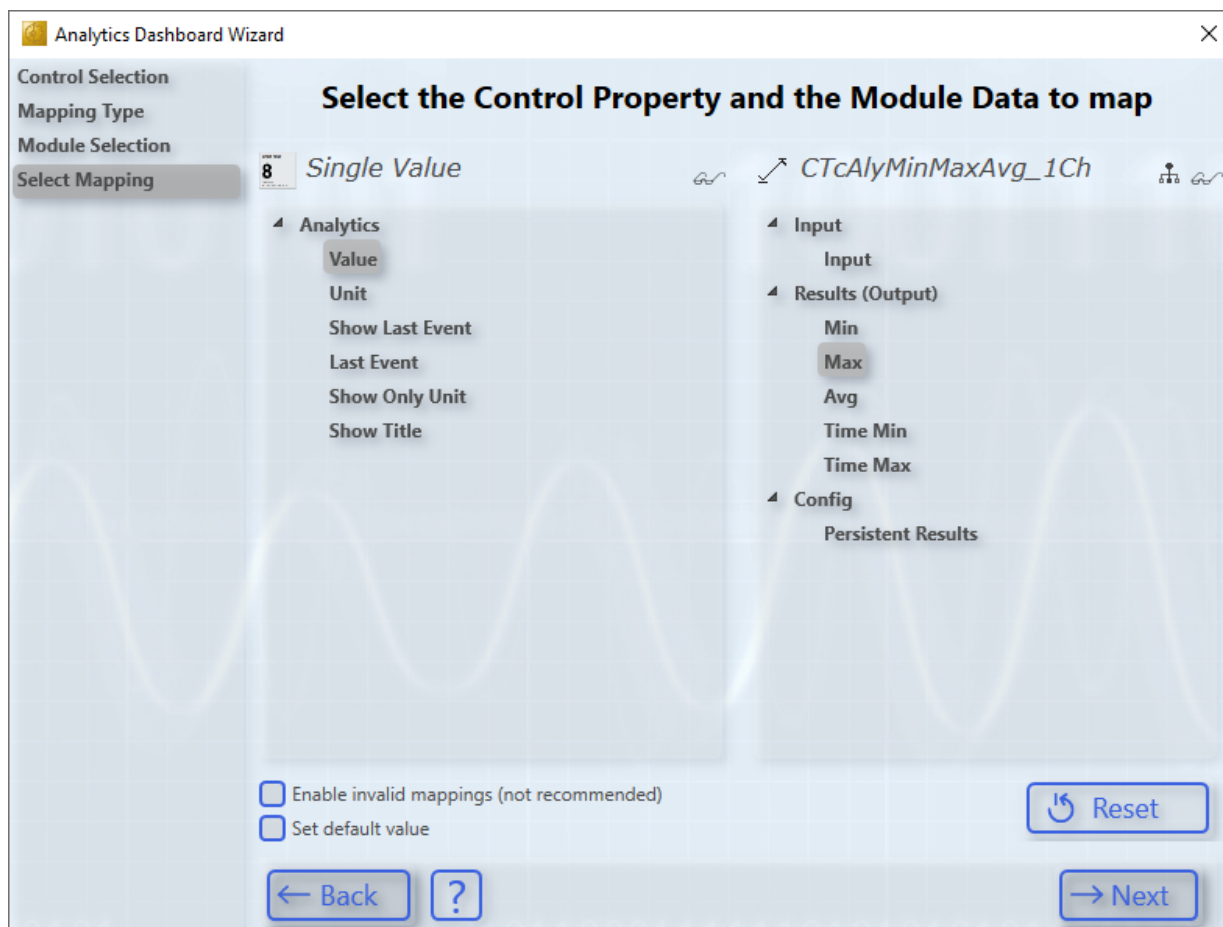
2. You can now link data from one module (**1-Module**), several modules (**N-Modules**) or from virtual inputs (**Virtual Inputs**). For this example, select **1-Module**.



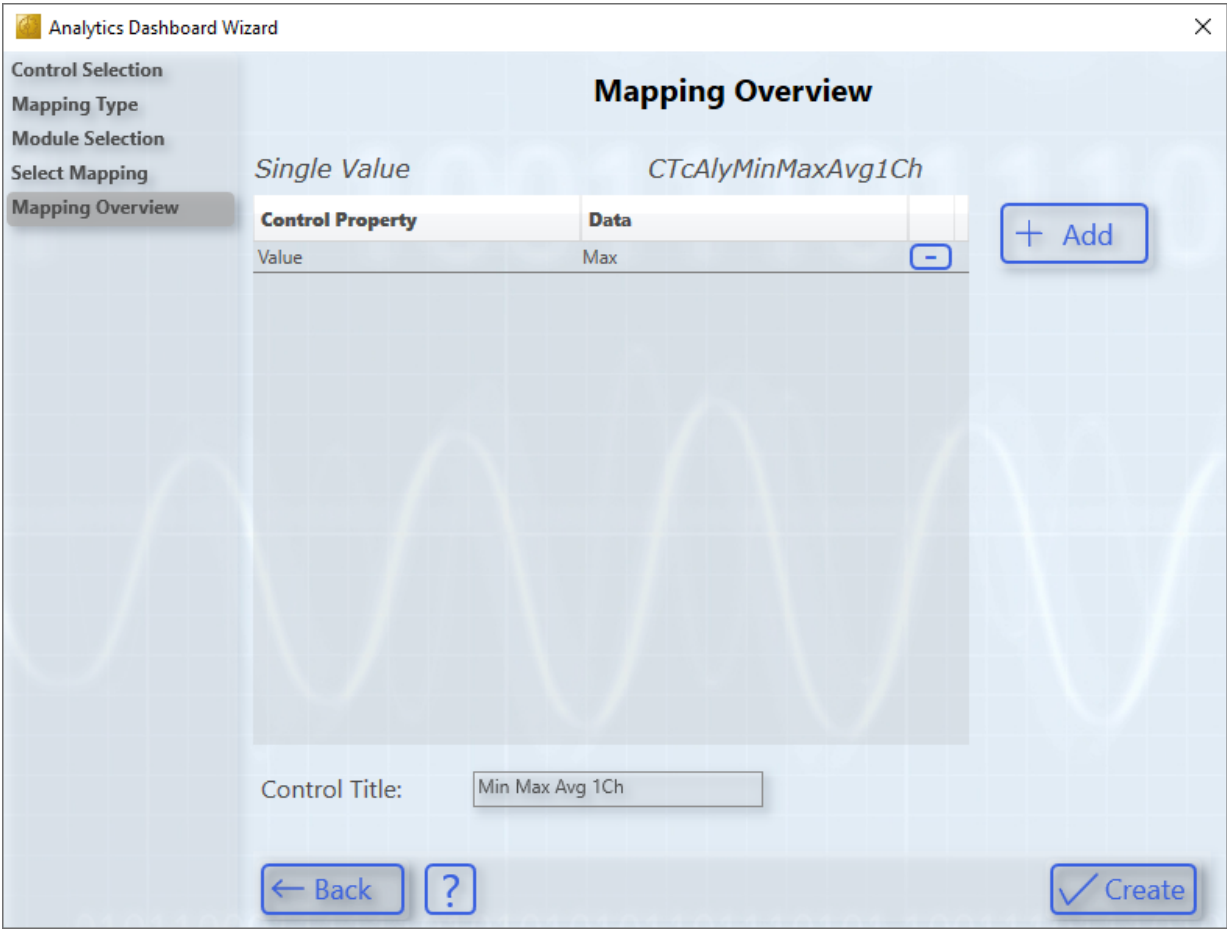
3. All modules from the Analytics project are listed. Select the module from which you want to display the data. A preview of the selected module is available on the right.



4. Here the first control property is linked to a module variable. To do this, select the **Value** to be displayed with the variable **Max**.



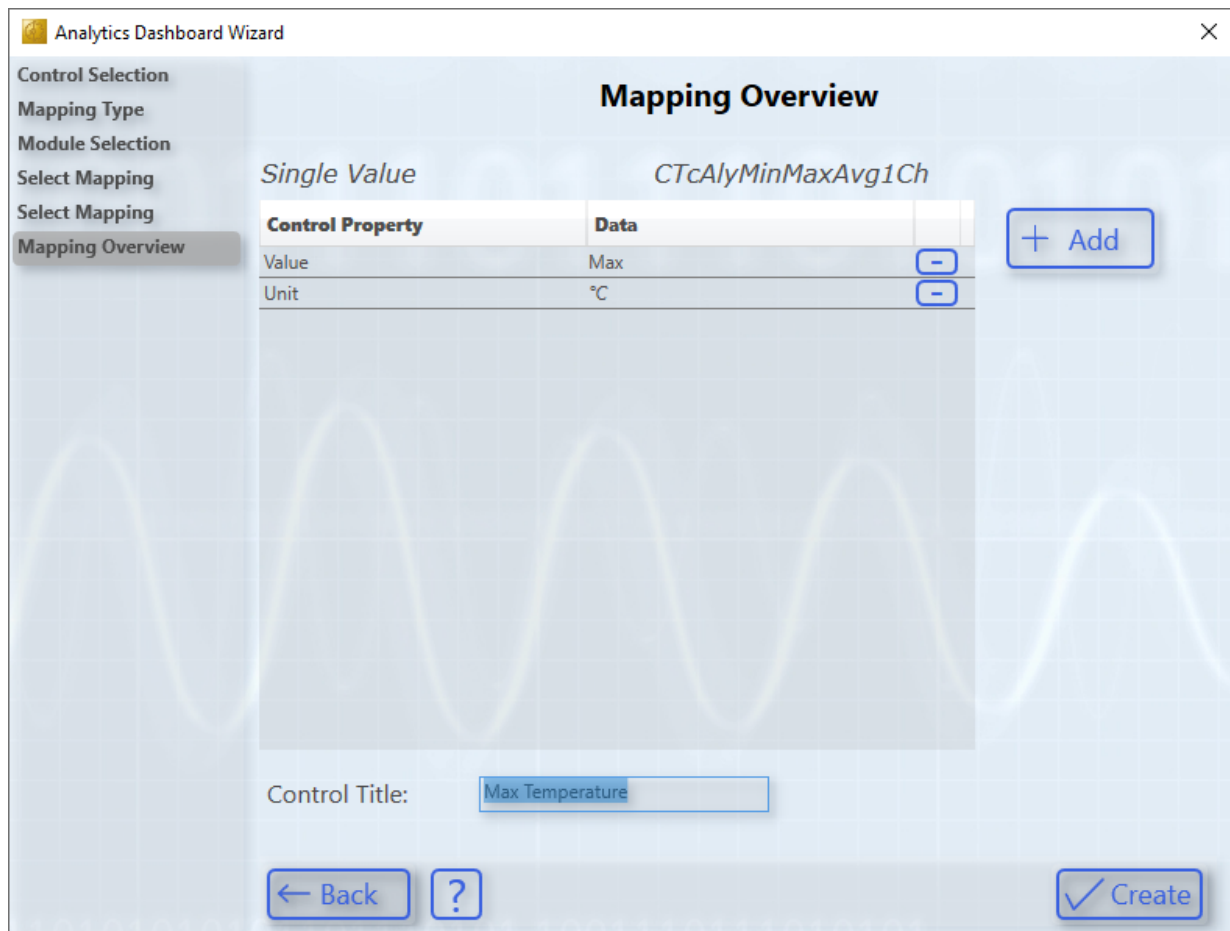
5. This overview page lists all existing links. Add another link with **Add**.



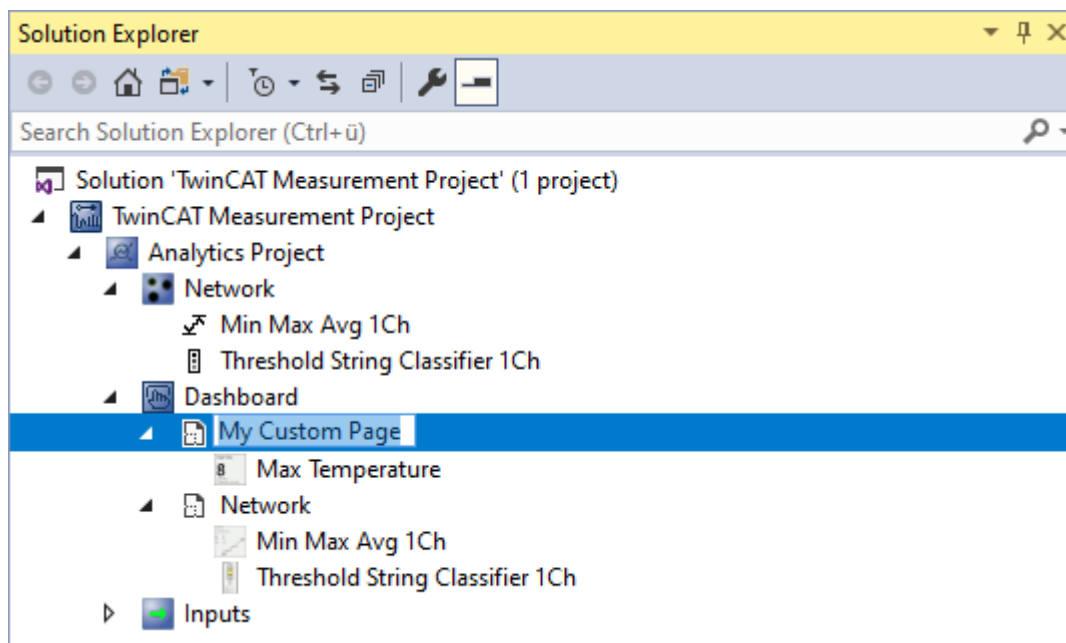
6. Select **Unit** and check **Set default value** to assign a static value. A text field opens on the right; enter **°C**.

The screenshot shows the 'Analytics Dashboard Wizard' window. The left sidebar contains the following steps: Control Selection, Mapping Type, Module Selection, Select Mapping, and Select Mapping (highlighted). The main area is titled 'Select the Control Property and the Module Data to map'. It shows a tree view under 'Analytics' with the following options: Value, Unit (selected), Show Last Event, Last Event, Show Only Unit, and Show Title. To the right of the tree, there is a 'Default Value:' label and a text input field containing '°C'. At the bottom, there are two checkboxes: 'Enable invalid mappings (not recommended)' (unchecked) and 'Set default value' (checked). There are also three buttons: 'Back', a help icon (?), and 'Next'. A 'Reset' button is also present near the bottom right.

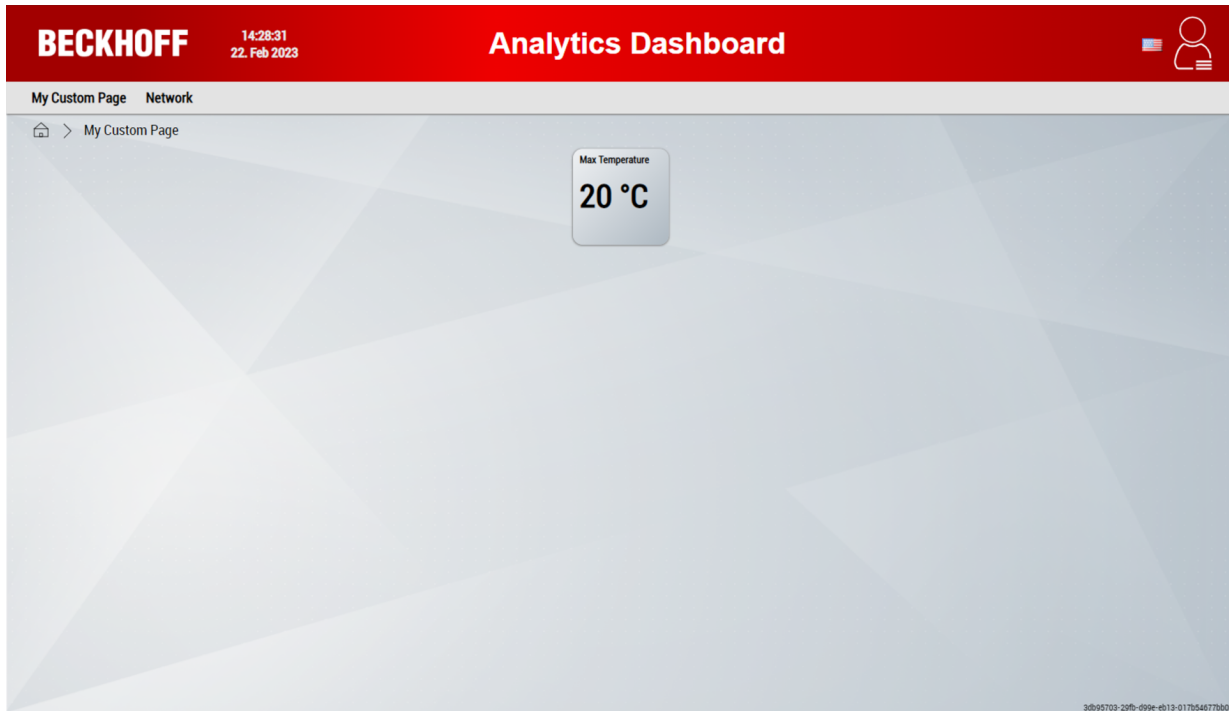
7. Change the title of the control to **Max Temperature** and add the control mapping via **Create**.



8. The **Max Temperature** control that was created appears in the **Dashboard** node. **Right-click** > **Rename** to rename the content.



- ⇒ After a successful HMI and PLC generation, you can open the previously created **My Custom Page** via the navigation. There you can see the manually created control.



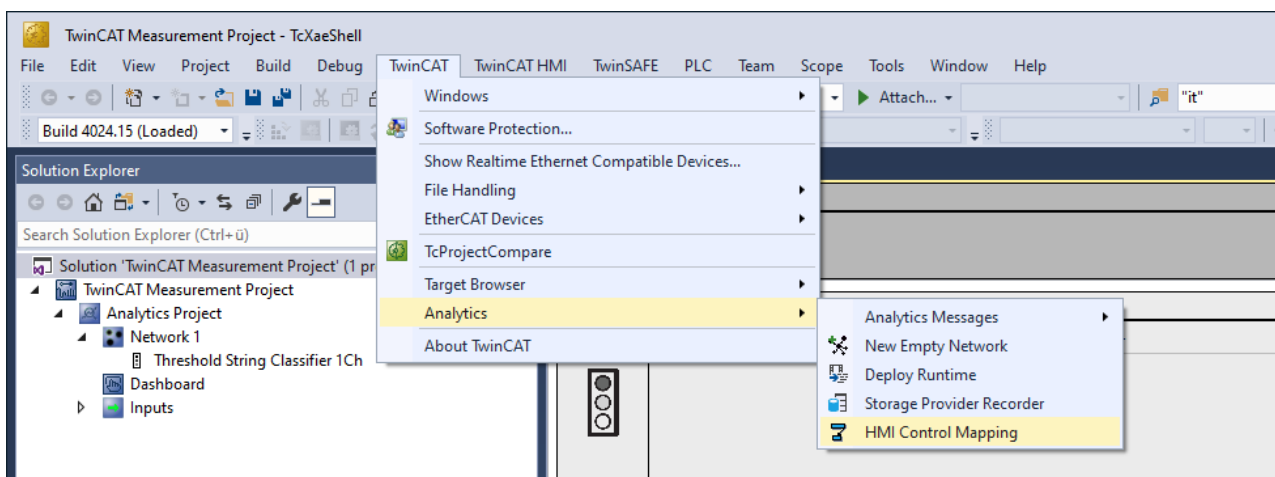
5.6.2 Use customized and own controls

HMI Control Mapping Wizard

The HMI Control Mapping wizard enables the following:

1. Adding your own controls.
2. Mapping controls to module classes or changing existing mappings.
3. Mapping controls to module instances or virtual inputs (Analytics project must be open)

Open the wizard via the tab **TwinCAT > Analytics > HMI Control Mapping**.

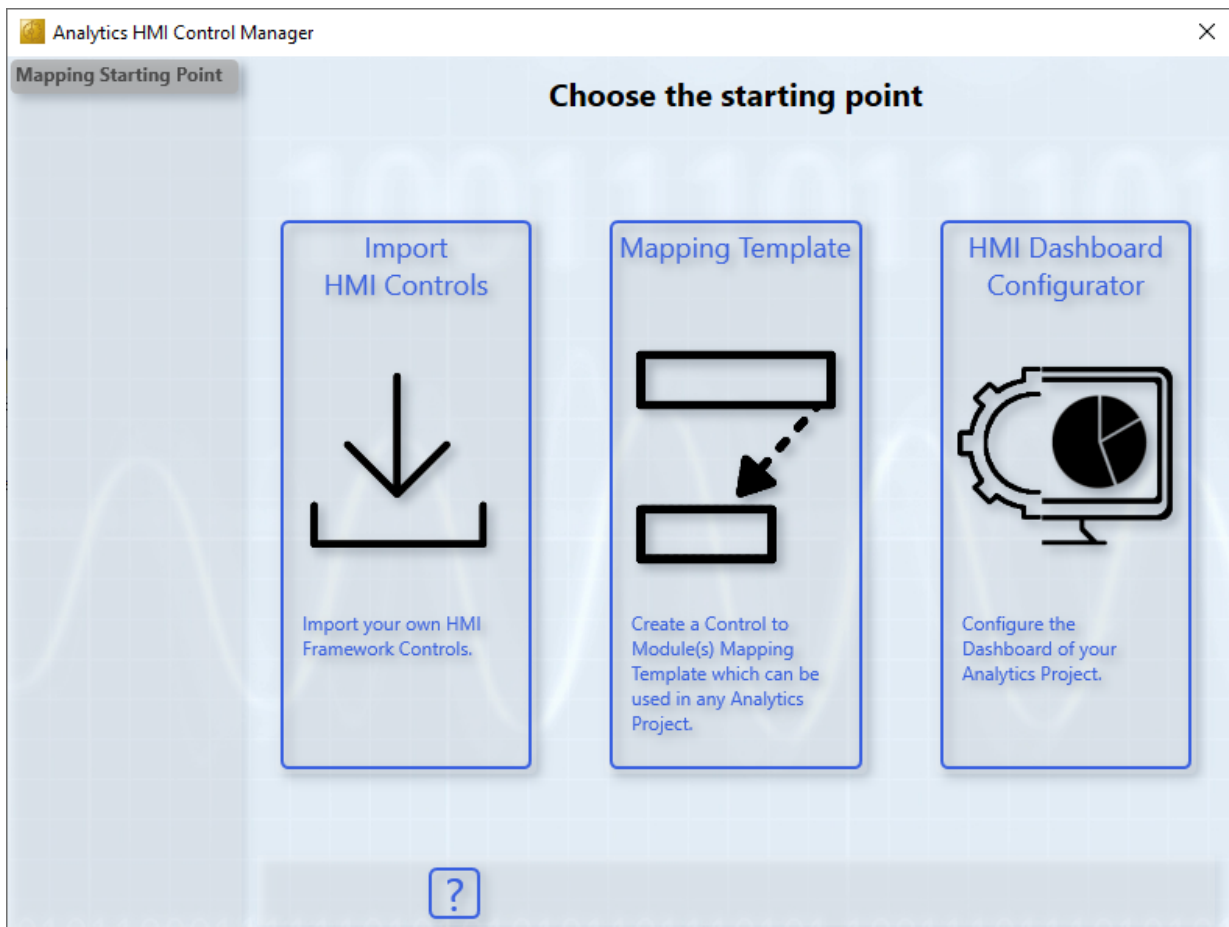


Additional help is offered via the question mark ? in the wizard.

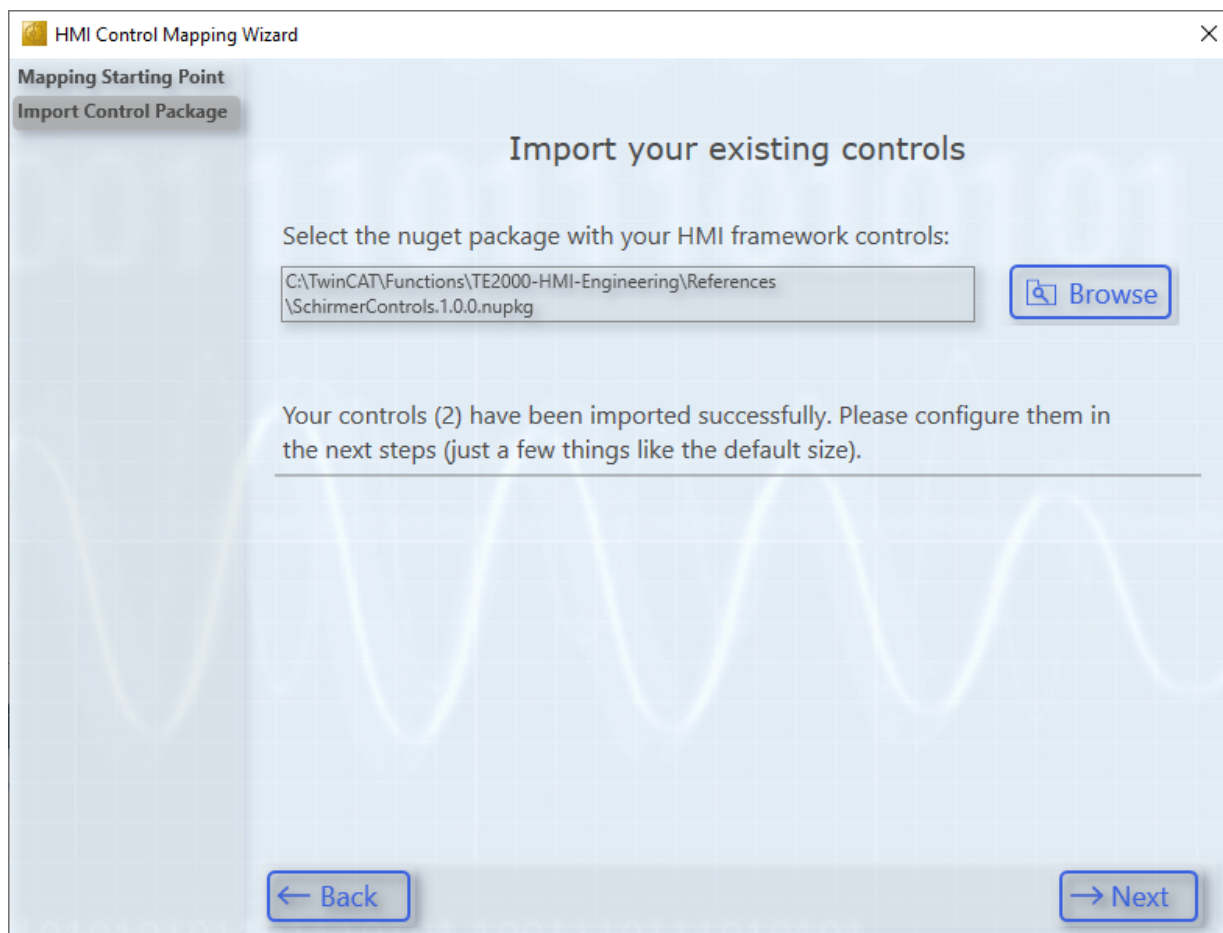
1. Adding your own Controls

The TwinCAT 3 HMI allows you to create your own HMI Framework controls and export them as a NuGet package.

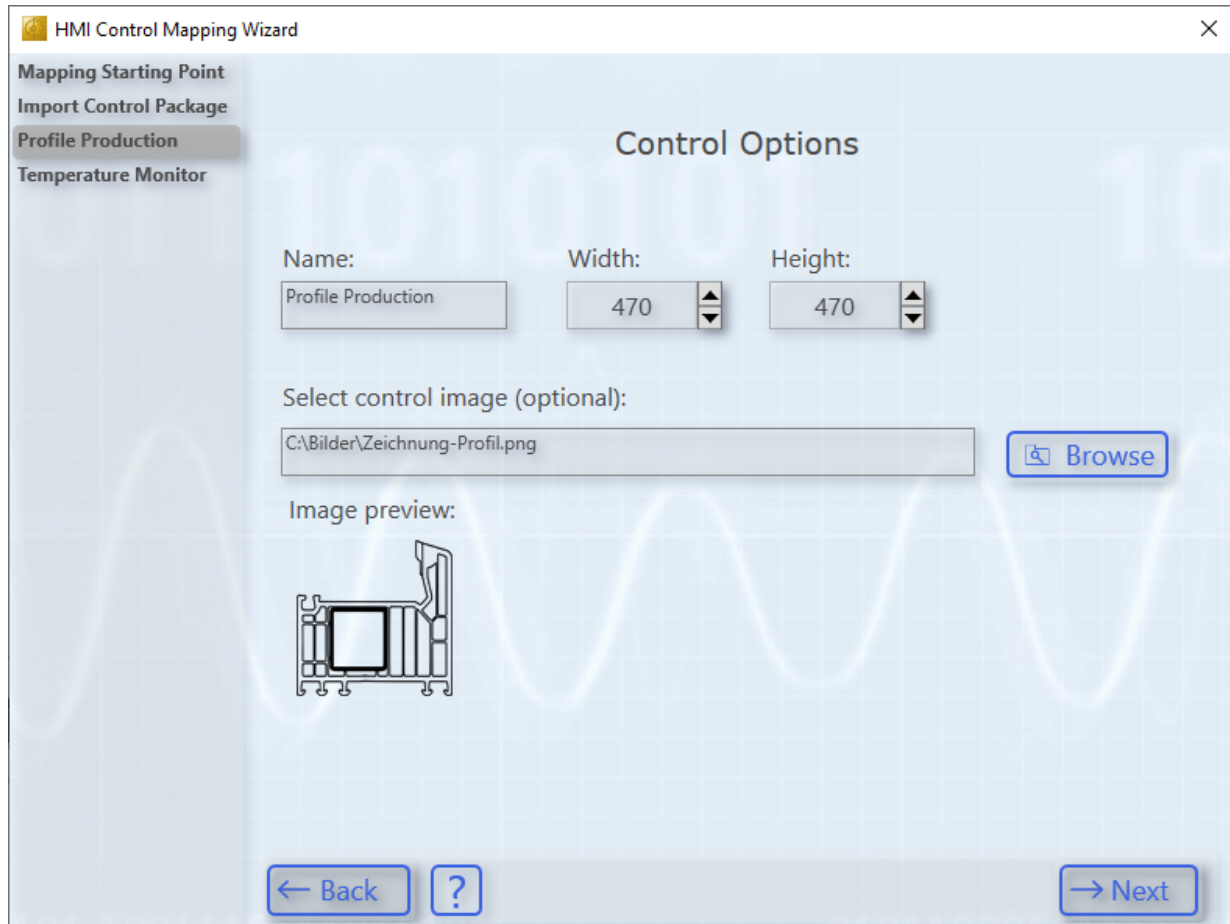
1. To assign your own framework controls to the Analytics modules, click **Import HMI Controls**



2. Select the NuGet package via **Browse**.



3. Next, choose a name, size and image for your control. Then click **Next** and for the last control click **Create**.



HMI Control Mapping Wizard

Mapping Starting Point
Import Control Package
Profile Production
Temperature Monitor

Control Options

Name: Profile Production Width: 470 Height: 470

Select control image (optional):
C:\Bilder\Zeichnung-Profil.png Browse

Image preview:

← Back ? → Next

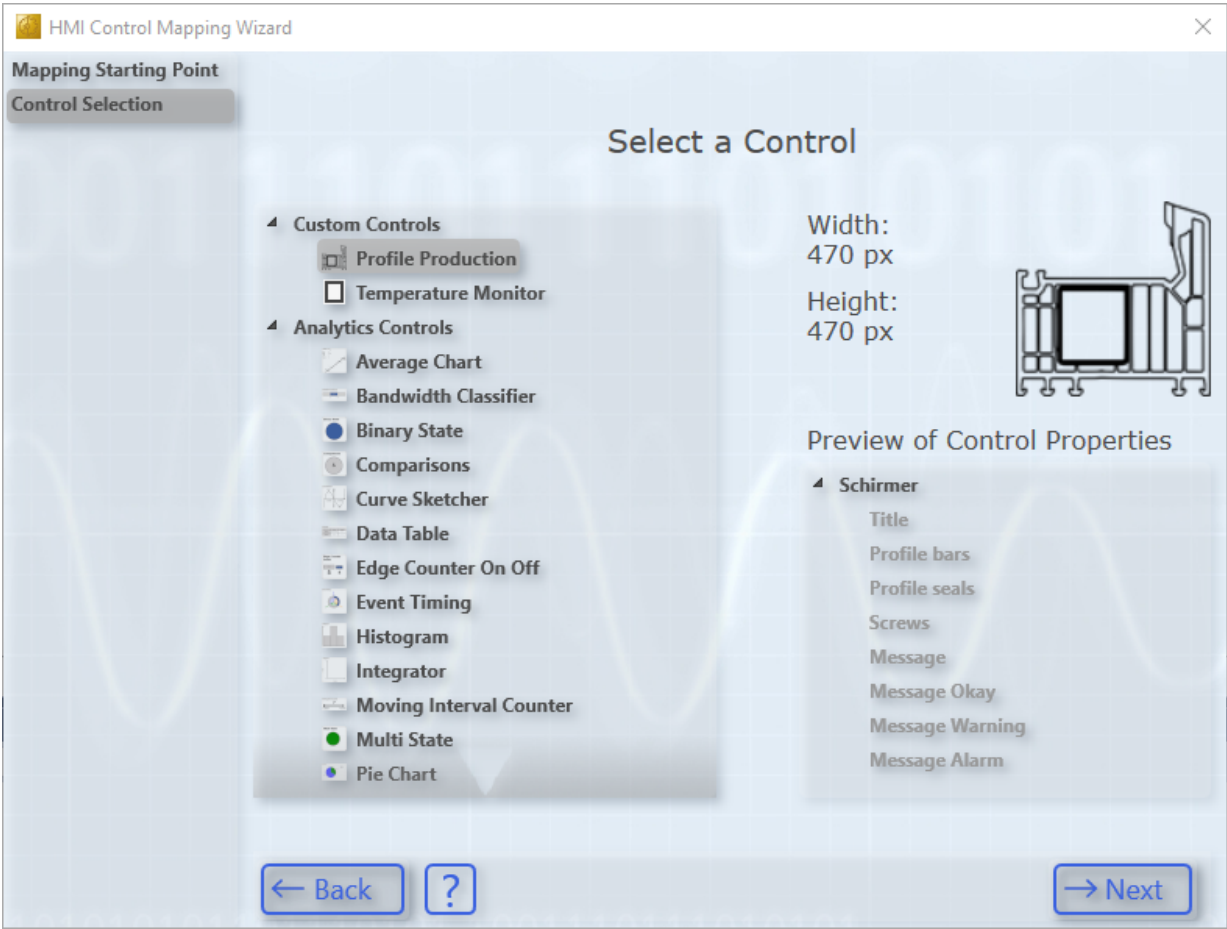
- ⇒ You will automatically be redirected so that you can create a mapping between your controls and Analytics modules.

2. a) Mapping of controls to modules (further to 1.)

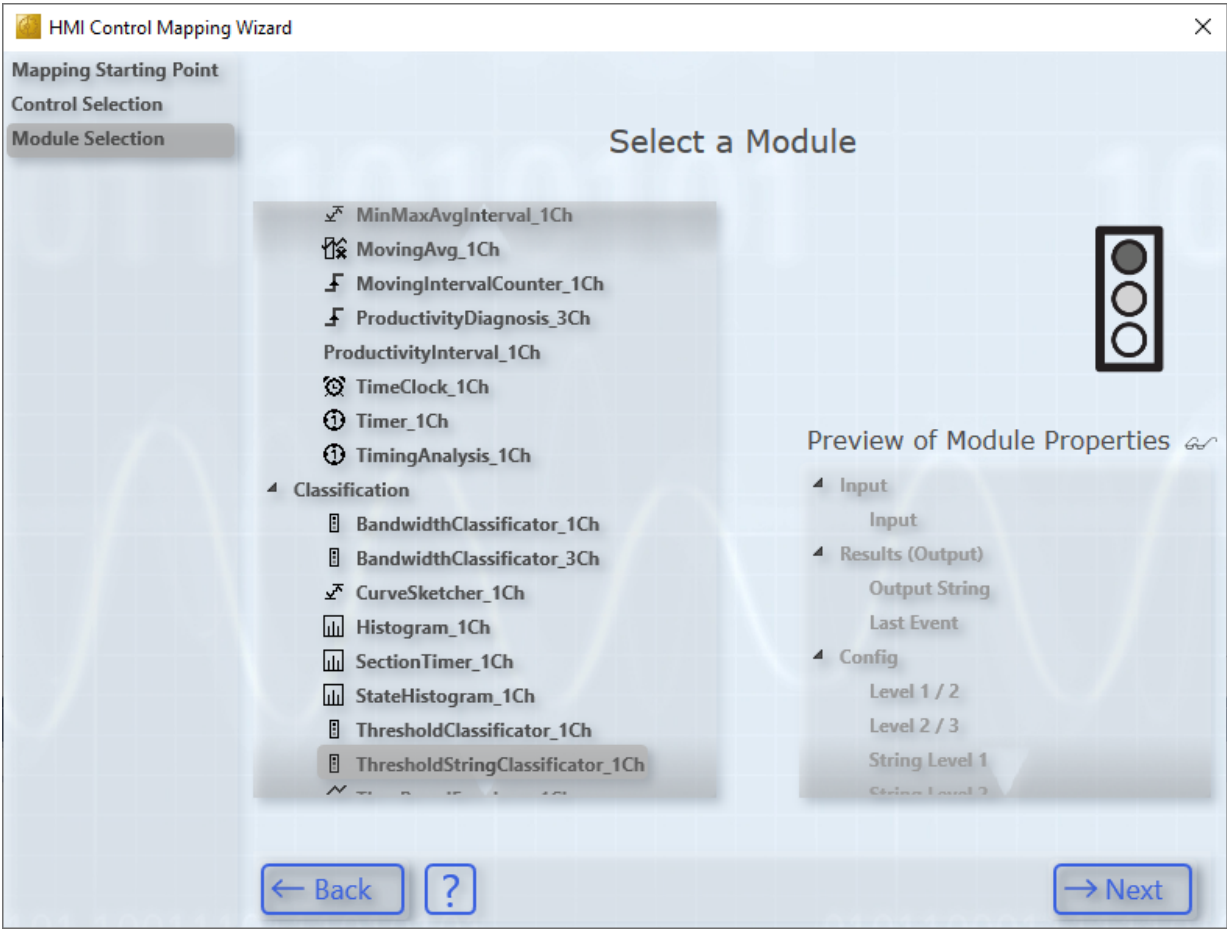


Under 2. b) this step is explained in more detail using "Binary State" control.

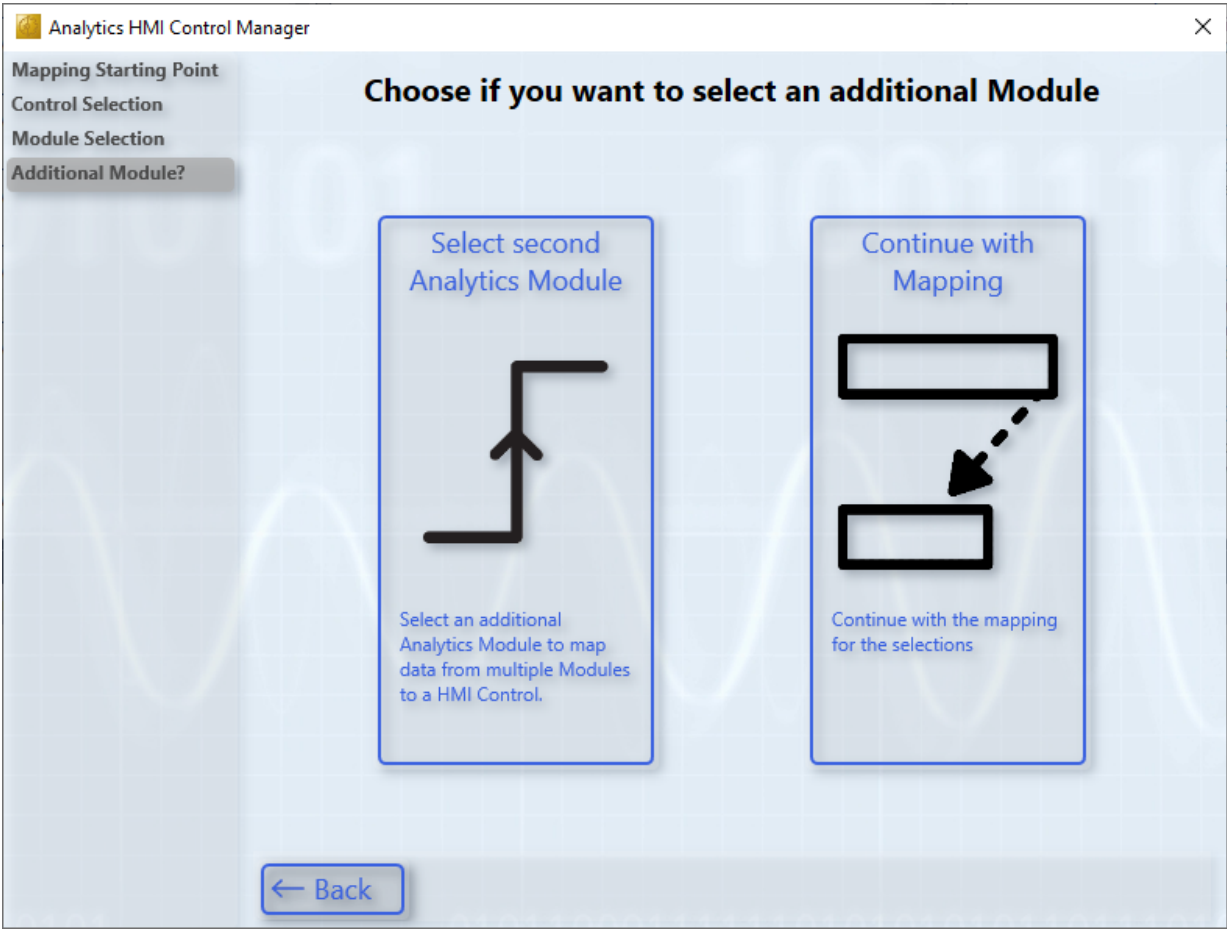
1. Now select your control.



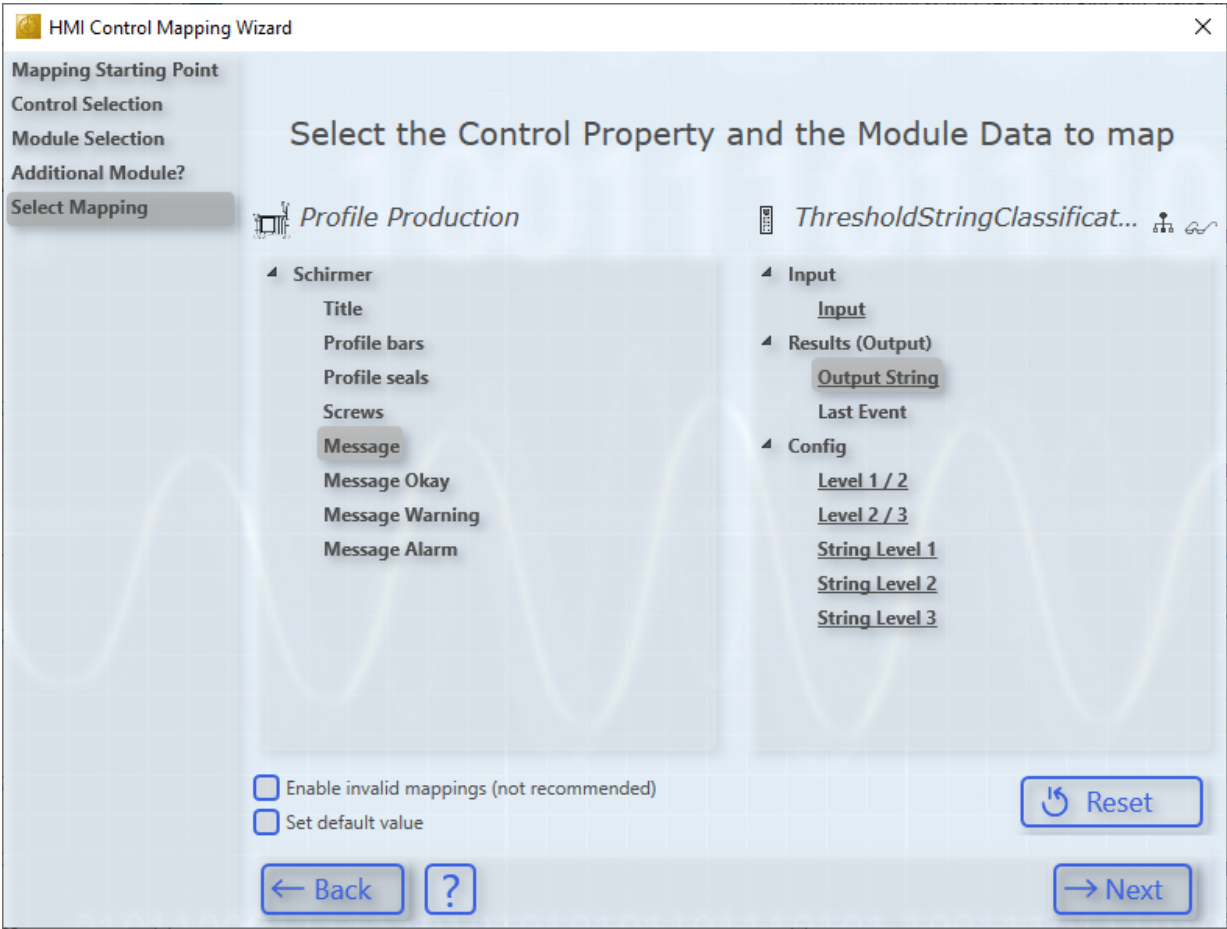
2. Now select the module to which you want to assign the control.



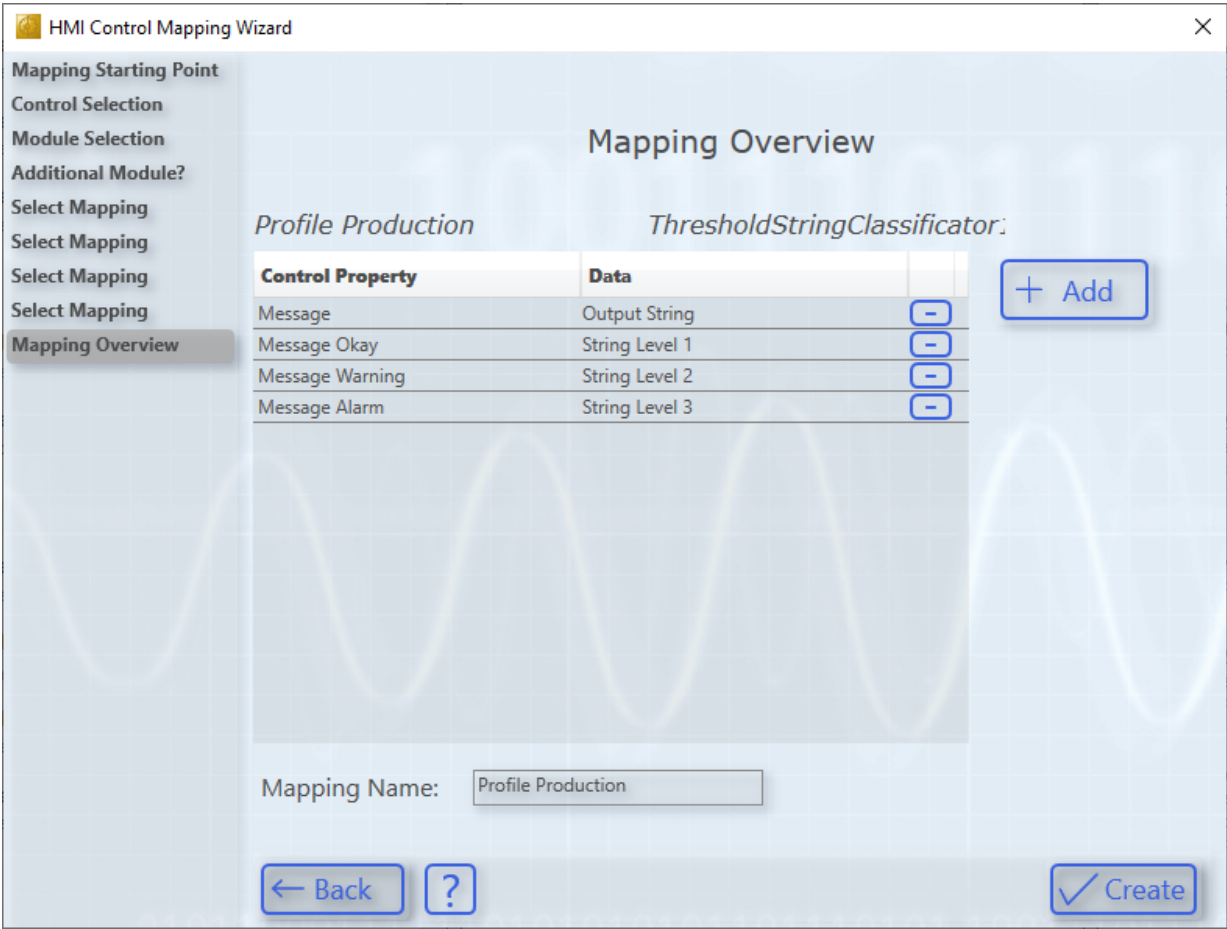
3. Select **Continue with Mapping**.



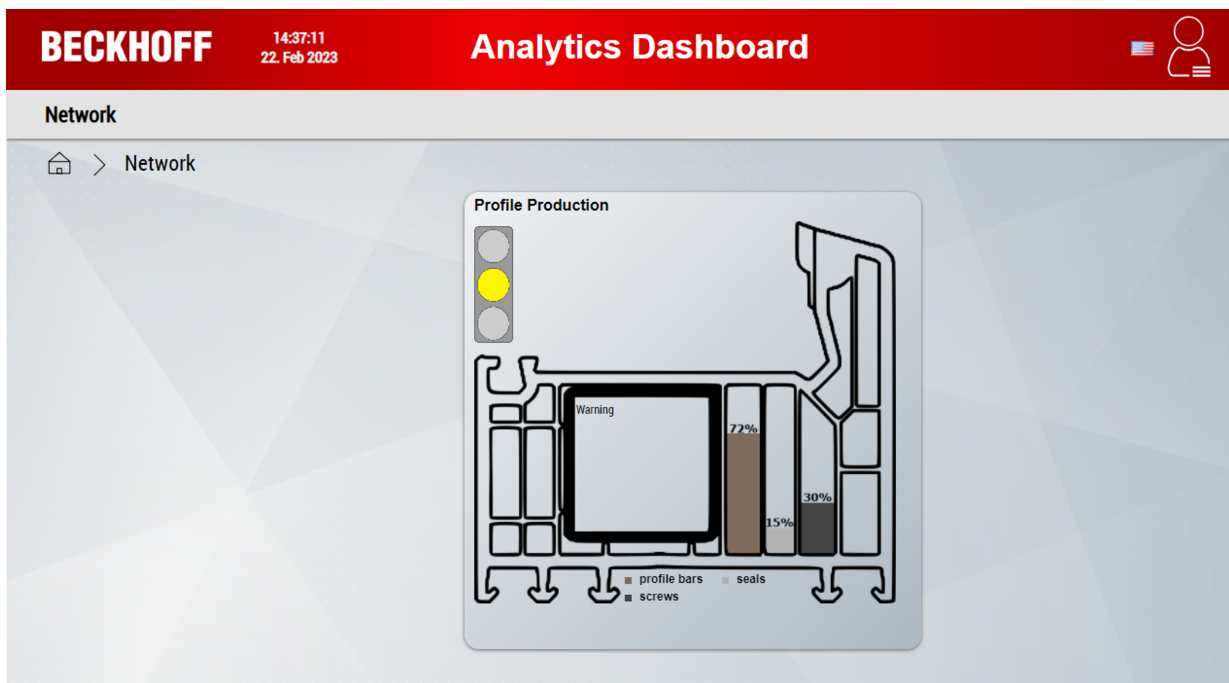
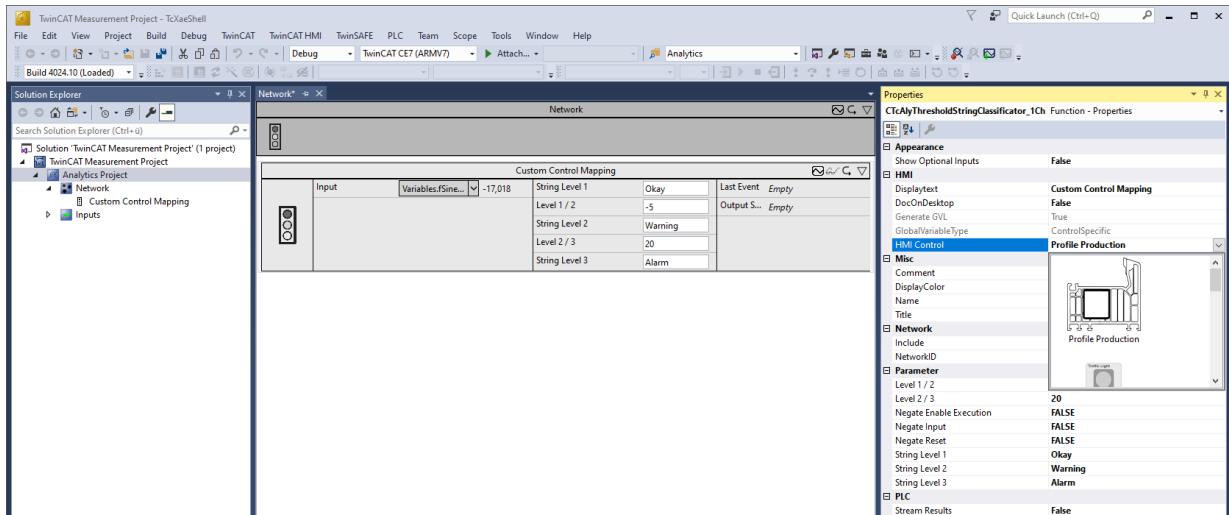
4. In the next steps, connect the control inputs to the module data.



5. Last, click **Create** to add the mapping to the Analytics workbench.



- You can now close the wizard and the mapping will automatically be available for the module. Select it and generate your dashboard.

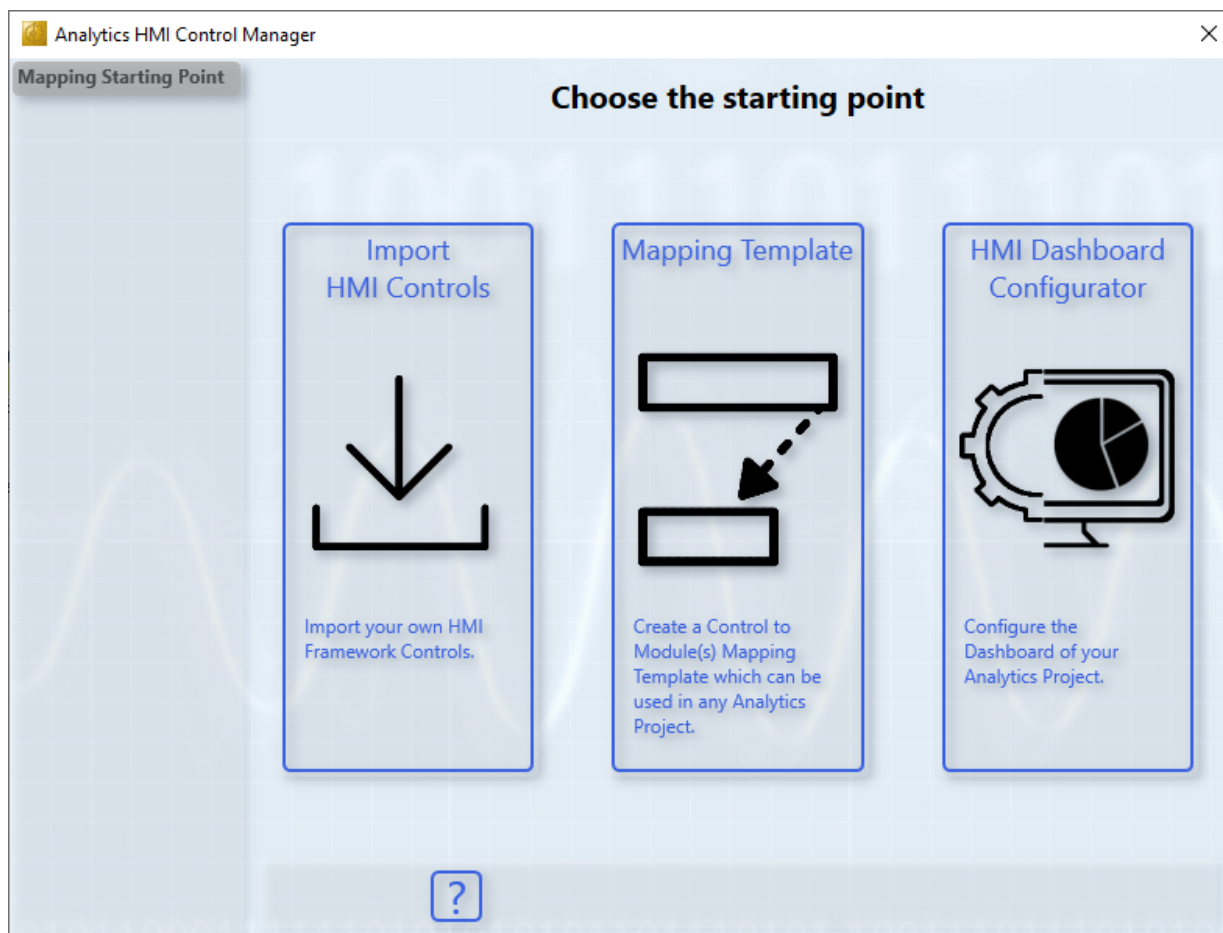


⇒ The dashboard is ready.

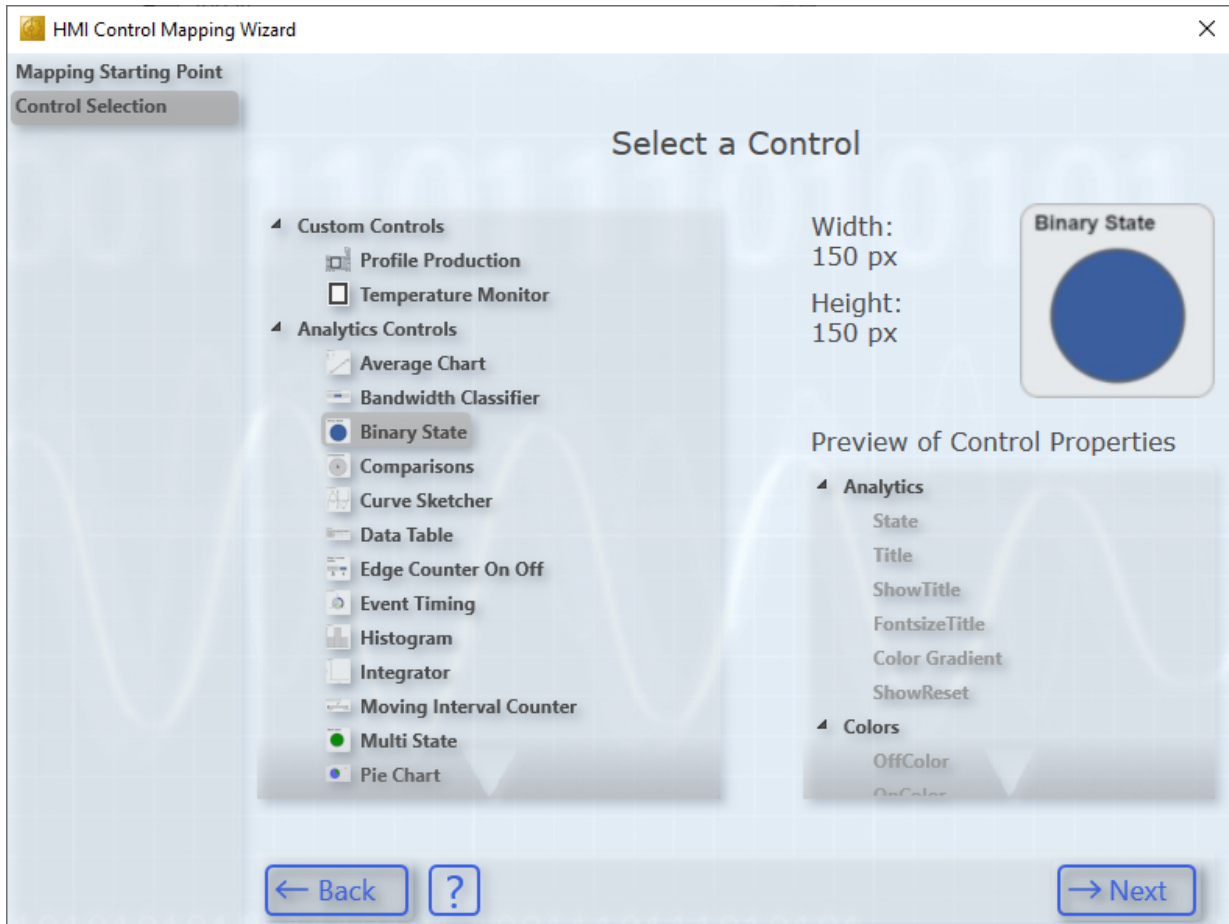
2.b) Mapping controls to modules (without own controls)

Analogous to 2.a), this section describes how to perform a mapping between a control and a module. An existing Analytics Control is used as an example. You can try this example directly.

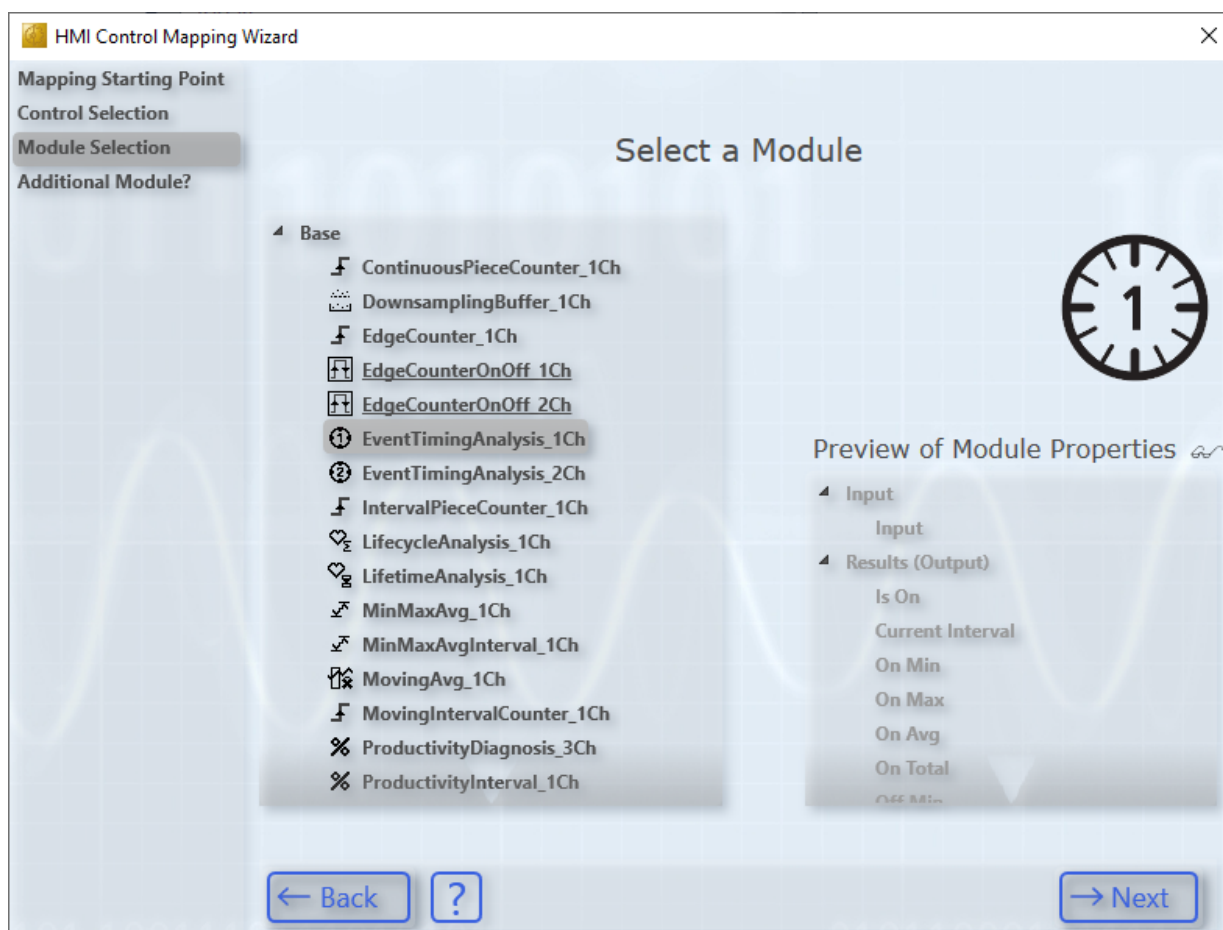
1. To do this, select the item **Mapping Template**.



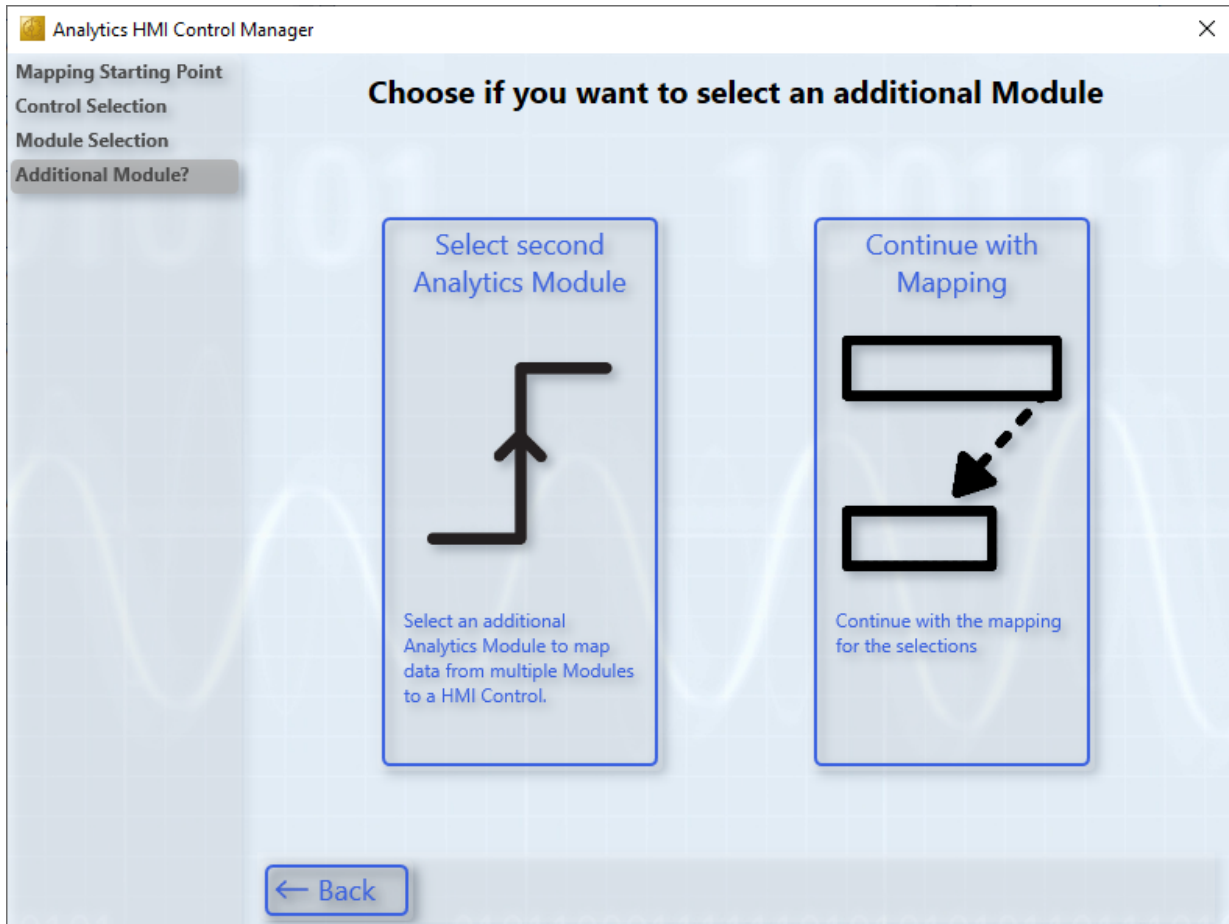
2. Select a control. The properties of the control are displayed on the right. It also shows how big the control will be on the dashboard.



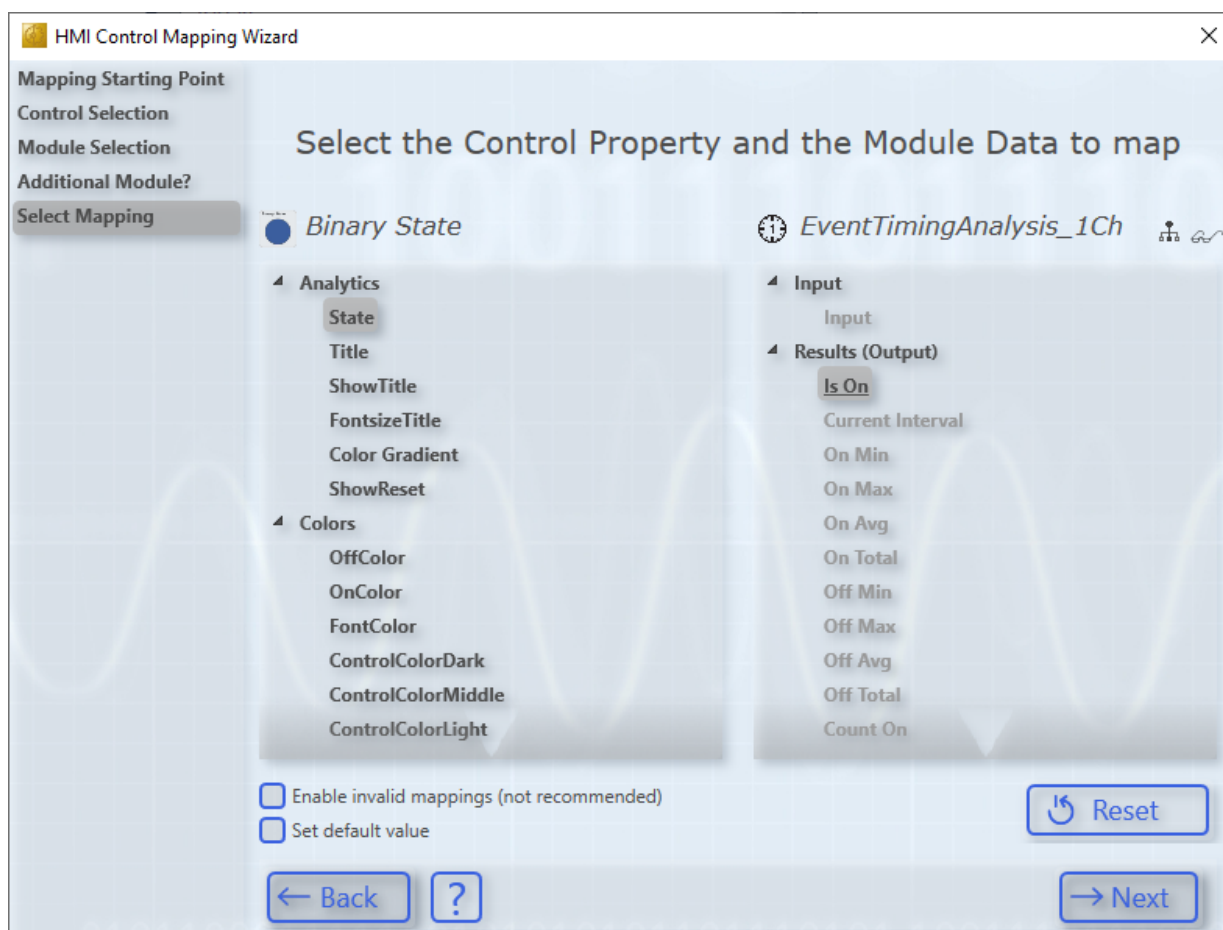
3. Select the module from which you want to display the data. The inputs, outputs and parameters are displayed as a preview, which can then be linked to the properties of the selected control. Modules that have already been mapped are underlined. These mappings can also be edited.



4. In the future, a control will be able to display data from different modules. Since this is not currently possible, select **Continue with Mapping**.

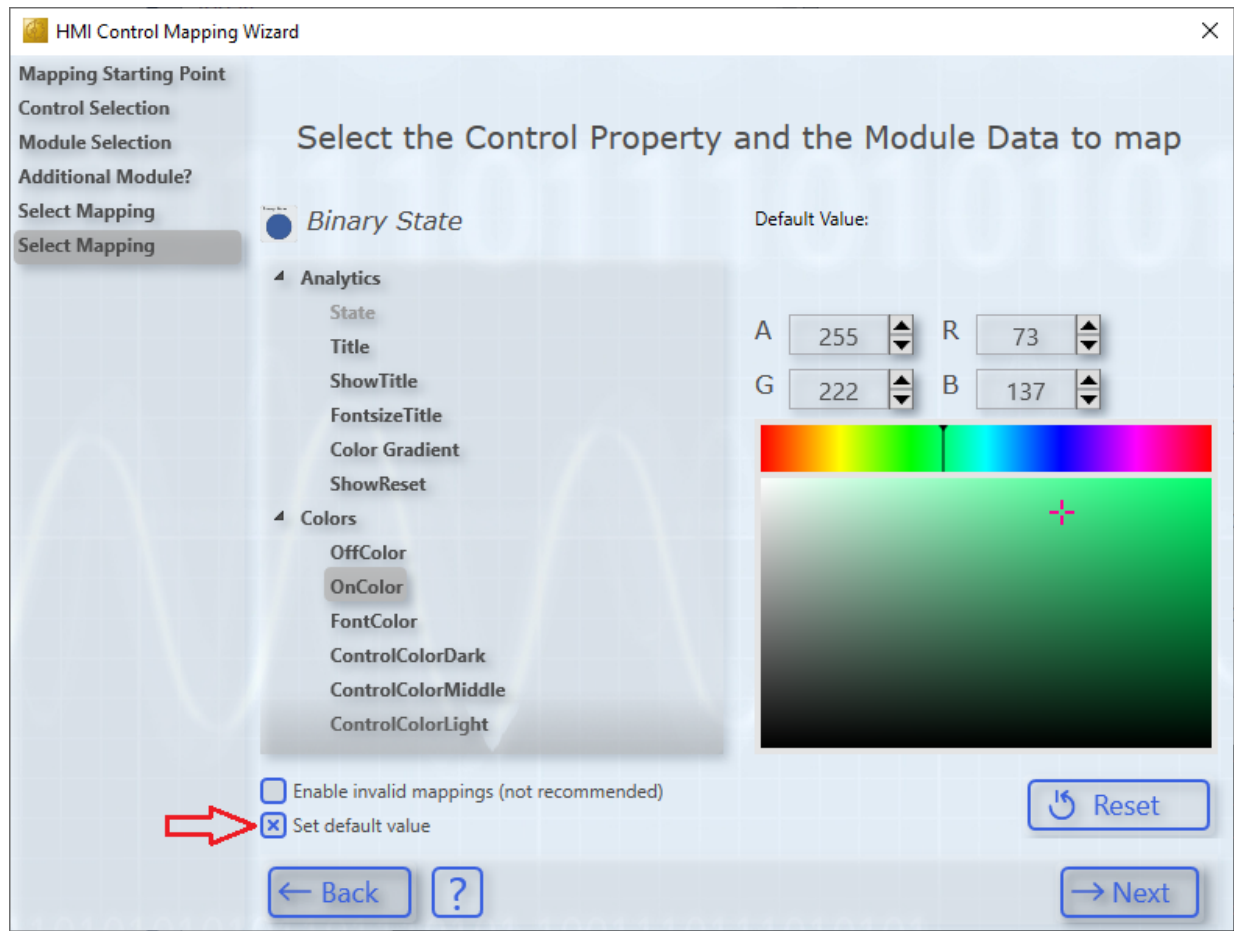


5. Next, select a control property to be mapped. All impossible variables that cannot be assigned due to their data type are grayed out. All others can be mapped.
Now select the module variable you want to map with the control property.

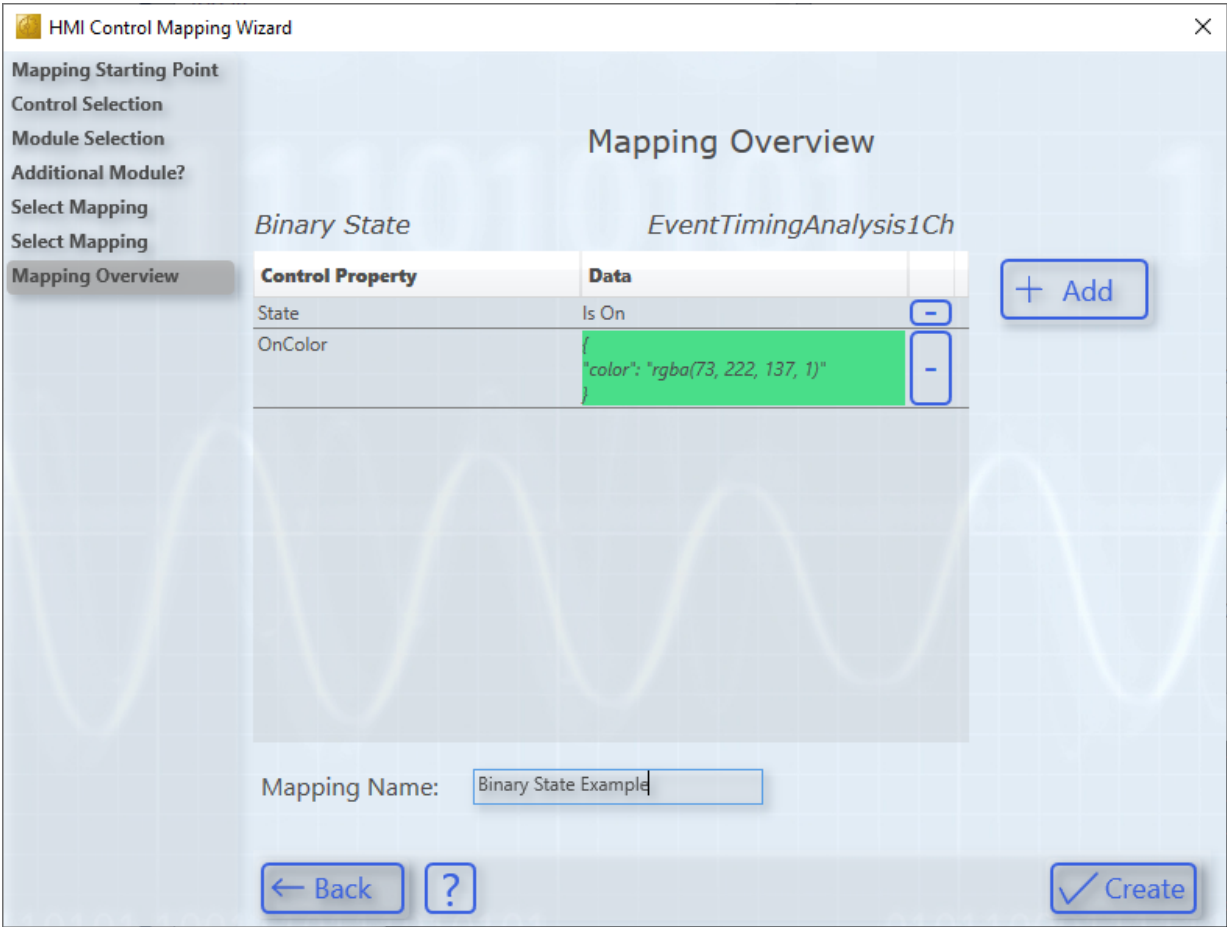


6. Likewise, you can check the **Set default value** checkbox to assign a default value to the control. This can be used, for example, to change default colors or to set Boolean values such as "ShowTitle" to "False" if no title is to be displayed in the Analytics control.

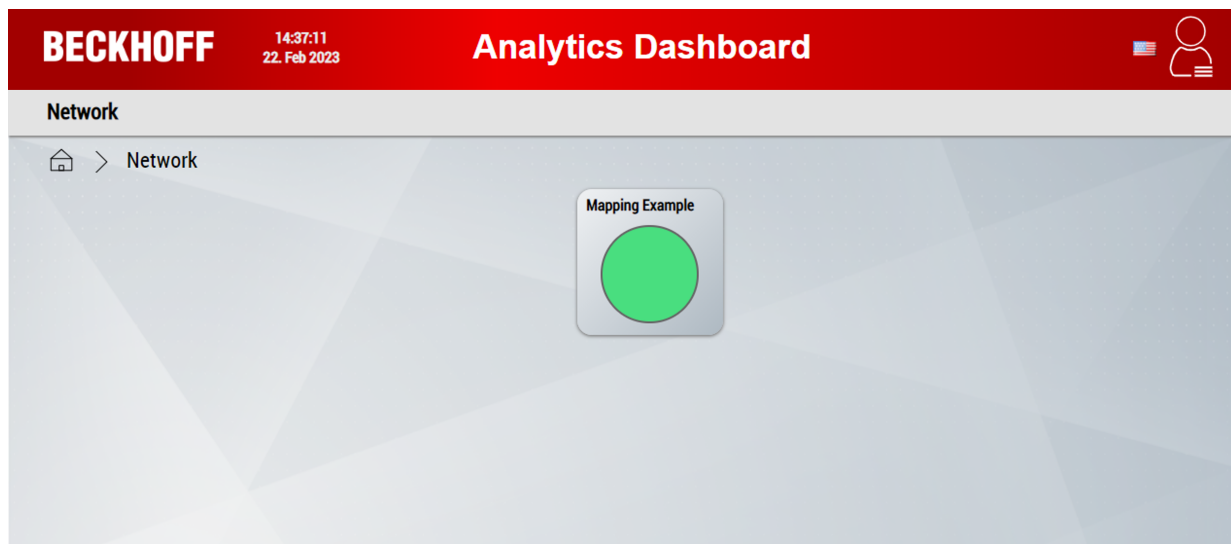
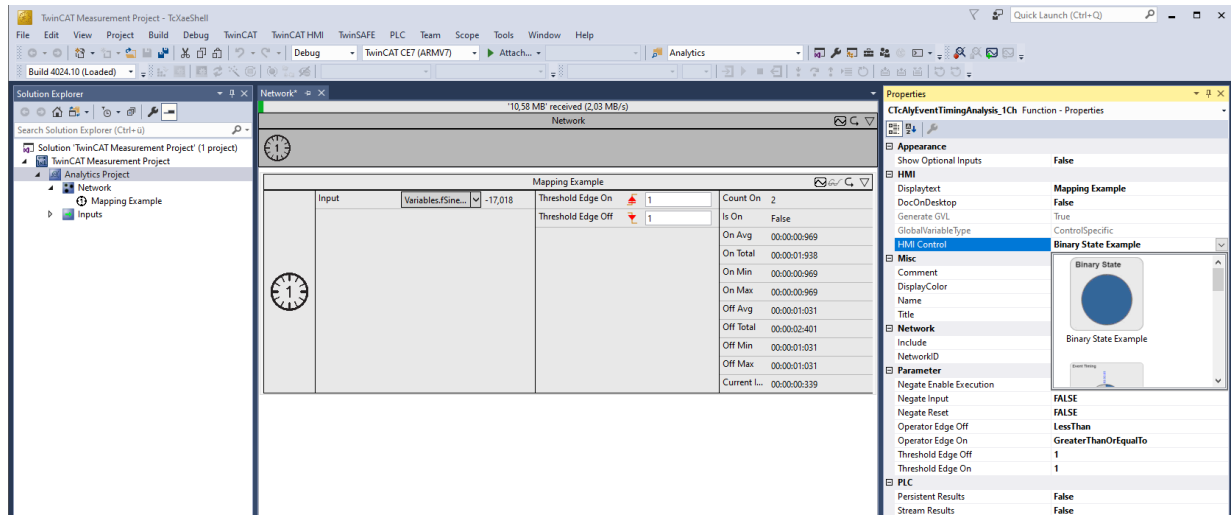
In this example, the "OnColor" is changed. Based on the data type, a specific selection option is provided.



7. As far as you do not want to add any more entries via **Add**, you can complete the mapping via **Create**. It is recommended to assign a meaningful mapping name.



8. You can close the wizard and the mapping will automatically be available for the module. Select it and generate your dashboard.

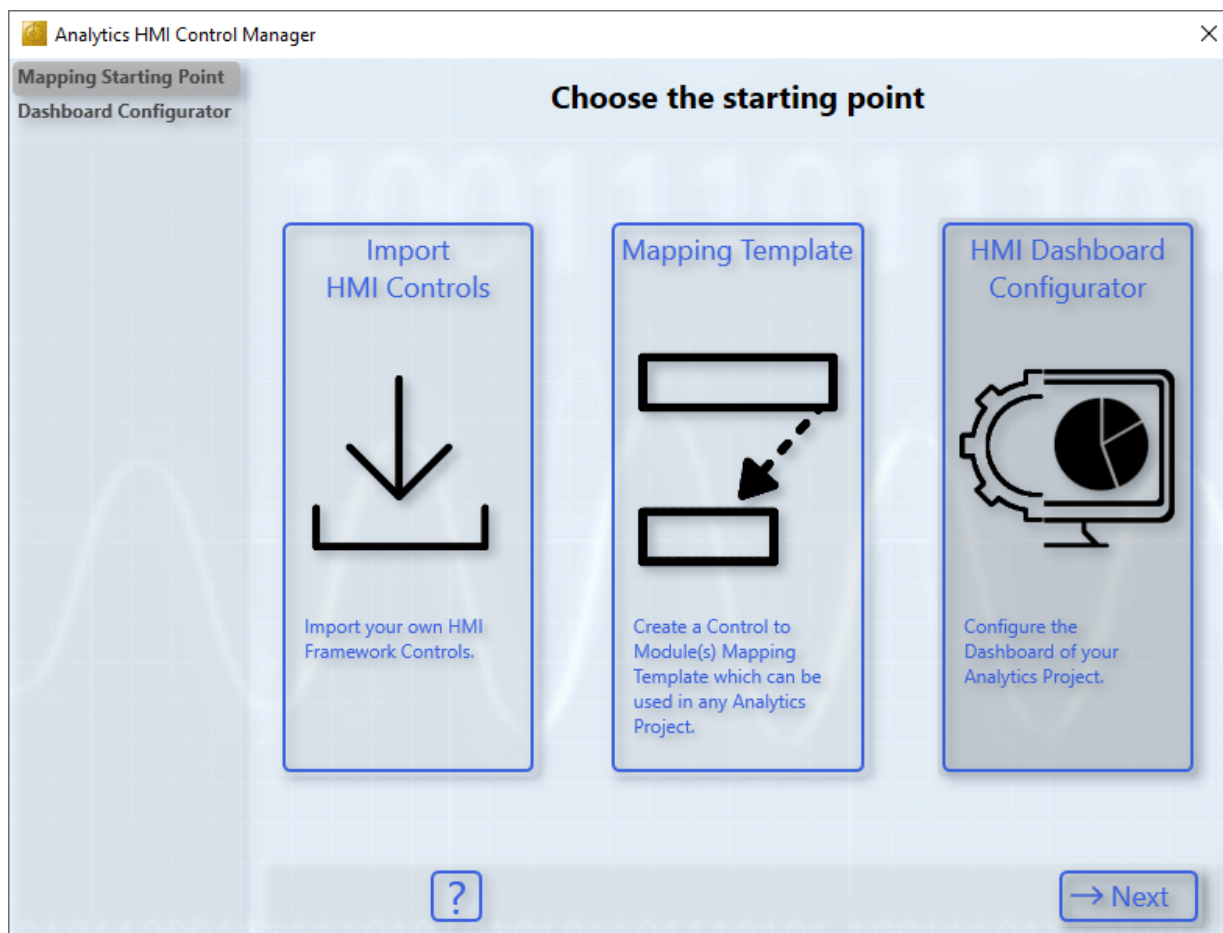


⇒ The dashboard is ready.

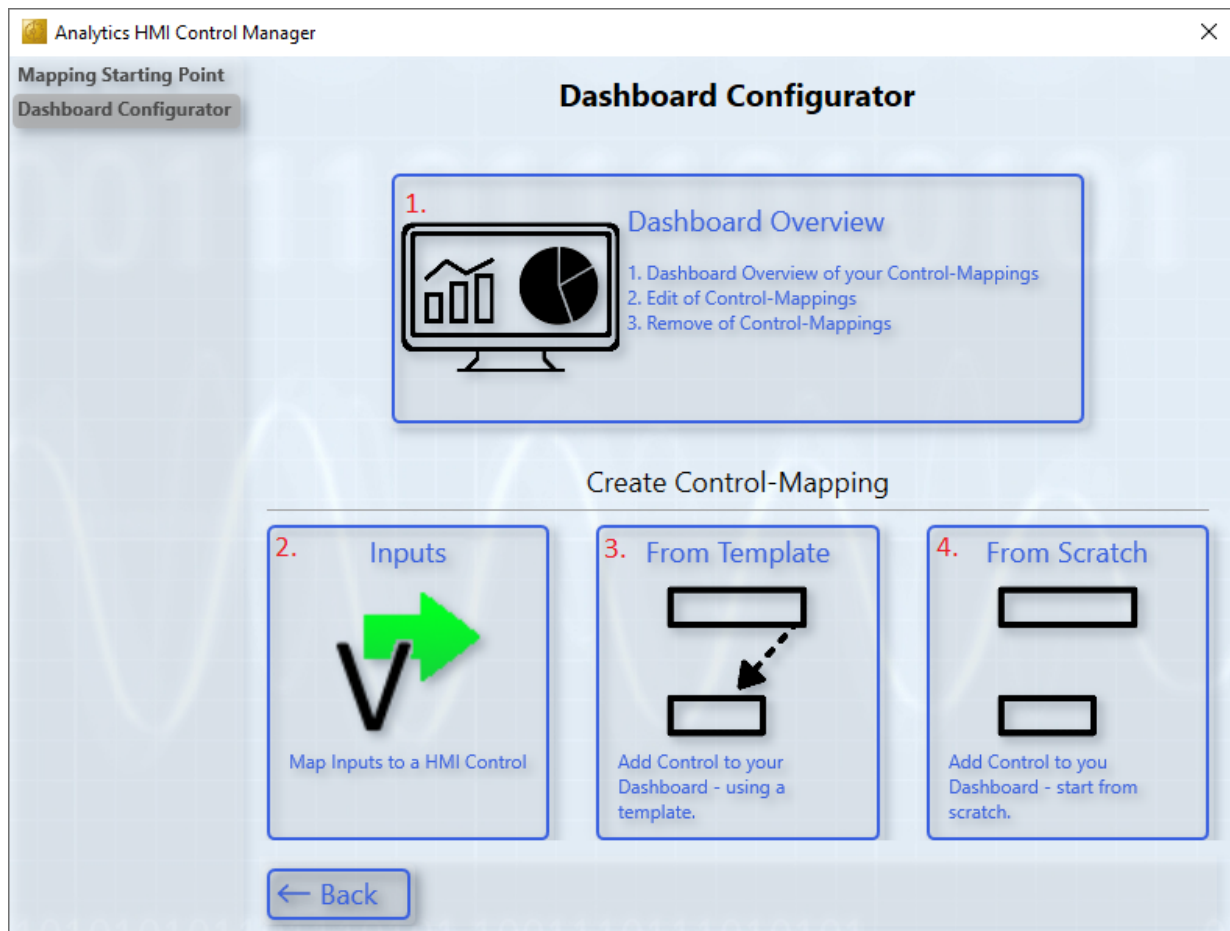
3. Mapping of controls to module instances

In addition to mapping controls to module classes, the module instances of a project can also be directly linked to controls. This is possible via the Manage dashboard structure and content in Analytics project [► 372] as well as via the wizard.

1. At the beginning of the wizard, click **HMI Dashboard Configurator**.



2. There you have several choices.



1. Overview and editing options for all controls
 - This function is also available by double-clicking a control in the Solution Explorer.
2. Adding a control that displays your input data
 - This function is also possible via the Manage dashboard structure and content in Analytics project [► 372].
3. Adding a control that displays module data (selection via a template)
 - This function can also be set for an individual module via the *Properties* window. This is the only way to display data from several modules.
 - You can see directly which data from the template are linked and can adjust them directly.
4. Adding a control that displays data from modules (fully manual)
 - This function is also possible via the Manage dashboard structure and content in Analytics project [► 372].

For the individual points you have to follow the steps of the wizard. The steps are the same or very similar to section "2. Mapping of controls to modules".

5.6.3 Configure user management and access rights

Users and user groups can be generated individually via the user management. Access rights to contents and controls can be defined for user groups.

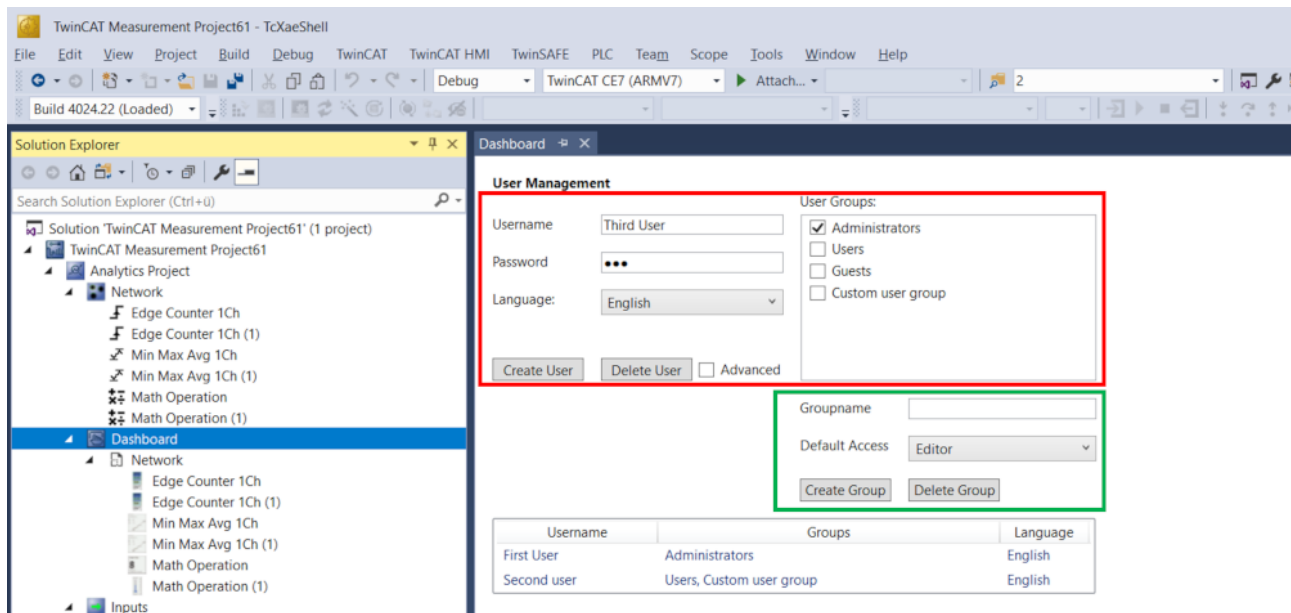


Note that user management and access rights are only transferred to the HMI project from Dashboard version 2.0.

5.6.3.1 Definition in the Analytics Project

Defined users, user groups and access rights are automatically transferred to the HMI project during code generation. Changes to users and access rights can be made in the web browser after generation.

5.6.3.1.1 User management



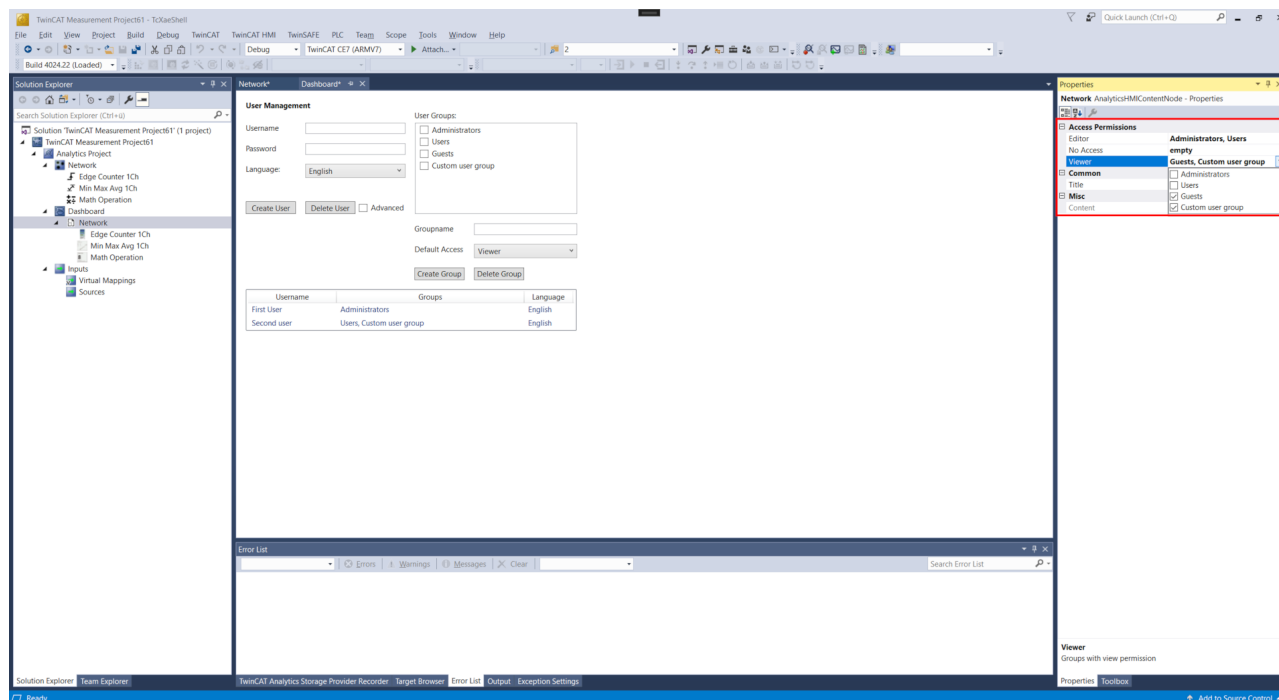
Users (outlined in red) can be created under the Analytics project dashboard node. For a new user, a username and password must be set, as well as a language. In addition, it is necessary to assign one or more user groups to the user, which gives the user the access rights of the respective group(s).

By default, there are the following user groups:

- Administrators
- Users
- Guests

For a more precise classification, additional user groups can be created (outlined in green). For this purpose, a name and the access authorization must be entered.

5.6.3.1.2 Configure access rights



Access rights can be configured for objects below the dashboard node (contents and controls). Access permissions (outlined in red) can be set for each user group.

The following access rights are available:

- Editor: Access to the object is permitted, and changes (such as layout adjustments for content objects) can also be made.
- Viewer: Access to the object allowed - but no changes can be made.
- No Access: No access allowed to the object and the objects subordinate to it in the Solution Explorer.



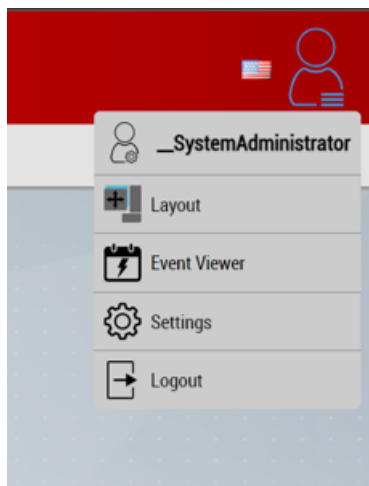
Access rights for content objects can be subsequently adjusted via the web browser.

5.6.3.2 Customize users and access rights in the web browser

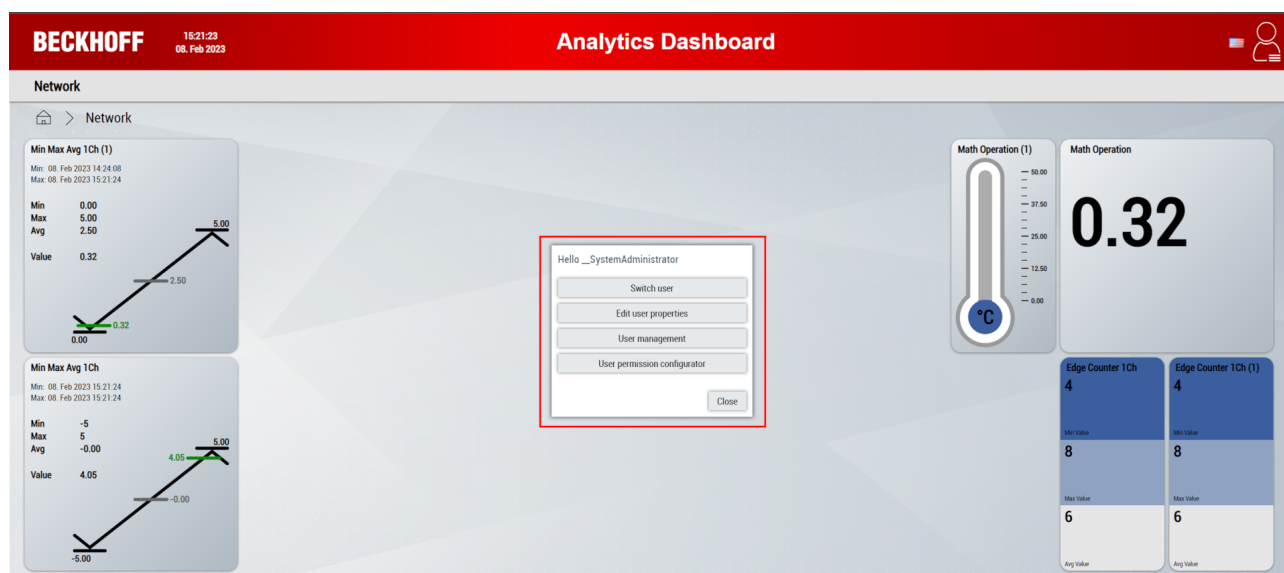


Note that the menu control and the configuration of users and access rights are only available from Dashboard version 2.0.

Configure user



The menu control can be used to make changes to the users and access rights. To do this, please click on your username (at the top of the menu).



The following options are available via the popup (outlined in red):

- **Switch user:** Change of the logged in user
- **Edit user properties:** Change the properties of the own user
- **User management:** Create new users, remove existing users and set user properties
- **User permission configurator:** Configuring the access rights of user groups to content objects

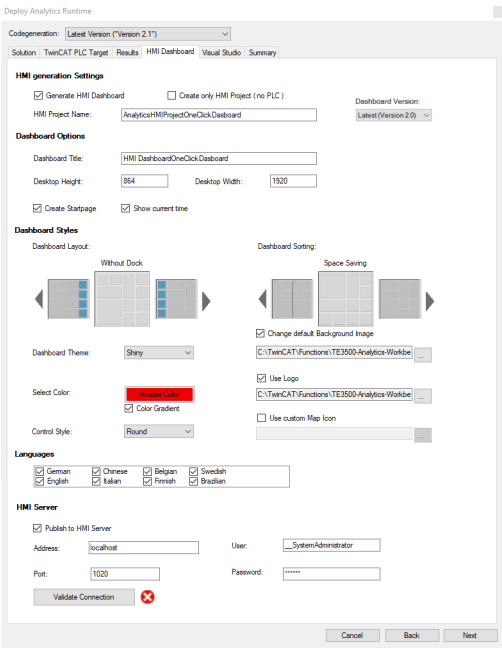


Note that when creating new users, individually configured layouts of the creating user are taken over.

5.6.4 Dashboard Configuration

HMI Dashboard tab

The **HMI Dashboard** tab contains all the configurations for the dashboard.



Configuration

HMI generation Settings

Generate HMI Dashboard	Enables the generation of the HMI Dashboard, if enabled. This automatically activates/deactivates the checkboxes Create Bootproject and Activate PLC Runtime on the TwinCAT PLC Target tab.
Create only HMI Project (No PLC)	Creates only one HMI project and not a PLC project, if enabled.
HMI Project Name	Name of the TwinCAT HMI project.
Dashboard version	Selection of the dashboard version.

Dashboard Options

Dashboard Title	Title of the HMI Dashboard, which is displayed in the dashboard header.
Desktop Width	Width of the target screen in pixels.
Desktop Height	Height of the target screen in pixels.
Create Start-Page	Creates a start page for the dashboard that displays a map with all machine locations. The location data is adopted from the machine management data.
Show current time	Creates a clock in the dashboard header that shows the current local time.

Dashboard Styles

Dashboard Layout	Defines the layout of the dashboard. Dock requires the "Dock" property of a module to be TRUE. Dock Left: fixed left column Dock Right: fixed right column Without Dock: no fixed column
Dashboard Sorting	Defines the sorting of the dashboard. Space Saving: arranges the controls without free space. Control Type: arranges the controls by Control type. Control Size: arranges the controls by size from large to small. Filled: arranges the controls so that the entire screen is filled. Network Groups: groups the controls by network and summarizes them on a screen. Grouping begins at the defined level (0 = All, 1 = First Level...)
Control Distance	Determines the minimum distance between controls in the grid.
Dashboard Theme	Defines the topic of the HMI dashboard, affects controls and backgrounds: Light: bright skin, especially for day mode. Shiny: similar to the bright skin, color gradient in the controls. Dark: dark skin, especially for night mode.
Header Color	Specifies the color of the dashboard header. This contains a color gradient if Color Gradient is enabled.
Control Style	Defines the style of the controls: Round: The controls have rounded corners. Flat: The controls have angular corners.
Change default Background Image	If enabled, a customer-specific background image can be set for the HMI Dashboard. If nothing is defined, the default image is used.
Use Logo	If enabled, a logo is added to the dashboard header. A customer-specific image can be defined for the logo.

Languages

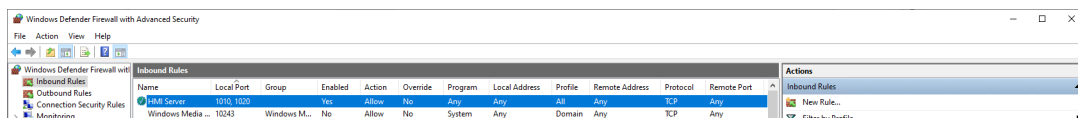
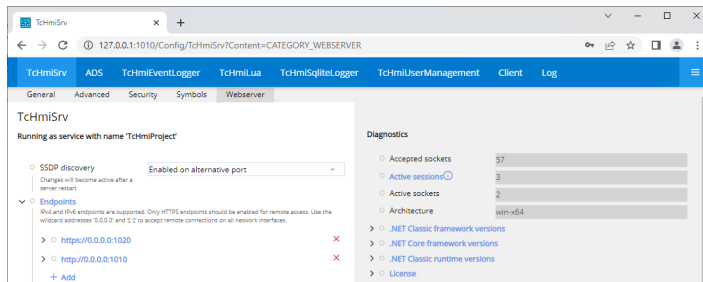
Languages	Selection of languages available in the HMI Dashboard.
Select Default Language	Default language for the dashboard.

HMI Server

<u>Publish to TwinCAT HMI Server</u>	Publishes the dashboard to a TwinCAT HMI server, if enabled.
Address	Enter the IP address or host name of the TwinCAT HMI server.
Port	Enter the port of the TwinCAT HMI Server (by default 1010 without encryption and 1020 with encryption).
User	Enter the user name.
Password	Enter the password. (The administrator password must be set once on the HMI server via the configuration page).
Validate Connection	Press the button to validate your server connection.

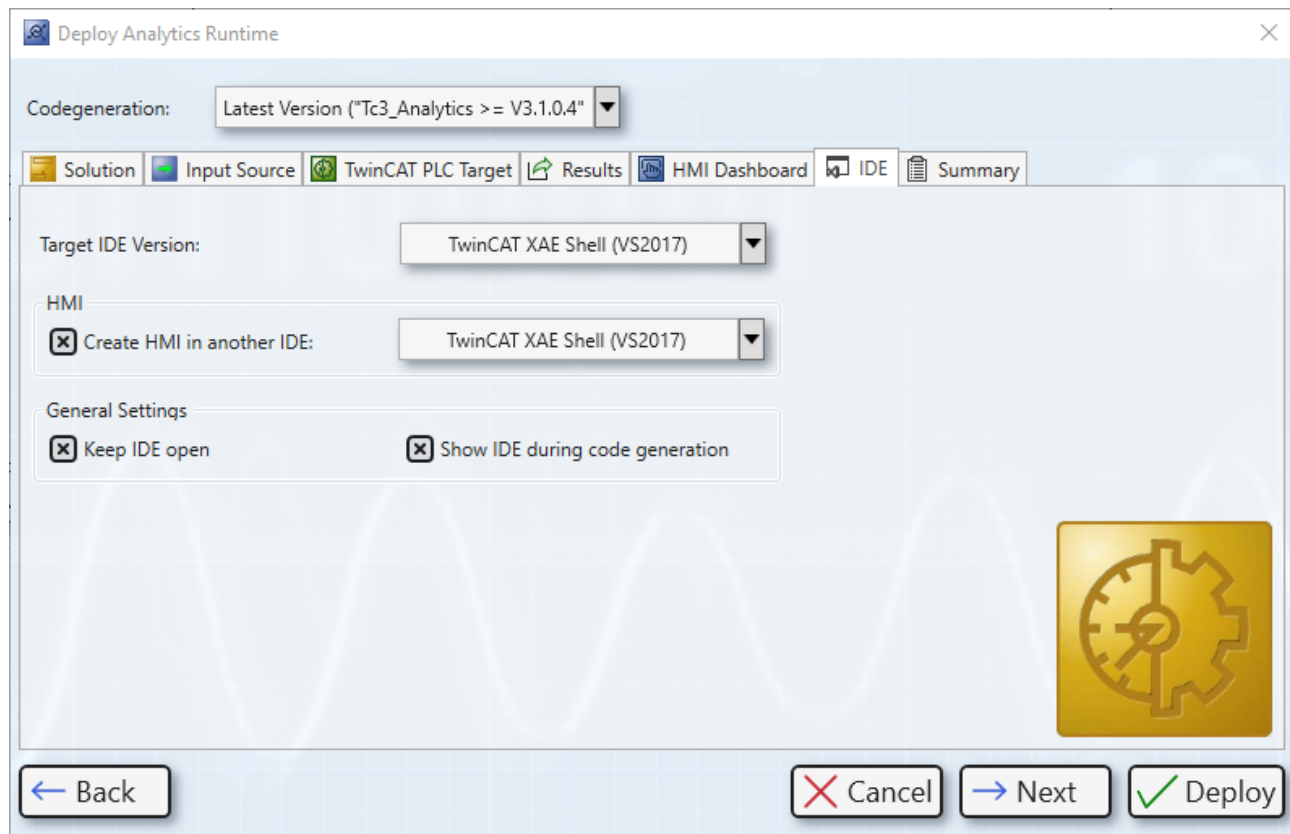


In order to publish to a remote HMI server, the HMI port and IP of the development computer must be shared in the HMI server. Also an inbound rule with the HMI ports must be set up in the Windows firewall from the remote PC.



IDE tab

On the **IDE** tab, you can select which IDE (Visual Studio or TwinCAT XAE Shell) is to be used for generating the PLC and HMI. In addition, it is possible to generate HMI and PLC in two different solutions.

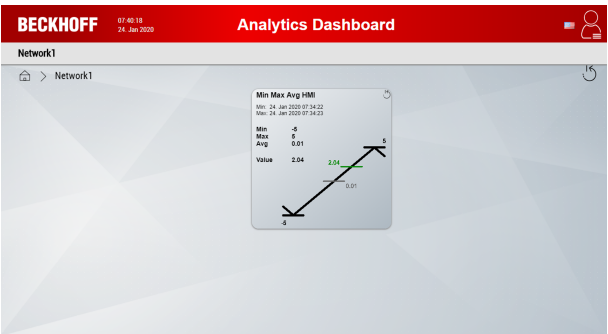
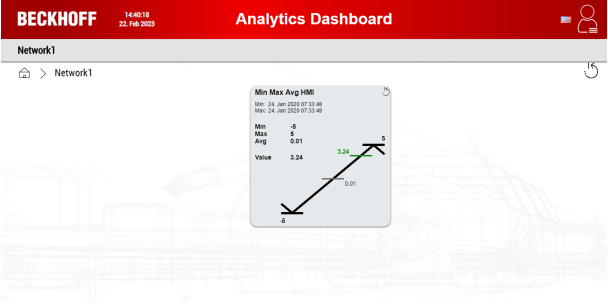
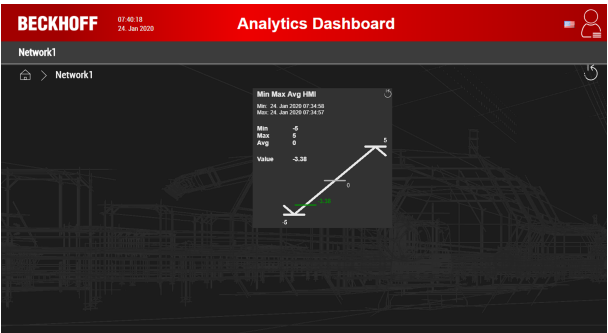


Keep IDE open	Keeps the IDE open after generation, if enabled.
Target IDE Version	The IDE target version for PLC and HMI.
Create HMI in another IDE	If enabled, the HMI is generated into a second solution. A different IDE can be selected.
Requirement: TwinCAT or TwinCAT HMI must be installed in the selected IDE.	

Click **Next** to display the **Summary** tab, then click **Deploy** to start generating the dashboard.

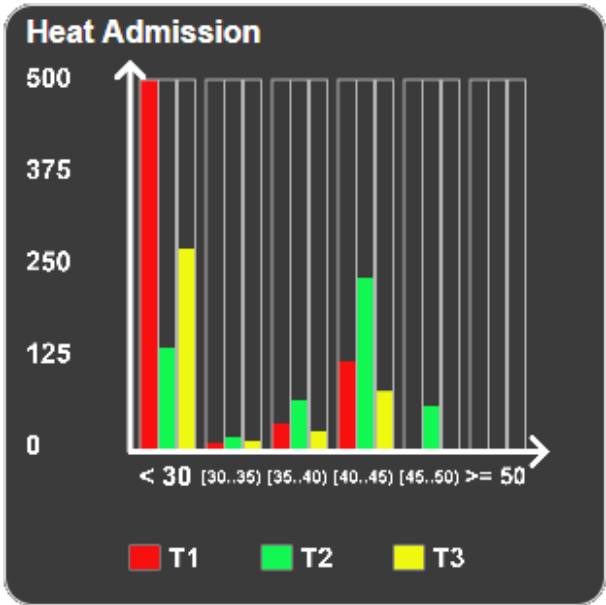
Impressions

Topics

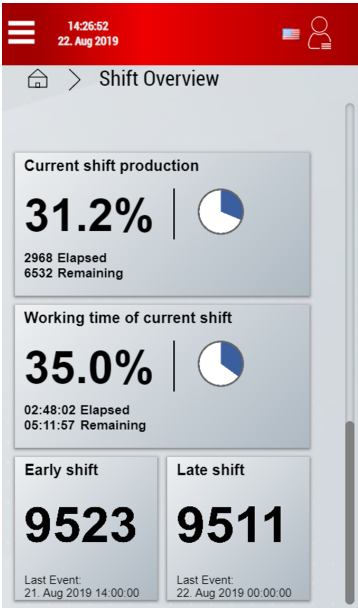


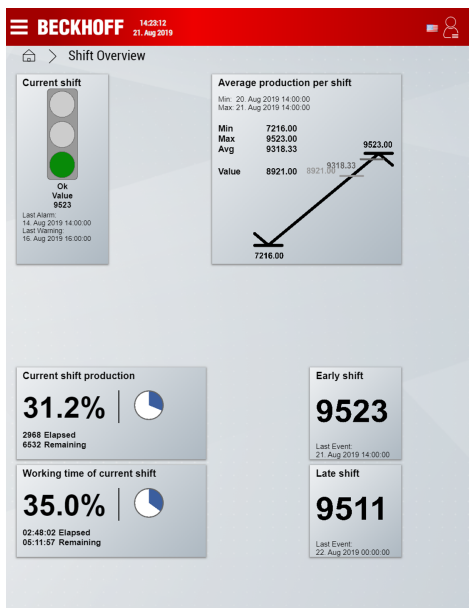
Styles



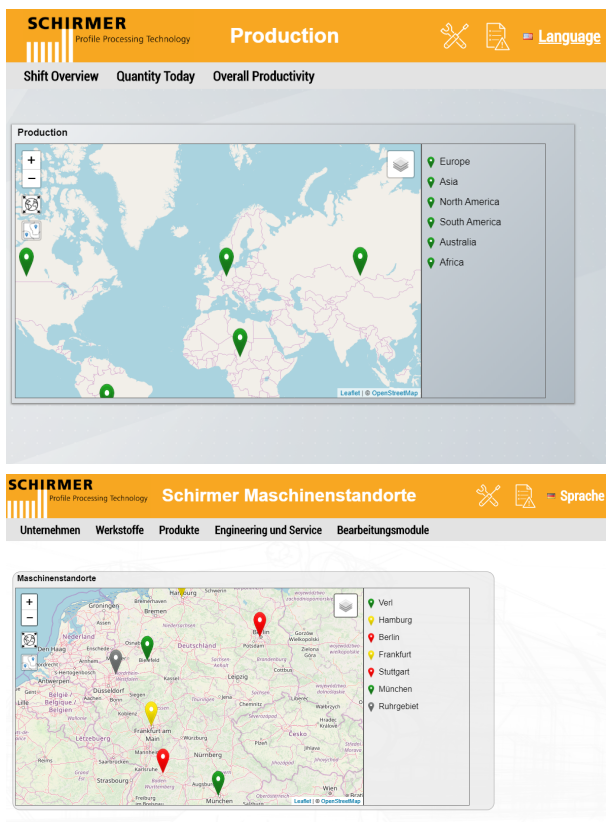


Views for mobile devices

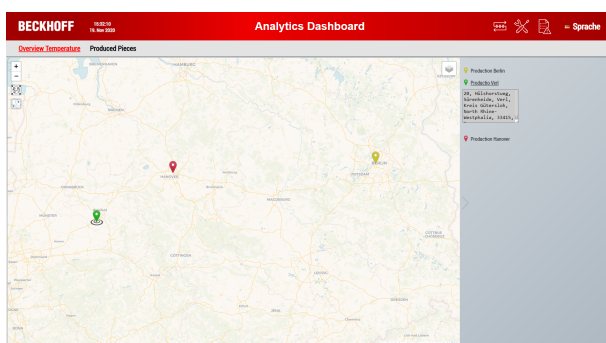


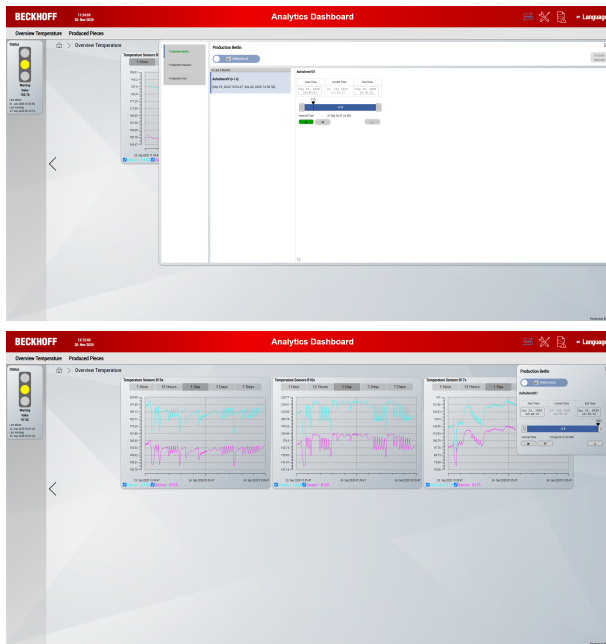


Map on customer-specific dashboard



Historical data and machine switching

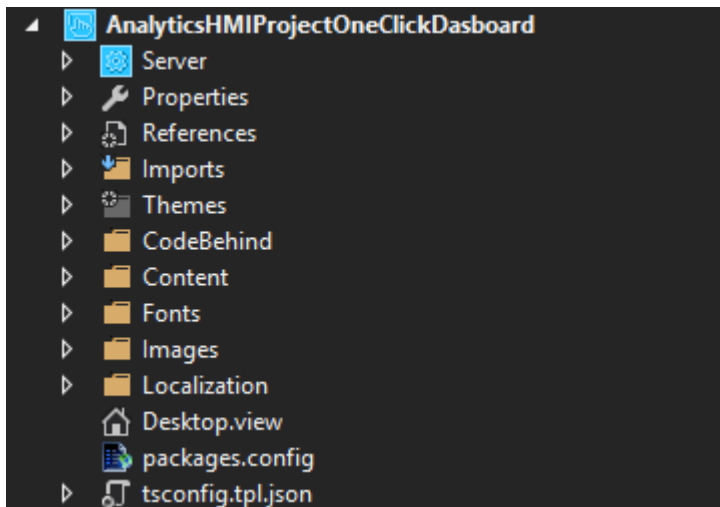




5.6.5 Modifying a generated dashboard in HMI engineering

TwinCAT 3 HMI project

The result of dashboard generation is a complete TwinCAT 3 HMI project. Therefore, all options offered by TwinCAT 3 HMI Engineering can be used and included.



Note that many dashboard customizations can also be made in the web browser at runtime.

5.6.6 Configuration of the dashboard at runtime in the web browser

At runtime, different configurations can be made in the web browser:

- Create individual user-specific layouts
- Manage users (see Configuring user management and access rights)
- Customize access rights to contents (see Configuring user management and access rights)
- Changing control properties in the browser
- Customize parameters

- Changing global dashboard options



Note that not all configuration options are available with Dashboard version 1.0.

5.6.6.1 Customizing layouts with the Interactive function

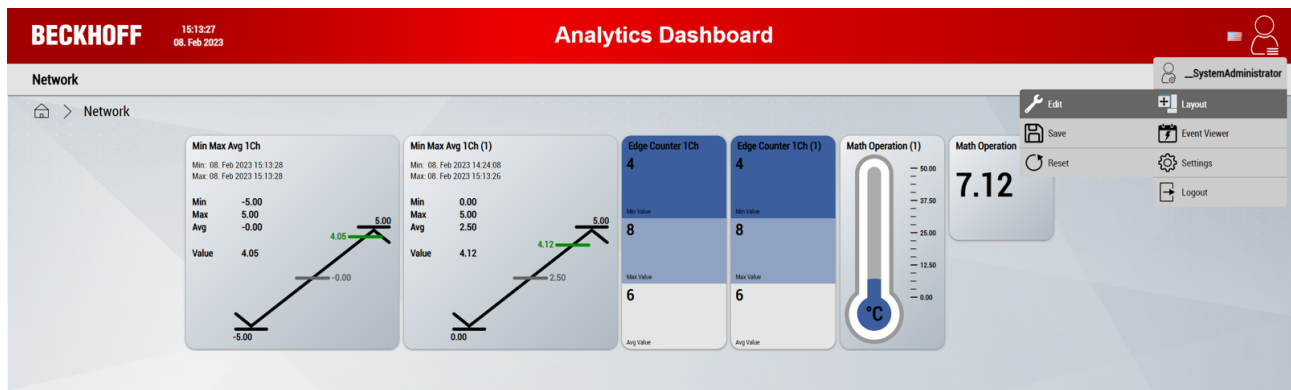
With the interactive function, controls can be arranged within a content in a user-specific way. The customized layout is stored centrally in the HMI server extension "TcHmiAnalytics".



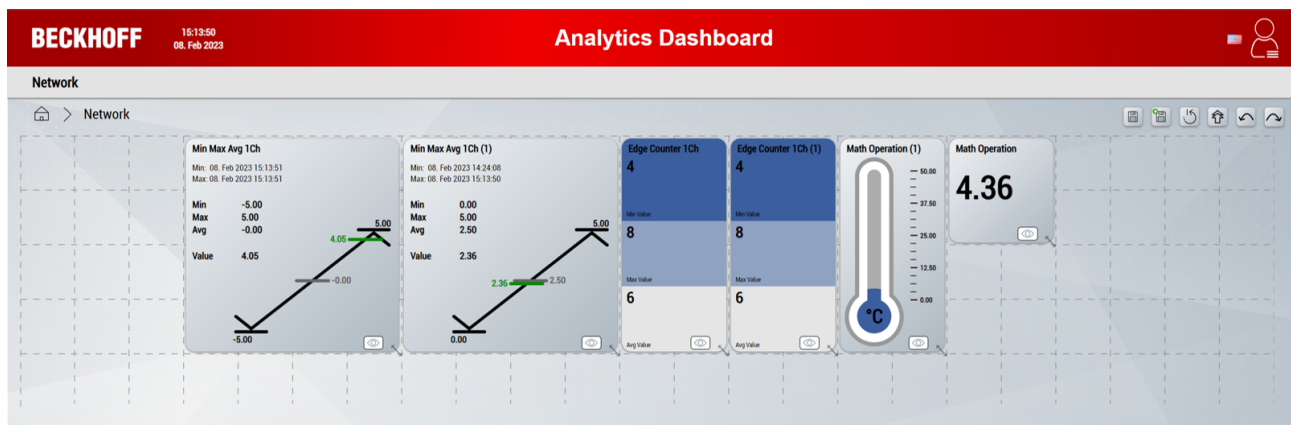
Note that layout editing is available only from Dashboard version 2.0. In addition, the layout can only be edited if the user has the necessary access rights.

Edit mode

Editing a layout is done in a special editing mode. The editing mode can be called via the menu control. Alternatively, the edit mode can also be started with the key combination "CTRL" + "E".



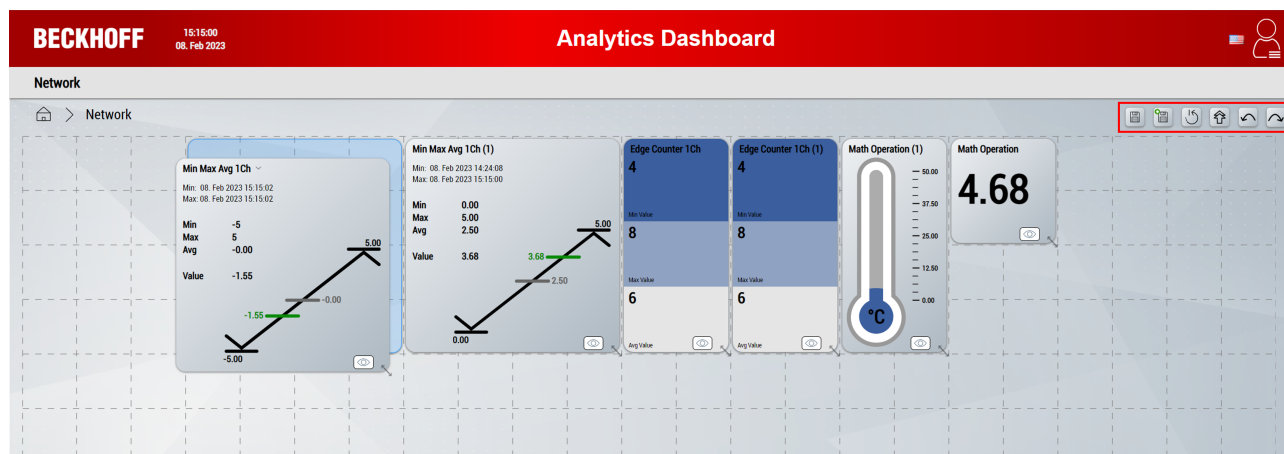
In edit mode, a grid is visible in the background, on which the controls can be arranged as desired.



Edit layouts

The following options are available:

- **Positioning:** For positioning, a control must be clicked or pressed. Thus, it is released from its anchorage and can be moved. The light blue frame indicates the new position within the layout. When the control is released, it anchors itself in its new position.
- **Scale:** Using the arrow in the lower right corner of a control, its size can be changed.
- **Hide controls:** By using the visibility button (bottom right inside a control) it can be shown or hidden. Hidden controls are displayed semi-transparently in edit mode.



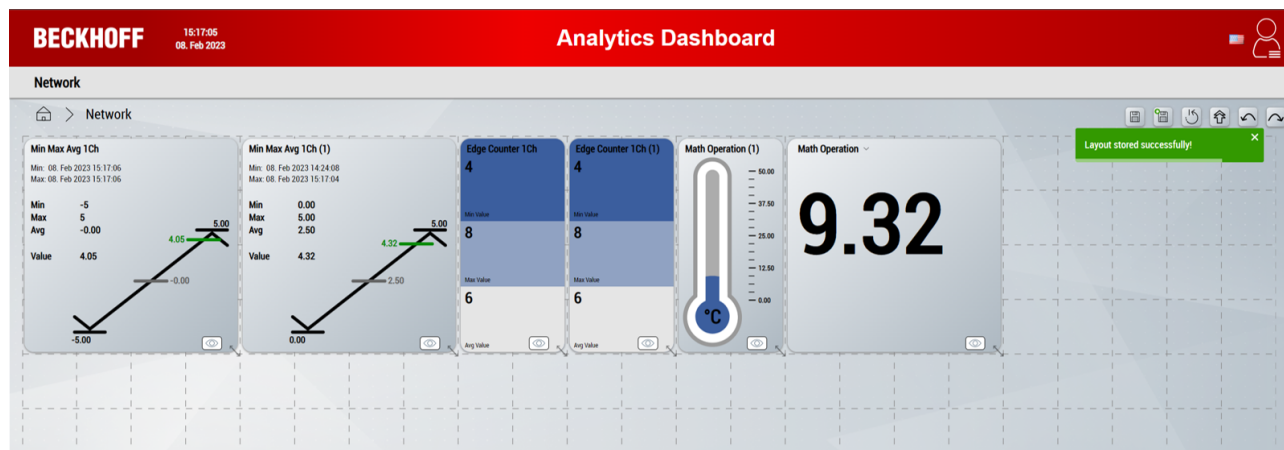
Editing menu

In edit mode, different actions can be performed via an additional menu (outlined in red):

- Save the layout (alternatively key combination "CTRL" + "S")
- Saving the layout and exiting the editing mode (alternatively key combination "CTRL" + "Q")
- Reset the changes made (alternatively key combination "CTRL" + "R")
- Move controls upwards so that free lines are removed (alternatively key combination "CTRL" + "U")
- Undo last action (alternatively key combination "CTRL" + "Z")
- Repeat last action (alternatively key combination "CTRL" + "Y")

Save layouts

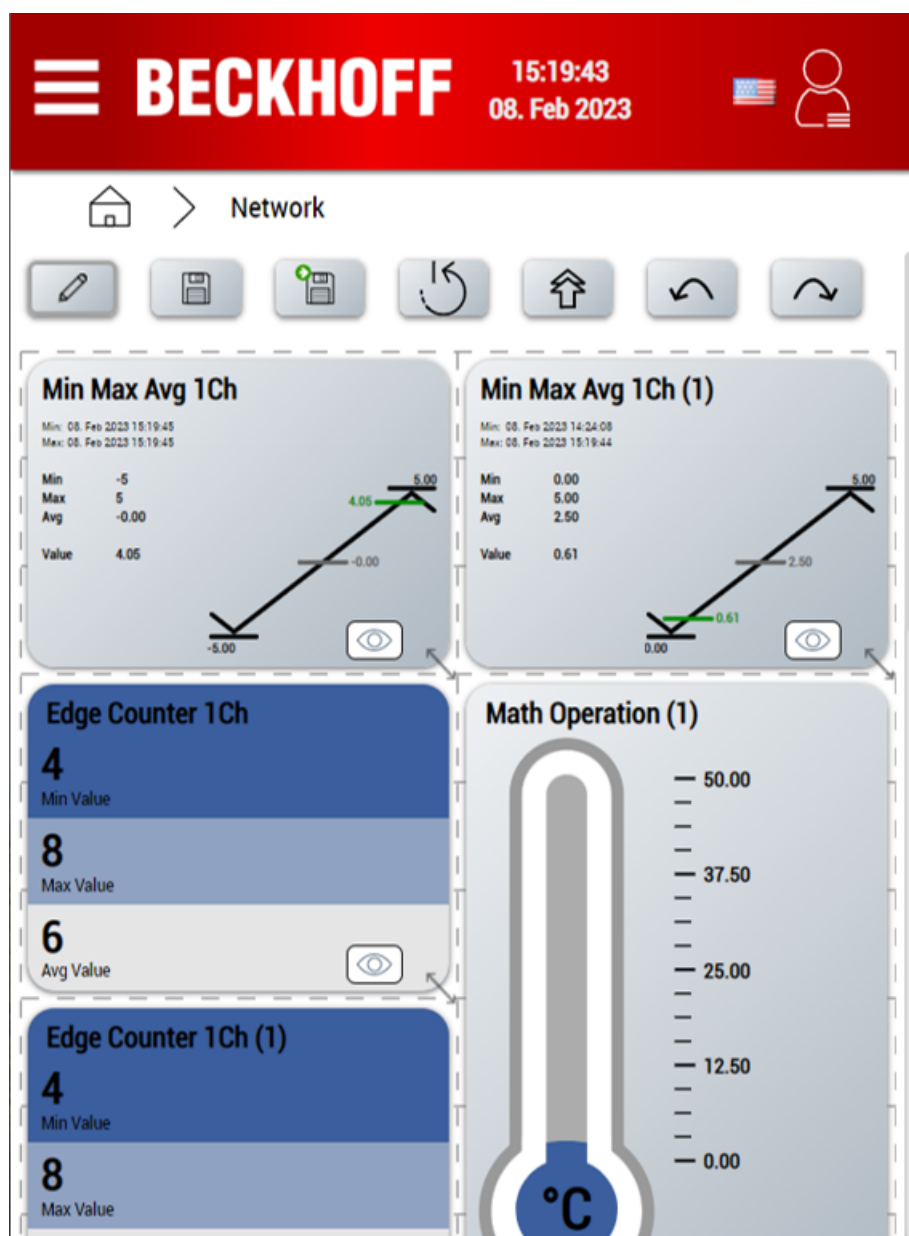
Customized layouts can be saved via the edit menu or via the menu control. A popup shows the result of the save operation.



If an error occurs during saving, check in the [server configuration](#) whether the "TcHmiAnalytics" extension is activated.

Mobile layout

For mobile devices, a user-specific mobile layout can additionally be generated. This is independent from the normal layout and has only 2 columns. Editing and saving is done in the same way.



5.6.6.2 Changing control properties in the browser



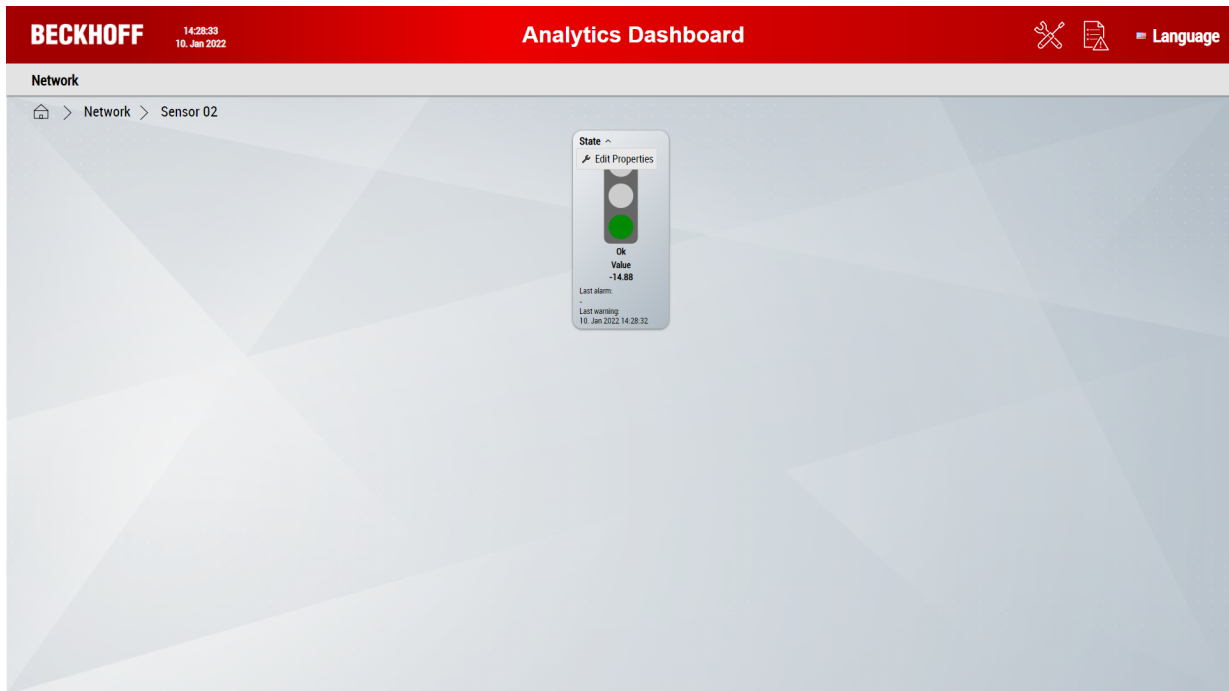
Recommendation: make changes in the Analytics Workbench project

Usually, the changes made here can already be made in the Analytics Workbench before you generate the dashboard (see Mapping Wizard or also Dashboard node). This has the advantage that the changes are saved when the dashboard is generated again.

Each Analytics Control has selected properties that can be changed dynamically in the browser. This option is available from Analytics version 3.4.3145.0 or with Control Package version 1.1.31. You can use it to easily update the NuGet package for older HMI projects.

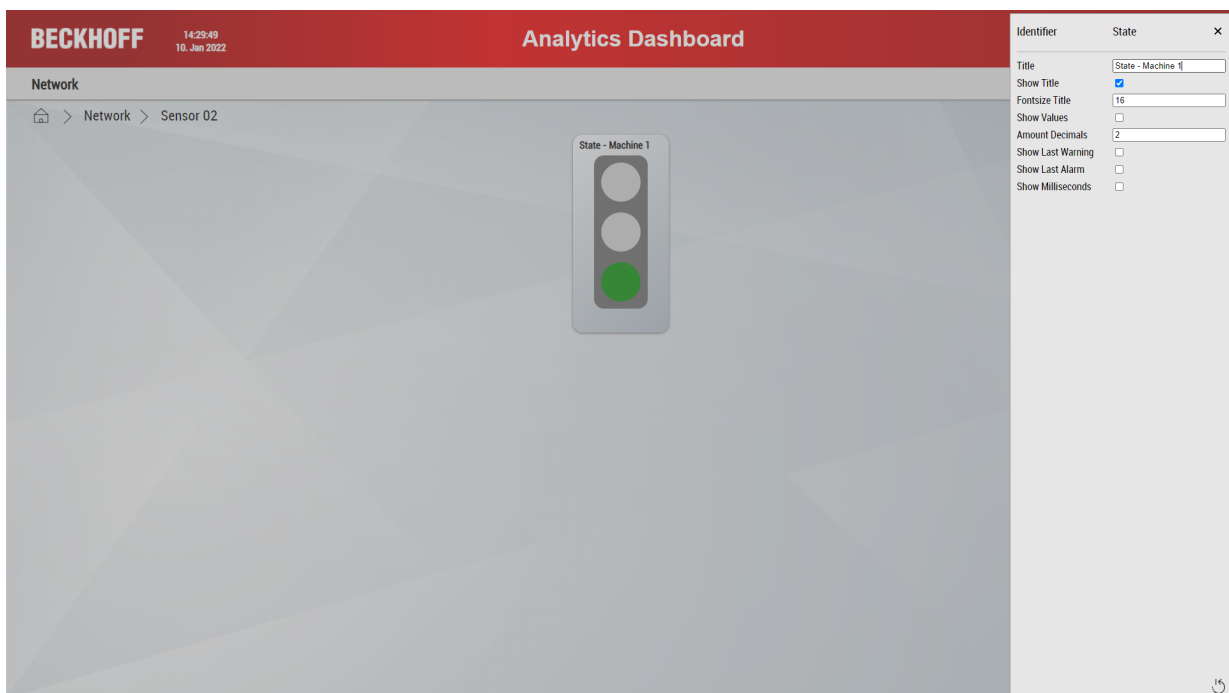
1. Open a generated HMI project and click on the title of a control where you want to change a property.

2. There a menu opens where you can click on **Edit Properties**.



⇒ Each control has different properties that can be customized. Here, for demonstration purposes, the properties of the *Traffic Light* control are changed

3. Change the title and uncheck the bottom three selection boxes to display only the traffic light with the changed title.

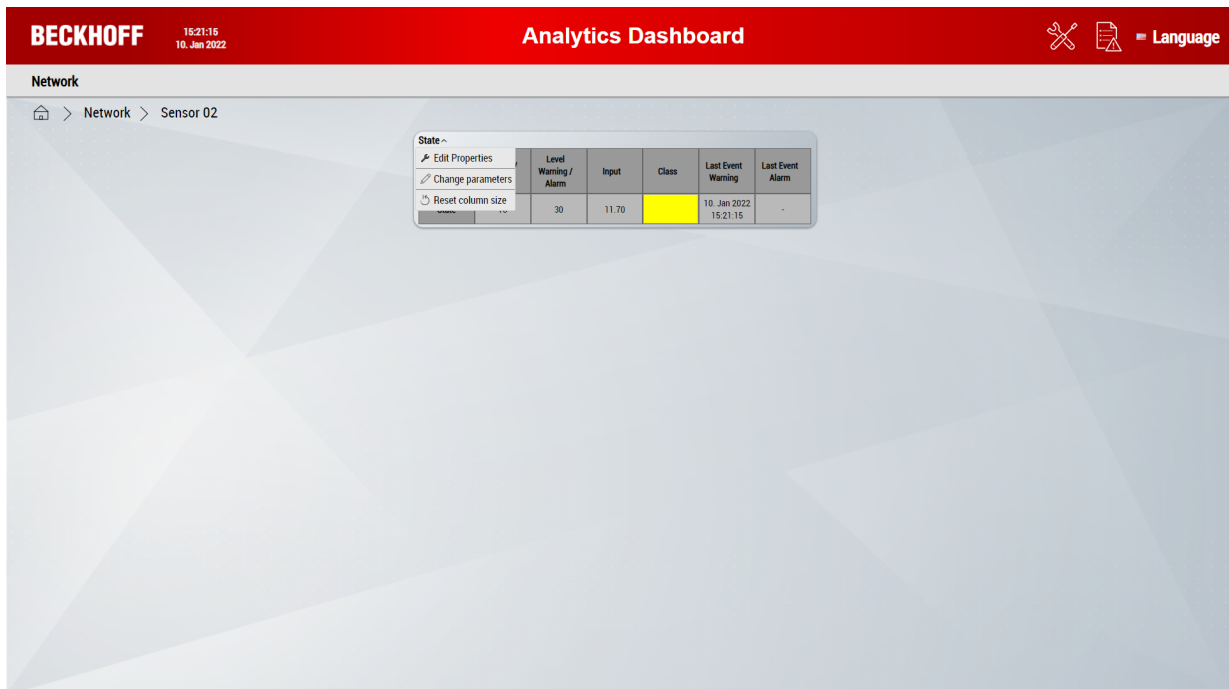


⇒ The changes made are saved client-side (on the device where you open the dashboard) and are only displayed there. In the future, storage will be performed via the HMI server, which means that changes will be made globally on each device.

5.6.6.3 Changing parameters in the browser

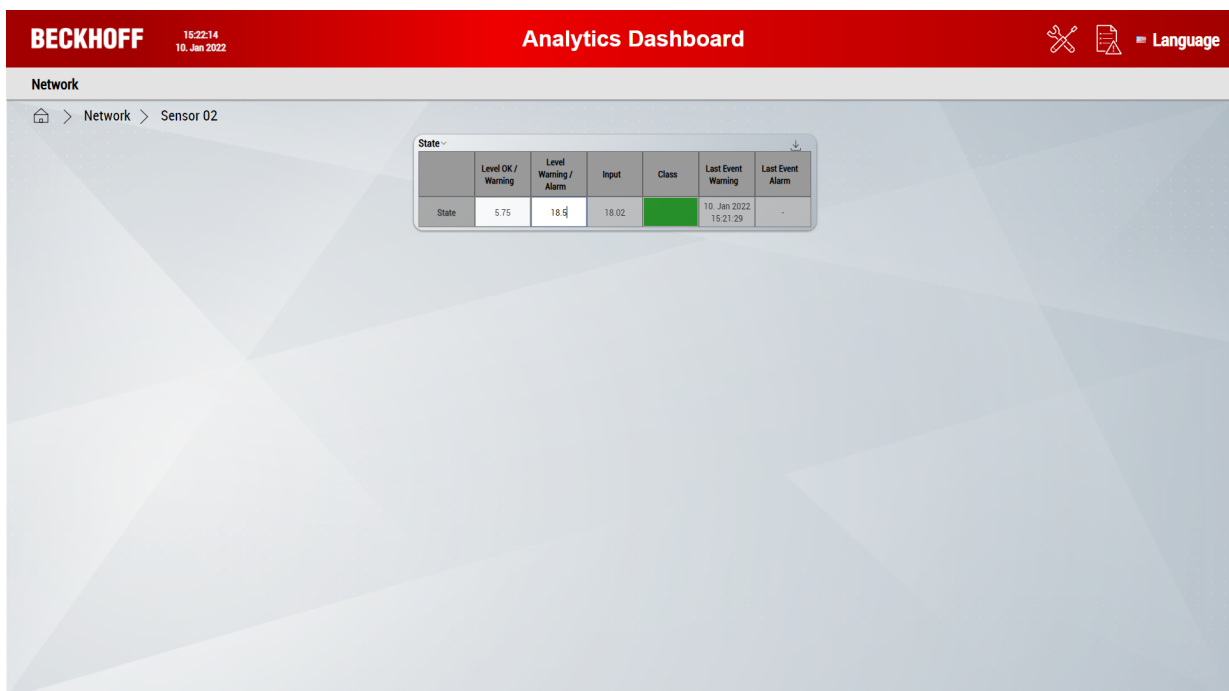
Numerical parameters such as limit values can be changed dynamically in the dashboard. This is possible with the Data Table Control. The values are persistently written back to the PLC there and are thus also available after a restart.

1. Open a generated HMI project and click on the title of the table control where you want to change a parameter.
2. There a menu opens where you click on **Change parameters**. A module without parameters does not have this menu item.



⇒ The changeable fields change to an input field with a white background.

3. Change the values there and then click the save icon in the top right corner.



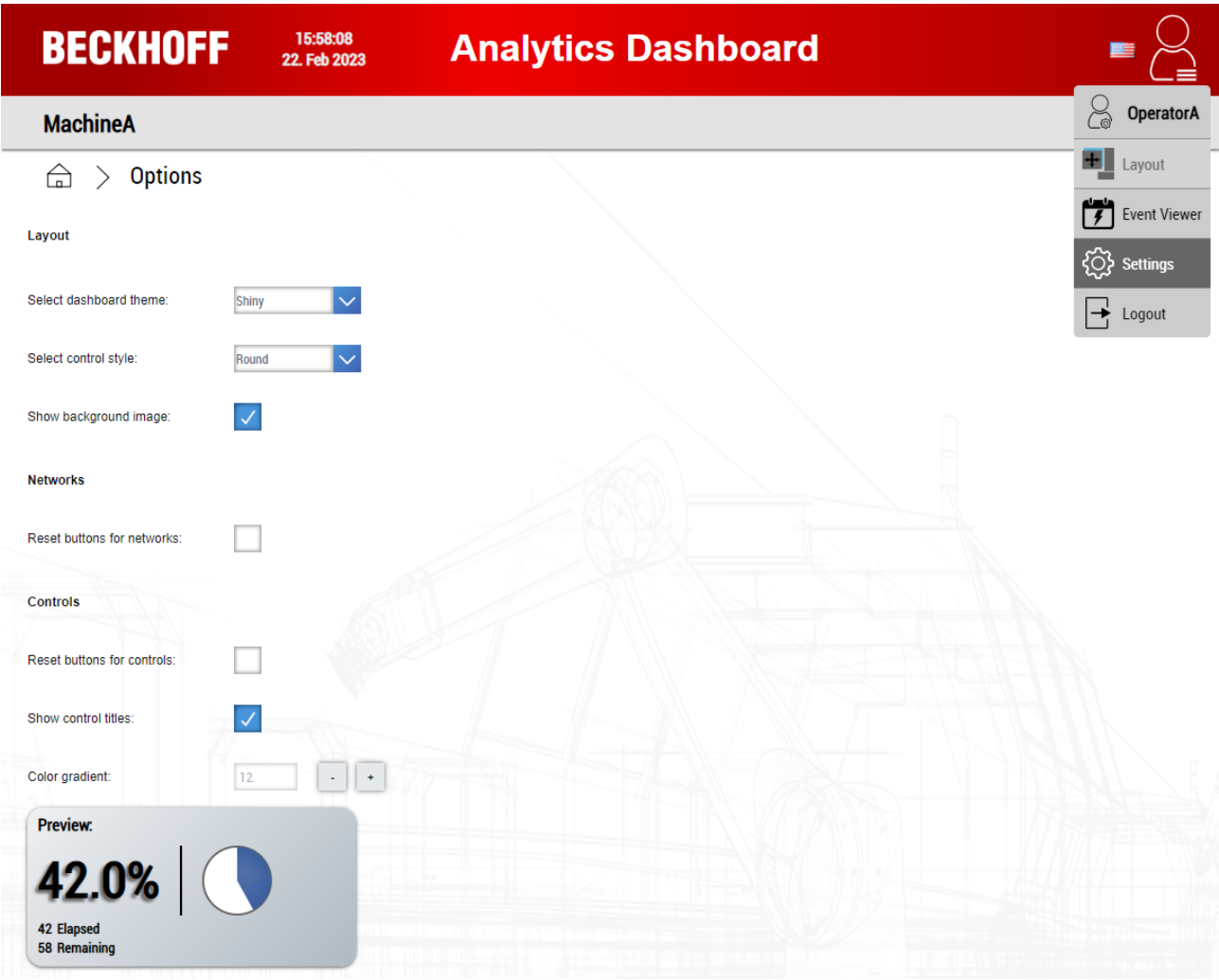
- ⇒ After saving the parameters, you will see a message in the upper right area of the dashboard. There you will be told if the reconfiguration worked.

The screenshot shows the Beckhoff Analytics Dashboard. At the top, there is a red header bar with the Beckhoff logo, the date and time (16:23:23, 10. Jan 2022), and the title "Analytics Dashboard". On the right side of the header, there is a notification box that says "Parameters has been reconfigured successfully!". Below the header, there is a navigation bar with the text "Network" and a breadcrumb trail "Home > Network > Sensor 02". The main content area displays a table titled "State" with the following data:

	Level OK / Warning	Level Warning / Alarm	Input	Class	Last Event Warning	Last Event Alarm
State	5.75	18.5	19.42		10. Jan 2022 15:23:23	10. Jan 2022 15:23:23

5.6.6.4 Customizing global dashboard options

On the Options page, general settings for the dashboard can be edited.



Layout

- Select dashboard themeChange the dashboard theme between shiny, dark and light.
- Select control styleChange the control style between flat and round.
- Show background imageDisplays the standard or customer-specific background image.

Networks

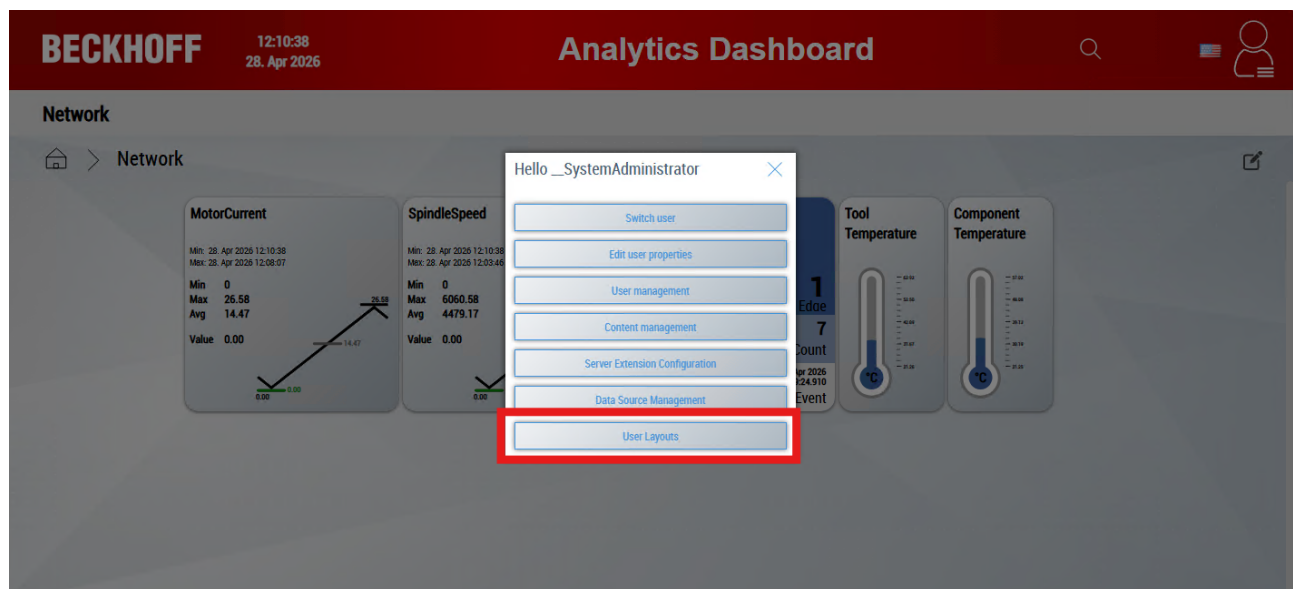
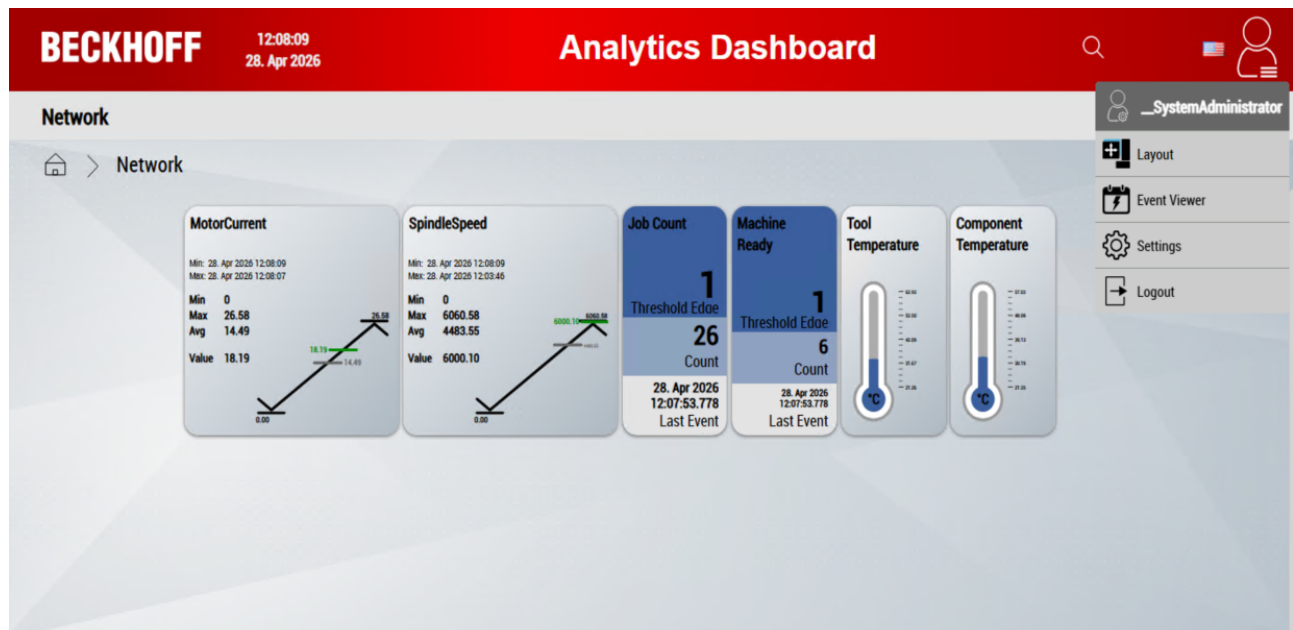
- Reset buttons for networksEnables reset buttons for entire networks. Reset all controls within the selected network.

Controls

- Reset buttons for controlsEnables reset buttons for controls. Reset a single control.
- Show control titlesEnables control titles for all controls.

5.6.6.5 Managing layout configurations

The **User Layouts** button allows you to manage layout configurations for individual users. These configurations include both the layout items saved in the server extension and the interactive charts generated.



User Layouts

User	Selected User:	Guest
Analytics	Layout Mode	Default layouts
Guest		
LayoutBackup		
User		
__SystemAdministrator	Import layouts from user: Analytics	Reset Layouts
__SystemGuest	Share layouts from user:	Import Layouts
__SystemUser		Share Layouts

There is an overview of all users on the left-hand side. Choose a user to manage their layout configurations.

The selected user and the current layout mode are displayed on the right-hand side. The following layouts are possible:

Default layouts

The user has no individual layout configurations. The standard layouts are used.

Custom layouts

The user has their own layout configurations that are used.

Shared layouts

The user shares his layout configurations with another user. This allows multiple users to use the same layouts.

The following actions are available to you:

Reset layouts

Resets all layouts saved by the user, including the interactive charts created.

Copy layouts

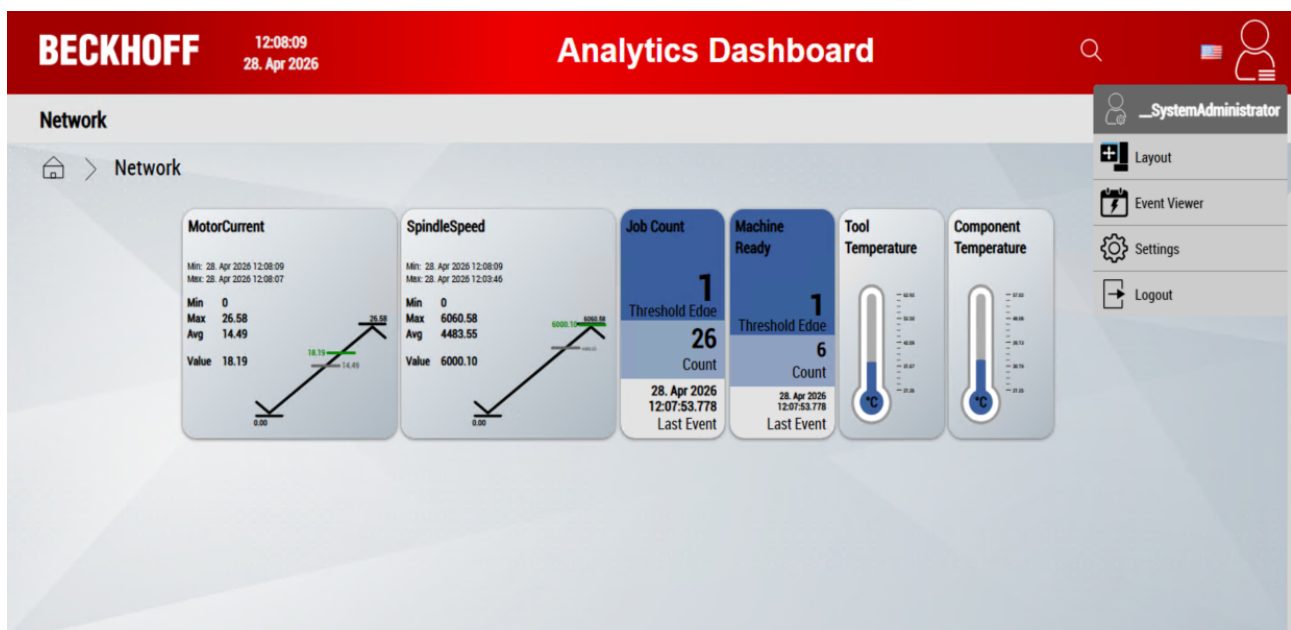
Choose a user whose layout configurations are to be copied. Confirm the process with **Copy Layouts**. The selected user receives an independent copy of the layout configurations.

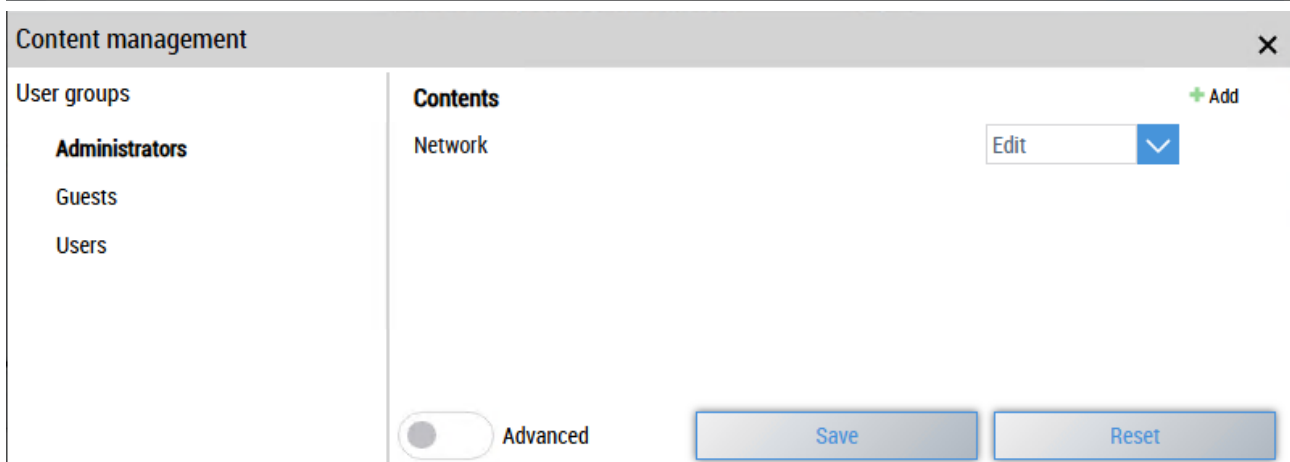
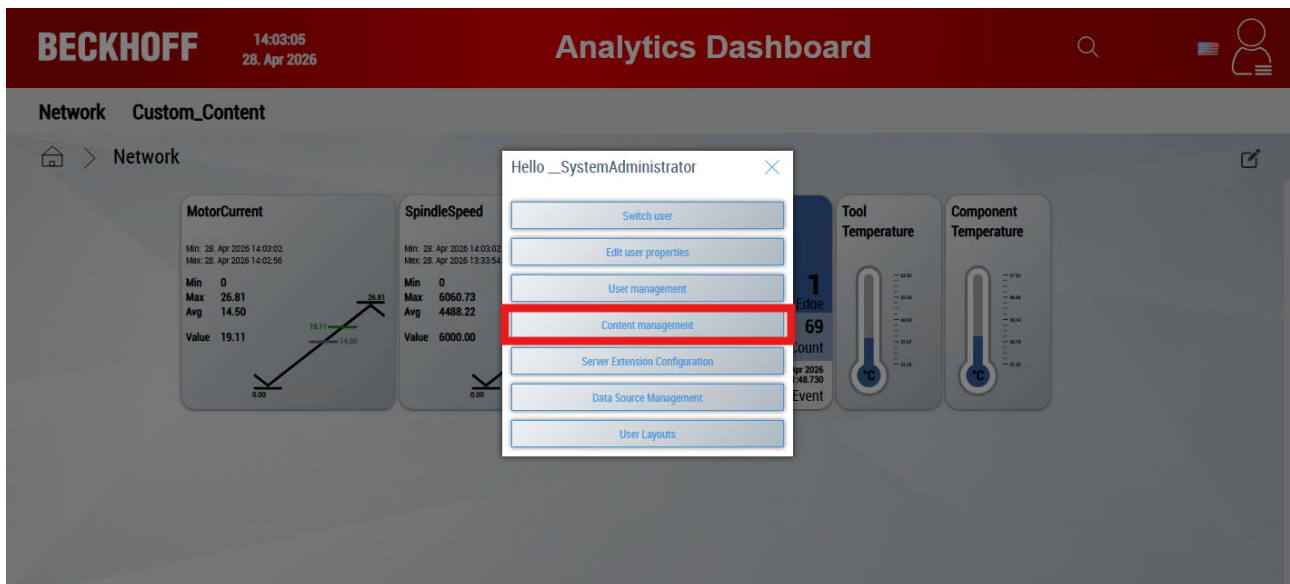
Share layouts

Choose a user whose layout configurations are to be shared. Confirm the process with **Share Layouts**. The selected user receives a reference to the layout configurations of the other user. If one of the users changes a layout configuration, the other user automatically receives the current version after a reload.

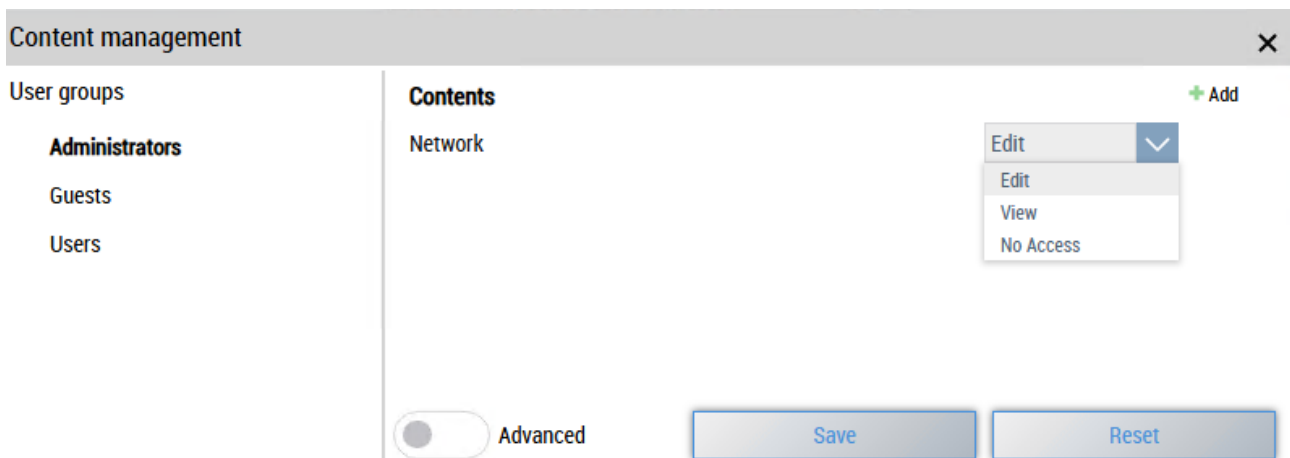
5.6.6.6 Management of the content pages

You can use the **Content Management** button to define the access rights of individual user groups for existing content pages. You also have the option of adding new content pages and activating them for selected user groups.





Select a user group on the left-hand side. You can then define the access rights for the respective content pages on the right-hand side.



The following access rights are available:

Edit

Users in this group can customize the layout of the content, e.g. Move controls, change their size, show or hide them and – if supported – add new controls.

View

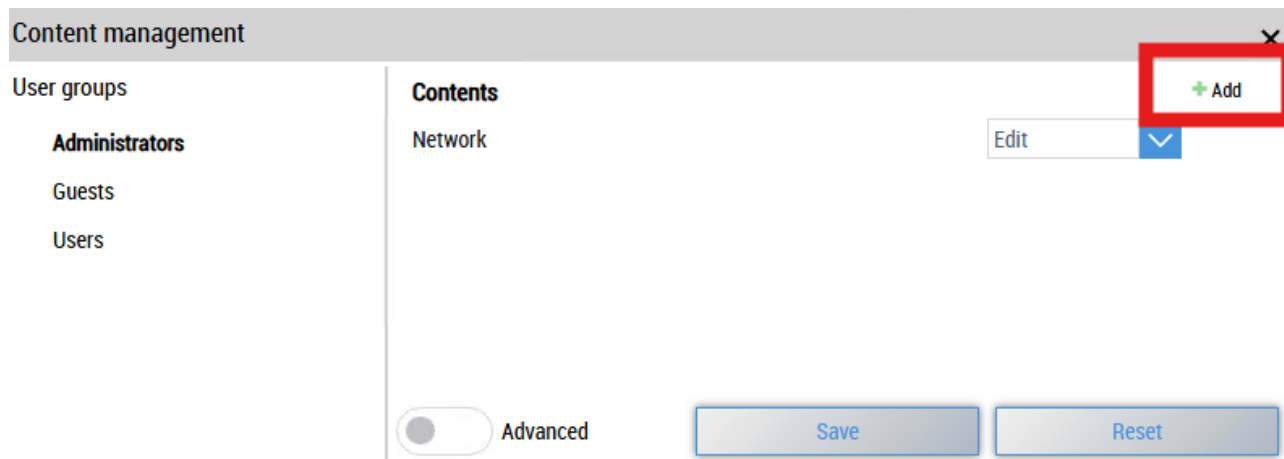
Users can view the content, but cannot make any changes.

No Access

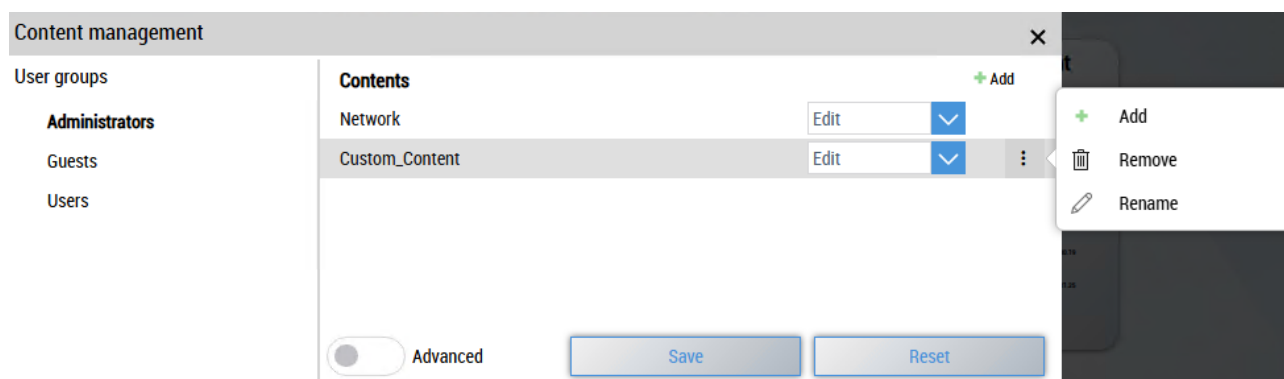
Users cannot view the content. The content is also not displayed in the navigation.

Adding dynamic content

Users have the option of adding new dynamic content. New content can be created using the Add button. This is enabled by default for all user groups with the View access right and is displayed in the navigation bar. The access rights can then be adjusted as required.



You can rename or remove the content using the three-dot menu on the right-hand side.

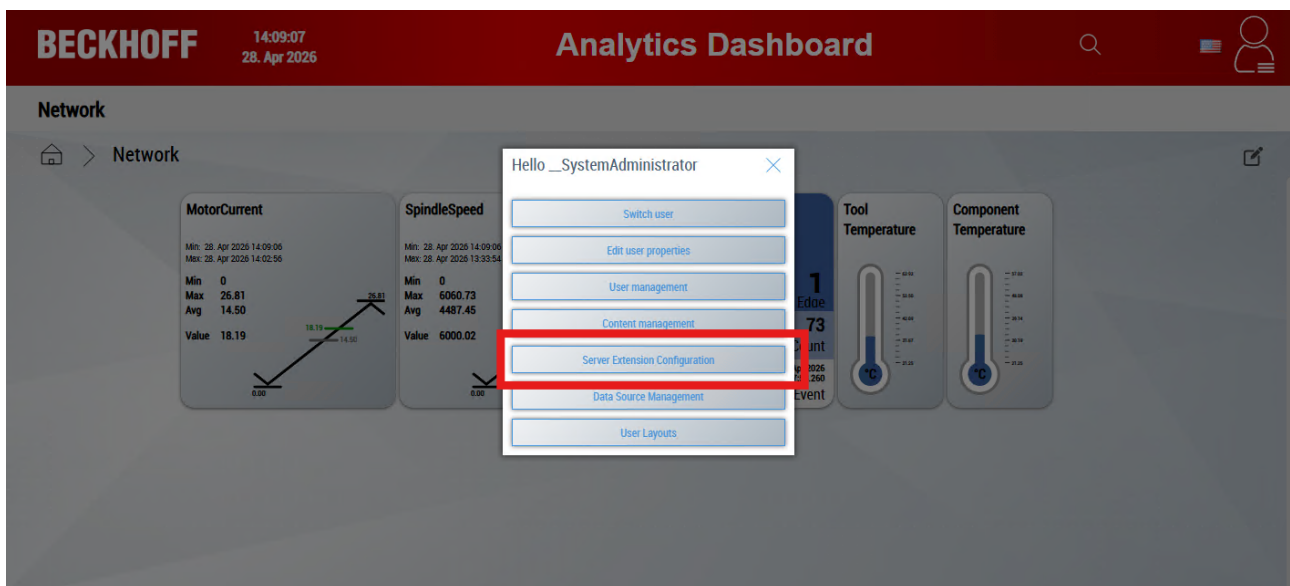
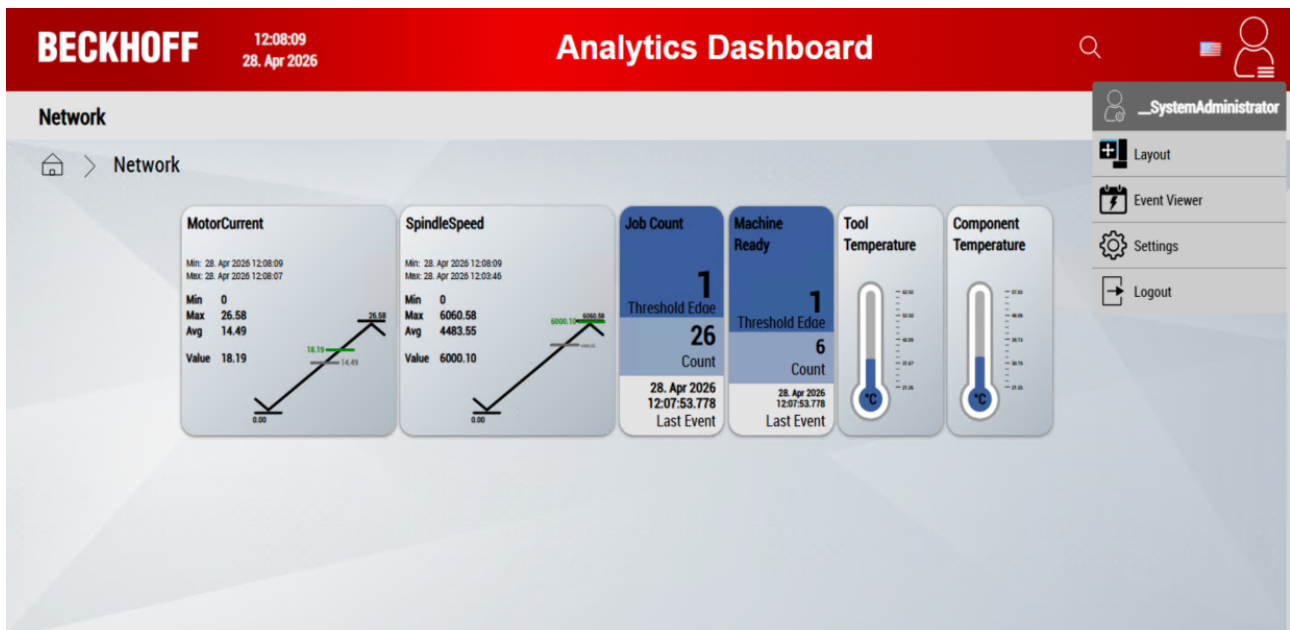


These options are only available for dynamically added content.

As an alternative to the Add button at the top right, you can also add a new sub-content via the three-dot menu of an existing content.

5.6.6.7 Exporting and importing layout configurations

You can export the current configuration of the TcHmiAnalytics extension or import a new configuration via the “Server Extension Configuration” area. The configuration includes all data stored in the server extension, such as. Layout configurations and user-specific settings (color theme, language, etc.).



Server Extension Configuration

Export TcHmiAnalytics Configuration [Export](#)

Import TcHmiAnalytics Configuration [Import](#)



When importing a configuration, the existing configuration is irrevocably overwritten.

5.6.7 Creating charts in the web browser

With the new Interactive Chart function, you can add charts to your dashboards as required. Charts can be added and configured directly in the web browser. Engineering is not required for this. Data that has been historized with the Analytics Storage Provider can be used as a data source for the charts. During configuration, you can select the variables to display, the time ranges and the chart type.

● Analytics Storage Provider Update required

i A TwinCAT Analytics Storage Provider **>= version 1.0.12.3** is required so that the data can be loaded correctly.

i The following NuGet packages are required to use the Interactive Chart function:

Beckhoff.TwinCAT.Analytics.Controls >= 1.35.0

Beckhoff.TwinCAT.Analytics.ControlsExtension >= 1.35.0

The Interactive Chart function can be integrated into already generated HMI projects by updating the NuGet packages.

5.6.7.1 Managing the data sources

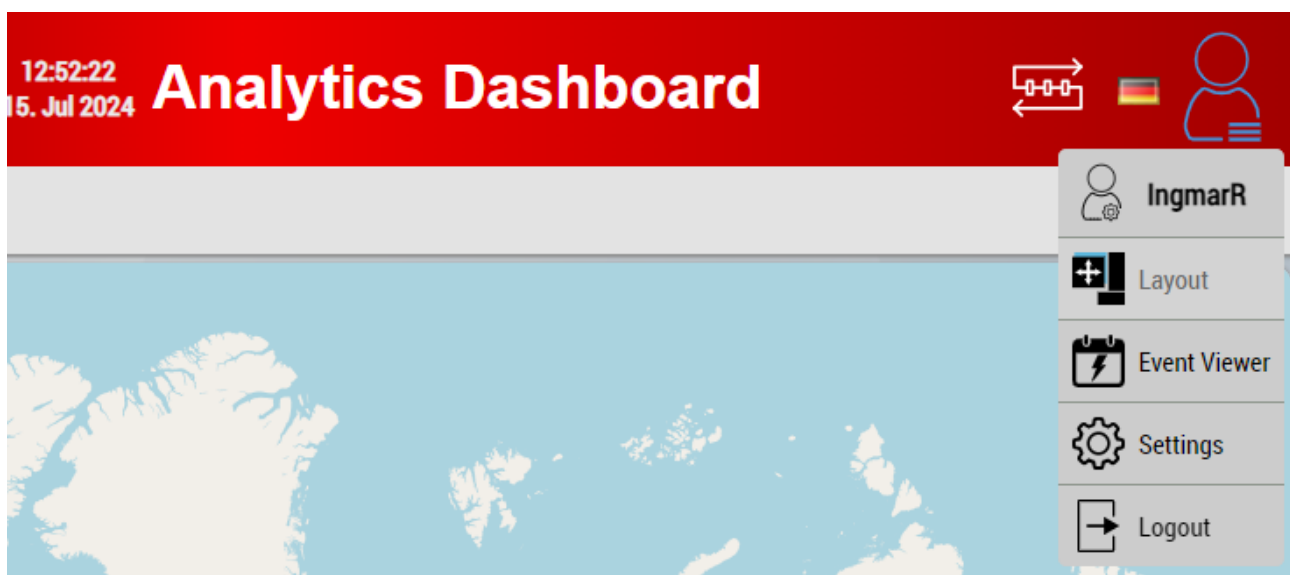
Data that has been historized with the Analytics Storage Provider can be used as a data source for the charts. The data sources can be managed directly via the dashboard.

● Automatic addition of historical sources during code generation

i From TwinCAT Analytics Workbench version **>= 49.0**, all historical data sources from the project are automatically transferred to the dashboard during code generation. It does not matter whether these are actively used in the project or have merely been added as sources. Please note that live data sources cannot be used.

Open of the Data Source Management

Data sources can only be managed by users in the "__SystemAdministrators" or "Administrators" group. You can open the Data Source Management via the menu icon in the top right-hand corner: **Menu - <username> - Data Source Management**.



Energy Data from Headquarter in Verl

Tue, December 19, 2023: 13:59:42.864

GVL_EnergieDaten_F_Halle_Verpackung.stMesswerte_KL3403_F_Verpackung_Beleuchtung.fPg	6.71
GVL_EnergieDaten_F_NSHV.stMesswerte_KL3403_F_NSHV_gesamt.fPg	37.54
GVL_EnergieDaten_F_NSHV.stMesswerte_KL3403_F_NSHV_Ladestation.fPg	1.14

Hello IngmarR

- Switch user
- Edit user properties
- User management
- User permission configurator
- Data Source Management

Close

Data Source Management

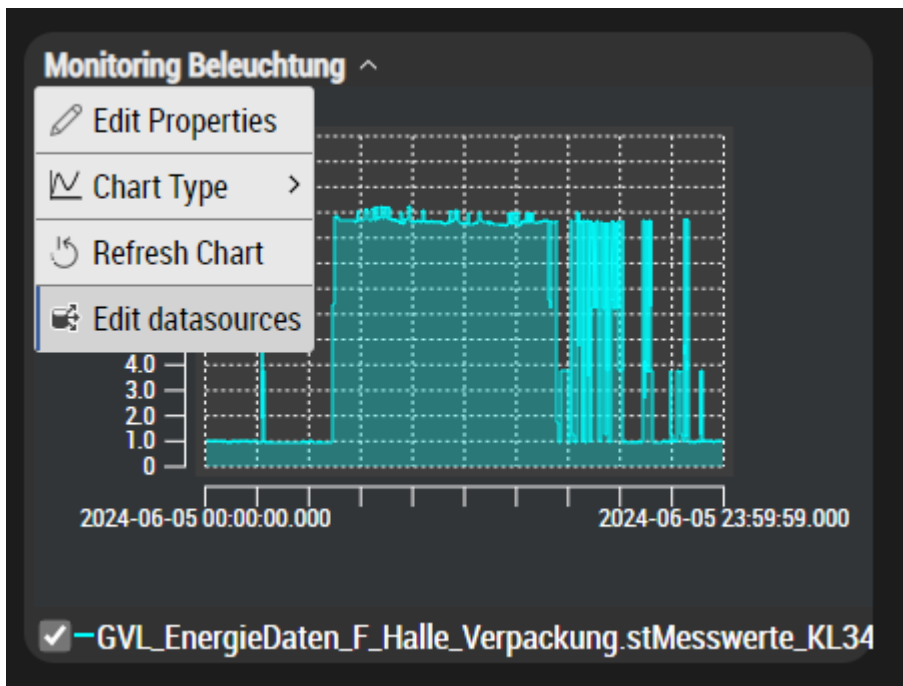
Data Sources

- messagebroker2.beckhoff-cloud.com:8883:EnergyData/Building_F - Status: OK
- messagebroker2.beckhoff-cloud.com:1883:EnergyData/Building_F - Status: NOCONNECTION
- ▼ messagebroker2.beckhoff-cloud.com:8883:EnergyData/Building_T - Status: OBSOLETE

Source ID	d223bdb0-679b-4254-b32e-beac12641f84
Broker	messagebroker2.beckhoff-cloud.com
Broker Port	8883
Topic	IngmarR/OldASP/TcAnalyticsStorageProvider/078bfcff-c260-454f-9912-016a9ca2845f/Historical/Stream_1
Stream Tonic	EnergyData/Building_T

Drop a source from the target browser

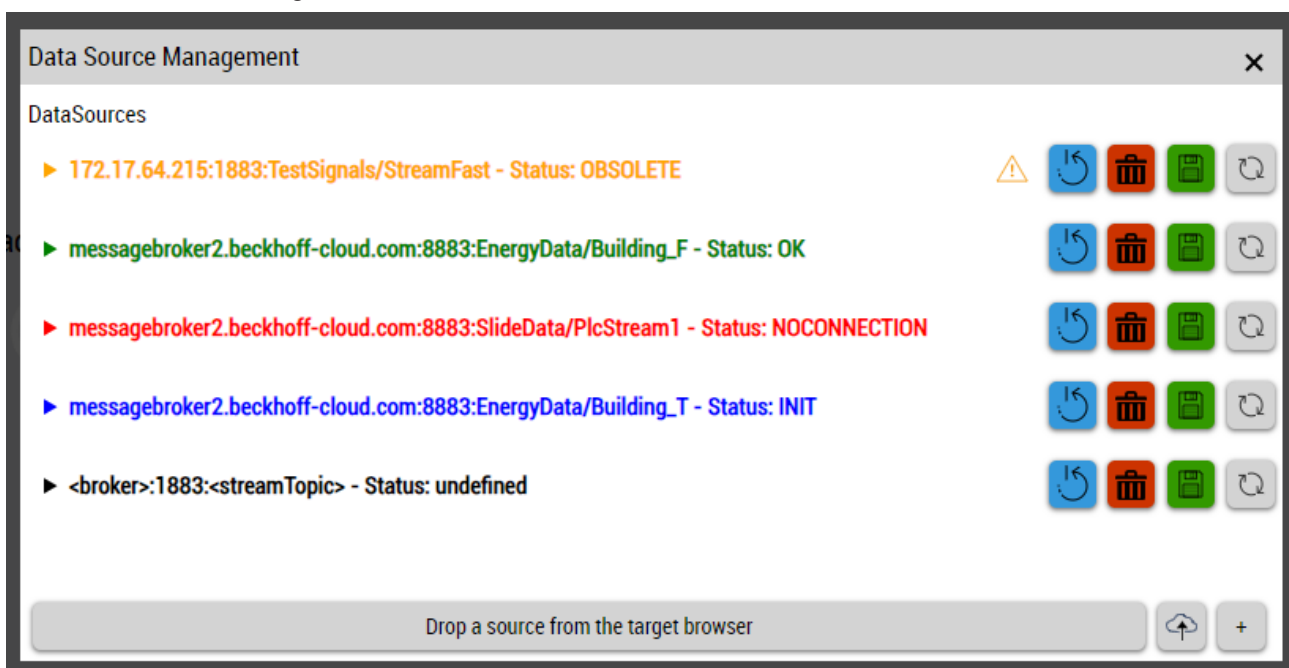
Alternatively, you can also open Data Source Management via an existing **Interactive Chart**.



Overview of data sources

All existing data sources are displayed in the Data Source Management. The status of each data source is also displayed. This has the following meanings:

- OK: The data source has been configured correctly.
- NOCONNECTION: The data source was configured incorrectly. In this case, the connection parameters should be checked and corrected.
- OBSOLETE: The Storage Provider of the data is outdated and should be updated to at least version $\geq 1.0.12.3$. Otherwise, data cannot be queried correctly.
- INIT: A new configuration has been sent to the server extension. This attempts to connect to the data source.
- UNDEFINED: The configuration has only been created locally and has not yet been saved in the server extension.
- NOSOURCEHANDLER: The configuration could not be loaded correctly in the server extension and should be saved again.

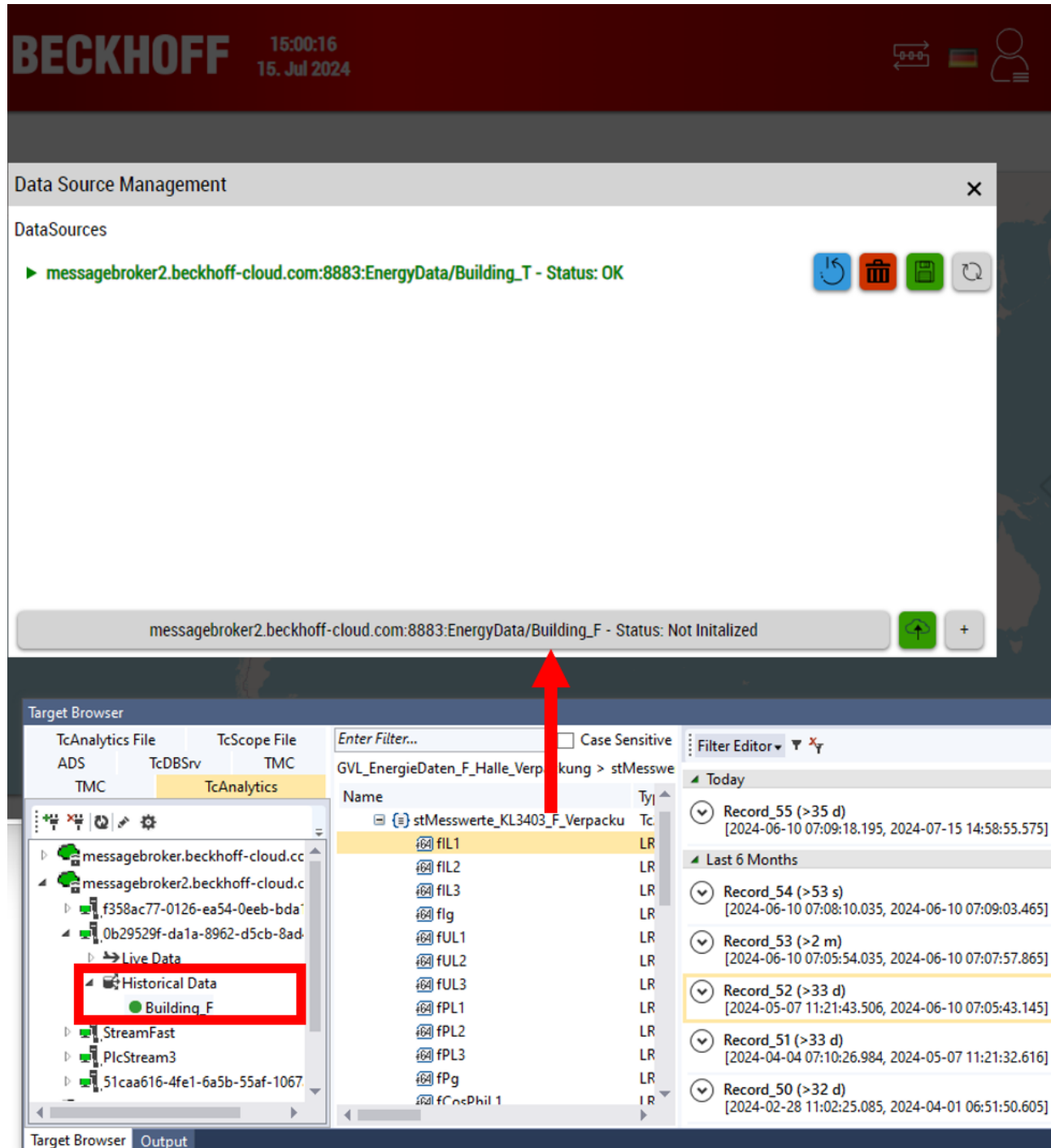


Creating a data source

New data sources can be added simply by drag and drop from the TargetBrowser. In the **TcAnalytics** tab, select Historical Data sources.

All symbols are automatically added to the data source.

If the parameters have been read successfully, the upload icon lights up green. Click on the upload icon to send the configuration to the server extension.



If the Target Browser is not available on the device, you can create an empty data source using the plus icon in the bottom right-hand corner. This must be configured manually.

Configuring and changing a data source

You can use the arrow to the left of a data source to expand and edit it.

Data Source Management

DataSources

▼ <broker>:1883:<streamTopic> - Status: undefined

Source ID: 00000000-0000-0000-0000-000000000000

Message Broker: <broker>

Broker Port: 1883

Topic: <topic>

Stream Topic: <streamTopic>

CycleTime:

Layout: 00000000-0000-0000-0000-000000000000

Stream SystemID: 00000000-0000-0000-0000-000000000000

Drop a source from the target browser

The following parameters can be set:

Parameter	Description
Source ID	Unique ID of a data source. This is unique and is assigned by the server extension.
Message Broker	Address of the message broker
Broker Port	Port of the message broker
Topic	MQTT topic of the historical stream „.../TcAnalyticsStorageProvider/.../Historical/Stream...“
StreamTopic	Original topic of the recorded stream.
Cycle Time	Cycle time of the data source in 100 nanoseconds.
Layout	Layout of the stream
Stream SystemID	SystemID of the recorded stream.
UserID	User name for accessing the message broker.
Password	Password for access to the message broker.
With certificates	If the value is checked, the selected certificate files are used.
CA	CA file (can be uploaded to the server extension).
CERT	CERT file (can be uploaded to the server extension).
KEY	KEY file (can be uploaded to the server extension).
Key Password	Password for the KEY file.

The changes can be sent to the server extension via the save icon (green). Alternatively, the changes made locally can be undone using the reset icon (blue). The status of the data source can be queried again via the refresh icon (gray).

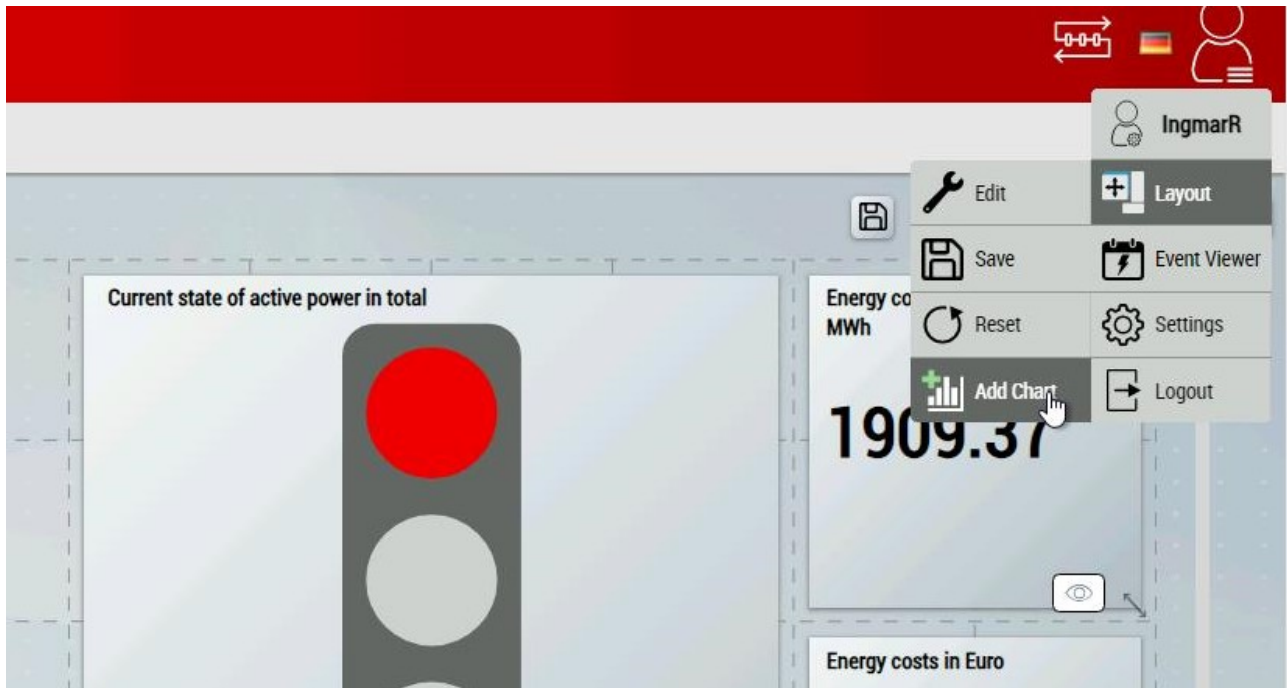
Deleting a data source

A data source can be deleted using the corresponding delete icon (red). Please note that all charts using this data source will no longer work. If parameters are changed, such as the broker address or the renewal of certificates, it is advisable to adapt existing data sources and not to delete them.

5.6.7.2 Adding and removing charts

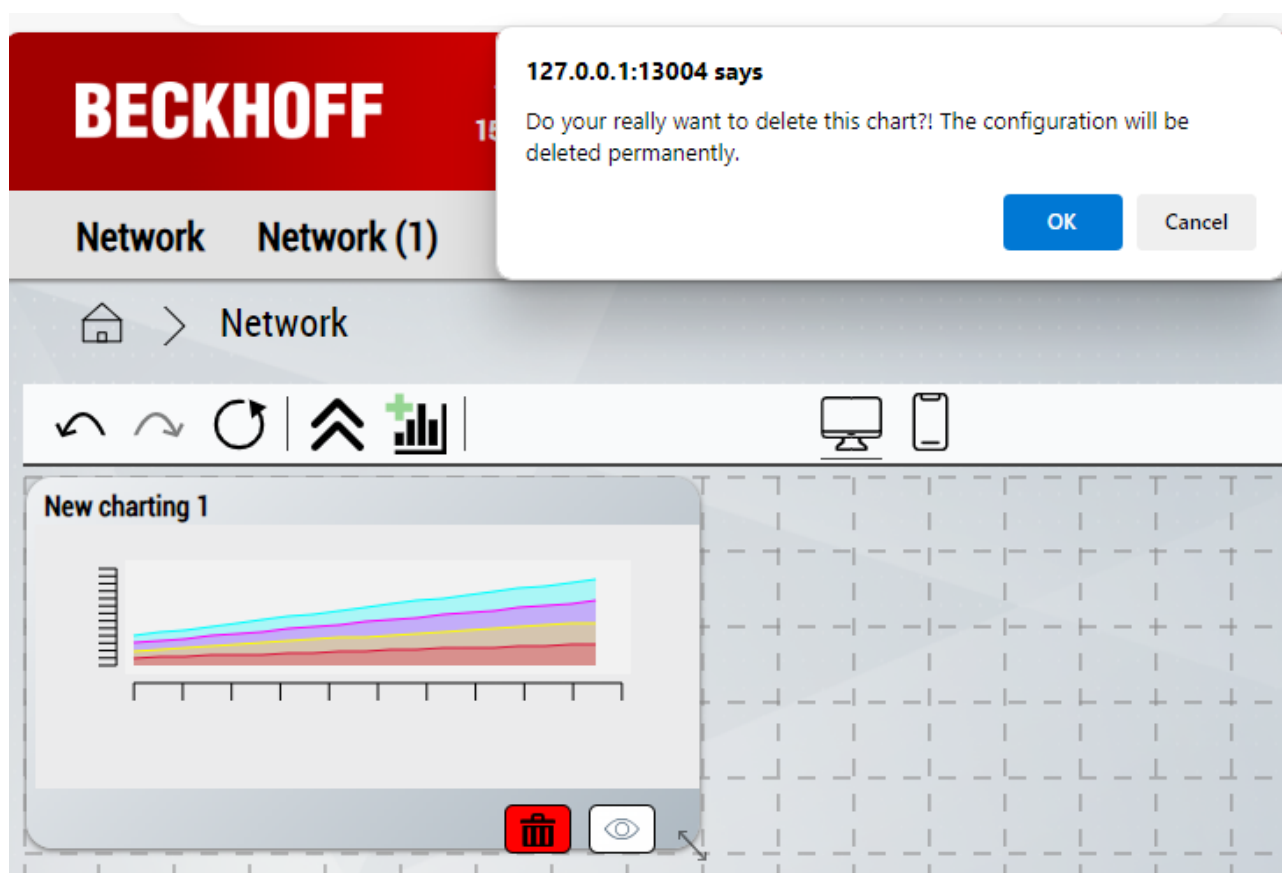
In the web browser

Add new charts via the menu icon. Click on **Add Chart** under the Layout category. The new chart is inserted at the first free position in the layout.



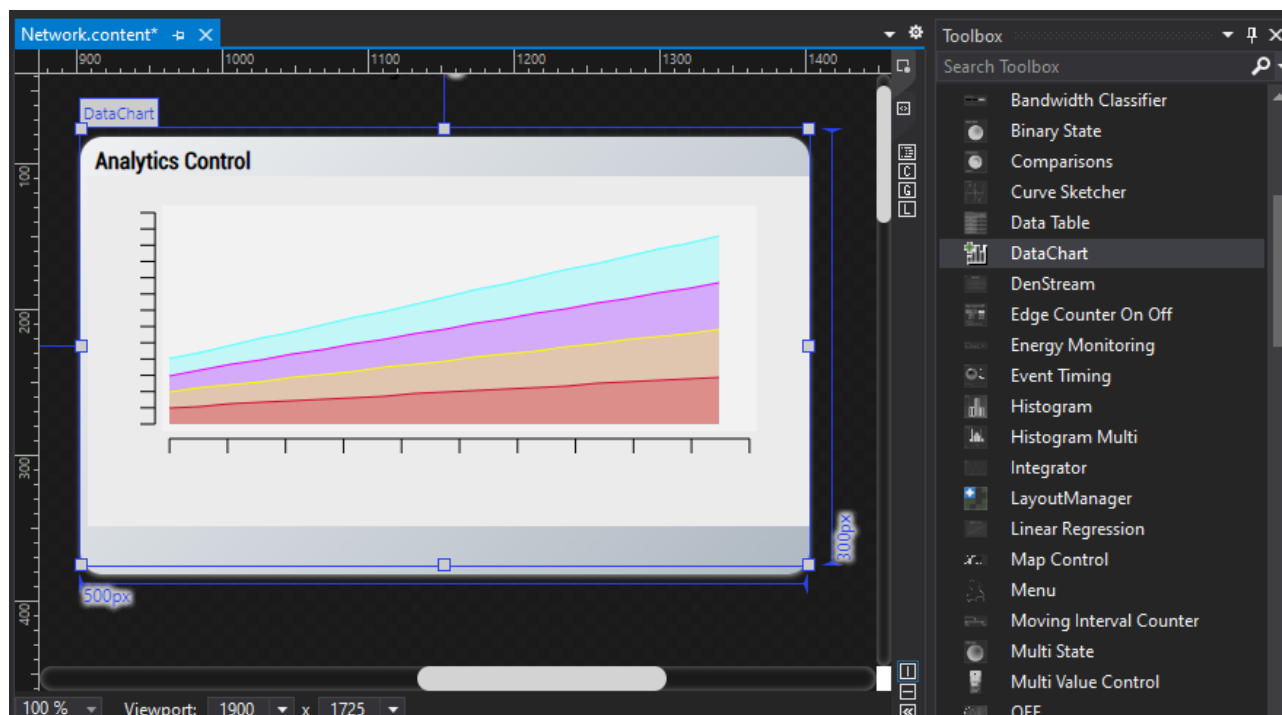
Charts can only be added if the user has authorization to edit the layout.

The added chart is user-specific and can only be viewed by the current user. Its position and size can be moved within the layout as required. You can remove a chart in Edit mode using the recycle bin icon in the bottom right-hand corner.



In HMI engineering

Charts can also be added via the HMI engineering. These charts are available to all users by default and cannot be removed. The [configuration](#) [► 431] of charts added in Engineering is carried out in the web browser.

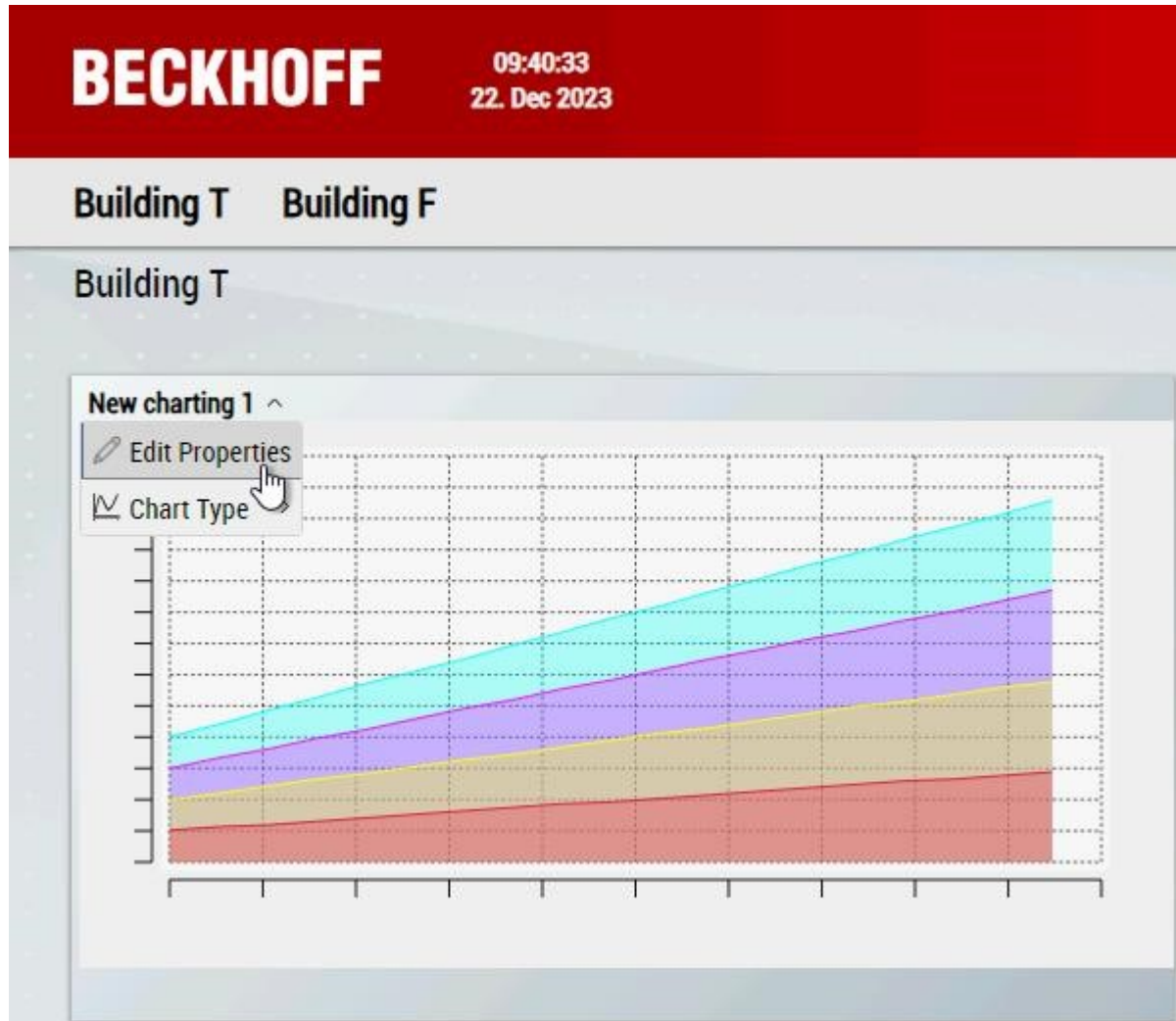


5.6.7.3 Configuration of a chart

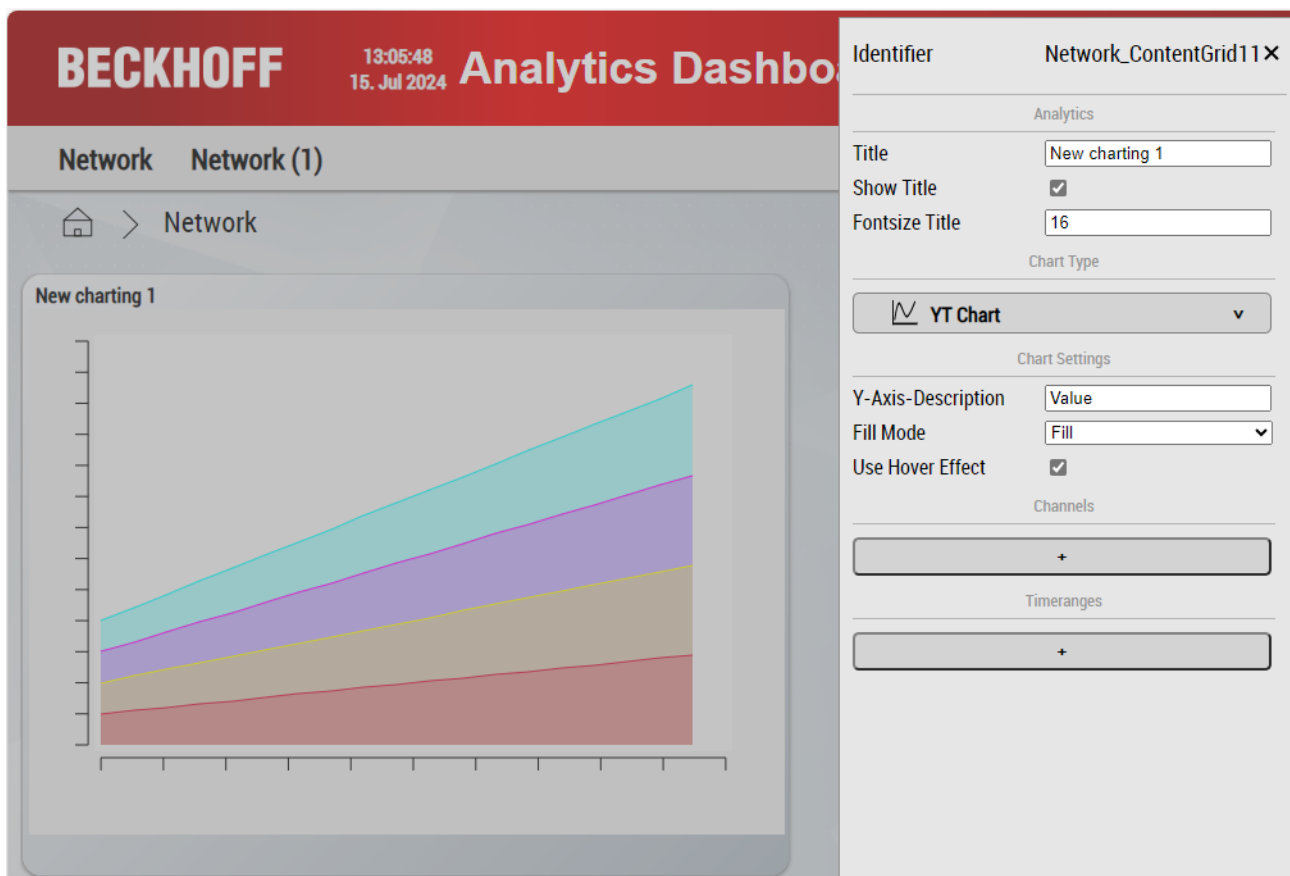
A chart is configured in the web browser. You can edit all settings (such as chart type, variables and time periods) via the properties window of the chart.

Open of the chart properties

Open the chart properties by clicking on the title of the Chart control.



Click on **Edit Properties** to open the chart properties.

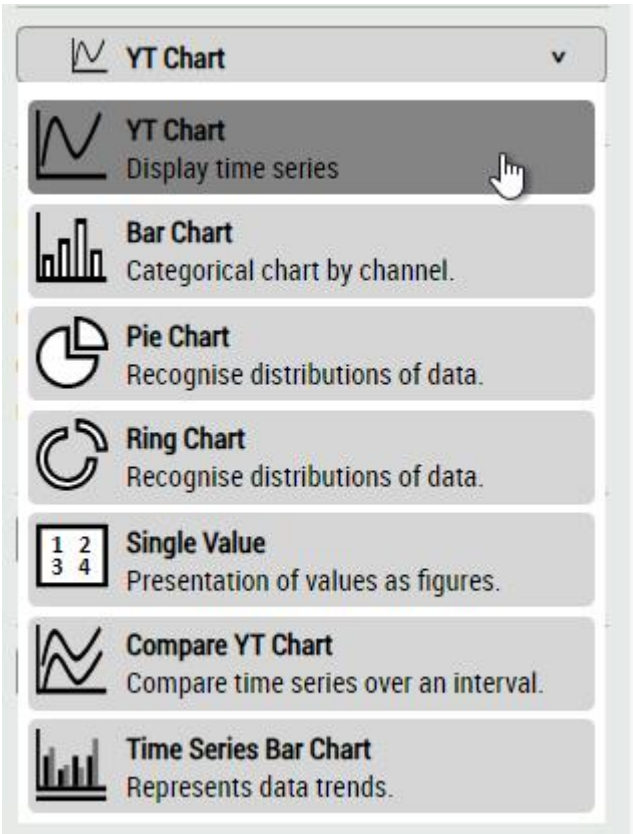


The chart properties are divided into 4 categories.

- Chart Type: Selection of the chart type
- Chart Settings: Configuration of the chart
- Channels: Selection of variables and their display.
- Timeranges: Selection of time ranges to be displayed.

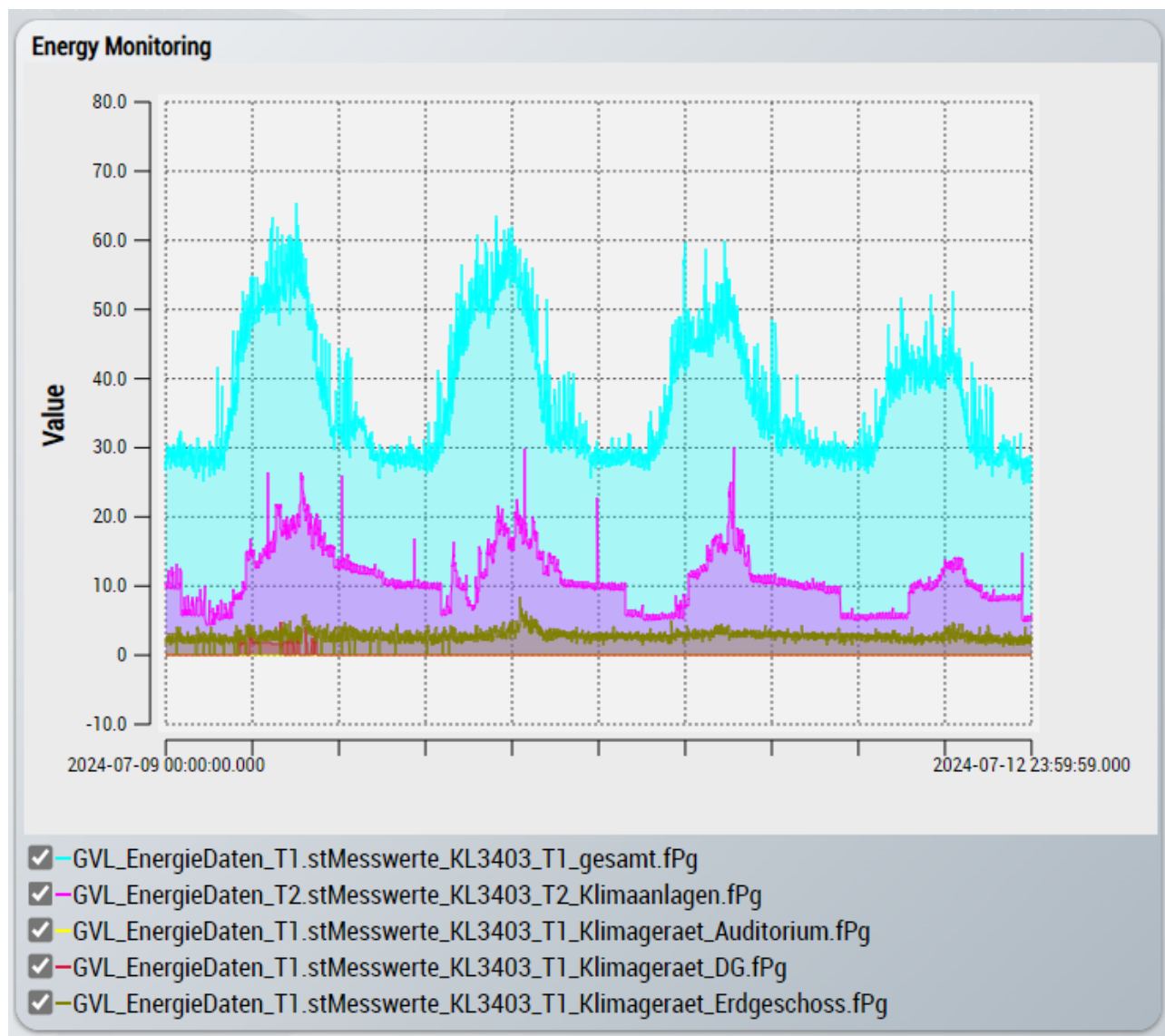
5.6.7.3.1 Chart type

You can set the chart type via the properties of the chart.



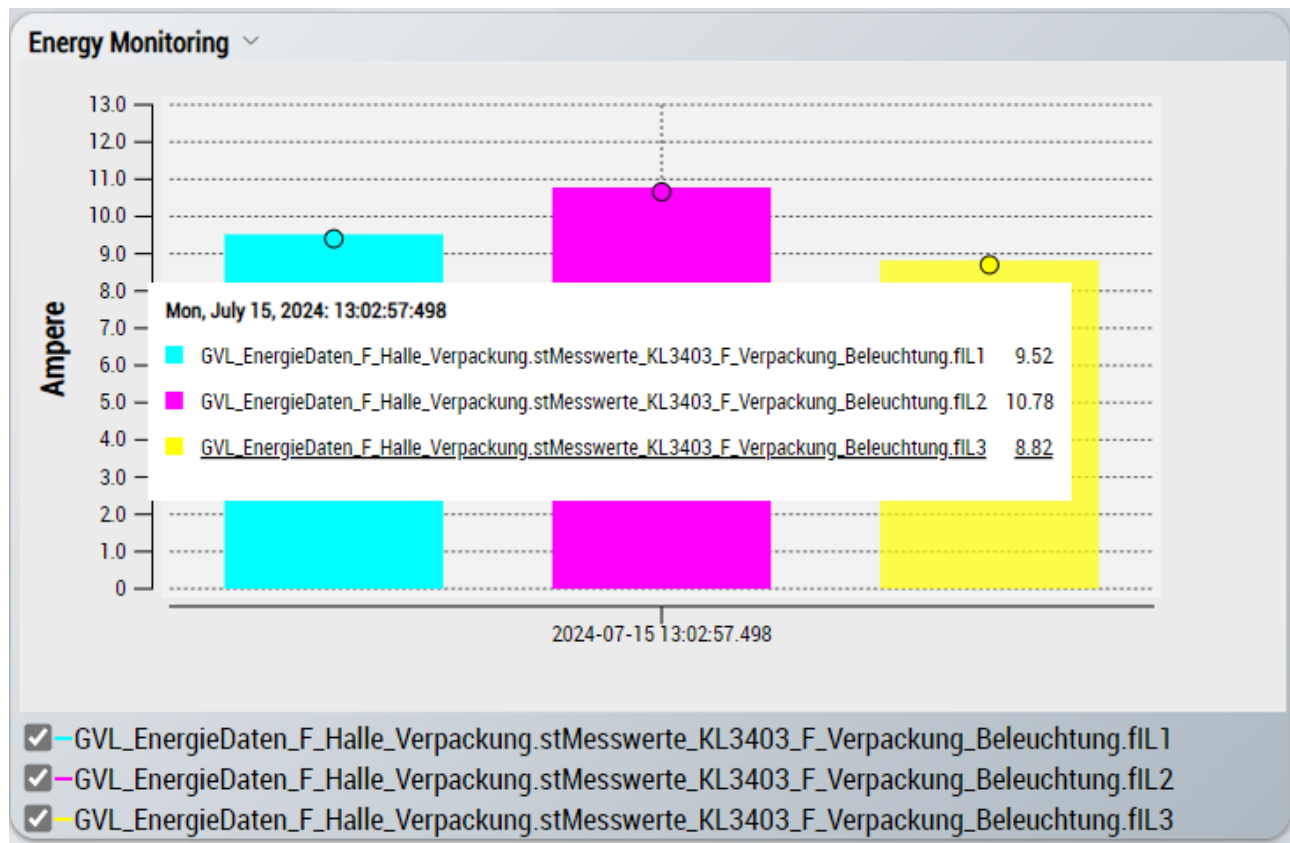
You can use the following chart types:

YT-Chart



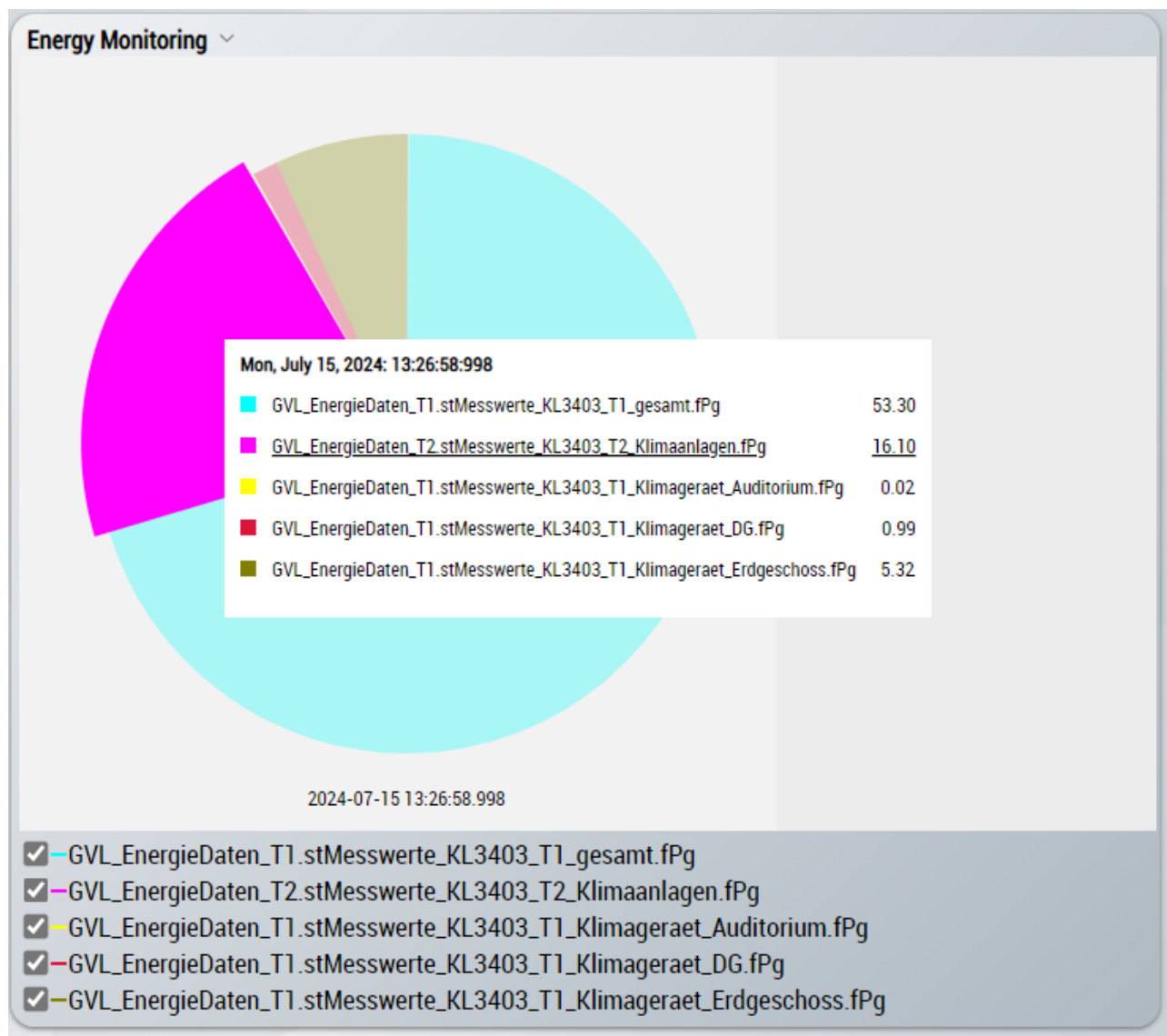
Display of the time on the X-axis and the corresponding values on the Y-axis.

Bar-Chart



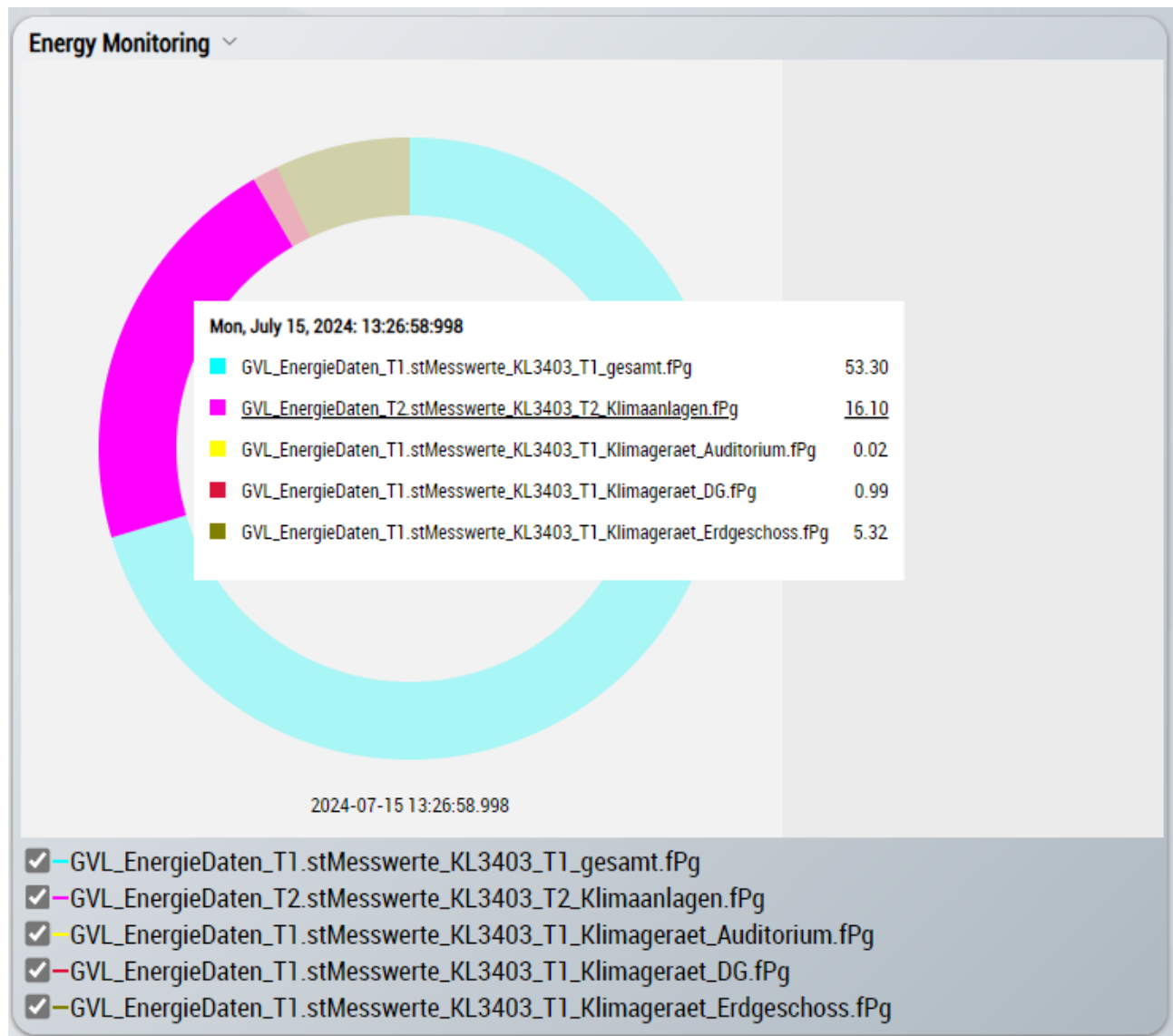
One bar is displayed per selected time range and per variable. The bar represents the last value of the time range.

Pie-Chart



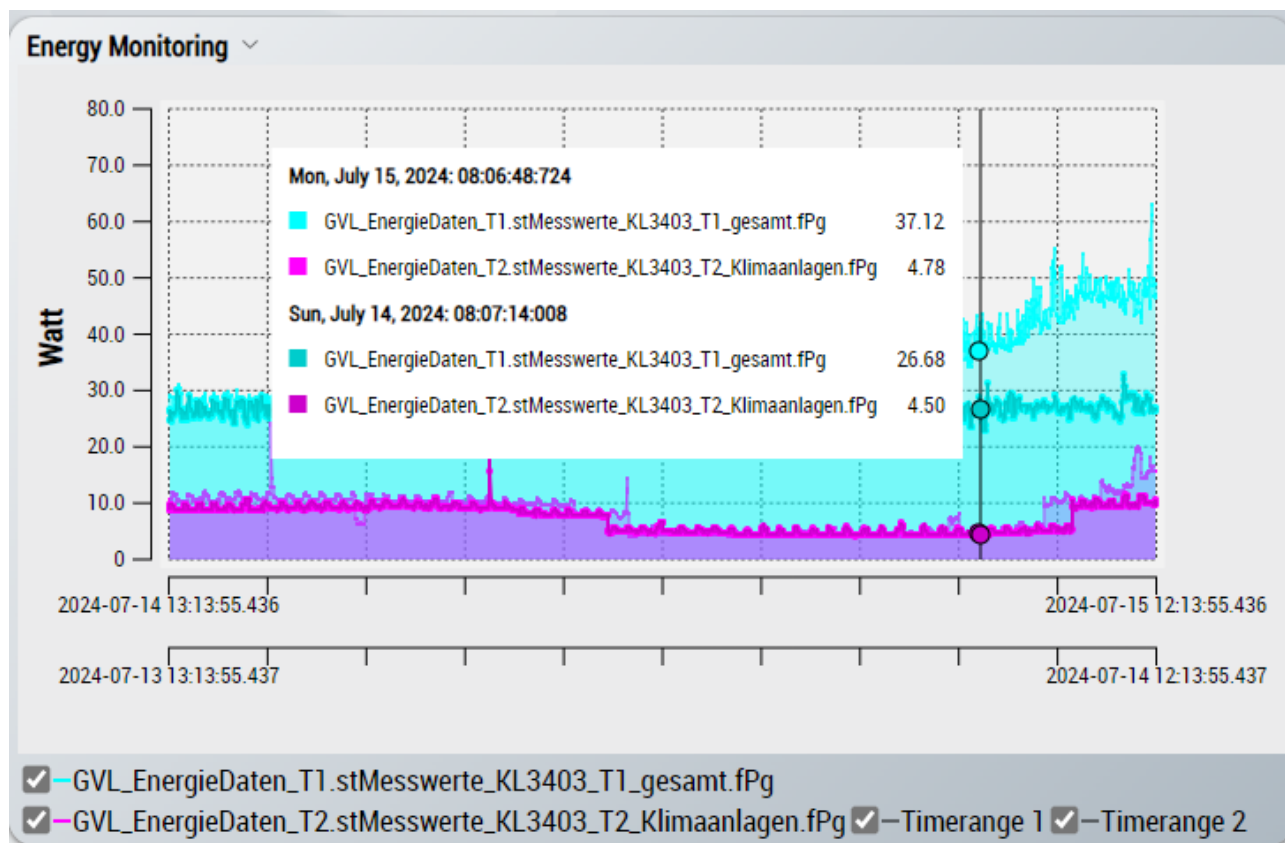
A pie is displayed for each selected time range, which represents the last sample of the time range. Negative variable values are not displayed.

Ring-Chart



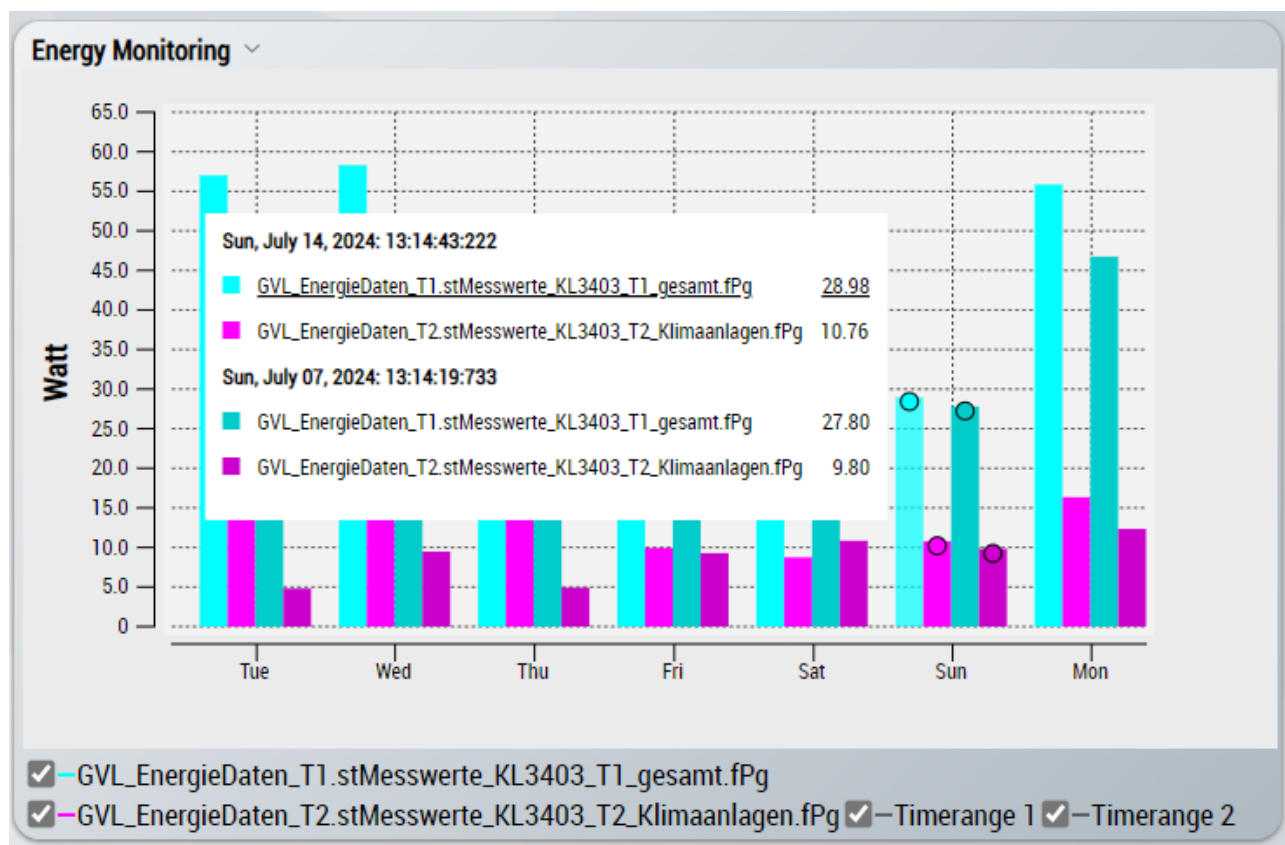
A ring is displayed for each selected time range, which represents the last sample of the time range. Negative variable values are not displayed.

Compare-YT-Chart



Suitable for comparing several time series. This involves comparing the data over the selected comparison interval. Several comparison modes are available for the comparison. The comparison period is shown on the X-axis.

Time-Series-Bar-Chart



Suitable for comparing several time series. This involves comparing the data over the selected comparison interval. Several comparison modes are available for the comparison. Bar charts are generated per selected timespan and per variable based on the comparison timespan.

5.6.7.3.2 General settings

You can make general chart settings in the **Chart Settings** area.

Identifier
NetworkContentGrid0 ✕

Analytics

Title
Energy Comparison

Show Title
☒

FontSize Title
16

Chart Type

 Compare YT Chart

Chart Settings

Y-Axis-Description
Power [W]

FillMode
Fill

Use Hover Effect
☒

Channels

Channel 1
GVL_EnergieDaten_T1.st

Channel 2
GVL_EnergieDaten_T1.st

+

Timeranges

Refresh Interval
1
Minute(s)

Compare Type
Absolute

Compare Interval
10
Minute(s)

Timerange 1 (Reference) ✕

From
Now - Last 1 day

To
From + Compare Interval

Last 1 day

Timerange 2 ✕

From
Now - Last 1 hour

To
From + Compare Interval

Last 1 hour

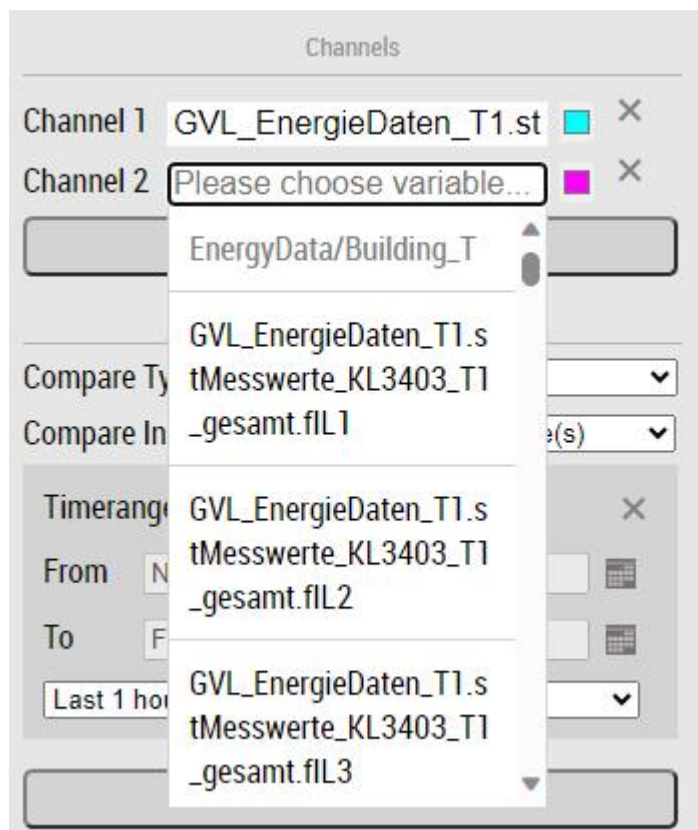
+

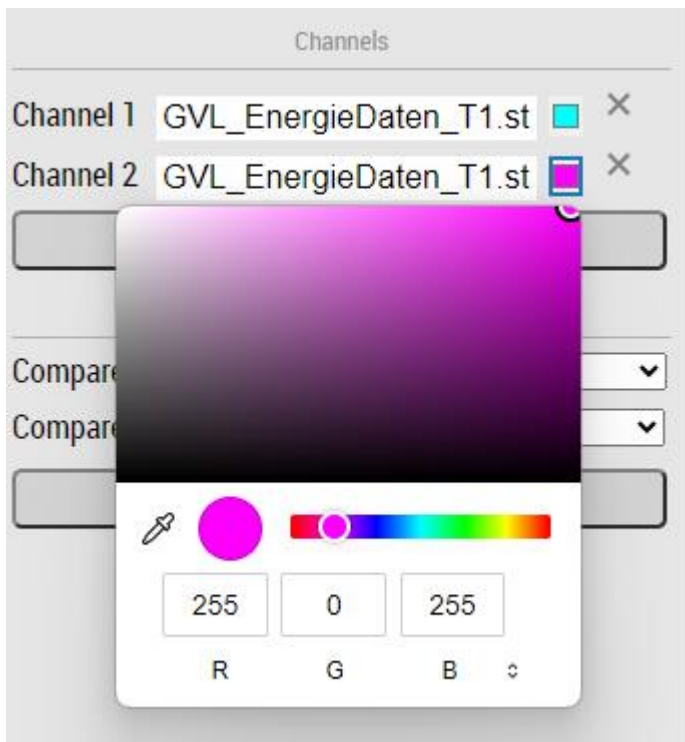
The following options can be set here:

Property	Description
Y-Axis-Description	Labeling the Y-axis
FillMode	Selection of whether the graphs should be filled.
Use Hover Effect	If this option is enabled and the mouse pointer is in the chart, the exact time points and variable values are displayed in a window.

5.6.7.3.3 Adding variables

In the **Channels** area, you can select the variables to be displayed. Use the plus icon to add a new channel for the chart. Define a variable for the channel via a drop-down window. Use the color icon to assign an individual color for the channel. Remove the channel using the X icon.





5.6.7.3.4 Adding time ranges

The time ranges to be displayed can be selected in the "Timerranges" area. A new time range can be added to the chart using the plus icon.

Absolute time ranges

For a time range, the start and end time can be selected using the "From" and "To" parameters. The date can be selected from the calendar (opened by clicking on the calendar icon). The days are marked in color in the calendar:

- Green: Day has been fully recorded.
- Yellow: Day was only partially recorded.
- Gray: No data available for this day.

Timeranges

Refresh Interval

1

Minute(s)

Compare Type

Absolute

Compare Interval

10

Minute(s)

Timerange 1 (Reference)

From

Now - Last 1 day

To

From + Compare Interval

Last 1 day

Timerange 2

From

Now - Last 1 day

To

From + Compare Interval

Last 1 hour

Timerange 3

From

Now - Last 1 hour

To

From + Compare Interval

Last 1 hour

Relative time ranges

As an alternative to absolute time ranges, relative time ranges can also be selected. The selection of a relative time range can be made via the drop-down window. As soon as at least one relative time range has been selected, the "Refresh Interval" property can be used to specify the time interval at which the chart is updated.

Timeranges

Refresh Interval Minute(s) ▼

Compare Type ▼

Compare Interval Minute(s) ▼

Timerange 1 (Reference) ✕

From 📅

Choose interval

- Last 1 minute
- Last 3 minutes
- Last 5 minutes
- Last 10 minutes
- Last 15 minutes
- Last 30 minutes
- Last 1 hour
- Last 2 hours
- Last 3 hours
- Last 6 hours
- Last 12 hours**
- Last 1 day
- Last 2 days
- Last 3 days
- Last 1 week
- Last 2 weeks
- Last 3 weeks
- Last 4 weeks
- Last 6 weeks

▼

● Automatic setting of the end time

i For the Compare-YT-Chart and Time-Series-Bar-Chart, the end time of a time range is set based on the configured comparison interval. Manual selection of the end time is not possible in these chart types.

5.6.7.4 Comparing data

The CompareCompare-YT-Chart and TimeSeries-Bar-Chart can be used to compare data from different time ranges. Up to three time ranges can be added in the settings for this purpose. The first time range serves as a reference for the display.

The data is compared over a defined timespan (e.g. 3 hours, 1 week, ...) The length of the comparison period can be configured in the "Timeranges" area if Compare-YT-Chart or TimeSeries-Bar-Chart is selected as the chart type.

Timeranges

Refresh Interval Minute(s) ▼

Compare Type ▼

Compare Interval Minute(s) ▼

Timerange 1 (Reference) ✕

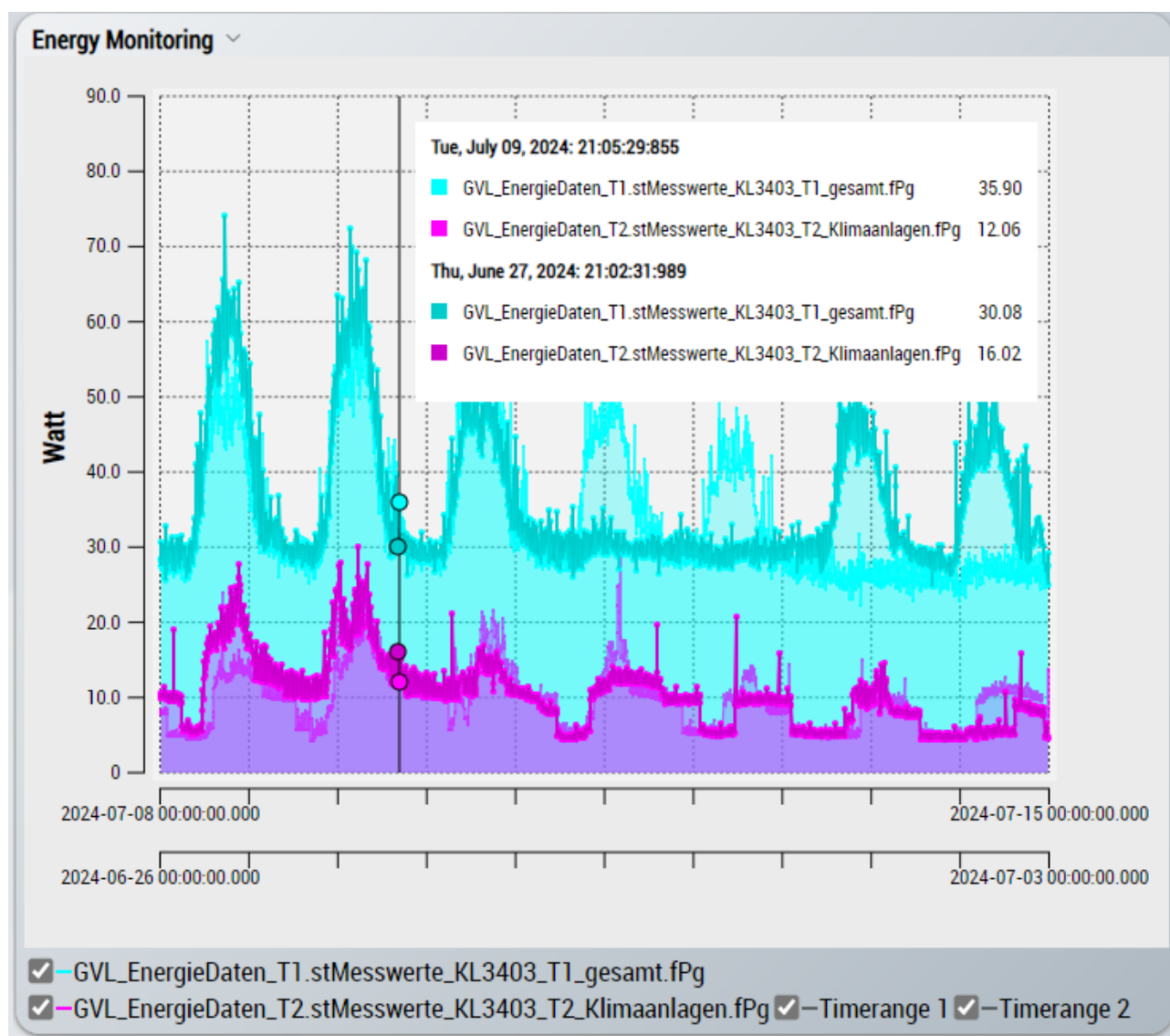
From 📅

To 📅

▼

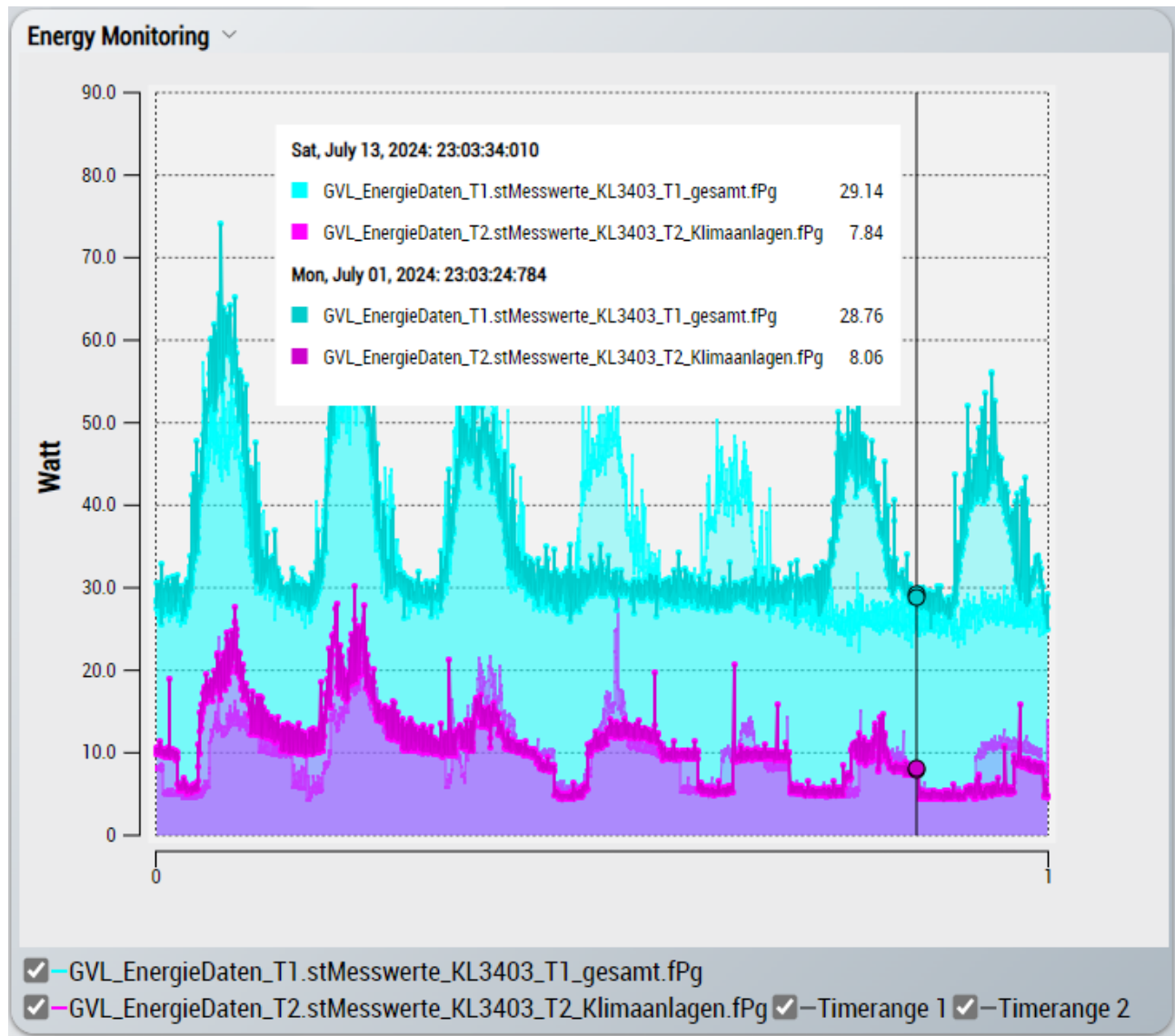
In addition to the timespan, a comparison mode can be set. This can be used to set whether the data should simply be displayed on top of each other or whether they should be displayed synchronized with each other. This corresponds to a shift on the time axis in the diagram. The available comparison modes are:

Absolute



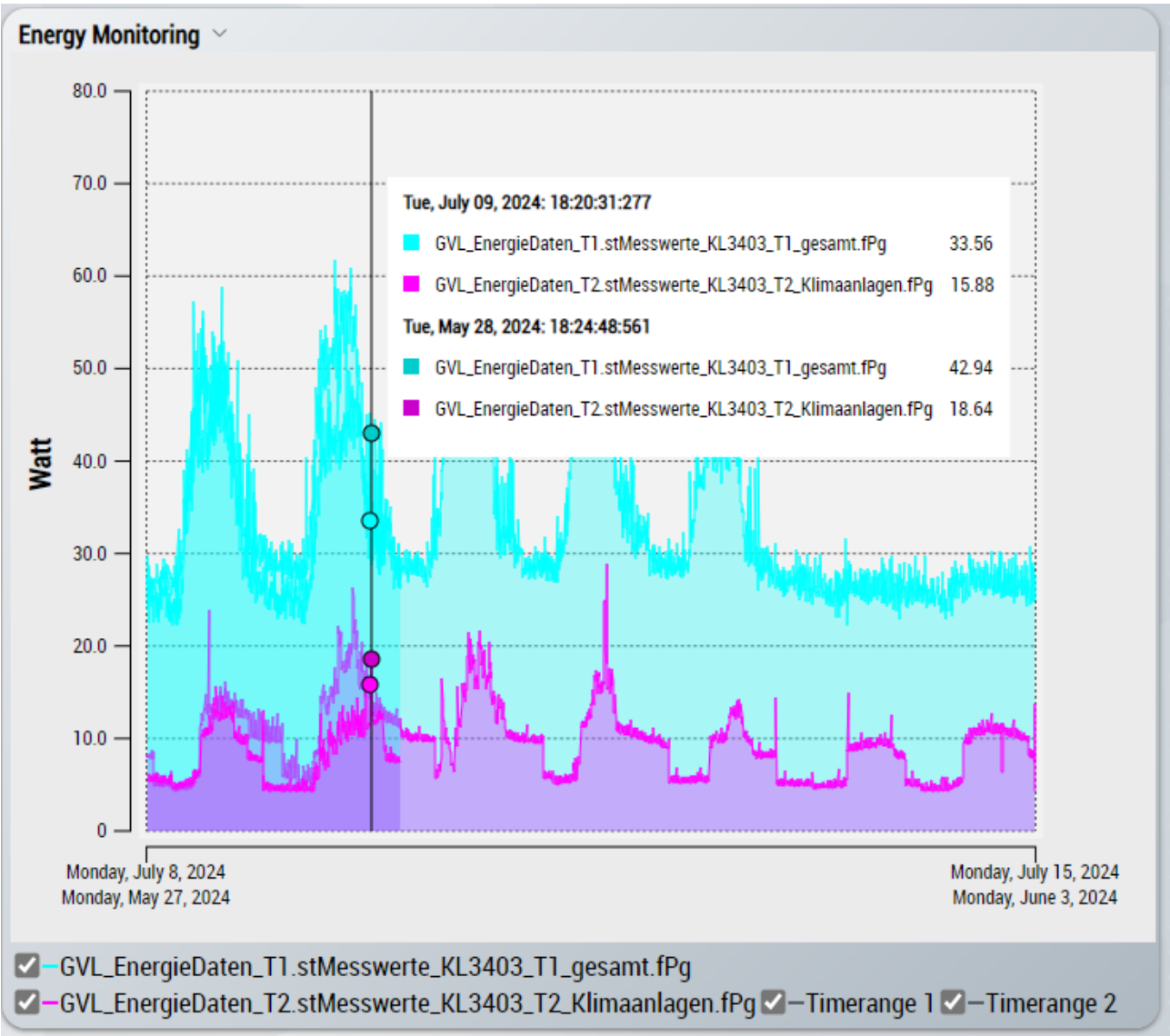
In this mode, all selected time ranges are visualized one above the other. The absolute time is displayed on the X-axis. There is no shift on the time axis. The origin is always the start time of a timespan.

Relative



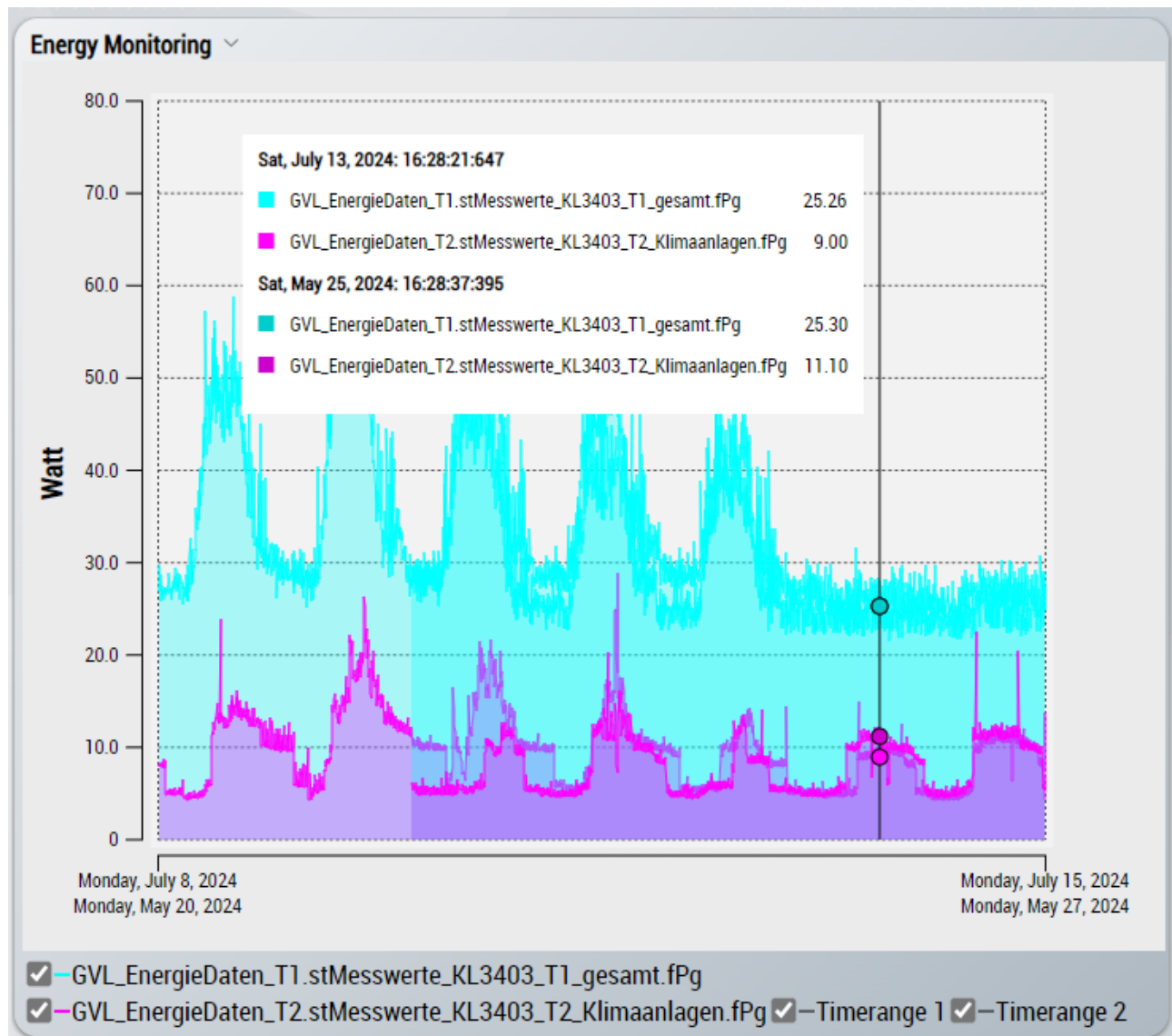
In this mode, all selected time ranges are visualized one above the other. The relative elapsed time is shown on the X-axis. There is no shift on the time axis.

Match Start



In this mode, the data is superimposed at the correct time. The start time of the first time range serves as a reference. The data for the other points in time are shifted so that the data at the beginning of the chart are aligned. The basis for this is the selected comparison period (see table).

Match End



In this mode, the data is superimposed at the correct time. The end time of the first time range serves as a reference. The data for the other points in time are shifted so that the data at the end of the chart overlap in time. The basis for this is the selected comparison period (see table).

Comparison period	Procedure
Second(s)	Comparison of the milliseconds of data. As soon as the same millisecond occurs, the data is superimposed.
Minute(s)	Comparison of the seconds of data. As soon as the same second occurs, the data is superimposed.
Hour(s)	Comparison of the minutes of the data. As soon as the same minute occurs, the data is superimposed.
Day(s)	Comparison of the hour of data. As soon as the same hour occurs, the data is superimposed.
Week(s)	Comparison of the weekday of the data. As soon as the same day of the week occurs, the data is superimposed.
Month(s)	Comparison of the day of the data. As soon as the same day (date) occurs, the data is superimposed. One month always corresponds to 31 days.
Year(s)	Comparison of the month of data. As soon as the same month occurs, the data is superimposed.

- **Shifting the data**

I As it is not guaranteed that the data is recorded at the same cycle time or that the data is scanned at the same sampling rate, it is possible that the values do not always overlap exactly.

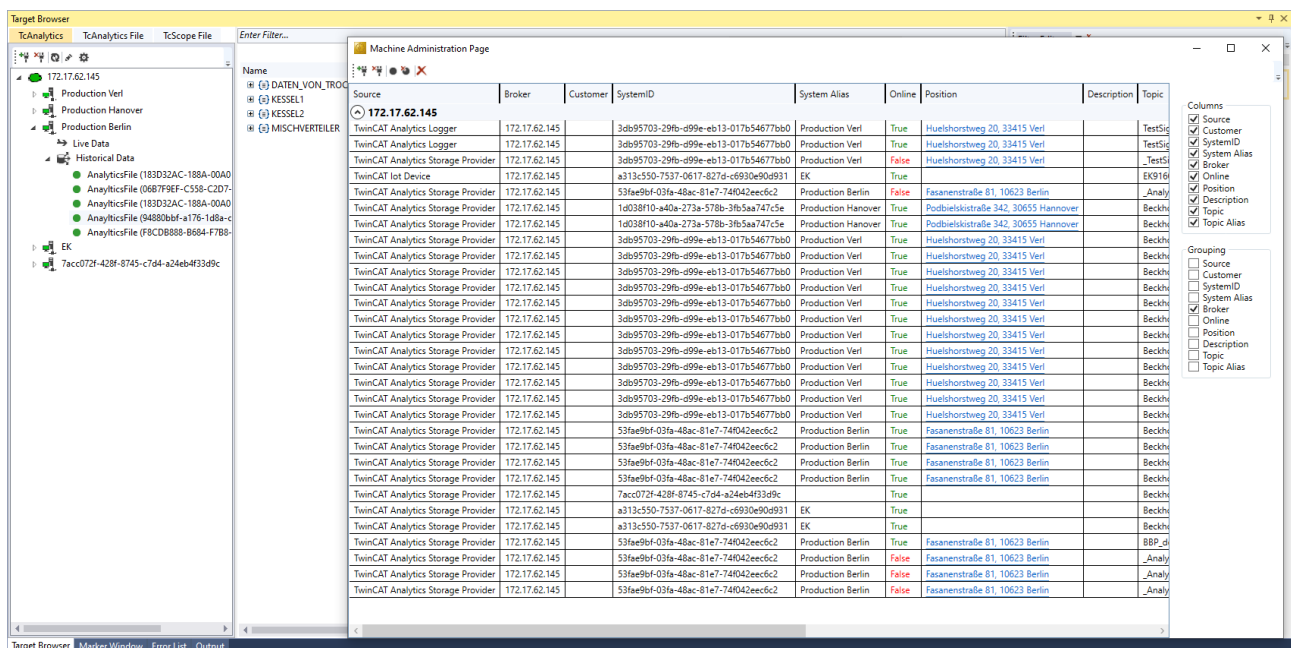
5.6.8 Switching multiple machines in the HMI Dashboard

In TwinCAT Analytics you can use different data streams from several machines, which you can switch in an analysis. This is possible both in the Analytics Workbench via the Virtual Input Source and in the fully generated PLC and HMI. Both live and historical data can be used. For each data stream, you can add a brief description and the location either in the [Analytics Logger](#) or in the Machine Administration.

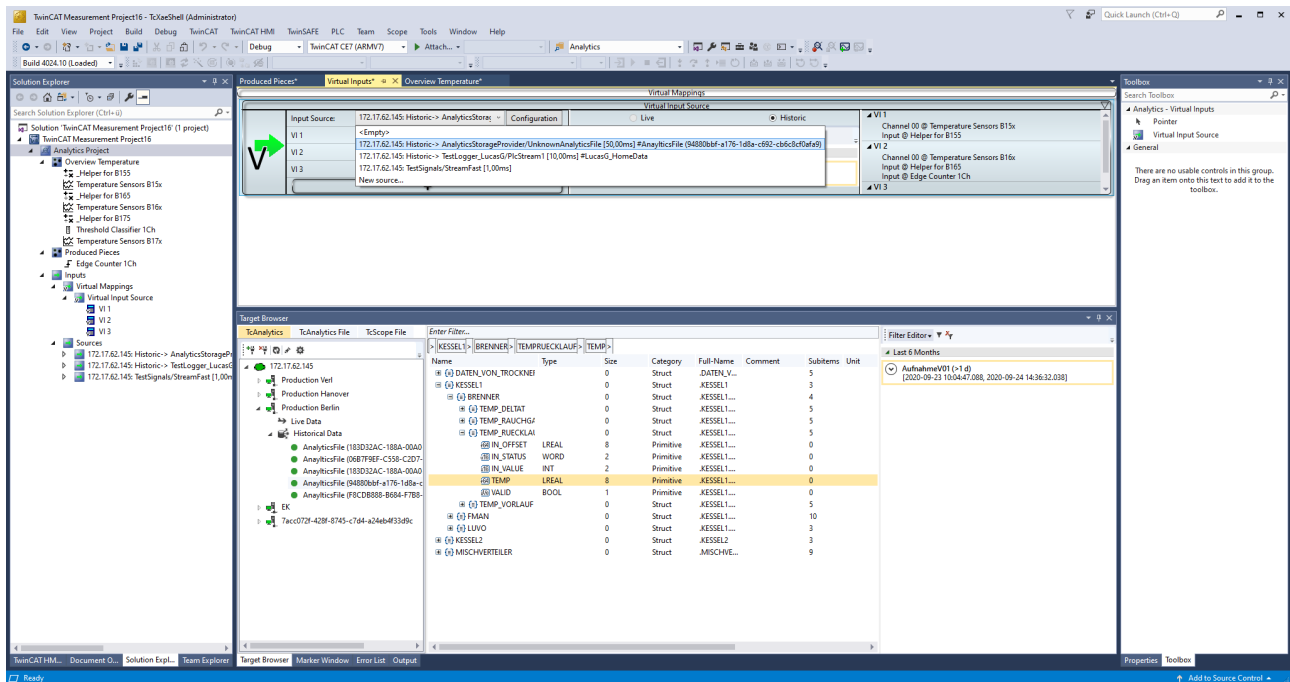
Machine Administration page

If not already done in the Overview, set the metadata of your machine on the Analyse data page. Open the TwinCAT Target Browser (**TwinCAT > Target Browser > Target Browser**) and click the gear icon. Now you can enter the location of your machine, a short description and the name of your machine. Note that existing data in an Analytics project is not assigned this meta information, since the information is only transferred from the Target Browser during drag and drop. In this case, you can delete the data stream and recreate it.

For each data stream you use in your Analytics configuration, the [Runtime deployment](#) [► 303] creates a map entry. These map entries are used as input variables for the general map on the start page of your HMI dashboard.

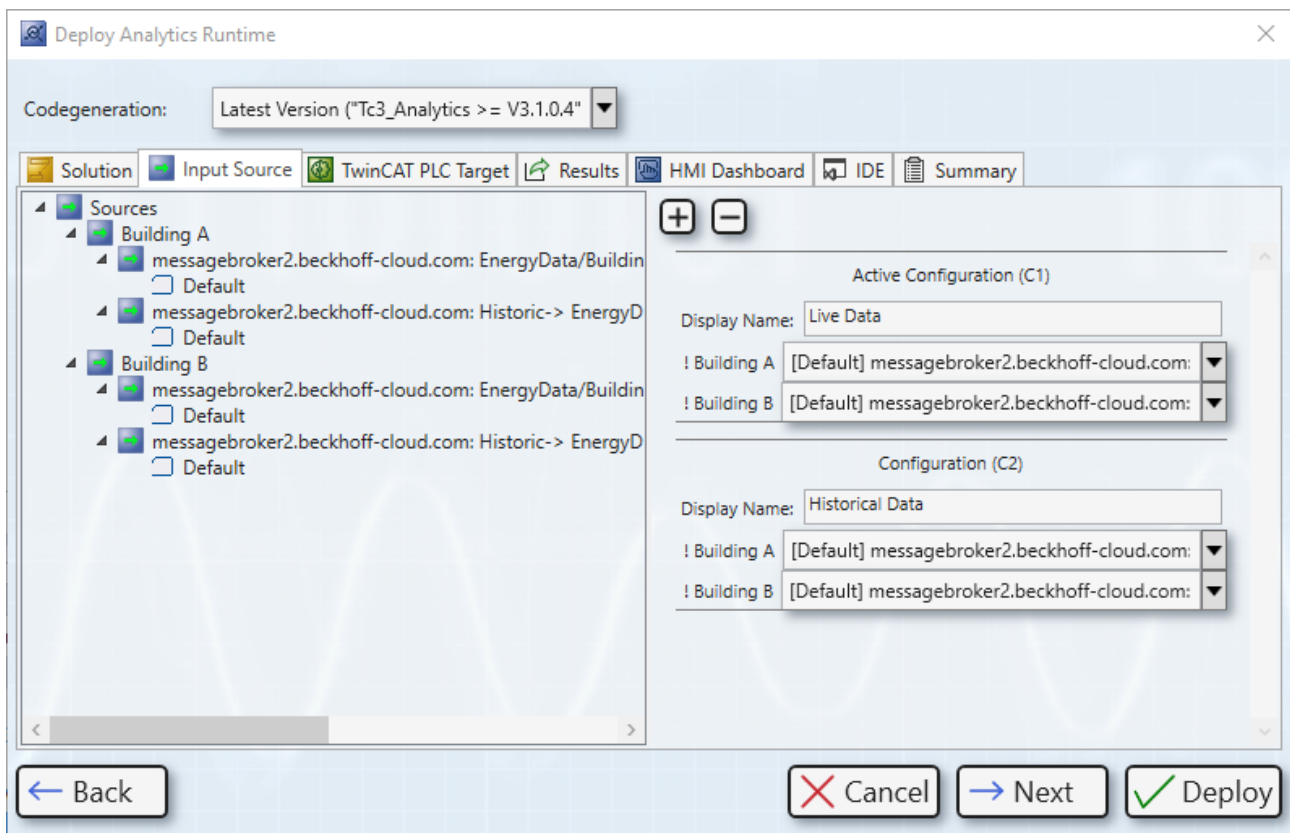


In the Analyse data the data streams are listed under **Sources**. These are created by using data from various sources from the TwinCAT Target Browser. These sources are listed in the Virtual Input Source. It is possible to switch between the sources. Check that all individual inputs are linked and that none is set to **Empty** by clicking each source once (which corresponds to a switch in the workbench). You can then generate an HMI with PLC.

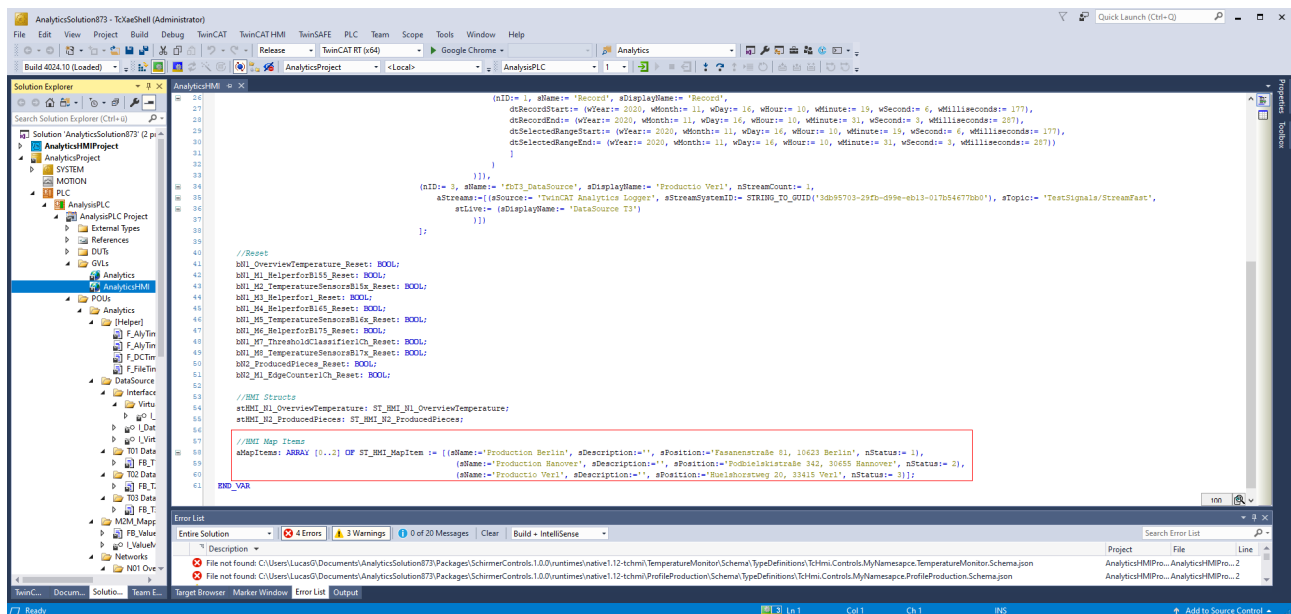


Deploy Wizard

In [Runtime deployment \[► 303\]](#), any number of configurations can be created at **Input Source**, which can be switched in the HMI. This makes it possible in the analysis HMI to analyze and switch live as well as historical data from possibly different machines. Each of the listed configurations can be analyzed in parallel. A configuration always has as many Virtual Input Sources as are configured in the Analytics Workbench.

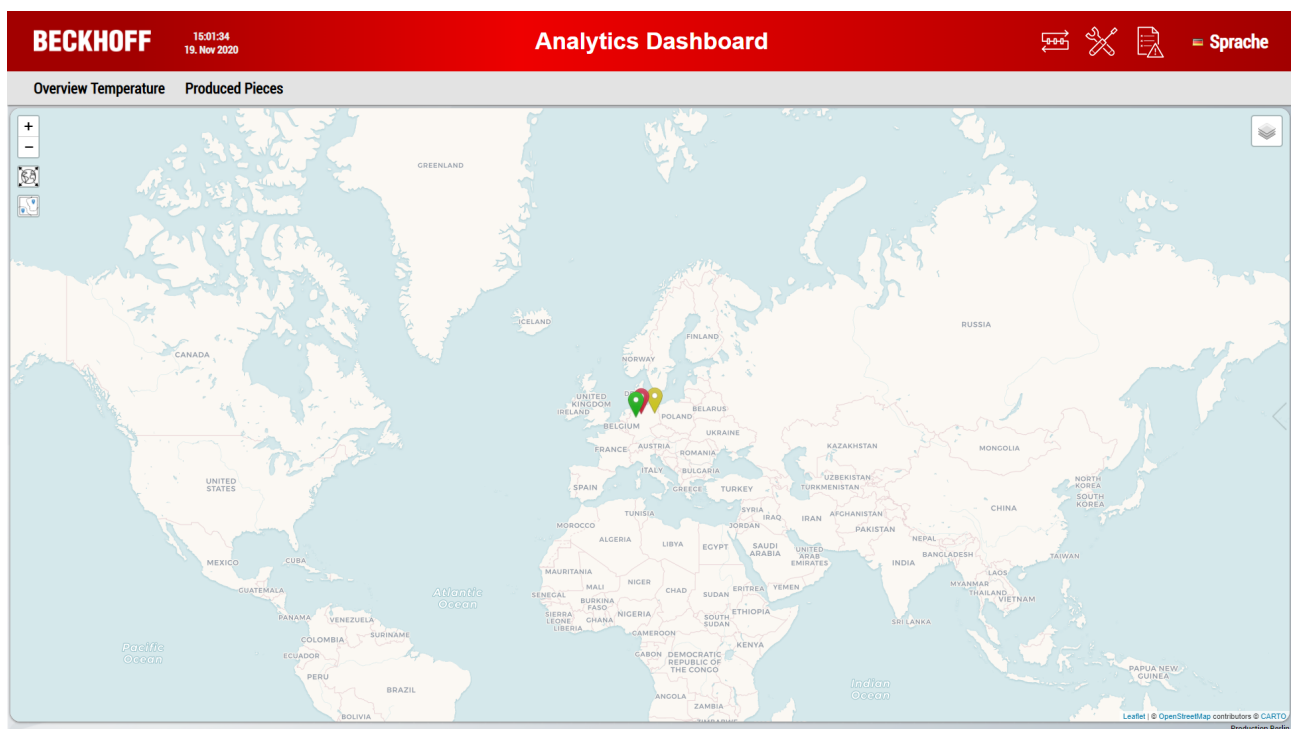


After a successful HMI and PLC generation, the names, positions, and descriptions of the input sources are entered into the global variable list of the PLC. If something is not right or you want to change a value afterwards, you can do this directly in the PLC.

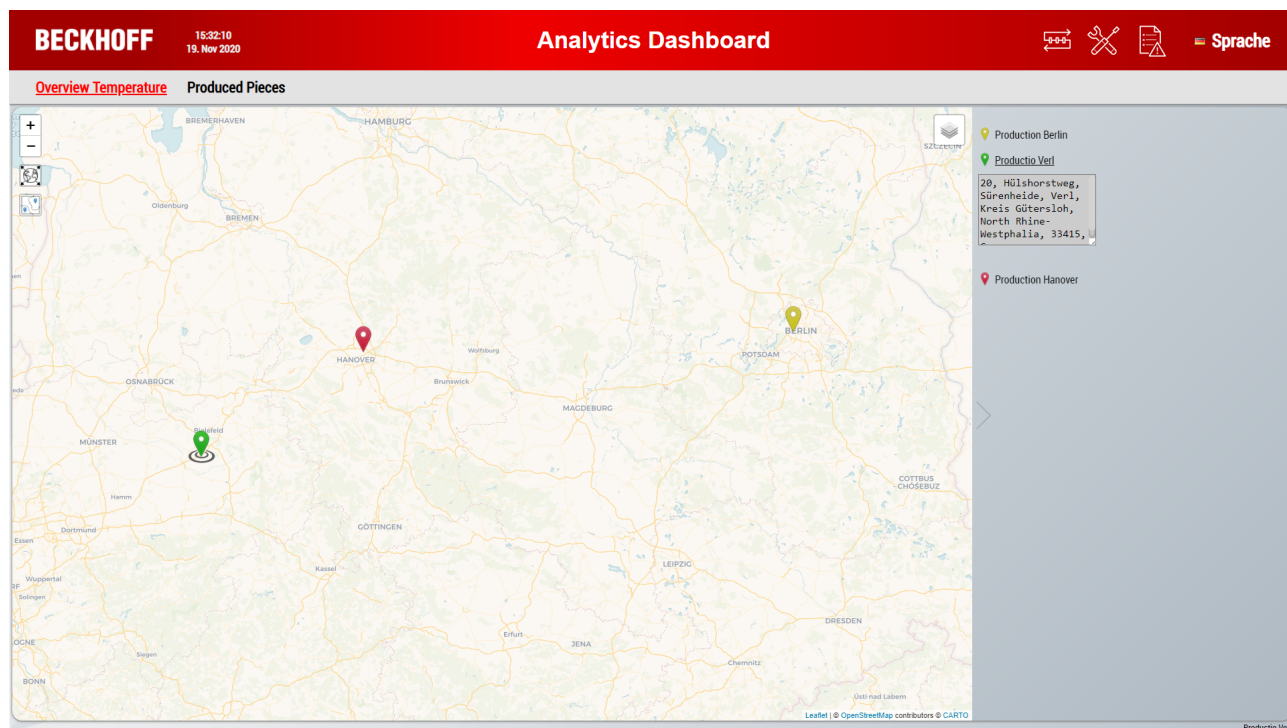


Map with machines

If you selected to create a start page, the *Map Control* will be created on the homepage of your HMI Dashboard. The map shows all machine locations and lists the names of the machines (system alias) in the legend. The legend can be opened and closed using the arrow on the right. The icon color indicates the current machine status: green = OK, yellow = Warning, red = Alarm.

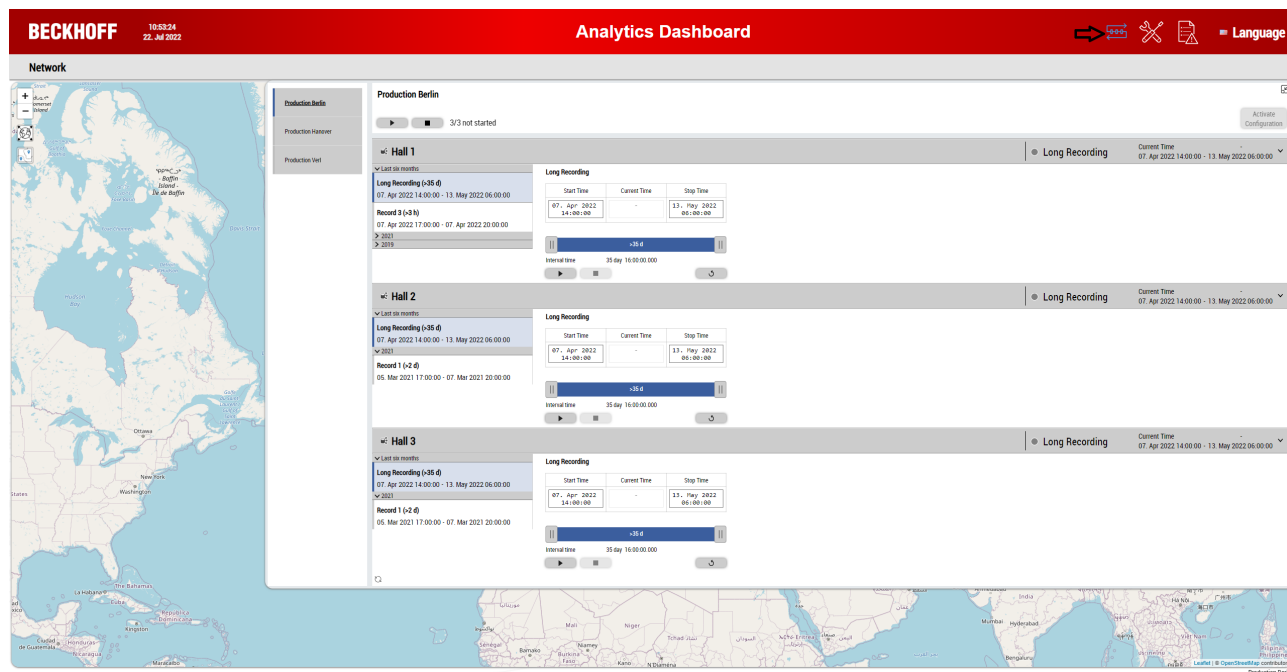


You can zoom in on a single icon by double-clicking it and zoom out to an overview of all machines using the icons on the left. It is also possible to click on an icon on the map. The corresponding legend entry is automatically highlighted. This also works the other way around, i.e. when you click on a legend entry, the corresponding icon is automatically highlighted.



Switching machine data

In addition to the map with the individual machine configurations, the analyses for the individual configurations can be switched. In the PLC all analyses run in separate tasks, which means that all analysis data is available at the same time. In the generated HMI, the various machines can be switched over in a specially developed control. The control can be opened and closed via the icon highlighted in the image below (black arrow).



The configurations that were previously shown on the map with the locations can be selected in the control. A configuration can consist of live and historical data, whereby only historical data is configured for the **Production Berlin**. Historical data can be analyzed equivalently as in [Working with Historical Data \[► 301\]](#) in the HMI.

When clicking on another machine, this configuration is only displayed in the control. To activate, click **Activate Configuration**. The control is explained below.

- BECKHOFF

 18:35:24
 22. Jul 2022

Analytics Dashboard

Language

Network

Production Berlin

10.

11.

3/3 not started

[Activate Configuration](#)

Hall 1

Last 36 months

Long Recording (-35 d)

07. Apr 2022 14:00:00 - 13. May 2022 06:00:00

Record 3 (x3M)

07. Apr 2022 17:00:00 - 07. Apr 2022 20:00:00

> 2021

< 2023

Long Recording

Start Time	Current Time	Stop Time
07. Apr. 2022 14:00:00:00	-	13. May. 2022 06:00:00:00

Internal time 35 day 16:00:00:000

▶
■
◀

Hall 2

Last 36 months

Long Recording (-35 d)

07. Apr 2022 14:00:00 - 13. May 2022 06:00:00

> 2021

Record 1 (x2 d)

05. Mar 2021 17:00:00 - 07. Mar 2021 20:00:00

Long Recording

Start Time	Current Time	Stop Time
07. Apr. 2022 14:00:00:00	-	13. May. 2022 06:00:00:00

Internal time 35 day 16:00:00:000

▶
■
◀

Hall 3

Last 36 months

Long Recording (-35 d)

07. Apr 2022 14:00:00 - 13. May 2022 06:00:00

> 2021

Record 1 (x2 d)

05. Mar 2021 17:00:00 - 07. Mar 2021 20:00:00

Long Recording

Start Time	Current Time	Stop Time
07. Apr. 2022 14:00:00:00	-	13. May. 2022 06:00:00:00

Internal time 35 day 16:00:00:000

▶
■
◀

Hall 4

Last 36 months

Long Recording (-35 d)

07. Apr 2022 14:00:00 - 13. May 2022 06:00:00

> 2021

Record 1 (x2 d)

05. Mar 2021 17:00:00 - 07. Mar 2021 20:00:00

Long Recording

Start Time	Current Time	Stop Time
07. Apr. 2022 14:00:00:00	-	13. May. 2022 06:00:00:00

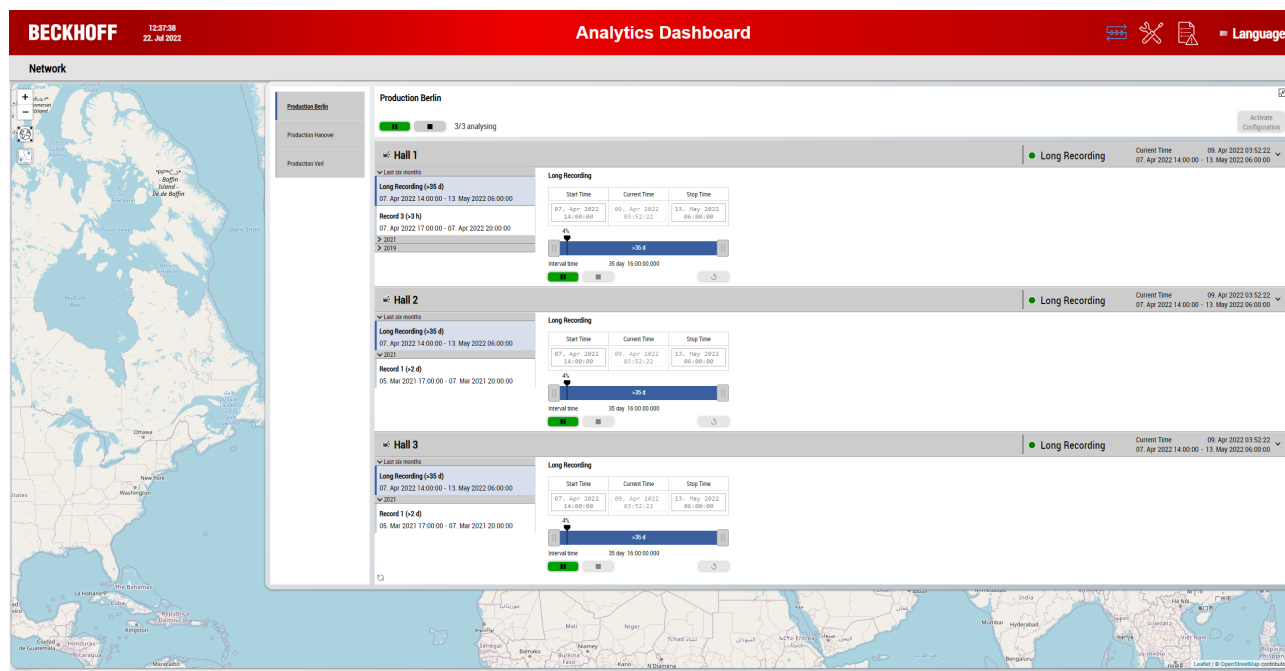
Internal time 35 day 16:00:00:000

▶
■
◀

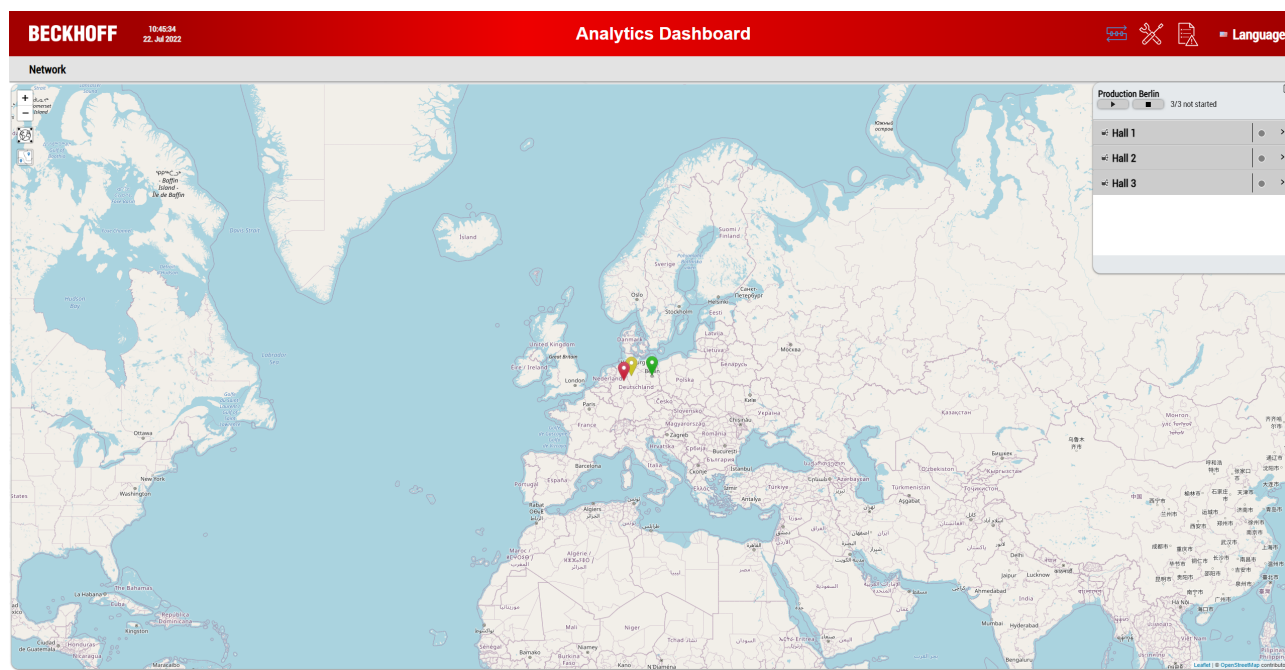
Laufzeit | © OpenStreetMap contributors | [Imagery by Mapbox](#)

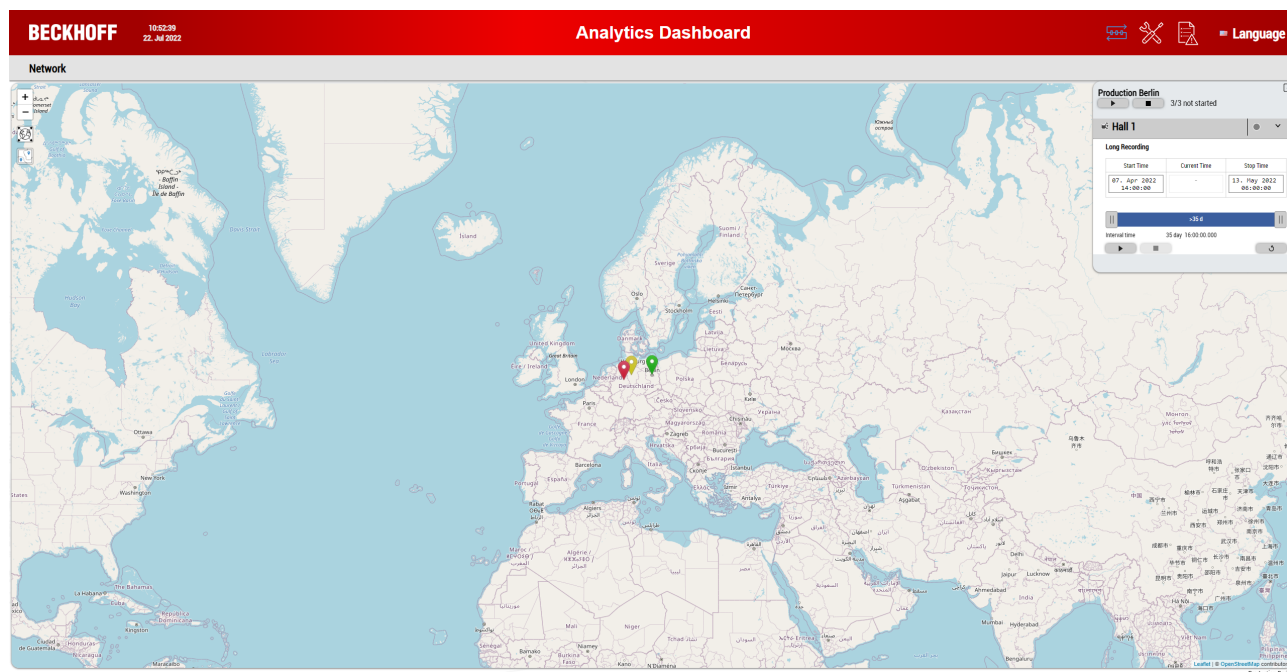
[illegible]

As soon as you click the button **Start**, the analysis is triggered with all input sources in the PLC. A loading symbol appears for the first time until the analysis starts. It turns green when the first data appears. Likewise, a marker shows the current time of the historical analysis in the slider. The process speed depends on the bandwidth, the number of data and the recorded cycle time. It is possible to analyze several historical recordings at the same time by simply switching machines. Internally, all analyses continue in parallel. This makes it easy to switch between analyses in the dashboard.



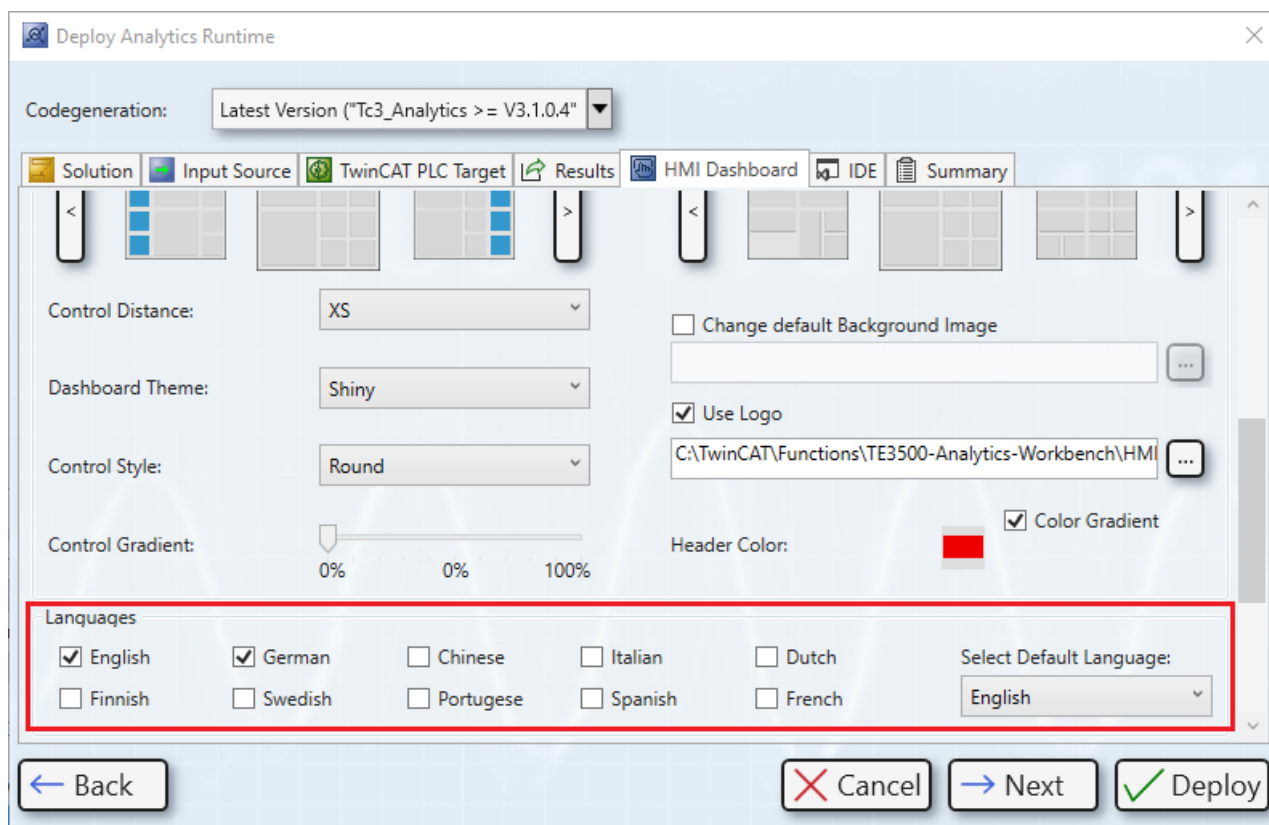
The button **Minimize** in the upper right corner hides the selection of configurations and records. This allows you to work with the records from the input sources and view the data at the same time. You can hide the window completely by clicking the blue icon at the top. Only the configuration name is always in the bottom right corner of the dashboard.





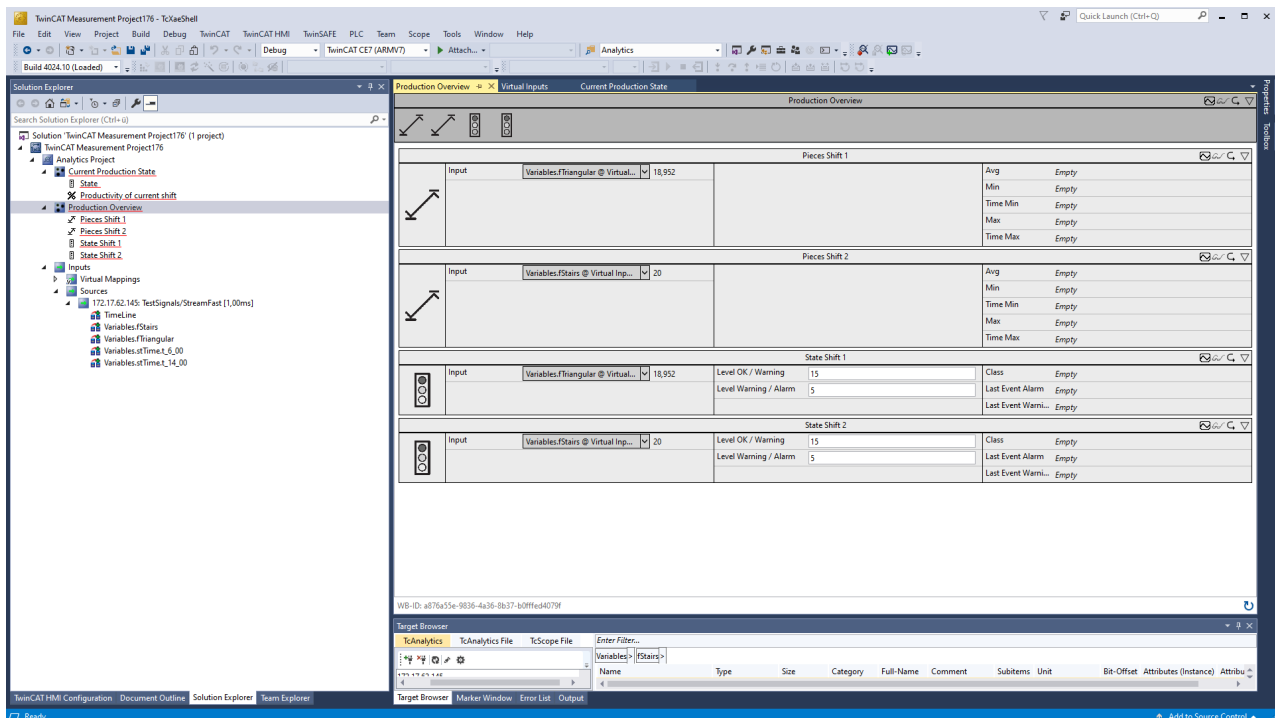
5.6.9 Integration of a language switch

In the Deploy wizard configuration window, up to 8 languages can be selected for language switching in the HMI area.

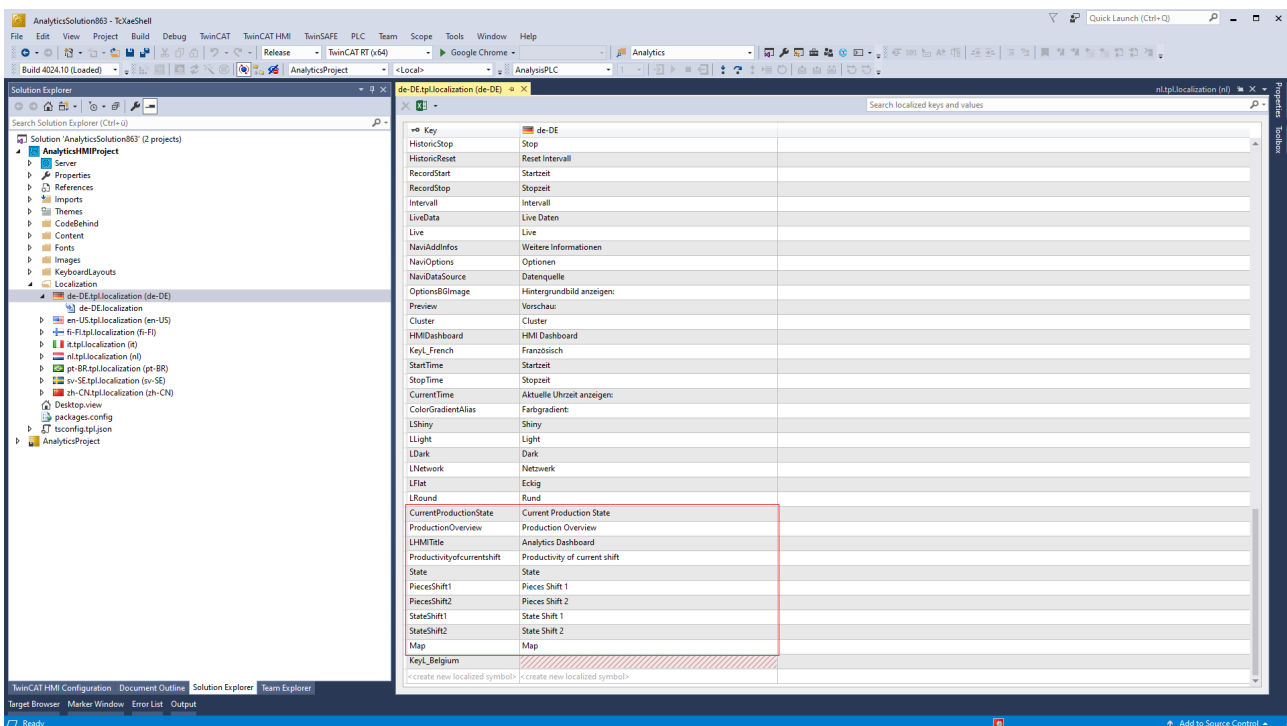


If you use the user administration, the languages of the created users are already selected. These cannot be deselected.

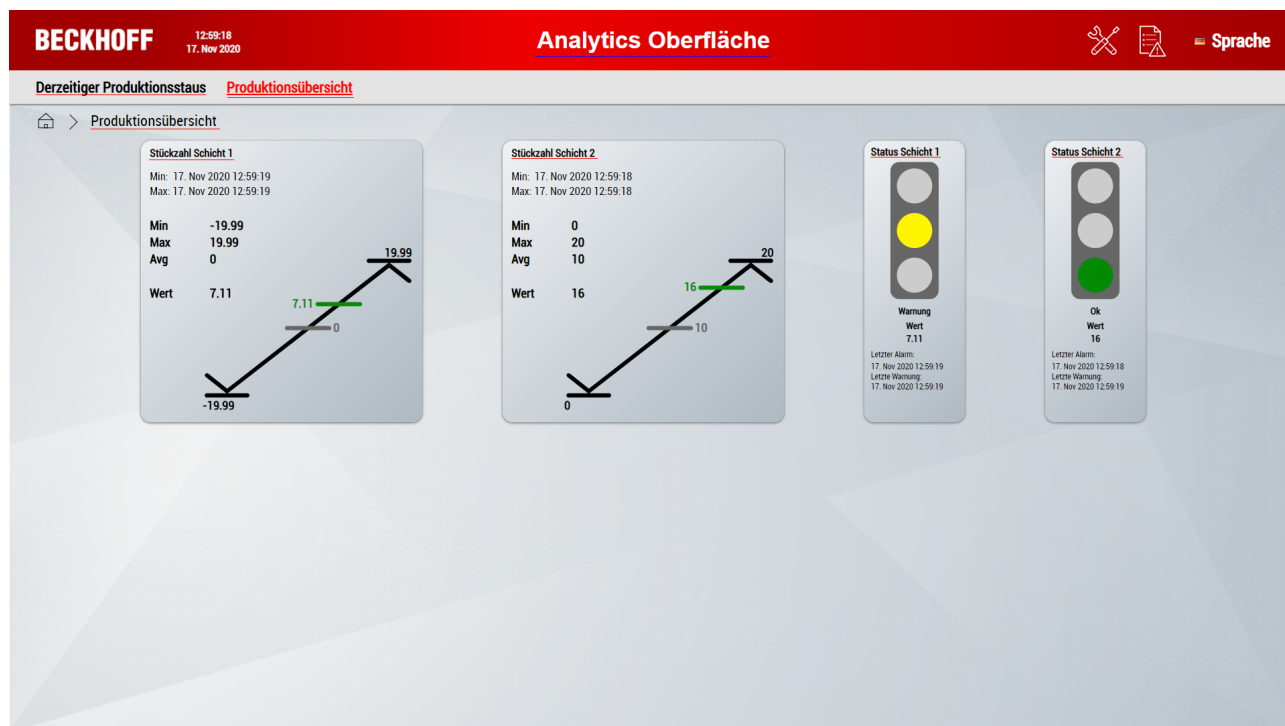
The texts in the supplied Analytics Controls and all other texts can be switched automatically. Only your network and module names have to be translated, if you want them to be included in language switching. In the following screenshot, the main affected names are marked in the **Solution Explorer**.



In the generated TwinCAT HMI Engineering project this is easily possible, since the translation entries are already prepared. The texts for the respective languages are stored in the **Localization** files. The names of your networks and modules are automatically entered there and only have to be translated. To do this, open all the files of the languages for which you need a translation. The following screenshot lists the entries that need to be translated for German. The number of entries to be processed varies depending on the complexity of the Analytics project.



Once completed, you can open the dashboard by clicking the **Google Chrome** button (or the name of your default browser). Simply reloading in the browser is not sufficient, because the project has to be rebuilt internally. In the following image the texts which are now also switched with the language switch are highlighted.



The language can be changed via the flag in the upper right corner.

The screenshot displays the 'Analytics Dashboard' with a red header bar. The header includes the Beckhoff logo, the time '14:54:55' and date '22. Feb 2023', the title 'Analytics Dashboard', and a language selection button with a flag icon. Below the header, the main content area shows 'Please select your language:' and a 'Close' button with an 'X' icon. The language selection interface lists the following options:

- German
- English (US)
- Chinese (CN)
- Italian
- Belgian
- Finnish
- Swedish
- Portuguese

Below the language selection interface, there is a map of Europe and Asia with a green location pin over Germany. The map is labeled with 'NORTH AMERICA', 'EUROPE', 'ASIA', 'AFRICA', 'SOUTH', 'Atlantic Ocean', and 'Indian Ocean'. The map is credited to 'Leaflet | © OpenStreetMap contributors © CARTO'. To the right of the map, there is a panel titled 'Beckhoff Automation'.



The language can be set and saved user-specifically from Dashboard version 2.0.

5.6.10 Using the interactive functions in standard TwinCAT HMI projects

Both the Interactive Dashboard and the Interactive Chart function can be integrated into any TwinCAT HMI project independently of the One Click Dashboard. This requires a valid TwinCAT Analytics Runtime or Runtime Base license (TF3550 or TF3551) on the target system.

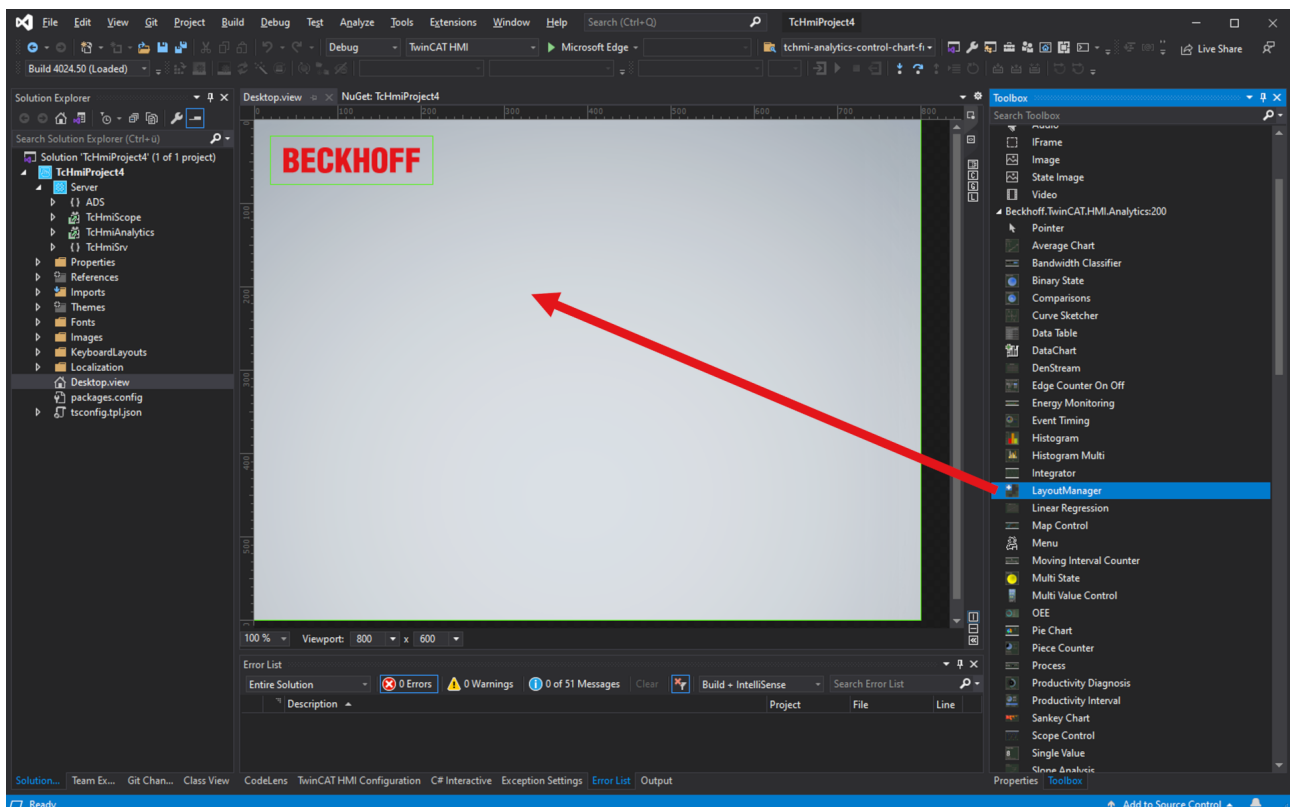
Installation of the NuGet packages

The required components (Analytics Framework Controls and the Analytics Server Extension) can be integrated into the HMI project via NuGet packages. To do this, the packages "Beckhoff.TwinCAT.HMI.Analytics.ControlsExtension" and "Beckhoff.TwinCAT.HMI.Analytics.Controls" must be installed in the NuGet Package Manager.

After installation, the "TcHmiAnalytics" server extension is located in the HMI project under the server node. If this does not start automatically, it must be started manually by right-clicking on the extension.

Adding the Layout Manager control

Dynamic and user-specific layouts can be created using the Layout Manager control. The Layout Manager control can be inserted into the HMI project via the toolbox using drag and drop. It is located in the "Beckhoff.TwinCAT.HMI.Analytics" area. The Layout Manager is similar to the Container Control. The difference is that the controls can be repositioned in the browser.



Setting the properties of the Layout Manager control

For the Layout Manager control to function optimally, the following properties should be set:

Name	Description
Identifier	Once layouts have been saved with the server extension, the identifier should no longer be changed. Otherwise, saved layouts can no longer be assigned.
WidthMode	Selection "Value" should not be changed.
HeightMode	Selection "Value" should not be changed.

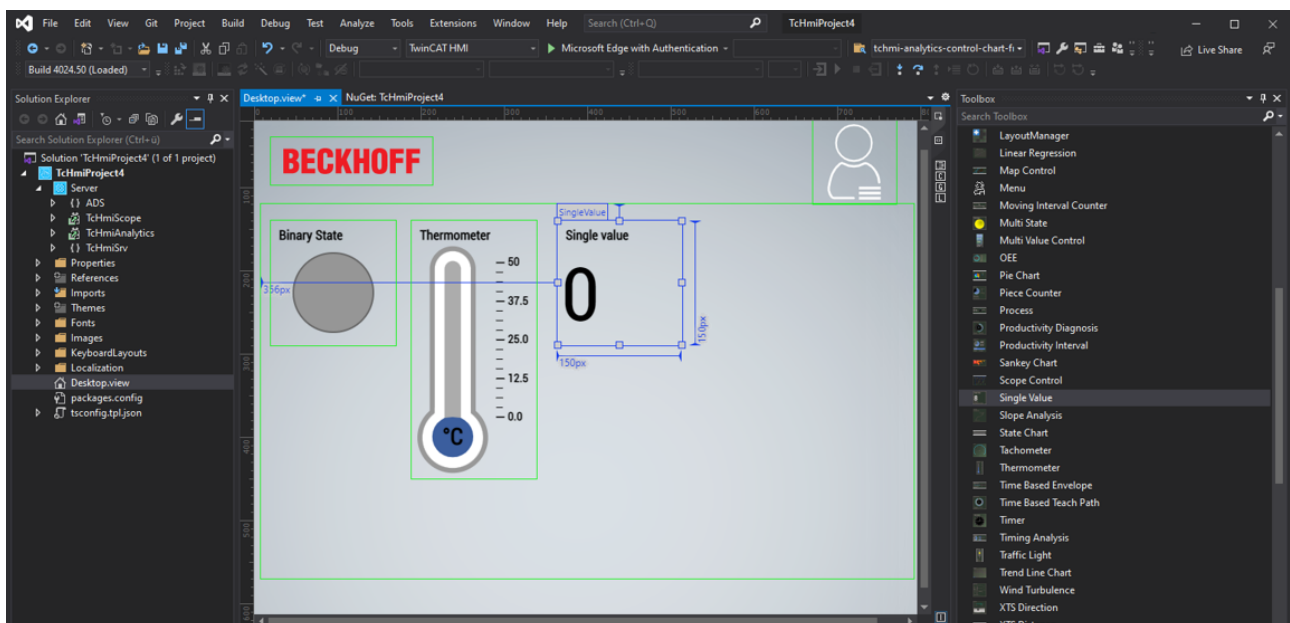
The following parameters can be set optionally:

Name	Description
DarkMode	Selects whether to display the control in DarkMode.
MobileViewWidth	Pixel width from which to switch to Mobile View.
Number of columns	Number of columns that the layout should include in the normal view.
Grid Gap	Percentage distance between the controls that are positioned on the Layout Manager.
UseCustomRowHeight	This parameter can be used to activate the fixed row height, which can be set using the RowHeight parameter. Otherwise, a dynamically scaled height is used.
RowHeight	This parameter can be used to specify the row height in pixels.
LockControls	This parameter can be used to prevent interaction with the existing controls in the layout in Edit mode.
IsAnalyticsUse	This property should only be activated if the Layout Manager control is used in an Analytics-One-Click-Dashboard. This positions the menu of the Layout Manager control appropriately on the layout.
MenuControlSymbol	A control of the "Menu" type (also from the "Beckhoff.TwinCAT.HMI.Analytics" area) can optionally be linked here. This should always be visible on the dashboard and placed on the "Desktop.view". The Edit mode can also be opened and closed via the Menu control.

Adding controls

Any controls can then be added to the Layout Manager control. These can be moved as required in HMI engineering.

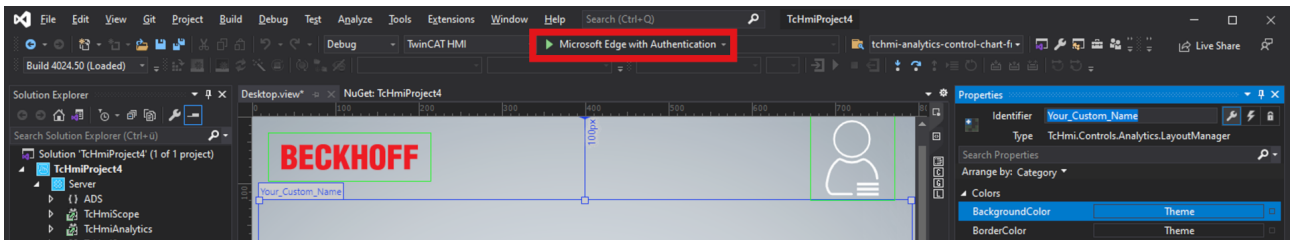
i If a layout has been saved for a user using the interactive function, the items are loaded from the saved layout. If further controls are added at a later date, they are sorted into a free space within the layout. Removed controls, on the other hand, disappear directly from the layout.



i Interactive chart controls can also be added in the web browser.

Building the project

To be able to use the functions of the Interactive Dashboard and Interactive Charts, the project must be built with user authentication.



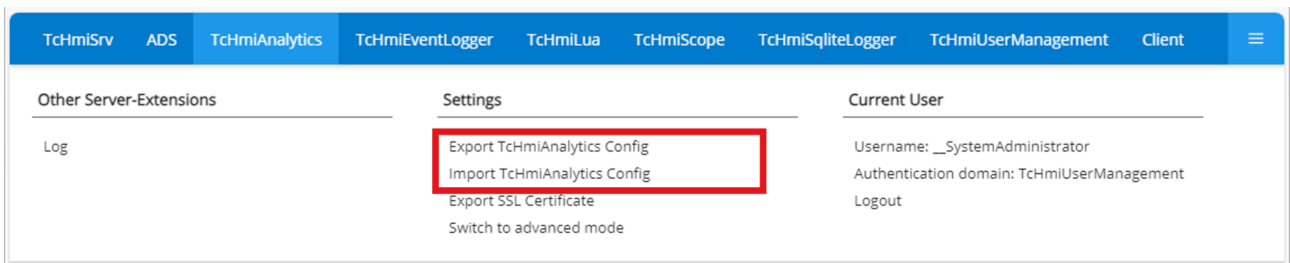
Making user-specific changes

The interactive functions can then be used in the web browser. Information on the use of the [Interactive Dashboard](#) [► 412] and the [Interactive Chart](#) [► 424] can be found on the following pages.

Exporting and importing layout and chart configurations

The configurations of layouts and charts saved in the "TcHmiAnalytics" server extension can be exported and imported. This is useful if the configurations are to be transferred from a remote project (e.g. running on an HMI server) to a local project. Alternatively, the configuration of a remote project can also be updated.

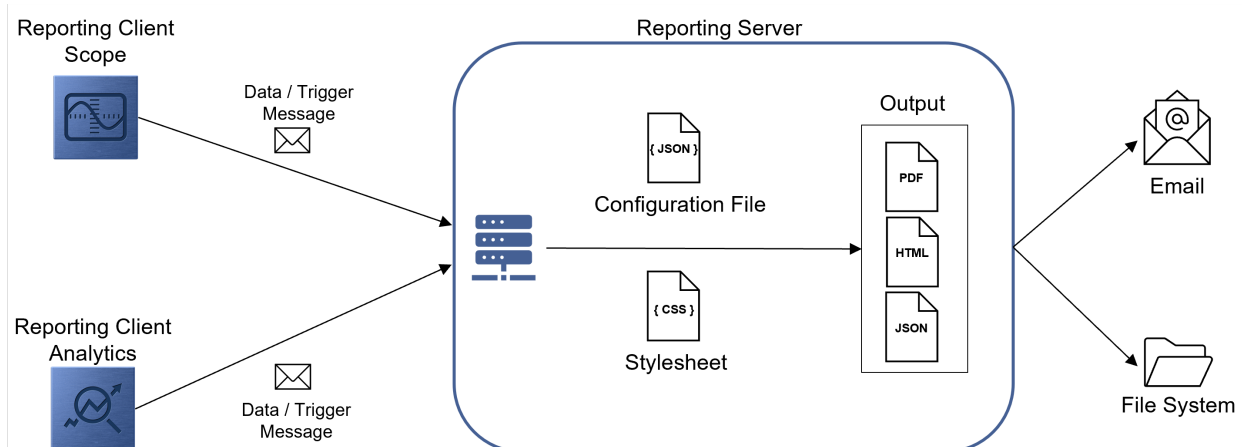
The configurations can be exported and imported into the server extension via the Config page of the respective HMI project.



5.7 Reporting Integration

5.7.1 Basic concept

The following figure shows the basic concept of TwinCAT Reporting from the Reporting Clients to the distribution of the generated Reports.



Reporting Clients

Currently, a Reporting Client is integrated in the TwinCAT Scope View as well as in the TwinCAT Analytics Workbench or in the TwinCAT Analytics Service Tool. These tools provide the ability to implement 24/7 reporting and on-demand reporting.

- Scope Reporting

- [Analytics Reporting \[► 460\]](#)

24/7 reporting is also possible via the PLC. The function blocks have the same functionality as the algorithms from TwinCAT Analytics.

- PLC Reporting

The Reporting Clients communicate with the TwinCAT Reporting Server. The Reporting Clients can send two types of messages. On the one hand, data messages can be sent with the associated data. On the other hand, so-called trigger messages can be communicated. This type of message triggers the generation of a report.

TwinCAT Reporting Server

The [TwinCAT Reporting Server \[► 468\]](#) has a variety of tasks. The server receives the messages from the Reporting Clients and stores the data in a DataStore. A report can be defined in an associated configuration file. The assignment of a data object to a report is established via a Report Name and a Data Key. The design of the report can be customized in a stylesheet. As soon as a trigger message is received, a report with the associated data is generated in the configured formats and distributed accordingly.

Glossary

Term	Explanation
Report	A report is a document available in PDF, HTML and JSON formats. It contains the information and images.
Configuration file	The configuration file describes the structure of a report. It also offers the possibility to integrate static information into a report by default. Furthermore, settings can be made in the Reporting Server for the corresponding report.
Reporting Server	The Reporting Server manages the configuration files, receives information and creates individual reports from this information.
Reporting Client	A Reporting Client can be used to send data to the Reporting Server or to trigger the creation of a report.
Data message	A Reporting Client can send information to the Reporting Server via a data message. The contained data is cached in the Reporting Server.
Trigger message	A Reporting Client can send a trigger message to the Reporting Server, triggering the generation of a report. For this purpose, the Reporting Server uses all data received up to this point and belonging to the corresponding report. After generation, the data is deleted.
Report Name	The Report Name is used to identify a report. The Report Name is derived from the name of the configuration file without the extension (e.g.: configuration file: Beckhoff Report Template.json; Report Name: Beckhoff Report Template)
Data Key	A Data Key identifies a data object of a report. It must be unique within a report and its configuration file. This can ensure the assignment and sorting of data objects in the report.
24/7 Reporting	24/7 Reporting provides the ability to collect data and automatically generate reports.
On-demand reporting	On-demand reporting provides the ability to generate custom reports on demand.

5.7.2 Analytics Reporting

5.7.2.1 24/7 Reporting

24/7 reporting can be implemented in TwinCAT Analytics using the algorithms in the *Reporting* category. The reporting collectors collect the data and send it to the reporting server. The reporting triggers trigger the creation of a report.

5.7.2.1.1 Reporting Collector

The Reporting Collectors collect data and send it to the Reporting Server in a data message after an event.

5.7.2.1.1.1 Reporting Collector Edge

Reporting Collector - Edge			
Edge	<Empty>	Threshold Edge	1
Channel 00	<Empty>	Report Name	Beckhoff Report Template
		Data Key	DataKey101
		Buffer Size	1
		Buffer on event	☒
		Num Channels	+ -
		Alias Name 00	Channel 1
		New Result	Empty
		Last Message	Empty
		Buffer Count	Empty

The Reporting Collector Edge collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the signal of the input channel passes the configured edge at a certain threshold.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.
- **Data Key:** the data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
- **Buffer Size:** specifies the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
- **Buffer on event:** specifies how the data is to be collected and buffered.
If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer.
If the parameter is False, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the TcReportingServer.
- **Num Channels:** the number of input channels from which the data will be collected. And the number of parameters to allow a description of the input channels.
- **Alias Name:** serves as a description of the input channel. BufferSize = 1: the alias name serves as a key in the key-value structure. BufferSize > 1: the alias name serves as the heading of the table column.
- **Include Timestamps (optional):** inserts a column with the timestamps.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Buffer Count:** specifies the number of elements in the buffer.

5.7.2.1.1.2 Reporting Collector Interval

Reporting Collector - Interval			
Timestamp	-	Interval	Seconds 1
Channel 00	<Empty>	Report Name	Beckhoff Report Template
		Data Key	DataKey101
		Buffer Size	1
		Buffer on event	☒
		Num Channels	+ -
		Alias Name 00	Channel 1
		New Result	Empty
		Last Message	Empty
		Current Interv...	Empty
		Buffer Count	Empty

The Reporting Collector Interval collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the timespan of the configured interval has expired.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Interval:** time interval at which the data should be sent to the Reporting Server.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.
- **Data Key:** the data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
- **Buffer Size:** specifies the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
- **Buffer on event:** specifies how the data is to be collected and buffered.
If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer.
If the parameter is False, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the TcReportingServer.
- **Num Channels:** the number of input channels from which the data will be collected. And the number of parameters to allow a description of the input channels.
- **Alias Name:** serves as a description of the input channel. BufferSize = 1: the alias name serves as a key in the key-value structure. BufferSize > 1: the alias name serves as the heading of the table column.
- **Include Timestamps (optional):** inserts a column with the timestamps.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Current Interval Time:** indicates the amount of time that has already elapsed from the current interval.
- **Buffer Count:** specifies the number of elements in the buffer.

5.7.2.1.1.3 Reporting Collector Time

Reporting Collector - Time		New Result Empty	
Timestamp -	Time On hh 1 mm 0 ss 0	Last Message Empty	
Channel 00 <Empty>	Day Of Week Mask ☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun	Next Trigger Empty	
	Report Name Beckhoff Report Template	Buffer Count Empty	
	Data Key DataKey101		
	Buffer Size 1		
	Buffer on event ☒		
	Num Channels + -		
	Alias Name 00 Channel 1		

The Reporting Collector Time collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the configured switch-on time is reached. The switch-on time and the days of the weeks can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Time On:** switch-on time.
- **Day of Week Mask:** weekdays on which the timer should be active.

- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.
- **Data Key:** the data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
- **Buffer Size:** specifies the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
- **Buffer on event:** specifies how the data is to be collected and buffered.
If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer.
If the parameter is False, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the TcReportingServer.
- **Num Channels:** the number of input channels from which the data will be collected. And the number of parameters to allow a description of the input channels.
- **Alias Name:** serves as a description of the input channel. BufferSize = 1: the alias name serves as a key in the key-value structure. BufferSize > 1: the alias name serves as the heading of the table column.
- **Include Timestamps (optional):** inserts a column with the timestamps.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Next Message:** indicates the remaining time until the next message.
- **Buffer Count:** specifies the number of elements in the buffer.

5.7.2.1.2 Reporting Trigger

The Reporting Triggers send a trigger message to the Reporting Server after an event and thus trigger the creation of a report.

5.7.2.1.2.1 Reporting Trigger Edge

Reporting Trigger - Edge			
Timestamp	-	Num Channels	+ -
Channel 00	<Empty>	Threshold Edge	1
Report Name		Beckhoff Report Template	
Last Trigger		Empty	
Edge Overview		Empty	

The Reporting Trigger Edge triggers the creation of a report after an event is triggered. An event is triggered when the input channel signal exceeds the configured edge at a specified threshold. Internally, the inputs that were once True remain True. The inputs are only reset to False as soon as all inputs were True at least once. This allows the output bNewResult to be used as one input by multiple Reporting Collectors and once all Reporting Collectors have sent a data message, a trigger message is sent.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Type of the edge:** Specifies whether the edge counter should respond to a rising or falling edge.
- **Threshold:** Threshold of the signal at the respective edge. The event is triggered when the signal passes this threshold.
- **Tolerance (optional):** Tolerance value for the Equal / NotEqual comparisons.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.

Output values

- **Last Trigger:** indicates when the last message was sent to the Reporting Server.
- **Edge Overview:** indicates which input channels were True at least once.

5.7.2.1.2.2 Reporting Trigger Interval

Reporting Trigger - Interval			
	Timestamp	-	
	Interval	Seconds	1
	Report Name	Beckhoff Report Template	
	Last Trigger	Empty	
	Current Interv...	Empty	

The Reporting Trigger Interval triggers the creation of a report after an event has been triggered. An event is triggered when the timespan of the configured interval has expired.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Interval:** time interval at which the data should be sent to the Reporting Server.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.

Output values

- **Last Trigger:** indicates when the last message was sent to the Reporting Server.
- **Current Interval Time:** indicates the amount of time that has already elapsed from the current interval.

5.7.2.1.2.3 Reporting Trigger Time

Reporting Trigger - Time			
	Timestamp	-	
	Time On	hh 1 mm 0 ss 0	
	Day Of Week Mask	☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat ☒ Sun	
	Report Name	Beckhoff Report Template	
	Last Trigger	Empty	
	Next Trigger	Empty	

The Reporting Trigger Time triggers the creation of a report after an event has been triggered. An event is triggered when the configured switch-on time is reached. The switch-on time and the days of the weeks can be configured.

Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

Configuration options

- **Time On:** switch-on time.
- **Day of Week Mask:** weekdays on which the timer should be active.
- **Report Name:** specifies the name of the report. The name must correspond to the name of the configuration file at the Reporting Server. The drop-down menu displays all report names that are located in the Configuration folder of the Reporting Server. The Edit button can be used to edit the selected configuration file.

Output values

- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Next Trigger:** indicates the remaining time until the next message.

5.7.2.2 On-demand reporting

On-demand reporting provides the ability to manually integrate algorithm data and additional data into a report.

Each algorithm has the same five icons in the upper right corner: The second icon from the left is called **Include to Report** icon. This icon allows you to add the algorithm to a manual report.

This icon symbolizes that the algorithm is added to a manual report:



This icon symbolizes that the algorithm is not included in a manual report:



A network also contains these icons. Via these, all containing networks and algorithms are added to or ignored from the manual report.

The functionality of the **Include to Report** icon can also be mapped via the properties.

For a network, the properties look like this:

Reporting	
Data Key	MyNetwork
Include in Report	False
Include SubMember in Report	True
Reporting visualization	Reporting Table Default

Closed networks are treated like algorithms in on-demand reporting.

Reporting

Data Key	The Data Key uniquely identifies the data object in the report. The Data Key is preset with the network name by default.
Include in Report	This option specifies whether the network is added to a manual report. The network inputs and outputs or the network parameters are used as contents to be displayed.
Include SubMember in Report	This option specifies whether the Analytics elements contained in the network should be integrated into the report.
Reporting visualization	This option specifies how the data of the network should be presented.

For an algorithm, the properties look like this:

Reporting	
Data Key	MyAlgorithm
Include in Report	True
Reporting visualization	Reporting Table Horizontal

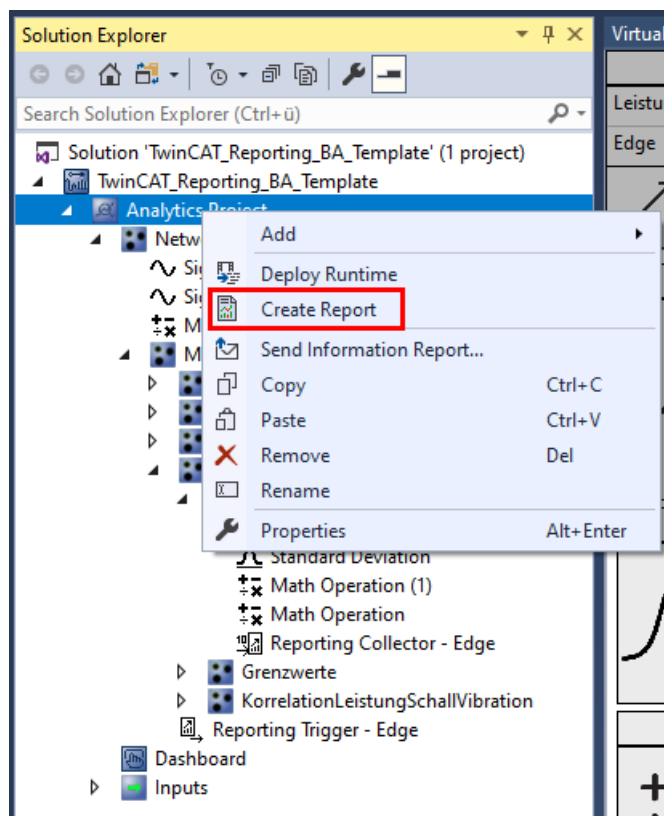
Reporting

Data Key	The Data Key uniquely identifies the data object in the report. The Data Key is preset with the network name and the name of the algorithm by default.
Include in Report	This option specifies whether the algorithm is added to a manual report.
Reporting visualization	This option specifies how the algorithm data should be presented.

On-demand reporting can be started in two ways. On the one hand, the reporting can be opened via the Reporting button in the toolbar:

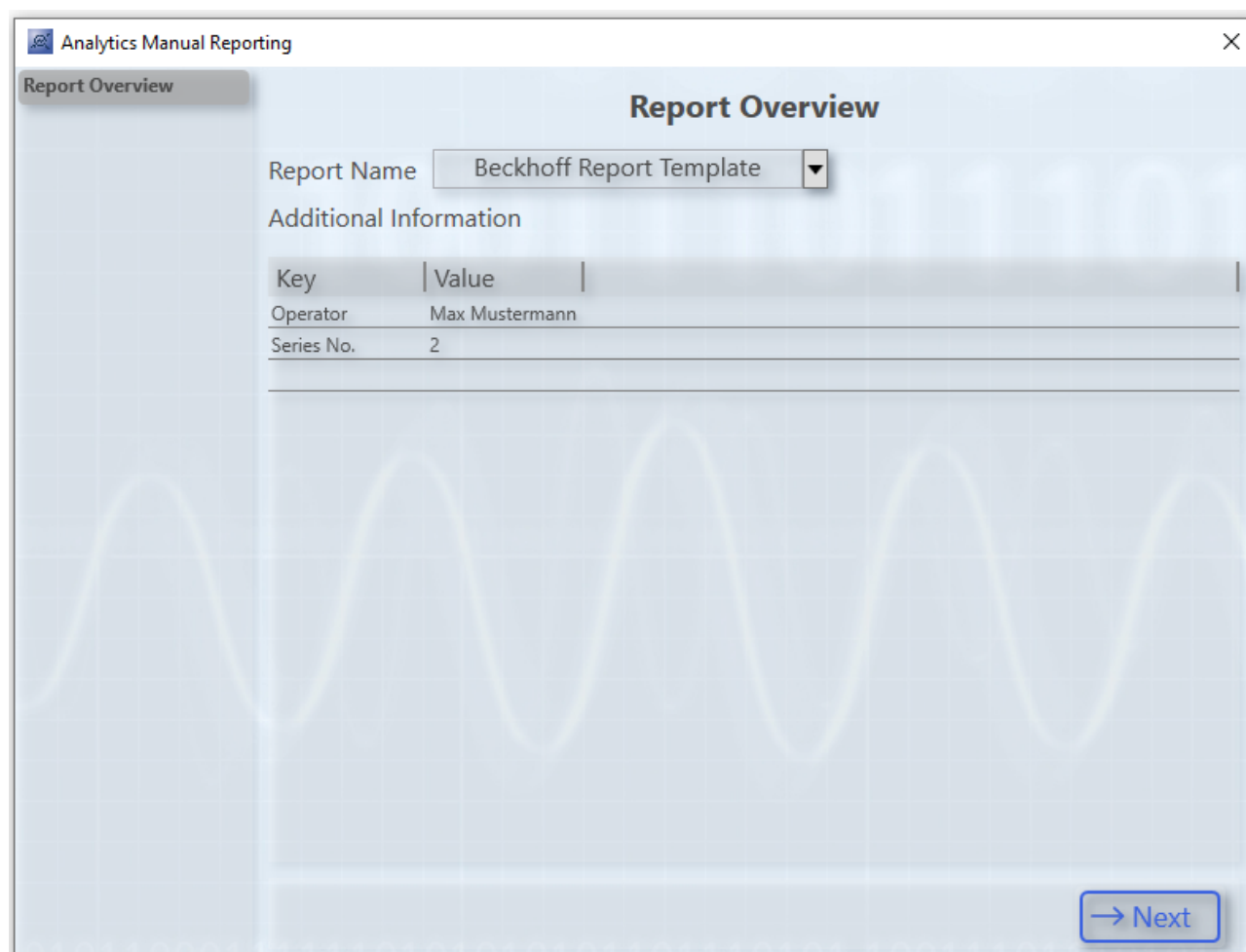


On the other hand, it can be started via the context menu on the Analytics project:



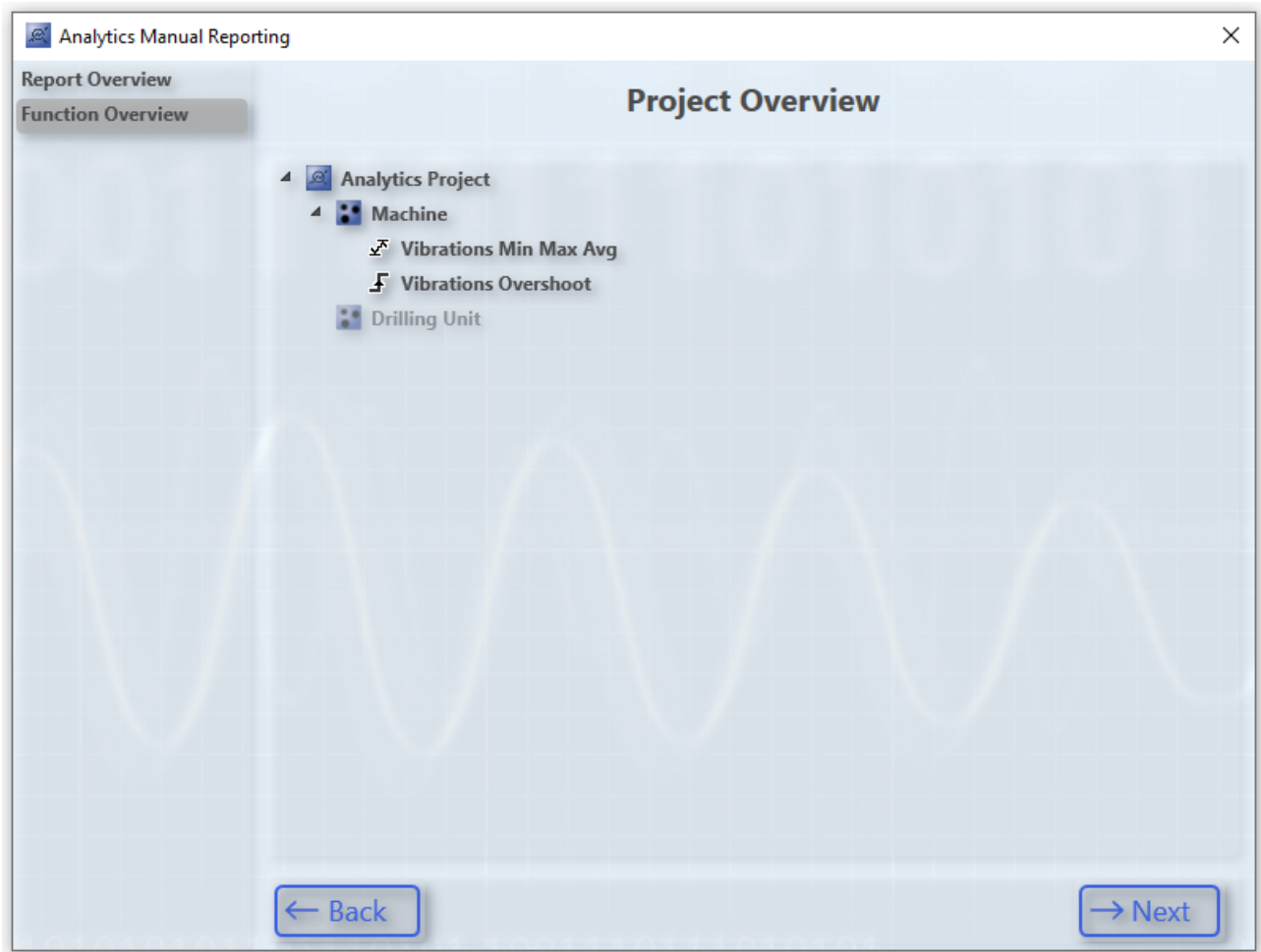
The wizard for on-demand reporting then opens.

On the first page, additional information can be added to the report.



The additional information is built in a key-value structure. Another line can be added by clicking on the empty line. A selected line can be deleted by pressing the Delete key.

An overview of the project is presented on the second page. The grayed out algorithms and networks are not included in the report.



On the third page, information about the analysis can be added to the report.

Include	Description	Value
☒	Analysis Start Time	15.06.2022 14:49:58
☒	Analysis End Time	15.06.2022 14:56:53
☒	Time In Run	00:06:54.5185000
☒	Processed Data	'17,06 MB' received (696,19 KB/s)
☐	Function Count	2
☐	Network Count	2
☐	Channel Count	2

☒ Generate Report
☒ Open Report

[← Back](#)
[✓ Create](#)

In the table, a lot of information is available for integration in a report. The checkbox in the Include column can be used to select or deselect individual pieces of information.

The Generate Report option defines whether the report should be generated with the configured data. If this option is not selected, only the data will be sent to the Reporting Server, but the report will not be generated yet. This option is useful when manually adding data from multiple products to a single report.

The Open Report option defines whether the report should be opened after generation. To use this option, a FilePublishLocation with PDF or HTML formats must be defined in the configuration file. If the report is generated in PDF and HTML format, the PDF report will be opened by default.

5.7.3 TwinCAT Reporting Server

5.7.3.1 Design

5.7.3.1.1 Configuration file

The configuration file offers the possibility to organize, sort and customize the report. The configuration files must be stored in the folder `.\TwinCAT\Functions\Reporting-Server\Configuration\ConfigFiles`.

The following options are supported in the Reporting Server.

Key	Description	Sample	Level
General			
Store Type	Specifies which store mode to use. Currently only "Ram" is supported.	"StoreType": "Ram"	1
Header	Provides the possibility to display three headers arranged one below the other. If the second and / or third header is not to be used, an empty string can be specified.	"Header": ["Header 1", "Header 2", "Header 3",]	1
Footer	The footer of a PDF page is divided into three areas. All three areas can be individualized with a free text. To display the date and time of creation, the placeholder "{date} {time}" must be specified. To display the page number, the placeholder "{page} of {total-pages}" must be specified. If a range is to be left blank, an empty string can be specified.	"Footer": ["Footer Left", "Footer Center", "Footer Right",]	1
Logo	The logo is displayed on the top right of the first page. The logo can be specified as a relative path to the folder <i>Reporting-Server\Configuration\ConfigFiles</i> . Alternatively, the logo can be specified as a Base64String with the preceding encoding.	"Logo": ".//<Bildname.png>" oder "Logo": "data:image/ <Encoding>;base64, <Base64String>"	1
StyleSheet	Offers the possibility to customize the design of a report. To use a user-specific style sheet, its name must be specified (see Style Sheet [► 472]).	"StyleSheet": "StyleBeckhoff.css"	1
SignatureSettingsFile	Provides the possibility to sign a report (see Signing [► 474]).	"SignatureSettingsFile": "BeckhoffSignatures.json",	
Email text	This text will be included in every email unless a specific text is specified.	"EmailText": "Custom of E- Mails send to all recipients with no or empty E-Mail-Text. Special. Ones"	1
Required Keys	Is a collection from Data Key. If the data objects of these data keys are not available in the Reporting Server, the report will not be generated.	"RequiredKeys": ["Data101"]	1
Trigger On Required Key Received	Is another way to trigger the generation of a report. If true is selected for the option and all data objects with the required Data Keys are present, the report will be generated. This eliminates the need for a trigger message from a Reporting Client.	"TriggerOnRequiredKeyReceived": false	1
Static Default Data			
Default Data	This object can be used to define data objects to be presented in each report.	"DefaultData": {}	1
Text Fields			
Text Fields	This element is an array and can have multiple text field objects with the following elements.	"TextFields": []	2
Key	Used as a heading for the text field.	"Key": "My Text Field"	3
Comment	Serves as a comment for the data object. The comment is not displayed in the report.	"Comment": "My Comment"	3
Sort Priority	Specifies the position of the data object in the report.	"SortPriority": 42	3
Value	Contains an individual value and must be specified as STRING.	"Value": "253-5553-9421"	3
Tables			
Tables	This element is an array and can have multiple table objects with the following elements.	"Tables": []	2

Key	Description	Sample	Level
Key	Used as a heading for the table.	"Key": "My Table"	3
Comment	Serves as a comment for the data object. The comment is not displayed in the report.	"Comment": "My Comment"	3
Sort Priority	Specifies the position of the data object in the report.	"SortPriority": 42	3
Value	This element is an object and contains the following elements.	"Value": {}	3
Header	This element is an array and receives the column titles of the table. The number of column titles must be equal to the columns of the table.	"Header": ["Table Header 1", "Table Header 2"]	4
DataTypes	This element is an array and receives the data types of the table column. The following data types are possible: BOOL, BYTE, SBYTE, CHAR, DECIMAL, DOUBLE, FLOAT, INT, UINT, LONG, ULONG, SHORT, USHORT, STRING. The number of data types must be equal to the columns of the table.	"DataTypes": ["string", "double"]	4
Values	This element is a 2D array and receives the value of the table. An array in the array represents a table column. The table column must match the specified data type. The number of arrays must be equal to the columns of the table. The arrays should be the same size.	"Values": [[], []]	4
Images	This element is an array and can have multiple image objects with the following elements.	"Images": []	2
Key	Used as a caption for the image.	"Key": "My Image"	3
Comment	Serves as a comment for the data object. The comment is not displayed in the report.	"Comment": "My Comment"	3
Sort Priority	Specifies the position of the data object in the report.	"SortPriority": 42	3
Value	Contains the Base64String with PNG encoding.	"Value": "<Base64String PNG-Encoding>"	3
Value Pairs	This element is an array and can have multiple key-value pair objects with the following elements.	"ValuePairs": []	2
Key	Used as a heading for the collection from key-value pairs.	"Key": "My Value Pair"	3
Comment	Serves as a comment for the data object. The comment is not displayed in the report.	"Comment": "My Comment"	3
Sort Priority	Specifies the position of the data object in the report.	"SortPriority": 42	3
Value Pair Collection	Serves as a collection of key-value pairs. This element is an array and can have multiple objects with the following elements.	"ValuePair_Collection": []	3
Key	Contains an individual value and is displayed presented on the left side.	"Key": "Serial number"	4
Data Type	Describes the data type of the "Value" element.	"DataType": "string"	4
Value	Contains an individual value and is presented on the right side. The value must be specified as STRING.	"Value": "253-5553-9421"	4
Data Configurations			
Data Configurations	Data Configurations can be used to sort and individualize the data that is sent dynamically from the Reporting Clients. This element is an array and can have multiple objects with the following elements.	"DataConfigurations": []	1

Key	Description	Sample	Level
Key	Corresponds to the Data Key of the data object. This ensures that this data configuration is applied to a unique data object. (Required)	"Key": "DataTable101"	2
Title	Provides the option of a better name as a heading. (optional)	"Title": "Rotations and vibrations"	2
SortPriority	Specifies the position of the data object in the report. (Required)	"SortPriority": 42	2
IsRequired	Indicates whether this data object is required for generating the report. (see: TriggerOnRequiredKey)	"IsRequired": false	2
StoreMode	Provides the option to store individual data objects in a specific way and display them accordingly. Currently, four different store modes are supported: • Blocking: The first data object is saved and all subsequent data objects with the same data key are discarded. • Overwriting: The newly received data object replaces the last data object with the same data key. • Appending: The newly received data object is saved. The previously received data objects are also held and presented in the report. In the case of a table, the data is merged. • Appending_Series: This store mode corresponds to the store mode "Appending". However, the data objects are numbered and presented in individual tables.	"StoreMode": "Overwriting"	2
RoundTo	Provides the option to round floating-point numbers to a specified decimal place.	"SortPriority": 4	2
TimestampFormat	Provides the option to convert Timestamp to a specified format. By default, the following formatting is selected: "yyyy-MM-dd HH:mm:ss.ffff" (See also: Custom date and time format strings)	"TimestampFormat": "HH:mm:ss.ffff"	2
IsAsc	Provides the option to specify the sort order of a data table. If the value is true, the first data set is presented in the first line.	"IsAsc": true	2
StyleClass	Provides the option to use a custom CSS style class. The CSS style class must be defined in the CSS file and the CSS file must be referenced under the StyleSheet option.	"StyleClass": "CustomClass"	2
Publish Locations			
Publish Locations		"PublishLocations": []	1
Email Publish Location	Used to forward the report as an e-mail. This element is an array and can have multiple objects with the following elements.	"EmailPublishLocations": []	2
Type	Specifies the type of distribution.	"Type": "File"	3
Format	Offers the option to send only a selection of formats by email.	"Format": ["PDF", "HTML", "JSON"]	3
Address	Provides the ability to send the report to multiple email addresses.	"Address": ["m.mustermann@tester.com "]	3
Use Bcc	Provides the option to set the email addresses in BCC.	"UseBcc": false	3

Key	Description	Sample	Level
Text	Provides the option to specify a specific email text. This text replaces the default email text.	"Text": "Hello, \n insert email text here.\n Your Reporting Service."	3
File Publish Location	Used to copy the report to a specified location. This element is an array and can have multiple objects with the following elements.	"FilePublishLocations": []	2
Type	Specifies the type of distribution.	"Type": "File"	3
Format	Offers the option, to send only a selection of formats by email.	"Format": ["PDF", "HTML", "JSON"]	3
Path	Provides the possibility to specify a destination path for the report. The destination path must not require administrator rights.	"Path": "C:\\tmp\\CpyFolder"	3

5.7.3.1.2 StyleSheet

The style sheets offer the possibility to customize the design of the report. The *StyleBeckhoff.css* style sheet serves as a template. This can be copied, renamed and customized. The name of the style sheet must then be specified in the corresponding configuration file. The style sheets must be stored in the folder `\\TwinCAT\\Functions\\Reporting-Server\\Configuration\\ReportStyleSheets`.

Basically, all HTML objects contained in an HTML report can be individualized in the custom style sheet. The following CSS objects are frequently used and are ideal for customization:

CSS element	Description
html	The HTML element refers to the entire report.
h1	Main heading of the header in the report
h3	Subheading of the header in the report; heading of a data object

The following CSS classes are used to design data objects:

CSS class	Description
reporting-header	Class for the header of the report.
reporting-header img	Class for the icon on the top right of the header.
reporting-header table	The header is tabular and can be customized via this class.
reporting-body	Class for the content of the report.
reporting-body caption	The headings of the data objects are located in a Caption element.
reporting-body valuetable	Class for a data table in the report.
reporting-body valuepair	Class for a collection from key-value pairs in the report.
reporting-body infotable	Class for the additional information and analysis or recording information from the manual report.
reporting-body figure	Class for images, such as charts from the scope in the report.
reporting-body img	This class contains the actual image. It is located in the figure element.

5.7.3.2 Report format

The reports are generated in the following formats.

1. PDF

2. HTML
3. JSON

5.7.3.3 Network Configuration

A possible use case could be that the Reporting Server and the Reporting Client are running on different systems. The Reporting Server and the Reporting Client must be on the same network. To be able to map this use case, a JSON file must be extended. The file should be located under the following path and have the appropriate name: `.\TwinCAT\Functions\Reporting-Server\Configuration\Network\ReportingNetworkConfiguration.json`

A network configuration looks like this:

Name	Description	Data type
ConfigIds	Report names of the corresponding reports	List<string>
AmsNetId	AmsNetId of the target system. The Reporting Server is executed on the target system.	string

5.7.3.4 Email-Client configuration

A TwinCAT Reporting Server can be connected to an SMTP server. The SMTP server must be set up by the user and is not part of TwinCAT Reporting. The TwinCAT Reporting Server sends the configured email with the report to the SMTP server. This forwards the email to the recipient.

A wizard is available for configuring the connection to an SMTP server. The wizard can be opened in two ways.

1. **TwinCAT > Analytics > Reporting Email**
2. **Scope > Reporting Email**

1. In the first dialog you can configure a connection to the SMTP server.

Reporting SMTP-Server Configuration

SMTP Server

SMTP Server

Connection

SMTP Server

Port

☒ Use Authentication

Username

Password

User

Sender Name

Sender Email

→ Next

- The second dialog offers the possibility to configure a connection to an HTTP proxy. This option can be enabled if the user's architecture requires it.

⇒ After clicking the **Create** button, the configuration will be saved in a corresponding file.

5.7.3.5 Signing

In order to verify reports, PDF reports can be signed in TwinCAT Reporting. Digital and handwritten signatures are supported. The wizard for configuring signatures can be opened in the following ways.

- **TwinCAT > Analytics > Reporting > Reporting Signatures**
- **Scope > Reporting > Reporting Signatures**

✓ The wizard offers the possibility to create multiple signature configurations, which can then be reused in multiple report configurations.

- The name of this configuration must be referenced in the configuration file via the key "SignatureSettingsFile" (see [Configuration file](#) [► 468]).
You can use the two checkboxes to select whether the configured signatures displayed on the next

pages should be used.
By clicking the button **Next** you can edit the selected configuration.

Reporting Signature Configuration

Configuration Selection

+ Add

- Remove

MySigningCfg

Signature Configuration

Name

MySigningCfg

☒ Use Digital Signature

☒ Use Handwritten Signature

→ Next

2. Here you can define a digital signature.

The screenshot shows the 'Reporting Signature Configuration' window with the 'Digital Signature Editor' tab selected. The window has a title bar with a close button (X). On the left, there is a 'Configuration Selection' sidebar with 'Digital Signature Editor' highlighted. The main area contains a list of digital signatures with '+ Add' and '- Remove' buttons. Below the list, there are input fields for 'Certificate', 'Password', 'Name', 'Contact', 'Location', and 'Reason'. The 'Certificate' field has a browse button (...). At the bottom, there are 'Back' and 'Next' buttons.

Configuration Selection	Digital Signature Editor
Digital Signature Editor	

+ Add - Remove

DigitalSignature1

Certificate: .\..\Configuration\Signatures\testCertificate.p12

Password: 123456

Name: DigitalSignature1

Contact: tester@beckhoff.com

Location: Verl

Reason: Test Reason

← Back → Next

⇒ Pfx and p12 certificates are supported by TwinCAT Reporting.

3. Here you can define a handwritten signature.

The image shows a 'Reporting Signature Configuration' window titled 'Handwritten Signature Editor'. On the left, there is a 'Configuration Selection' sidebar with three options: 'Digital Signature Editor', 'Handwritten Signature Editor' (which is selected), and 'Handwritten Signature Editor'. The main area contains two buttons: '+ Add' and '- Remove'. Below these are two slots for 'Handwritten Signature 0' and 'Handwritten Signature 1'. The 'Handwritten Signature 0' slot is active, showing a preview of a handwritten signature. Below the preview, there are input fields for 'Signature' (containing a file path), 'Name' (containing 'Handwritten Signature 0'), 'Owner' (containing 'User1'), 'Location' (containing 'Verl'), and 'Date Time Format' (containing 'yyyy-MM-dd HH:mm:ss'). At the bottom, there are four alignment and size controls: 'Left Align' (set to 25), 'Bottom Align' (set to 50), 'Height' (set to 100), and 'Width' (set to 100). At the very bottom, there are 'Back' and 'Create' buttons.

⇒ If you use handwritten signatures, an additional signature page is inserted into the report.

4. The signatures can be positioned via the left and bottom alignment. The origin is at the bottom left of the signature page.

The following image shows an exemplary handwritten signature:

Verl, 2022-12-19 09:54:51

Test Signing

Owner

Gittersloh, 09:54:51

Test Signing

User

⇒ The signature configuration information is stored in the Reporting Server folder in plain text in a file.

5.7.3.6 Licensing

TwinCAT Reporting can be licensed as a full version and as a 7-day trial version. The following products include the TwinCAT Reporting license.

Product number	Product name
TE1300	TwinCAT 3 Scope View Professional
TF3300	TwinCAT 3 Scope Server
TE3500	TwinCAT 3 Analytics Workbench
TE3520	TwinCAT 3 Analytics Service Tool

The following features are available with the full version or with the 7-day trial version in TwinCAT Reporting Server.

Features	TwinCAT Reporting Server	
	7-day trial	Full license
General:		
Reports	✓ (Max. 1)	✓ (unlimited)
Digital signatures	✗	✓
Handwritten signatures	✗	✓

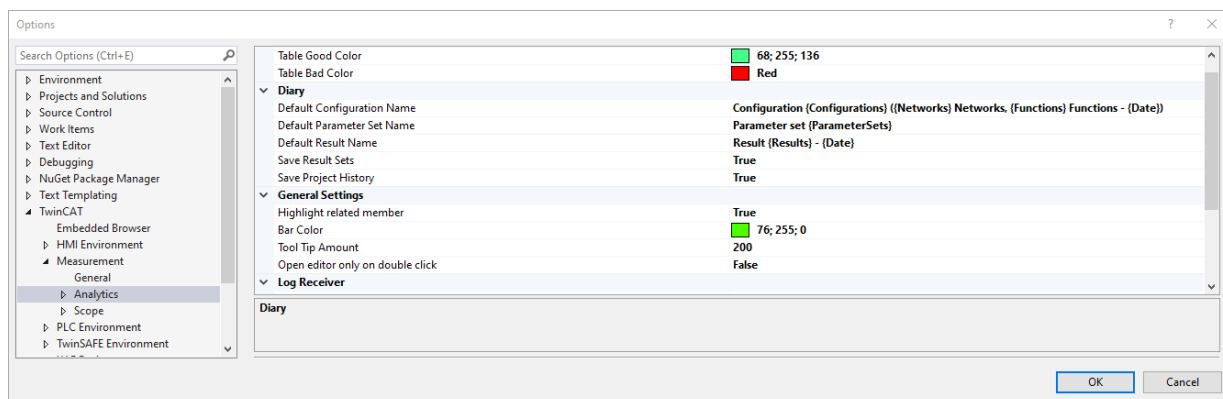
Please refer to the product overview page for Reporting Client trial limitations.

5.8 Comparison of analysis results

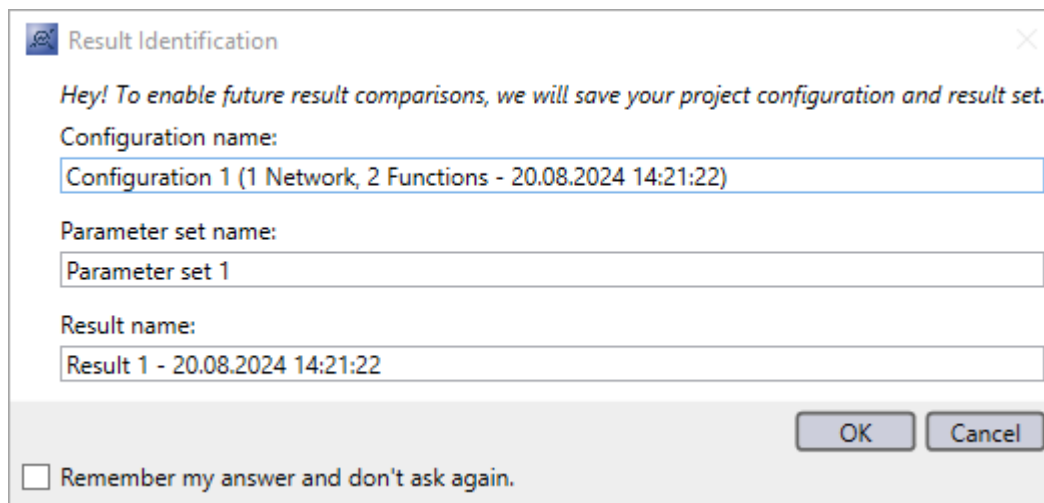
Various analysis runs can be compared with the Analytics Diary. The Analytics Diary offers several advantages for exploratory data analysis and identifying an appropriate analysis method, including algorithm parameters.

1. By default, all analysis results including the project configuration are saved with each analysis run. You can change this setting at

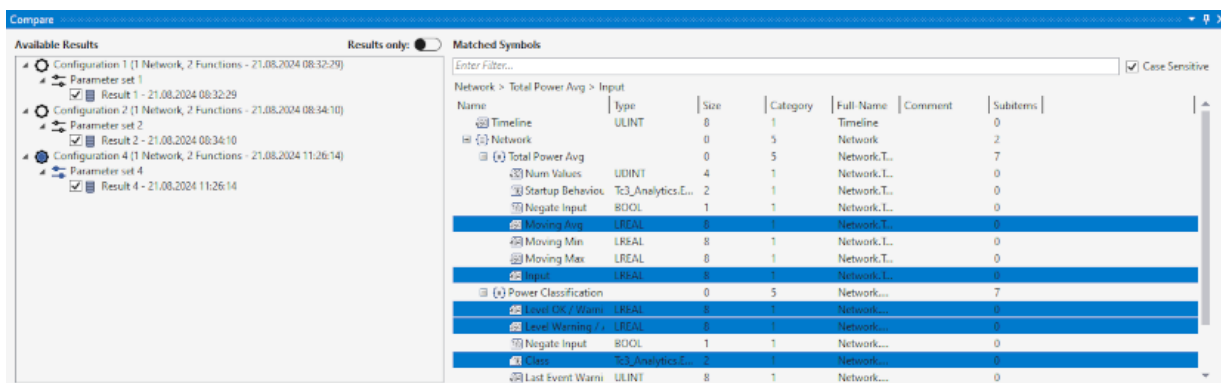
Tools > Options > TwinCAT > Measurement > Analytics > Diary .



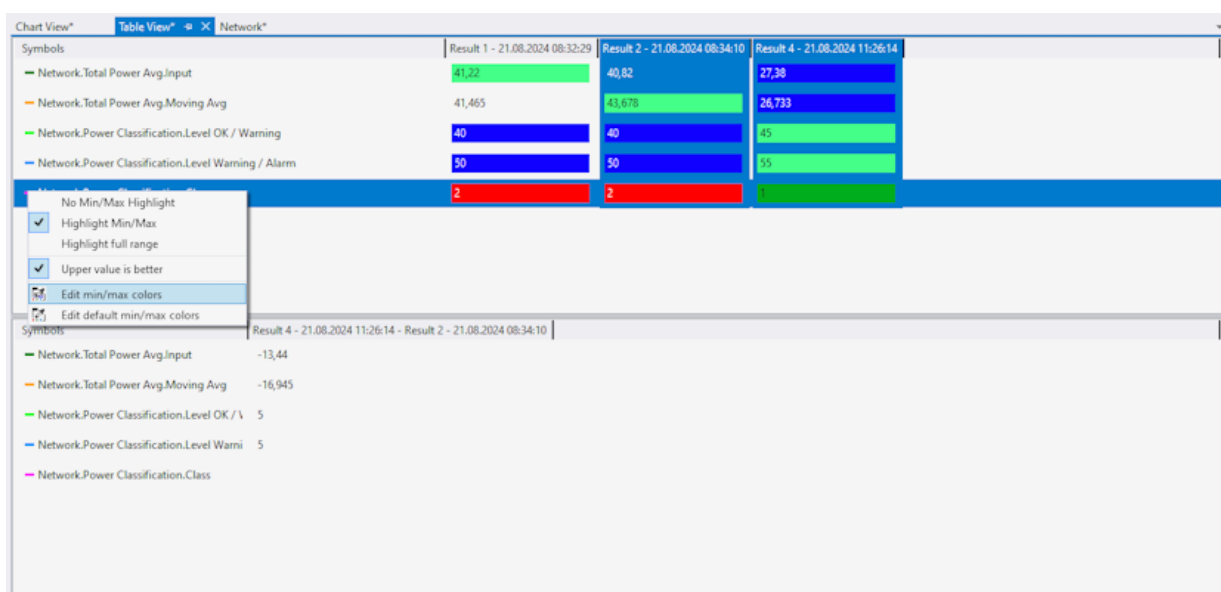
2. When you start an analysis, a Diary window appears. Here you can set the names of the configuration, the parameter set and the result. If nothing has changed in the project, only the data is saved. Likewise, no analysis configuration is generated if only the configuration of parameters changes (see also [Global parameters](#) [► 25]).



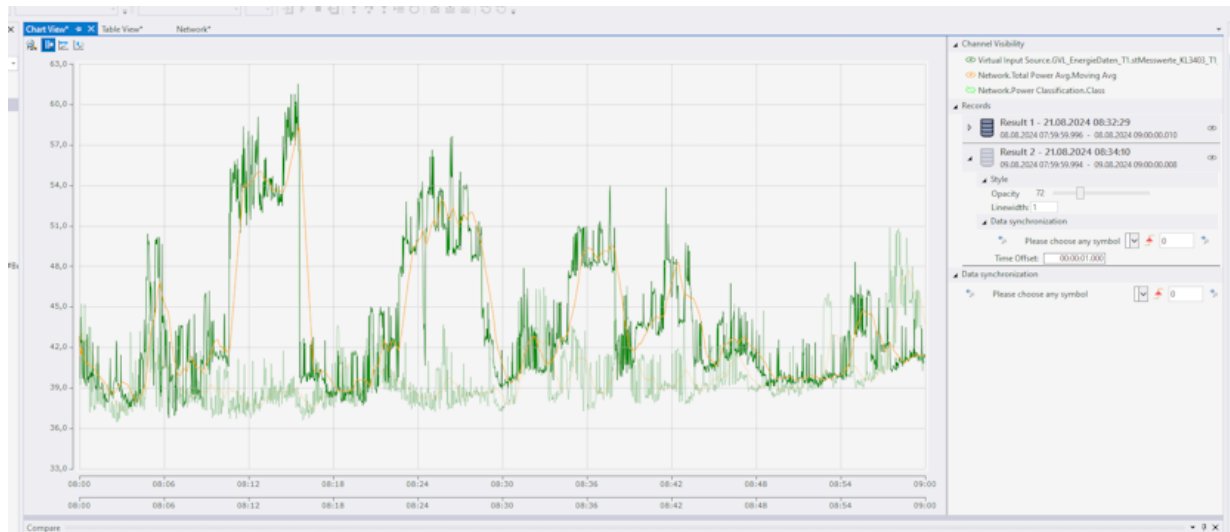
3. These entries will then appear in a tree structure in the **Compare** window, which you can open by clicking on them in the **Solution Explorer**. This shows the history of analysis runs. You can select multiple result data sets and matching symbols, which can then be viewed in the **Table View** and **Chart View** windows. The selection of data sets and symbols is therefore synchronized with the other two windows. The last data set is always selected by default.



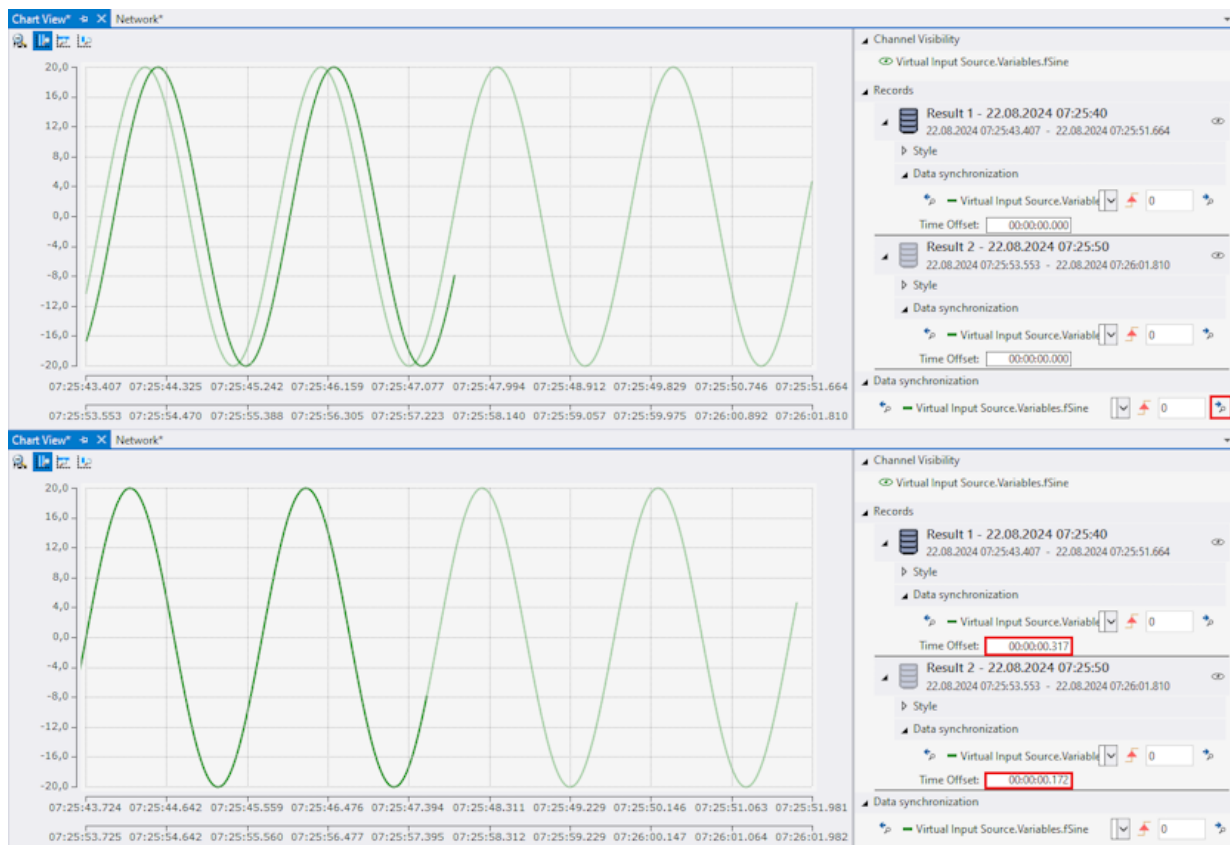
4. You can open the **Table View** and **Chart View** windows by clicking in the **Solution Explorer**. Two tables are displayed in the **Table View**. The first displays the last result of all selected data sets and symbols. The data sets are listed column by column and the symbols row by row. An optional color display can be used to visually highlight the differences between symbols. Furthermore, the differences between the symbols can be shown in a second table. At least two columns or data sets must be selected from the first table.



5. The *Chart View* is used to compare result data sets as time series. Different colors are assigned to the symbols. The symbols appear according to the number of selected data sets, whereby the opacity of the lines is reduced in each case. You can manually adjust the opacity and the line thickness to differentiate the data sets. To view a closer area, you can use a zoom equivalent to the scope.

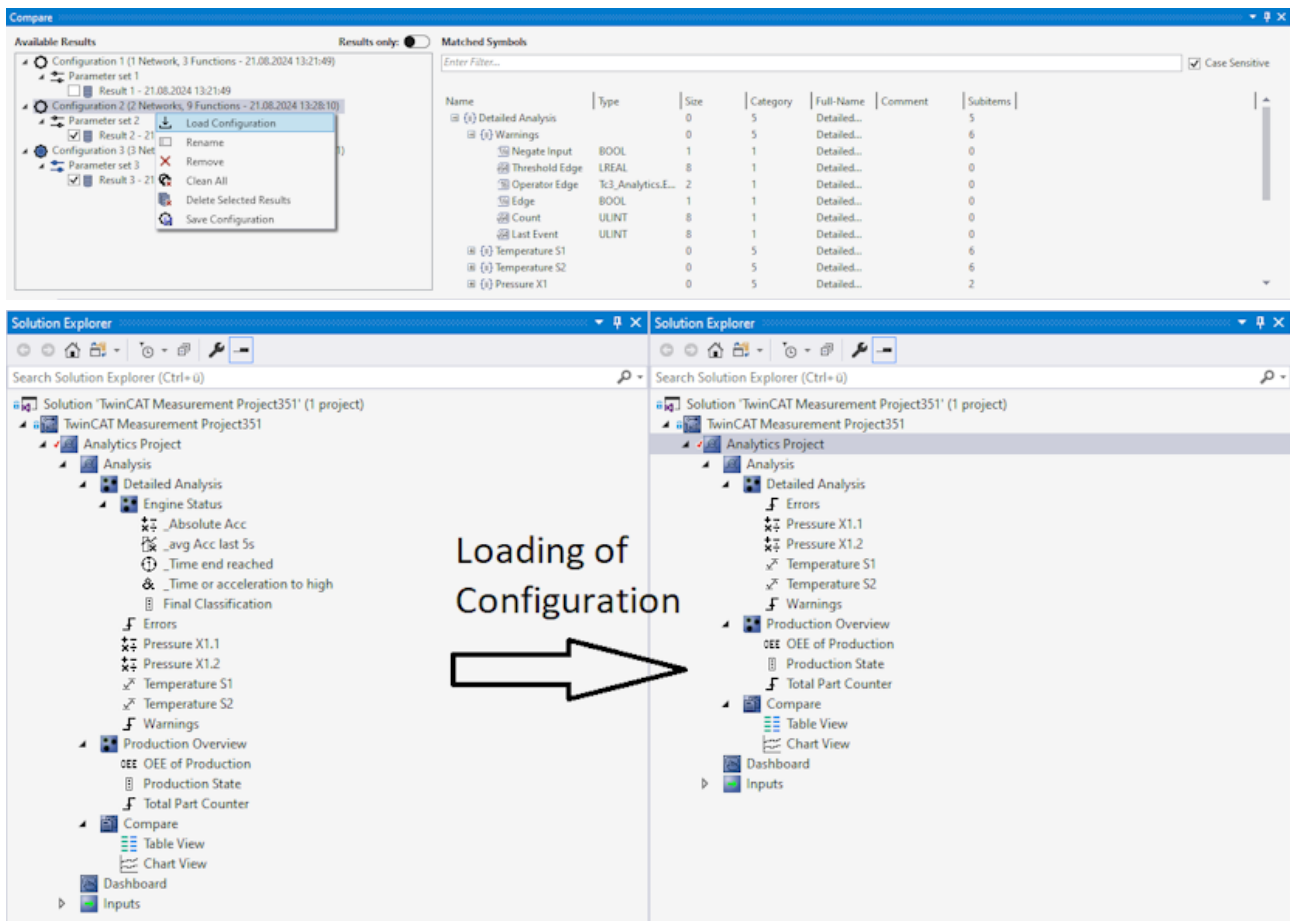


6. If the start times of the data sets do not match, you can adjust them for a direct comparison. This can be done either by manually entering the time offset or by an automatic edge search.



5.8.1 Loading and saving project configurations

The history of the analysis runs is listed in the *Compare* window. The corresponding project status is saved for a configuration and can be loaded. To load the configuration, right-click on the desired configuration (or parameter set) and select the *Load Configuration* option in the context menu. This restores the old project status. The configuration icon then turns blue, indicating the last loaded configuration.



In addition to automatically saving a configuration at the start of an analysis, you can also save it manually. To do this, simply select *Save Configuration* from the context menu.

6 Appendix

6.1 FAQ - Frequently asked questions and answers

This section answers frequently asked questions to make your work with TwinCAT Analytics Vision easier. If you have any further questions, please contact our support team at support@beckhoff.com.

1. I have purchased TE3510 Analytics Vision, but need additional Condition Monitoring algorithms for my application. Can I extend my package? [► 482]

I have purchased TE3510 Analytics Vision, but need additional Condition Monitoring algorithms for my application. Can I extend my package?

Yes, you can combine all TE351x products as you wish. If you specifically require Condition Monitoring algorithms, you can also license TE3512 in addition to TE3510.

7 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our [download finder](#) contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for [local support and service](#) on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963-157
e-mail: support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963-460
e-mail: service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963-0
e-mail: info@beckhoff.com
web: www.beckhoff.com

Trademark statements

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Chrome, Chromium and Google are trademarks of Google LLC.

Excel, IntelliSense, Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

MAX®, Stratix®, Cyclone®, Altera®, Agilex™, Arria®, Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

More Information:
www.beckhoff.com/te3510

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com

