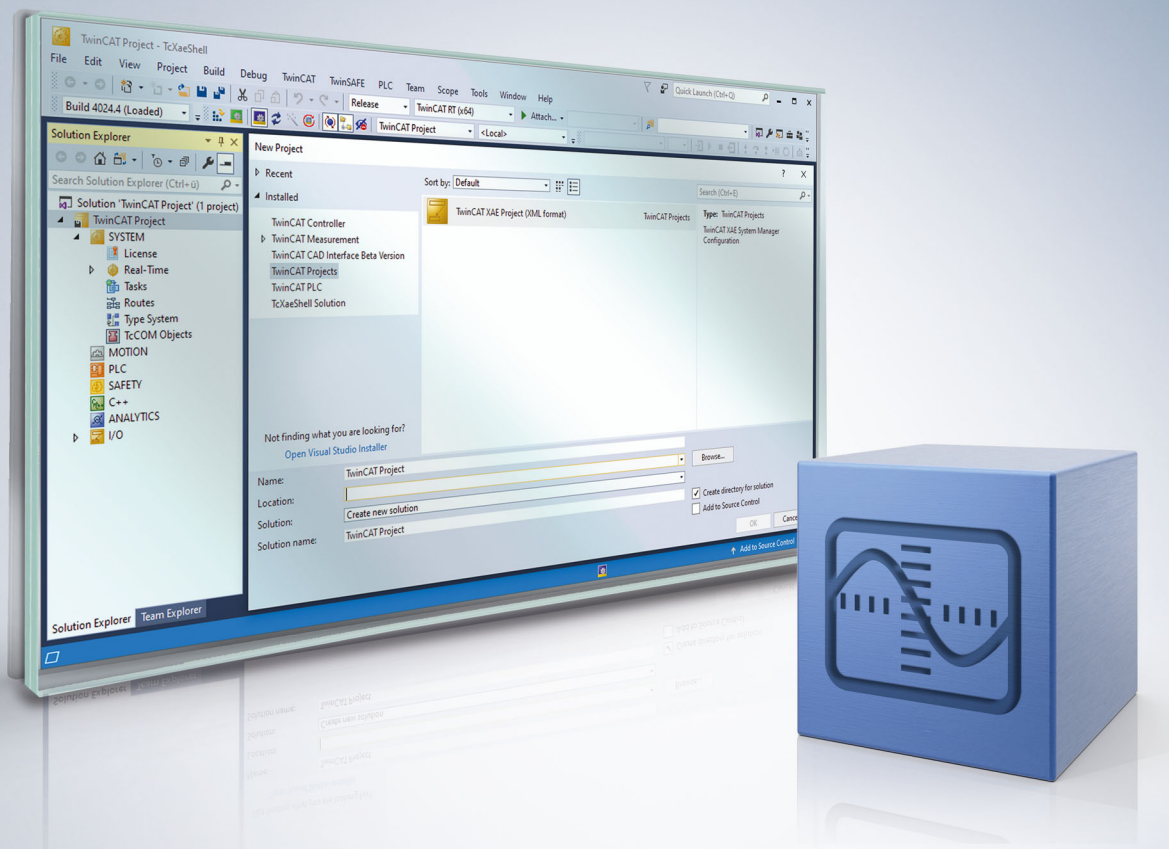
BECKHOFF New Automation Technology

Manual | EN

TE3520

TwinCAT 3 | Analytics Service Tool



2023-07-18 | Version: 1.4.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared 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 prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

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

Copyright

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

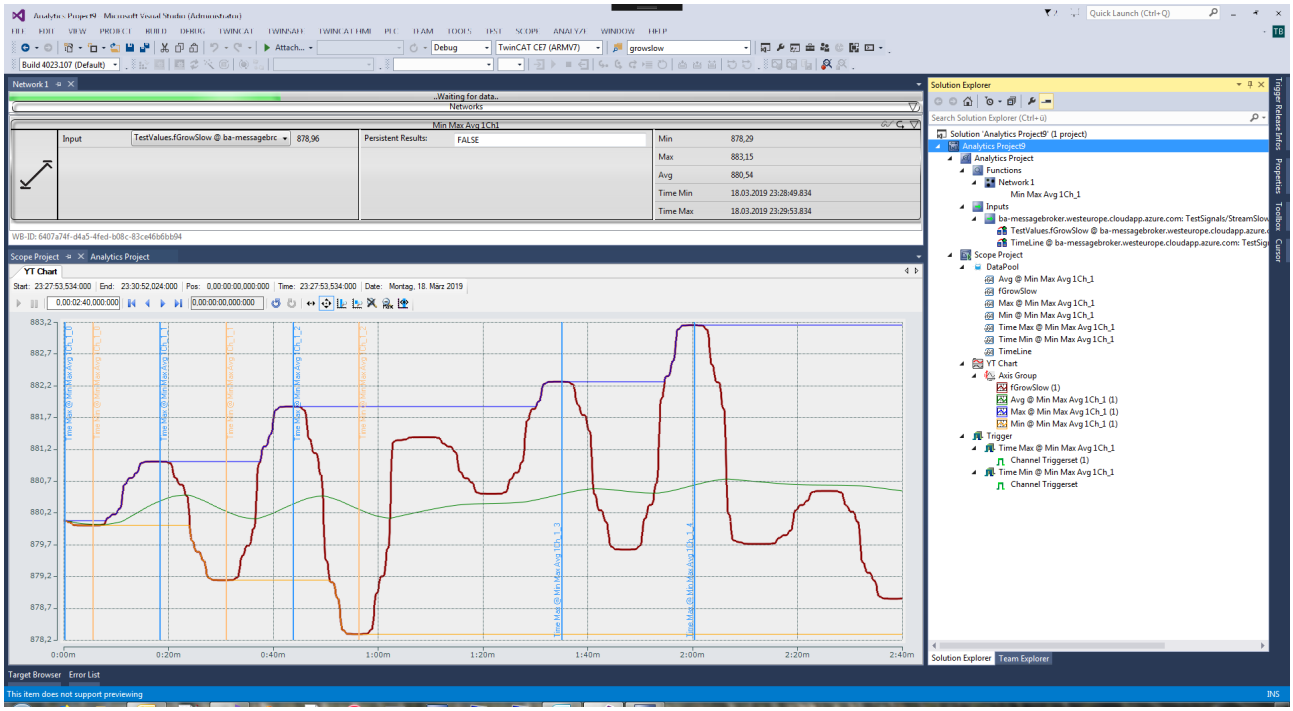
The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of 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



The TwinCAT 3 Analytics Service Tool is used for commissioning the machine and for service engineers. Live and historical data can be retrieved for an analysis via the IoT connection or offline via file copy. The analysis is configured in Microsoft Visual Studio where the user has access to a toolbox of algorithms for implementing the relevant life time, cycle time, envelope or component counter analysis. The outputs of the algorithms can be used as inputs for other algorithms or can be output as a result directly in the graphical editor. Signal paths can be visualised with ease by means of parallel recording with the TwinCAT Scope. Analysis results can be dragged by the user from the analytics configurator and dropped in the charting tool so as to mark the significant positions in the data stream. The interaction between the product components offers advantages in particular for diagnosing machine behavior and can highlight optimisation potential. The user's location is immaterial owing to the IoT technologies used, which means that service technicians can perform system and machine diagnostics from practically any location.

Components

- Analytics configurator
- Base Analytics algorithm
- Analytics Storage Provider Recorder
- TwinCAT Scope (TE1300 and TF3300)
- IoT Connectivity

Features	TE3500 Analytics Workbench		TE3520 Analytics Service Tool	
	7-Days-Trial	Full License	7-Days-Trial	Full License
General:				
Analysis configurator	✓	✓	✓	✓
Analysis channels	max 5	unlimited	max 5	unlimited
Analysis moduls/ algorithm	max 3	unlimited	max 3	unlimited
Long time records >1h	✗	✓	✗	✓
Interaction with Scope View	✓	✓	✓	✓
Storage Provider Recorder	✓	✓	✓	✓
MQTT	✓ (max 5)	✓	✓ (max 5)	✓
Analytics File	✓ (max 5)	✓	✓ (max 5)	✓
ADS	✓ (no auto deployment)	✓ (no auto deployment)	✓	✓
Data export tool	✓	✓	✓	✓
Basic data export formats	✓ (max 5)	✓	✓ (max 5)	✓
Extended data export formats	✗	✓	✗	✓
Analytics Data Scout:				
Load data	✓	✓	✓	✓
Toolbar snipping buttons	✗ *	✓	✗ *	✓
Export data to Analytics File	✓	✓	✓	✓
Deploy Runtime:				
Deploy Wizard	✓	✓	/	/
Add PLC Code to existing Sln	✗	✓	/	/
Merge PLC Code	✗	✓	/	/
Auto Create Bootproject	✗	✓	/	/
Auto Activate Runtime	✗	✓	/	/
Stream Results	✗	✓	/	/
HMI Dashboard	✓ (defaults)	✓	/	/
Reset Algorithm in PLC Code	✗	✓	/	/
Reporting Integration:				
24/7-Reporting with Reporting Algorithm	✓	✓	/	/

Features	TE3500 Analytics Workbench		TE3520 Analytics Service Tool	
On-Demand-Reporting				
Adding Additional Information	✔ (max 1)	✔	✔ (max 1)	✔
Select specific Analytics Information	✘	✔	✘	✔
✔	Full support			
✘	No support			
/	No feature of this product			
*	By existing TE1300 Scope Professional full support			

3 Installation

3.1 System requirements

The following system requirements must be fulfilled for proper function of TwinCAT Analytics.

Supported operating systems

Windows 10

TwinCAT

Minimum is TwinCAT 3.1 Build 4022.29 for engineering with TwinCAT Analytics Service Tool and Workbench.

.NET Framework

Engineering requires a .NET Framework 4.7.2.

Visual Studio development environment

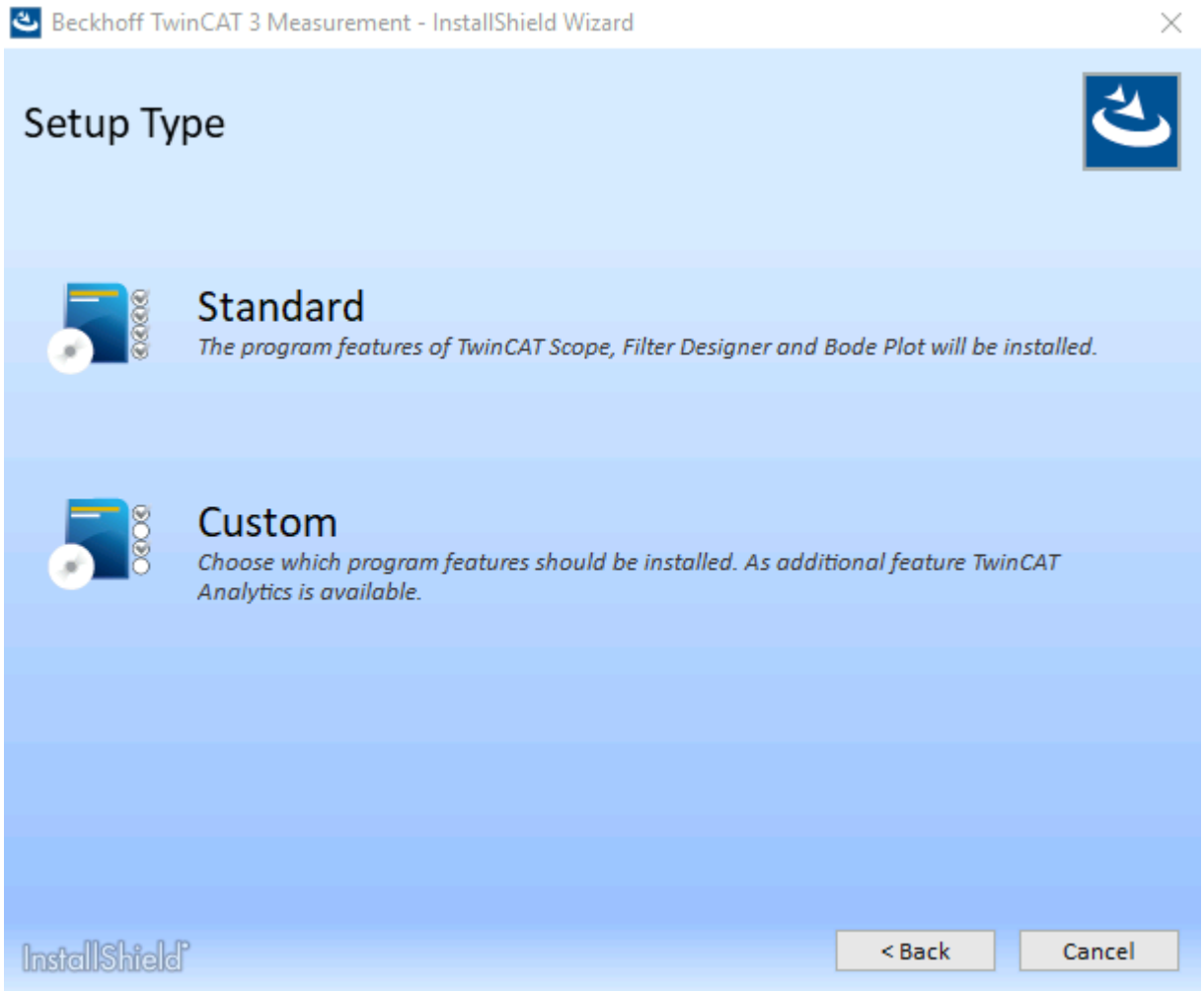
- Visual Studio® 2015
- Visual Studio® 2017
- Visual Studio® 2019
- TwinCAT XAE Shell

In general, using the Visual Studio® Shell is sufficient. If you select the "Full" setup, the TwinCAT XAE Shell is installed automatically. The "Update" setup only provides an update of the Analytics sources and not a Visual Studio® Shell.

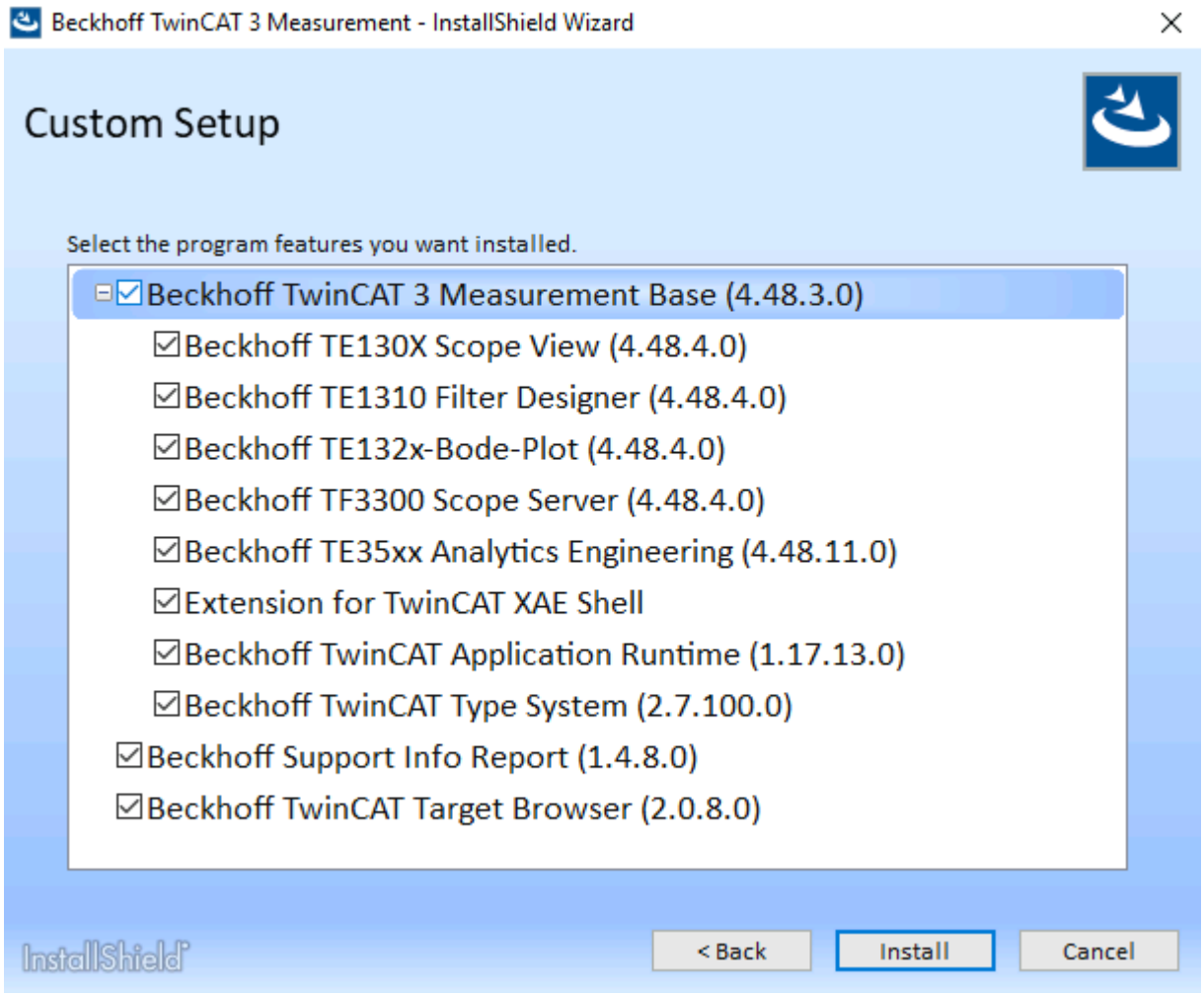
3.2 Installation and licensing

The TwinCAT Analytics setup is part of the TwinCAT Measurement Suite setup. You have the option of choosing between two setup variants:
The TC3-Measurement-Full Setup includes the TwinCAT Measurement components and a Visual Studio® environment.
The TC3 Measurement Update Setup, on the other hand, only includes the measurement components.

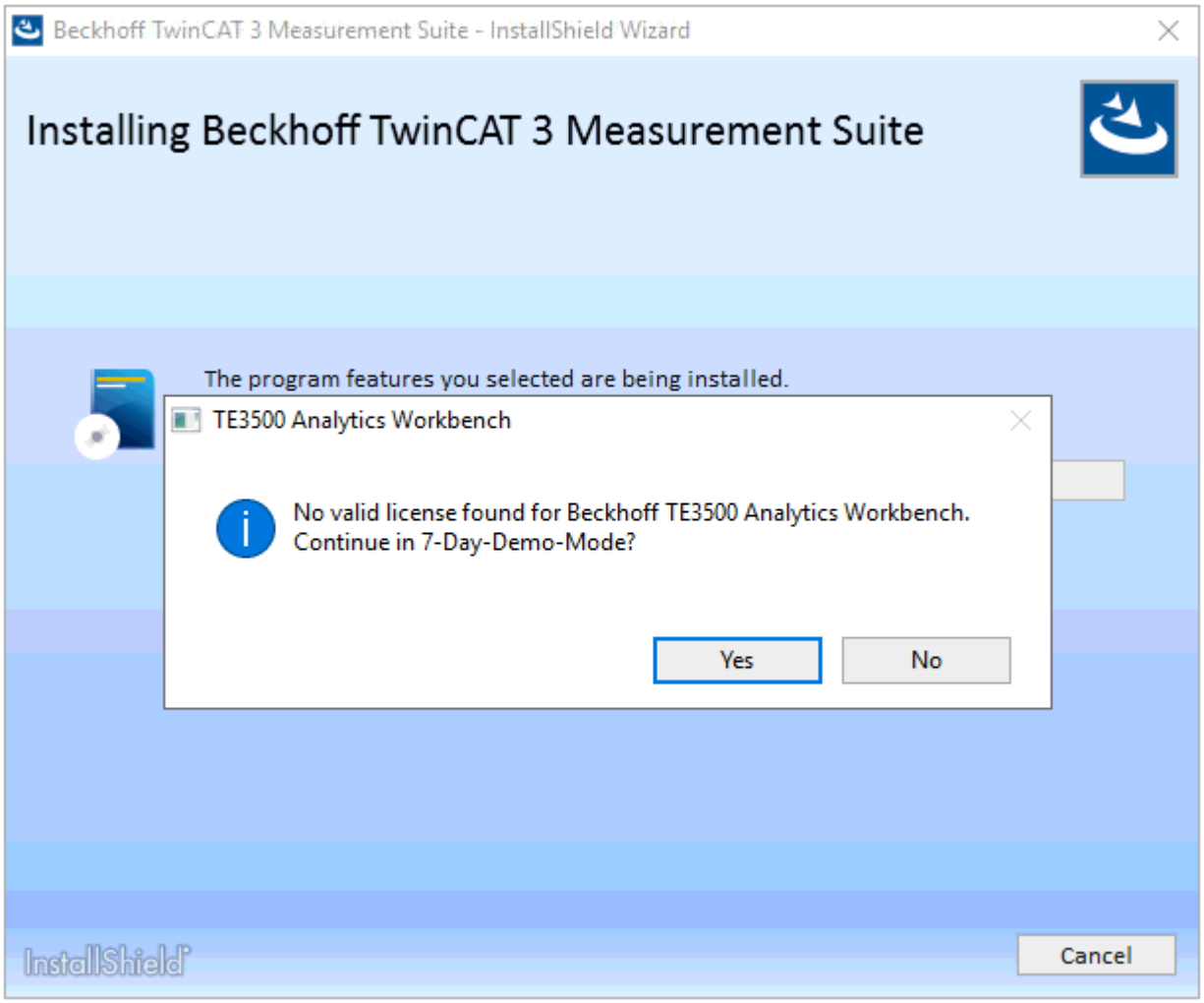
After confirming the terms in the license agreement, you can choose between Standard and Custom. By default, all measurement components including TwinCAT Analytics Engineering are automatically installed.



With Custom, you can deselect individual components so that they are not installed on your system.



Analytics Setup checks whether your system has the required Analytics Engineering licenses during the process. If not, a demo license can be activated. This demo license can be extended as often as you like. However, after the installation in demo mode, no further functional updates can be performed on this system through newer setups. In order to be able to import these updates, a license must first be purchased. An overview of the functional limitations in the demo version can be found here [▶ 9].



The licenses of the Analytics Engineering Tools TE3500 Analytics Workbench and TE3520 Analytics Service Tool are always associated with a maintenance license. After the initial purchase, the maintenance license is valid for 12 months. Functional updates of the software can be executed during this period. After 12 months, the software can still be used without any restrictions. However, new functions can no longer be imported without extending the maintenance license. The installation indicates this with a corresponding message. An extension of the maintenance license can be performed at any time via the products TE3501 and TE3521.

Please contact your Beckhoff sales employee for information about future functions of newer versions.

Setup requires license

Updates of TwinCAT Analytics engineering tools can only be performed with a valid maintenance license!

TwinCAT 3 licenses for non-Beckhoff devices

If you use an IPC from a manufacturer other than Beckhoff (TwinCAT 3 plattform level >= 90), aTwinCAT 3 licencse dongle is highly recommended, if not a prerequisite for successful licensing of TwinCAT Analytics!

3.3 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

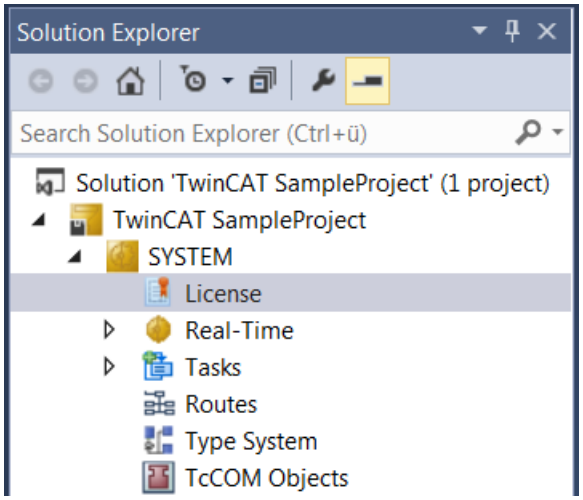
Licensing the full version of a TwinCAT 3 Function

A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

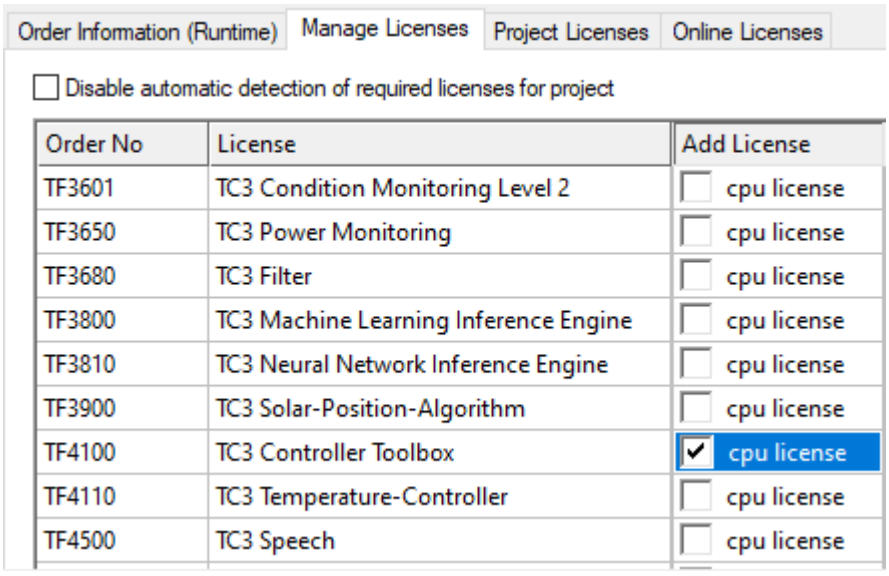
Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



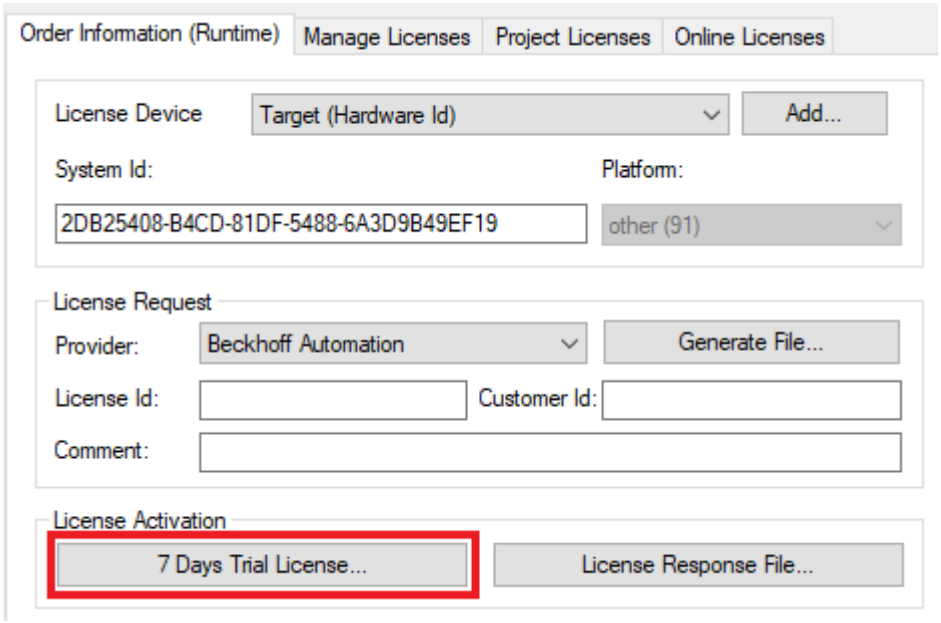
   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



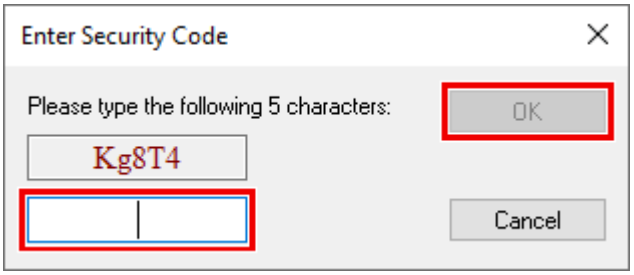
Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

6. Open the **Order Information (Runtime)** tab.
   ⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

7. Click **7-Day Trial License...** to activate the 7-day trial license.



⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

4 Analytics Workflow - First Steps

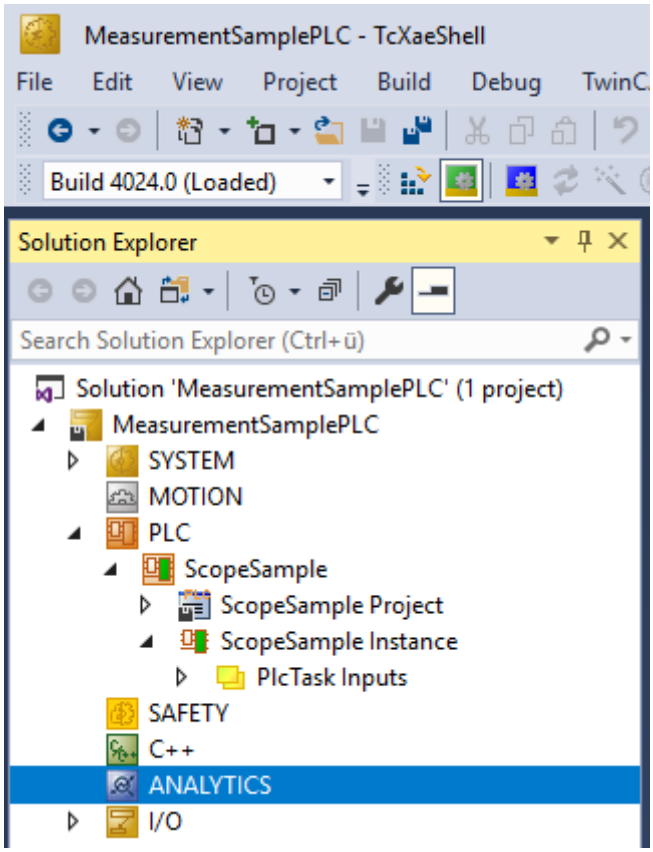
This step by step documentation presents the complete TwinCAT Analytics workflow. From the data acquisition over the communication and historizing up to the evaluation and analysis of the data and to the presentation of the data in web-based dashboard.

4.1 Recording data from the machine

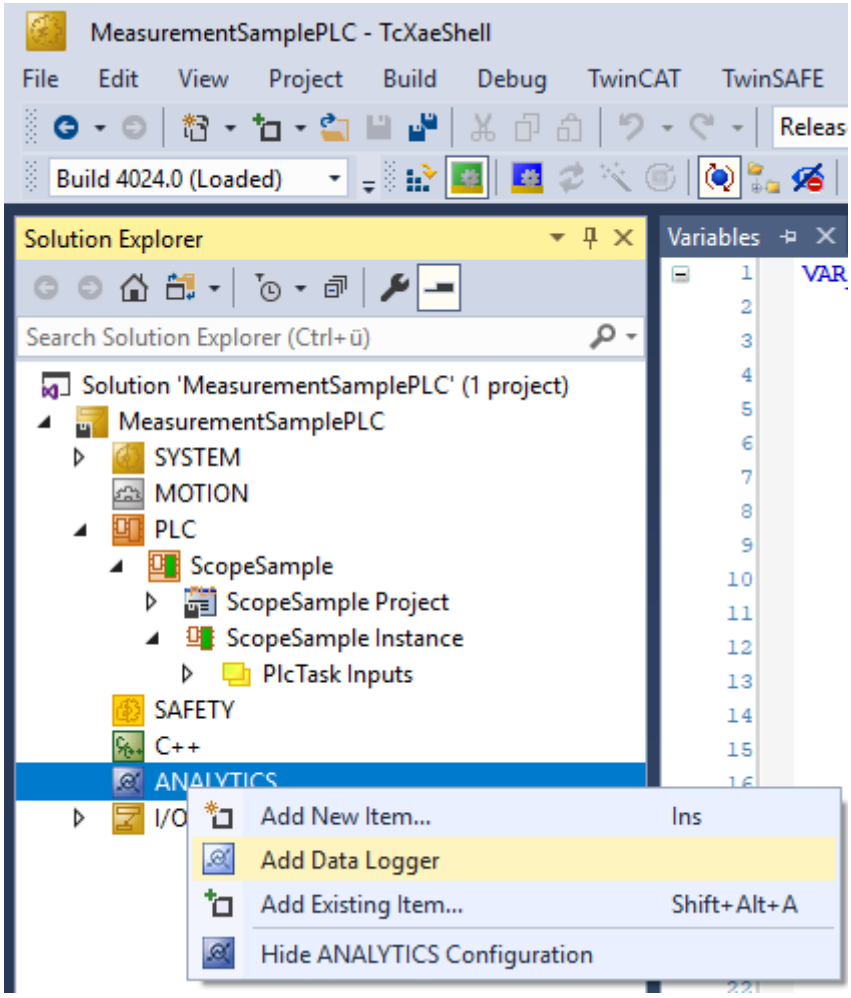
On the machine side is the Analytics Logger the recorder of process data from the machine image, PLC, NC and so on. The Logger is working in the real-time context of TwinCAT.

The TwinCAT Analytics Logger is installed with TwinCAT XAE and XAR. The Logger can act as MQTT Client to communicate the recorded data to a native MQTT Message Broker or store the data in the same data format in a local binary file. By the usage as MQTT Client the Logger is able to bypass short disconnects to the Message Broker with a ring buffer functionality. You can configure a ring buffer as well for the local binary file storage.

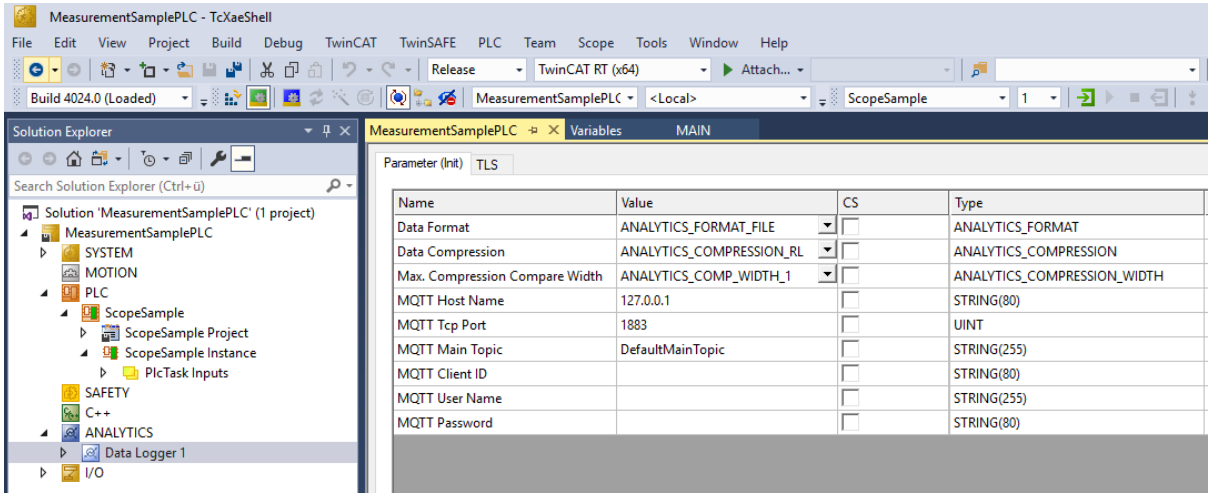
- To configure the Analytics Logger you have to navigate in your existing TwinCAT Project to the Analytics tree node



- Right click on this node and click on “Add Data Logger” to add one new instance to your configuration



- For configuring the base settings, please double click on the new tree item



You can make your specific Analytics Logger settings

-Data Format: Binary file or MQTT stream

-FILE format: Analytics Logger stores the data in local binary files and all other settings are not necessary anymore. The files will be stored in C:\TwinCAT\3.1\Boot\Analytics.

-BINARY: Data will be sent to the configured MQTT Message Broker. You can have multiple Logger in one TwinCAT project to communicate data to different MQTT Message Broker.

-Data Compression: on (default) or off

-Max Compression: mode of the compression

-MQTT host name

-MQTT Tcp port

-MQTT main topic for own hierarchical levels to keep the identification easy

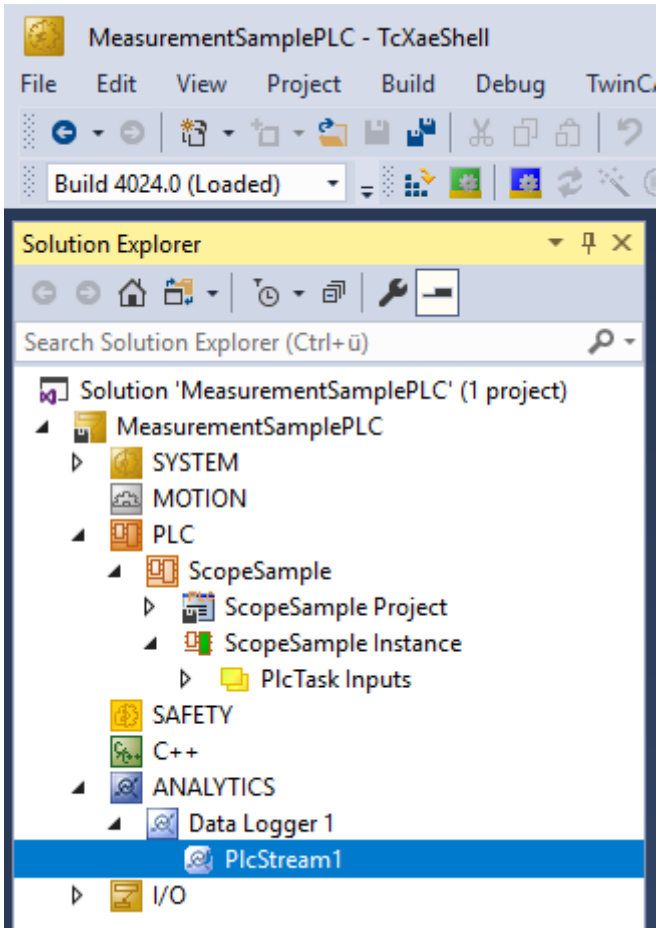
-MQTT Client ID should be unique in the network

-MQTT username

-MQTT password to make authentication at the message broker

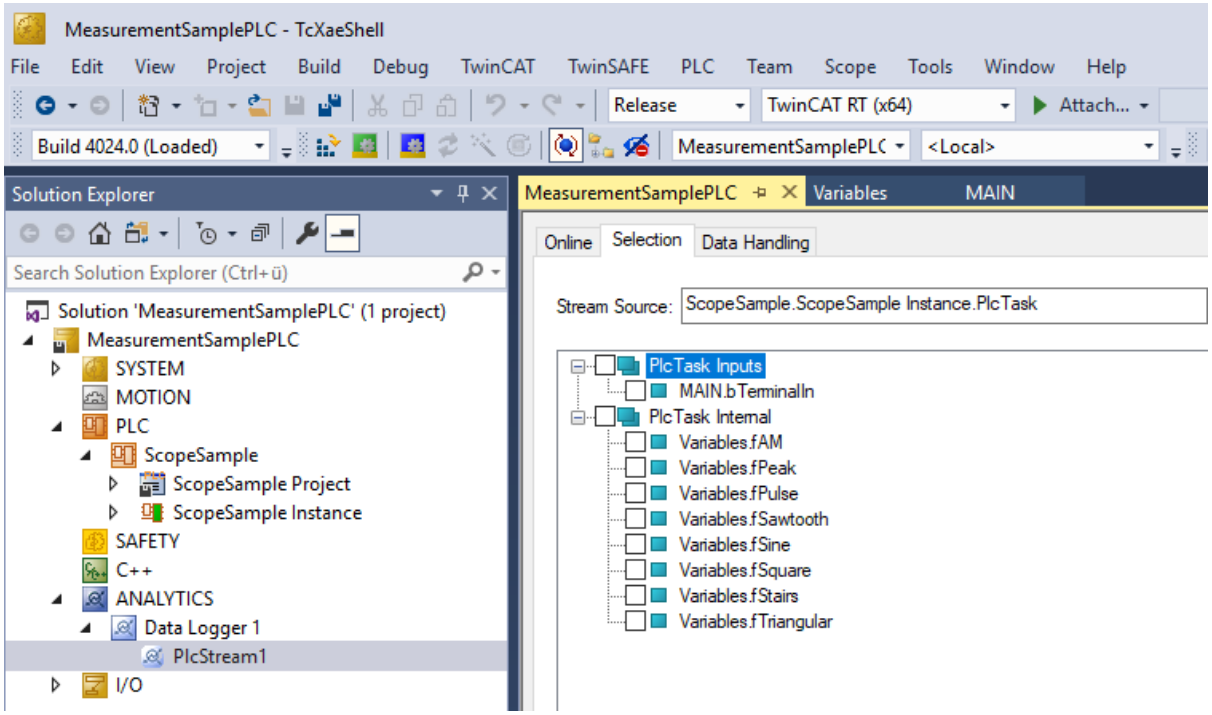
-At the TLS (Transport Layer Security) tab, security settings can be configured. TLS is a secure communication channel between client and server. By the usage of certificates, the TCP port 8883 is exclusively reserved for MQTT over TLS. Analytics Logger is supporting the modes CA Certificates, CA Certificates & Client Certificate and Preshared Key (PSK) mode.

- If variables in your PLC application are marked in the declaration with the attribute {attribute 'TcAnalytics'} they will be shown automatically as a stream below the Data Logger tree node.

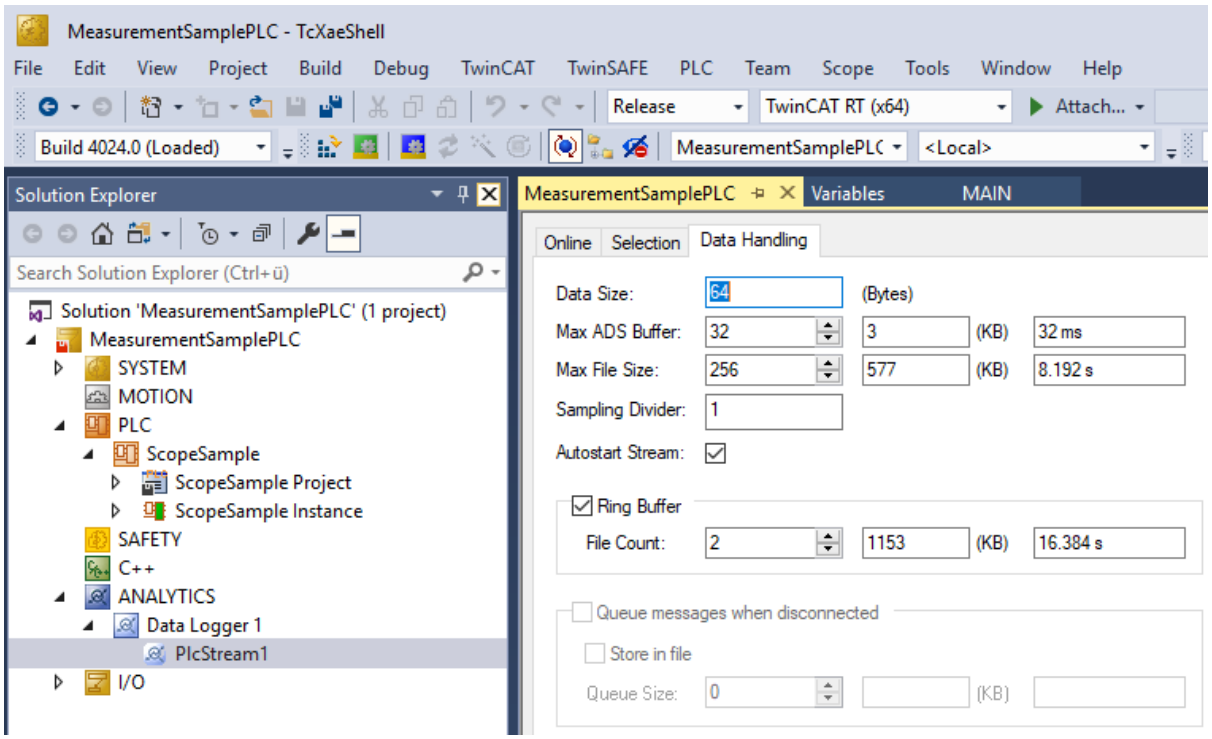


An additional device stream will be shown if your configuration provides an EtherCAT Process Image.

- In the stream a Selection tab is available to choose the variables that should be recorded

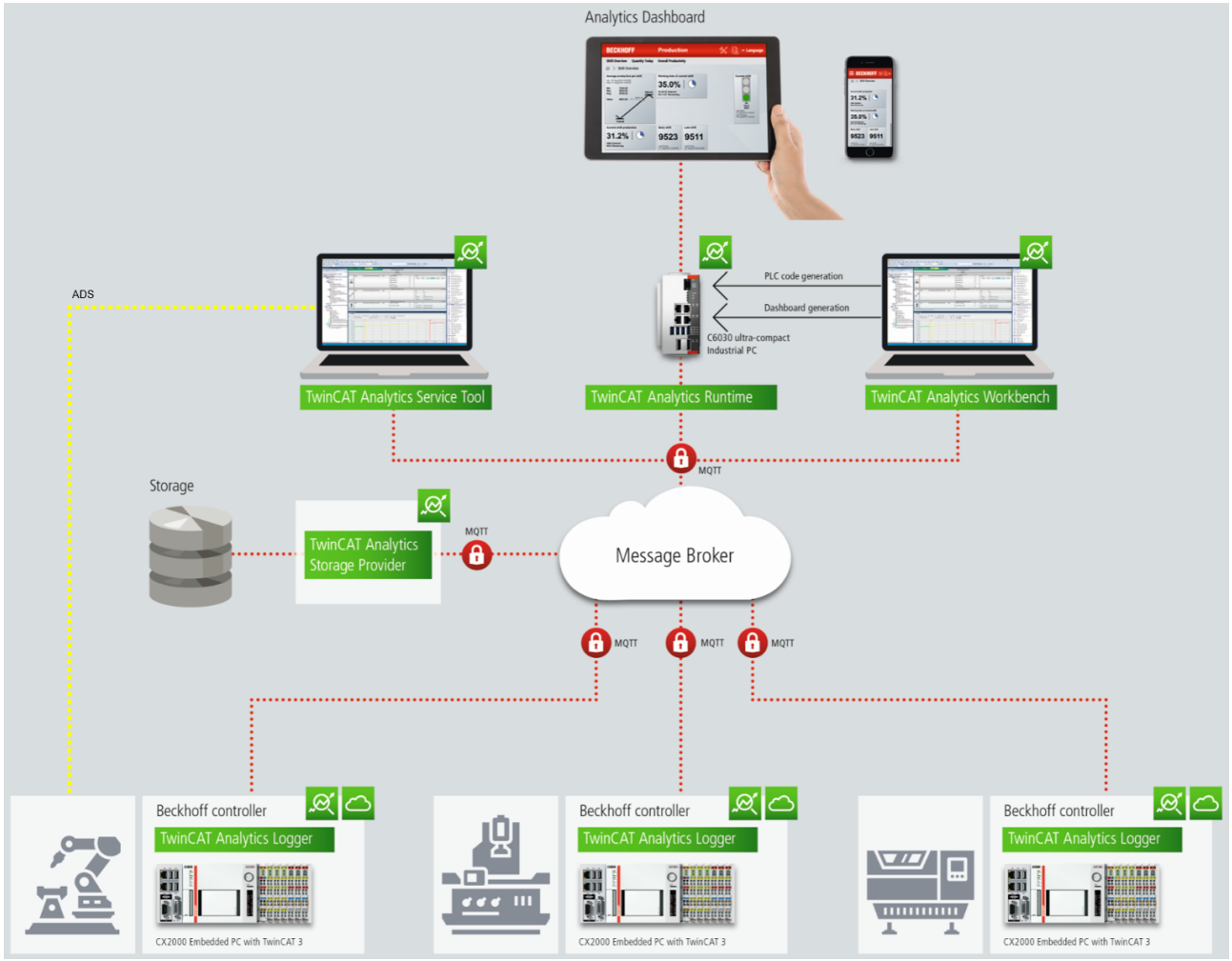


- Finally it is possible to change the package size for the frames or to configure the ring buffer for disconnects and file in the Data Handling tab.



4.2 Communication

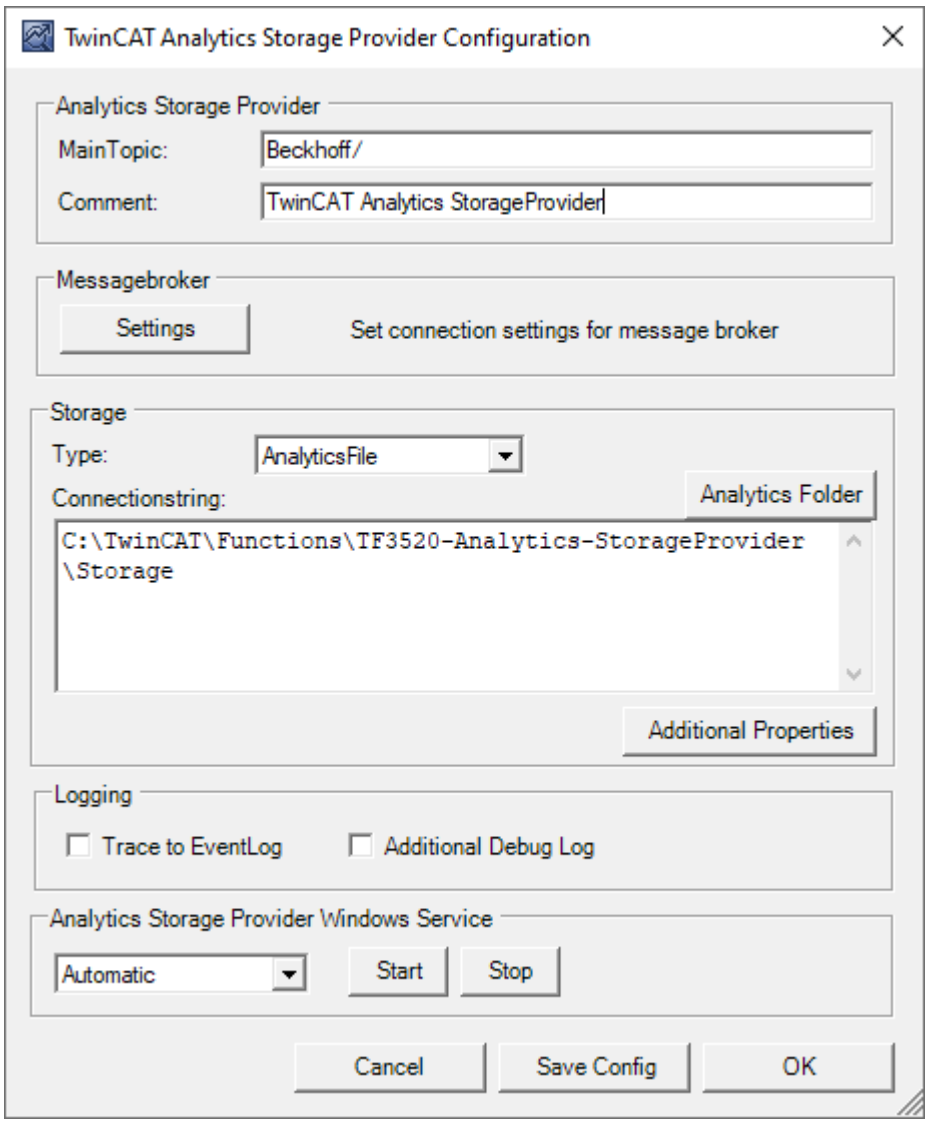
Currently, the Analytics workflow is fully mappable via MQTT. The engineering tools can also access the data of the machines via ADS and carry out analyzes.



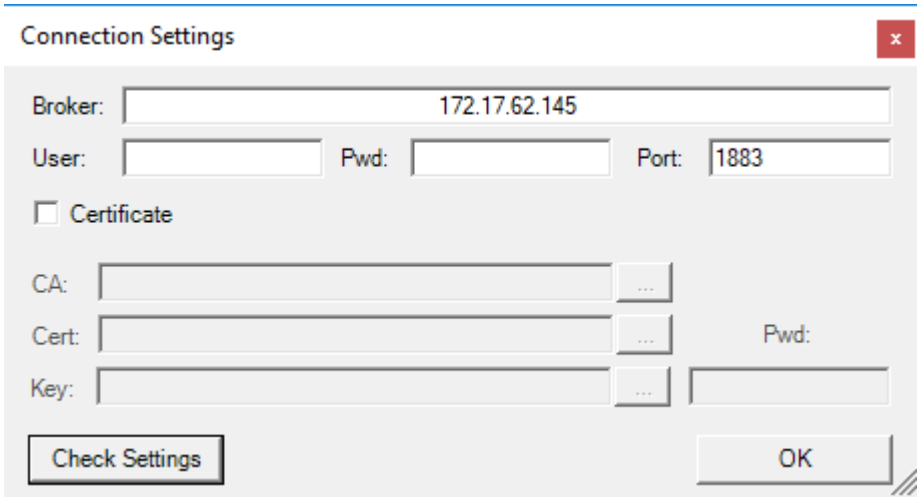
If you choose for the IoT communication protocol MQTT you have to setup a native MQTT Message Broker somewhere in the network (VM in a cloud system is also possible). This Message Broker provides a decoupling of the different applications in the Analytics Workflow.

4.3 Historicize data

After the TwinCAT Analytics Storage Provider has been installed, the service running in the background can be configured. For this purpose you can find the TcAnalyticsStorageProvider_Config application in the *C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\WinService* folder.



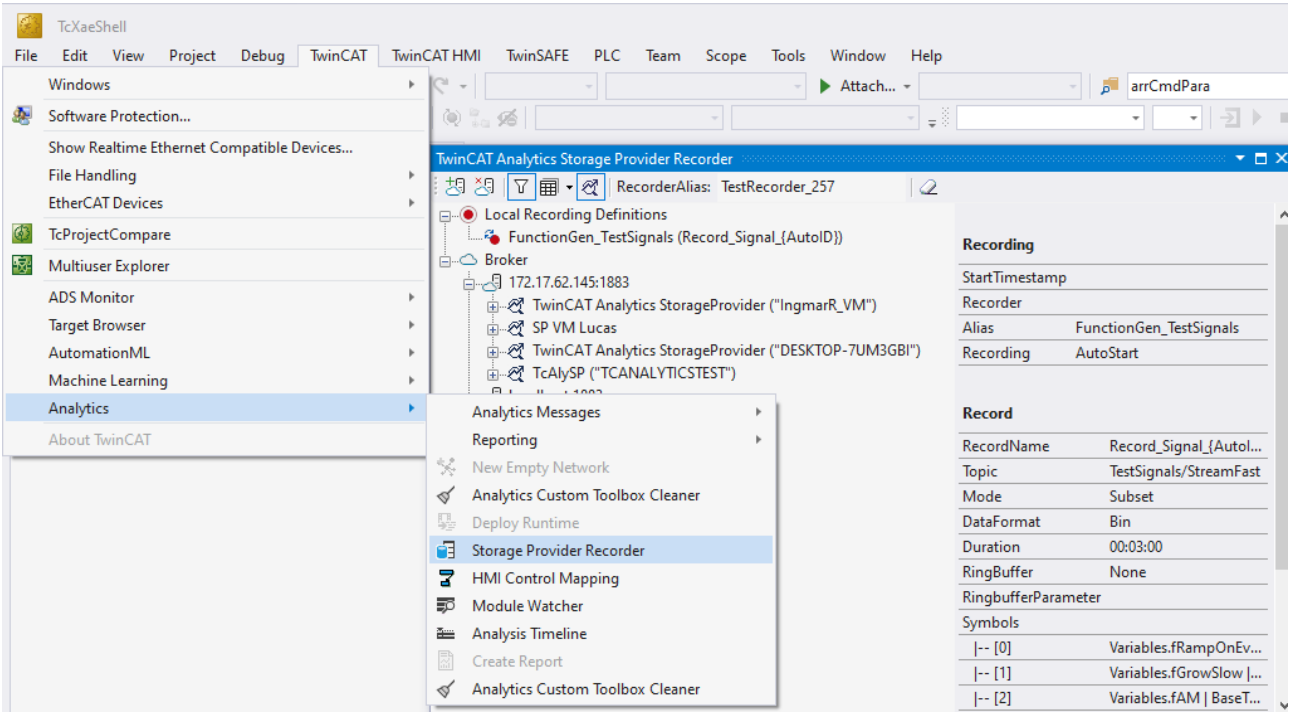
The main part of the topic can be defined in the configuration as well as the comment, which is used for identification if more than one Storage Provider is registered with the message broker.



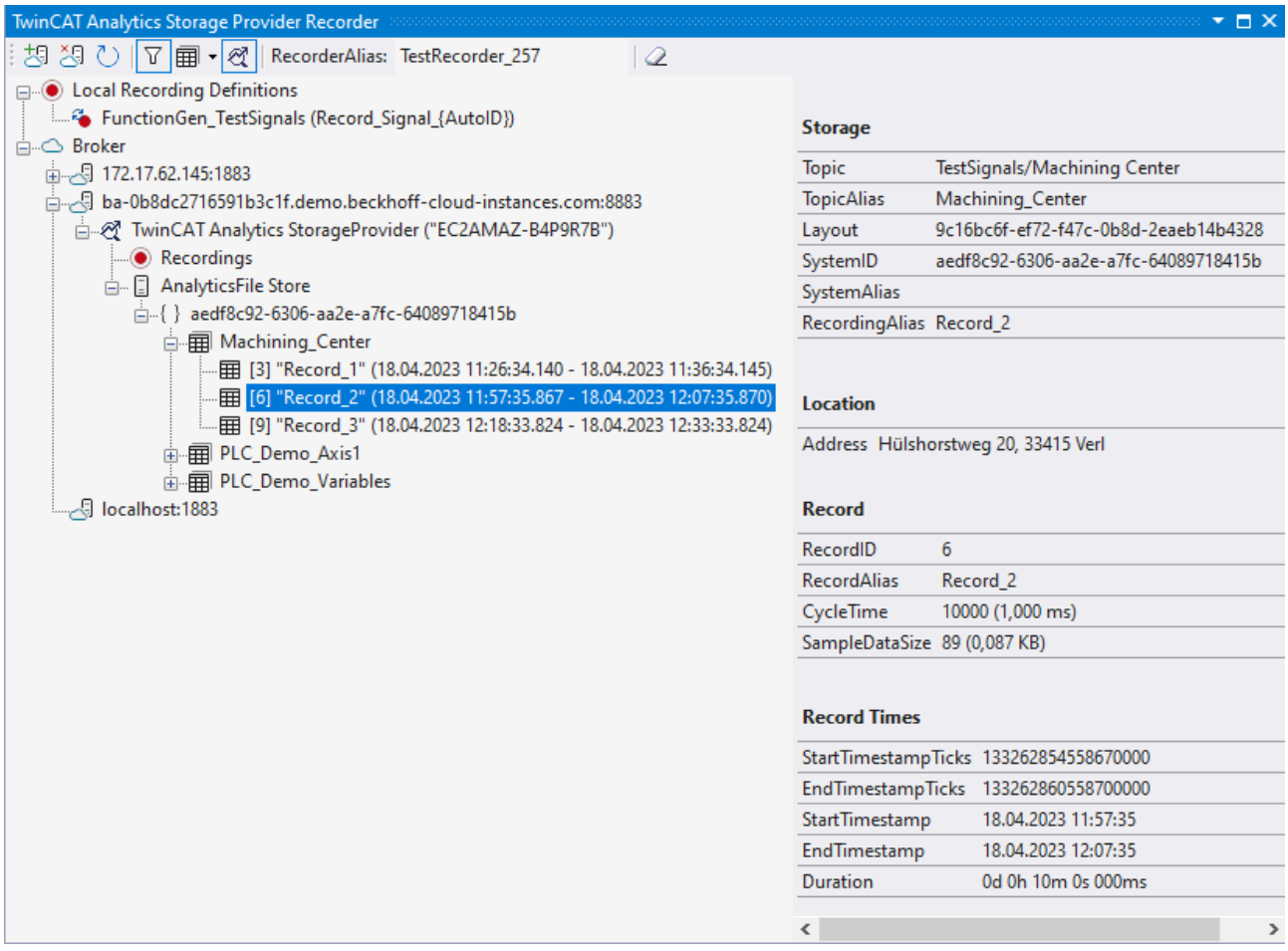
You can make the message broker settings and decide on a storage type:

- Analytics File (binary file)
- Microsoft SQL (binary)
- Microsoft Azure Blob (Azure Cloud required)

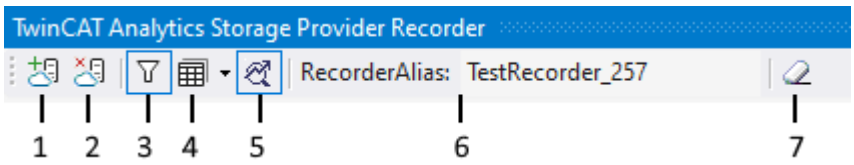
At last you can save the configuration and start the service. The next step is to configure the specific recording. For this you should select the **Storage Provider Recorder** in your development environment.



With the Storage Provider Recorder recording definitions can be created, started and managed. In addition, it is possible to manage the data memories of individual Analytics Storage Providers. All important properties of the found Analytics Storage Providers and historized data are clearly displayed.



Toolbar

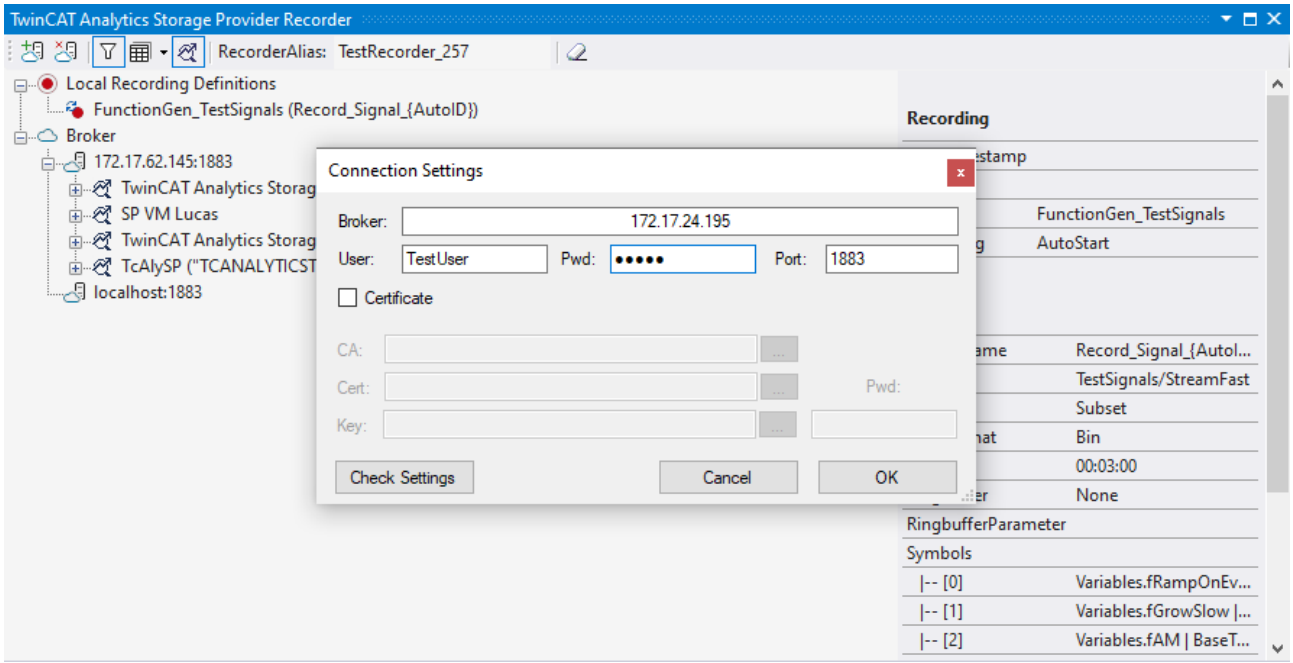


1	Add new broker
2	Remove selected broker
3	Display filters (All / Show my active recordings)
4	Select display type (Alias / Topics)
5	Show / Hide Offline Analytics Storage Provider
6	"RecorderAlias" - grouping name of recordings
7	Remove messages from error list

Recorder window setup

First assign a "RecorderAlias". This helps to group the started recordings and to find its self started ones again. The filter can also be used to display recordings started by others.

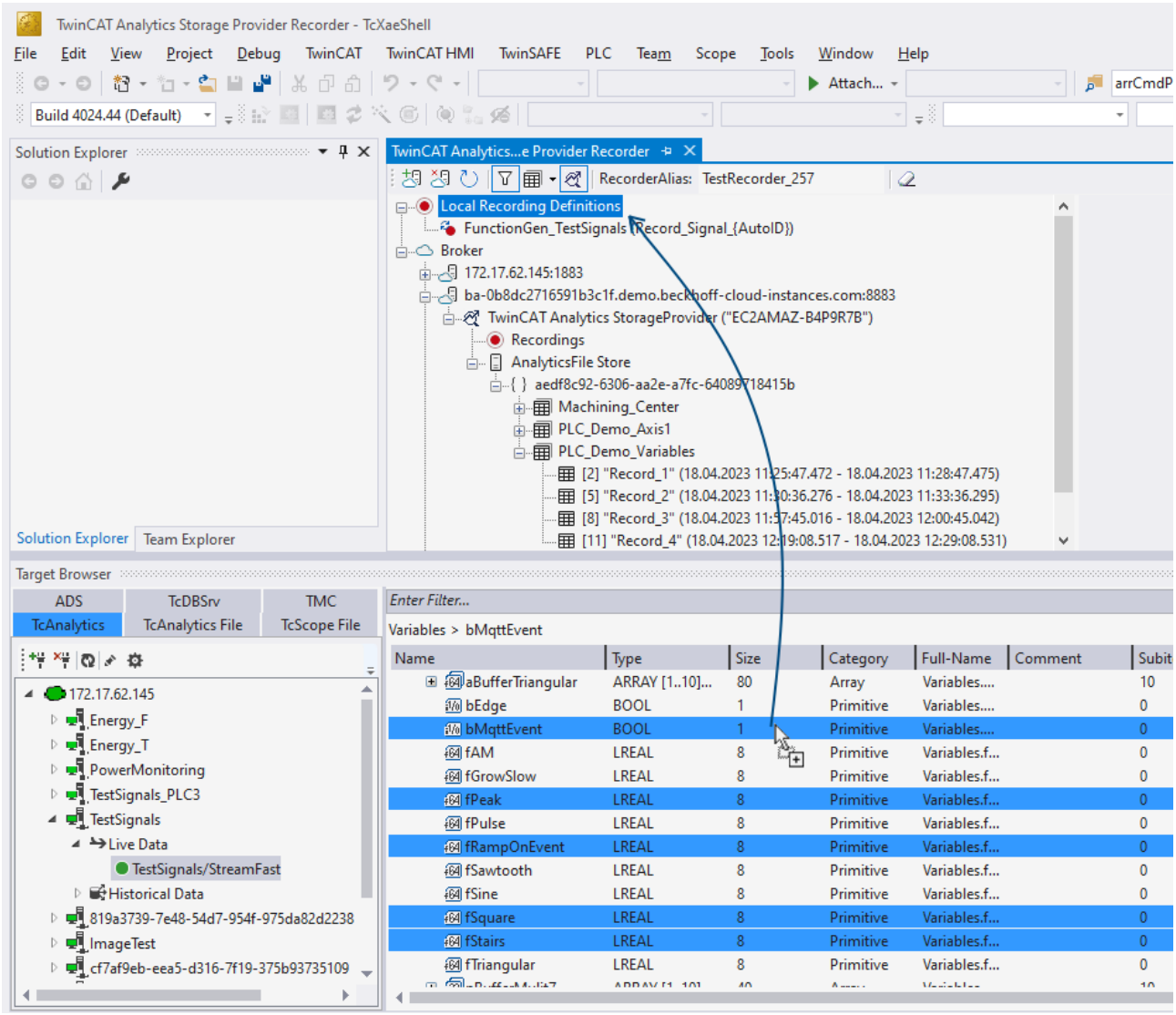
After that, one or more brokers can be set up. This is done via the already known input mask for MQTT connection properties.



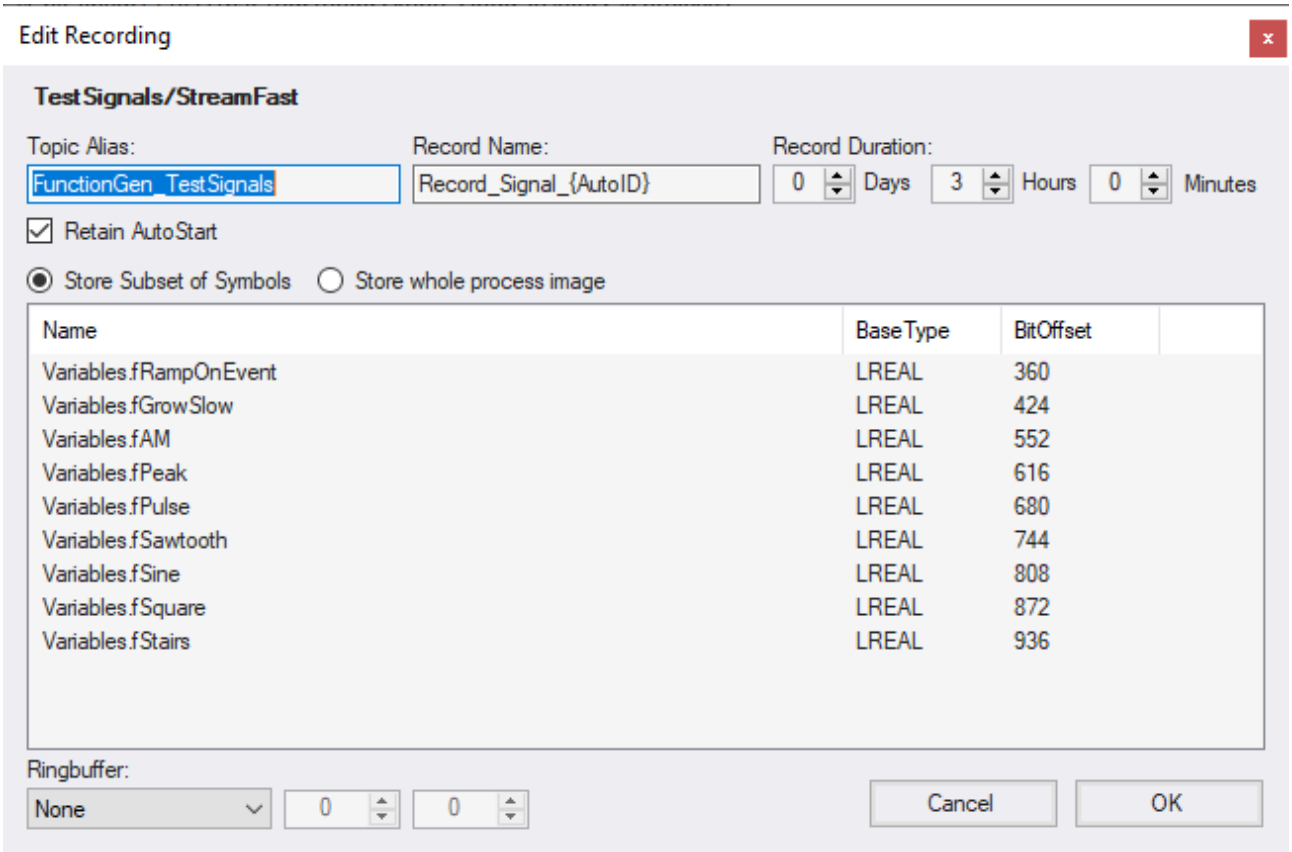
Once a connection to the broker could be established, all Analytics Storage Providers connected to it will be listed.

Create recording definition

To configure the recording, select your target in the Target Browser. Click **Live Data** and select one or more variables by multiple selection and drag and drop them to the **Local Recording Definitions** node in the Recorder window.



In the recorder you can add the selected variables or the complete source process image of the variables.

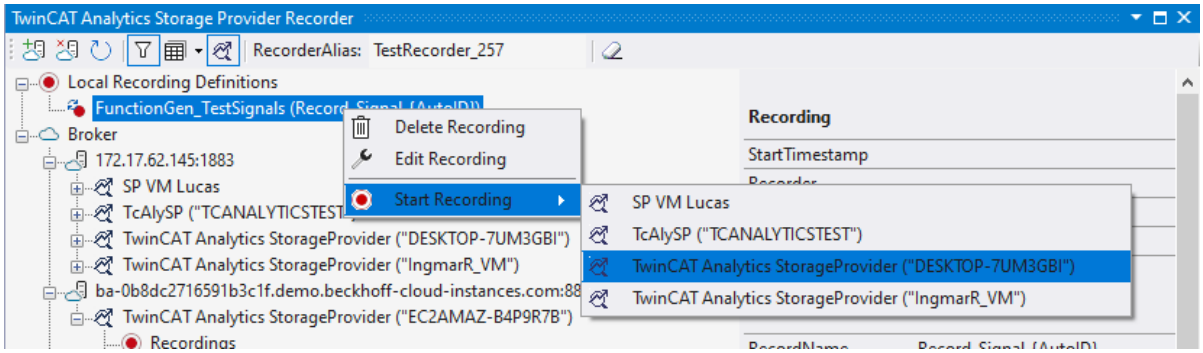


You can also configure recording names and a duration (otherwise the recording will run endlessly until it is stopped manually). A ring buffer can be set by storage or time.

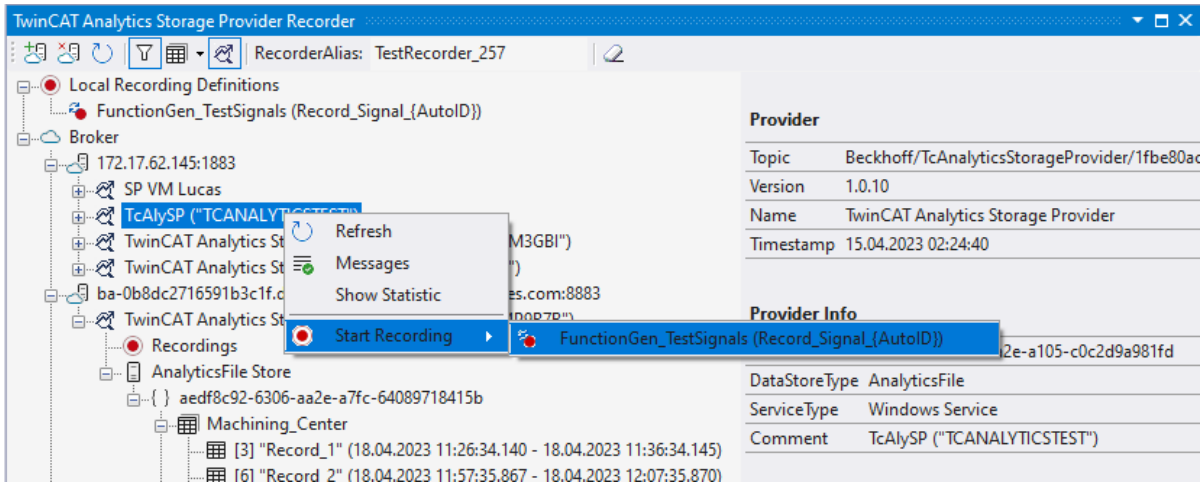
Start recording

There are three different ways to start a recording.

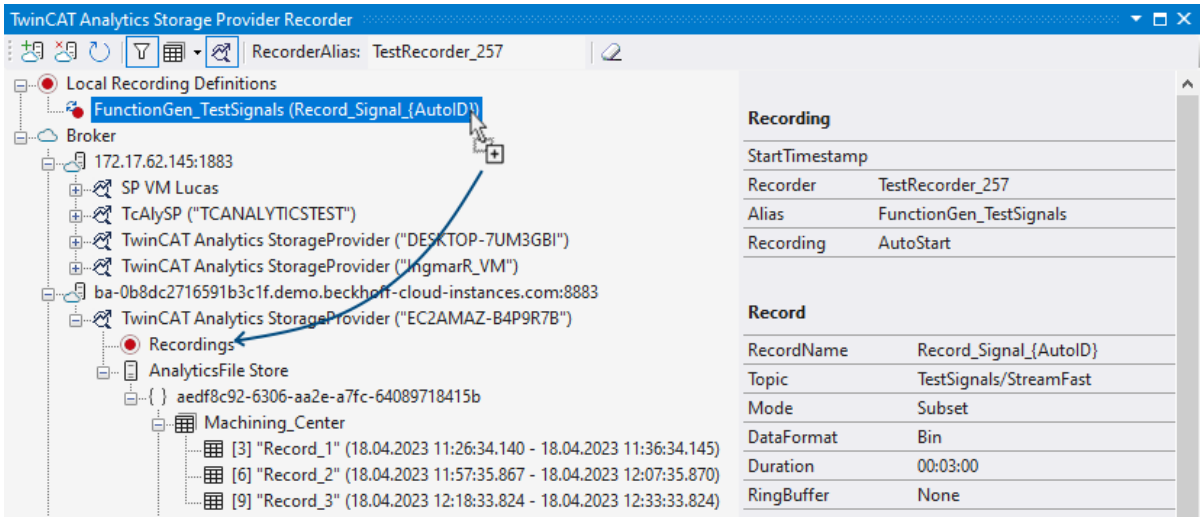
1. Via the context menu at a local recording definition node. Here you can select the desired Analytics Storage Provider, which should record the data. Only the storage providers that have access to the selected topic are displayed.



2. From the context menu on the Analytics Storage Provider node. The desired recording definition can be selected here. Only the definitions that can also be processed by the Analytics Storage Provider are displayed.

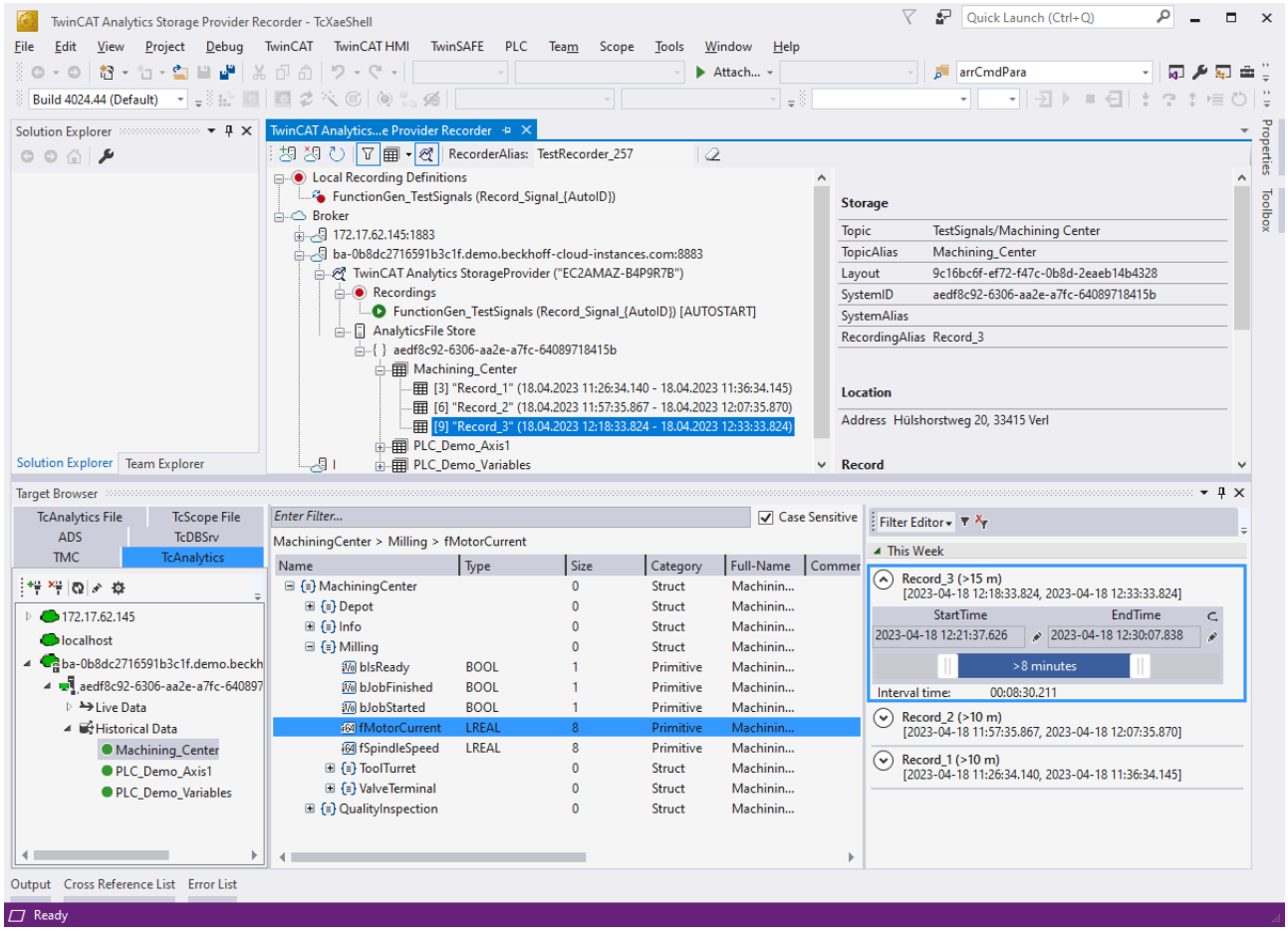


3. Drag and drop the local recording definition node. The desired recording definition can be started on the desired Analytics Storage Provider via drag-and-drop. If the Storage Provider is unable to execute the recording definition, an error message is displayed in the error list.



Use historized data

After and also during recording, you can select the historical data as input for your analysis in Target Browser. In the Target Browser, you will find a new control on the right side for the historical data. There you can select the timespan for your data.

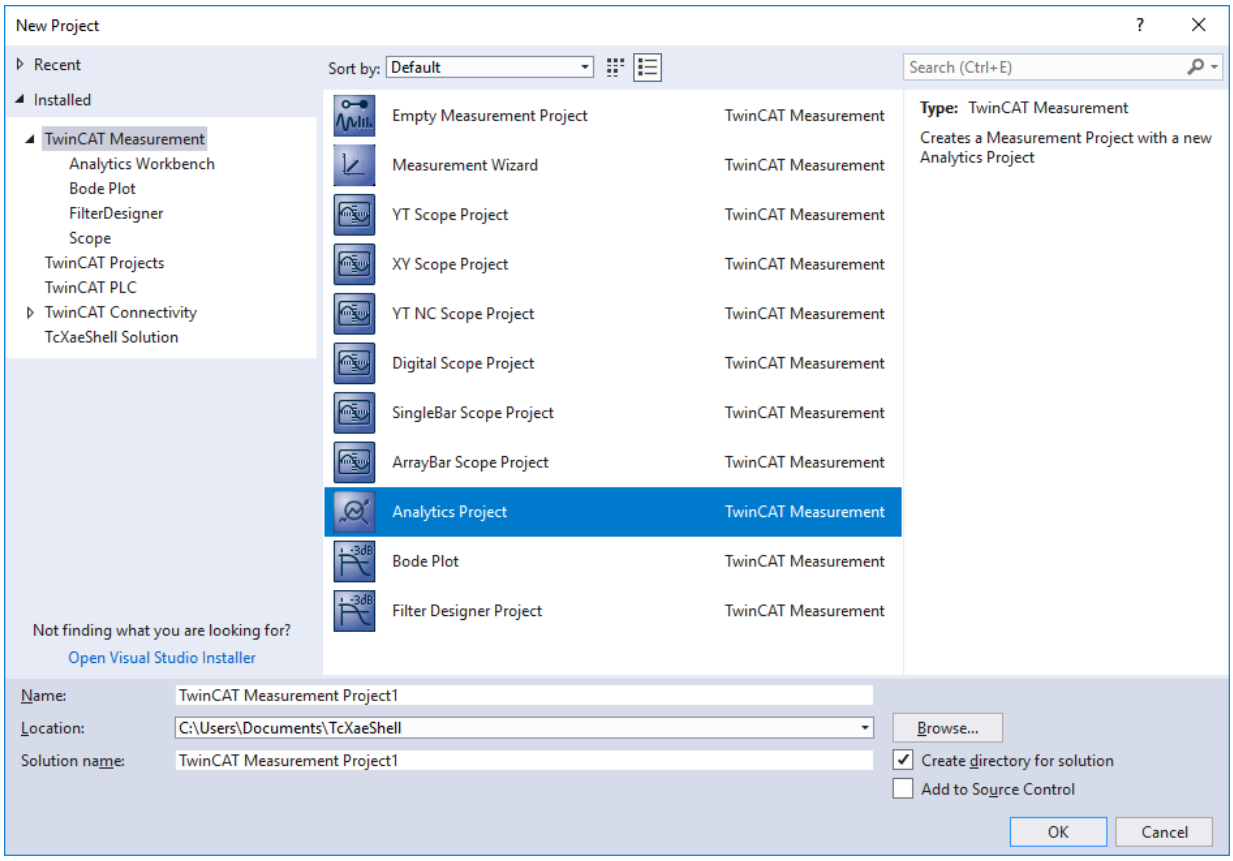


4.4 Analyse data

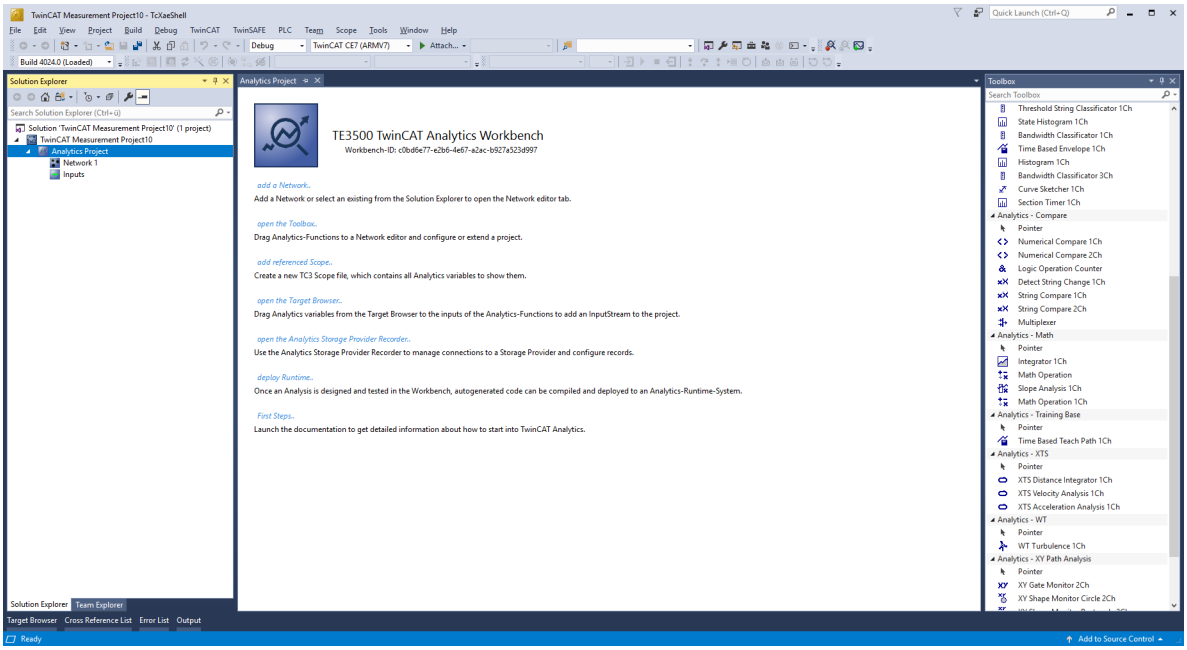
✓ Open your TwinCAT Engineering environment to start the data analysis.

1. Open **Visual Studio® > File > New > Project…**

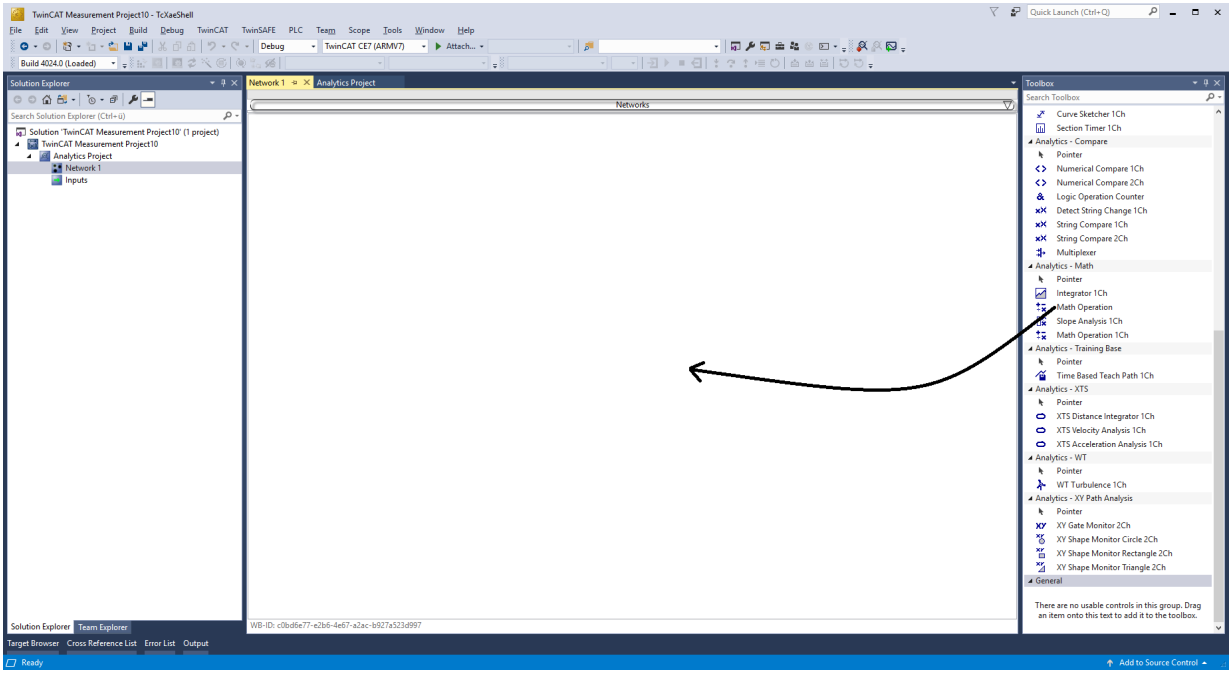
2. Select the **Analytics project template** from **TwinCAT Measurement**.



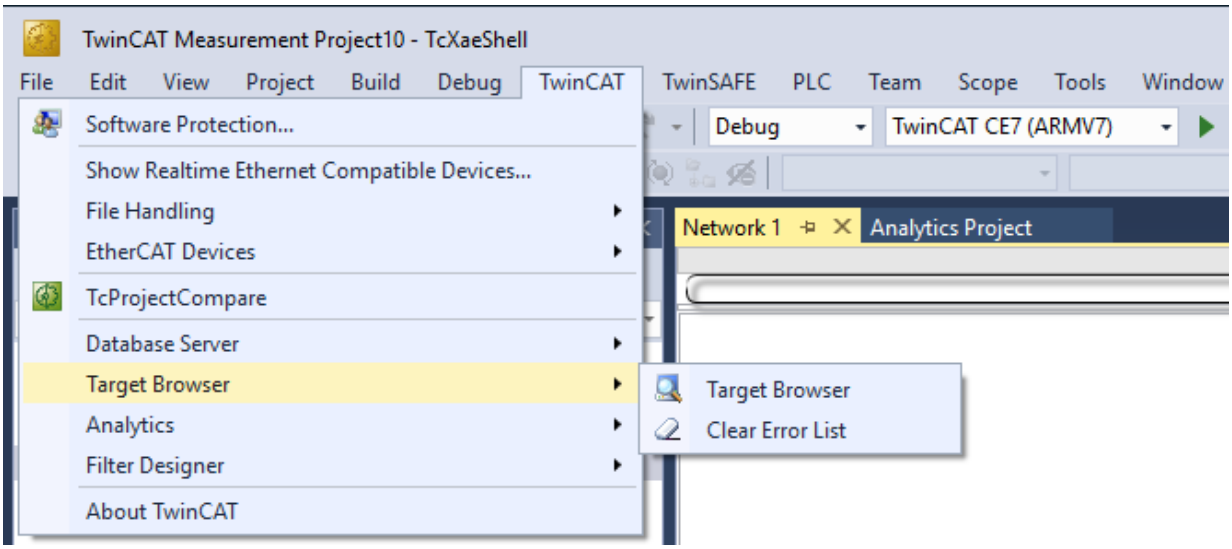
⇨ The new project is displayed in the Solution Explorer. After clicking the **Analytics Project** tree node element a start window opens where you can select your first action. From here you can add a network, open the **Toolbox**, open the **Target Browser** or open the **Analytics Storage Provider Recorder**. In the following steps you will perform all these actions.



3. It makes sense to open the **Toolbox** of Visual Studio® first. There you will find all the algorithms supported by TwinCAT Analytics. Algorithms need to be grouped and organized into networks. Right-click **Analytics Project** to add a new network, or add a network using the start page. The first network is always generated by default.

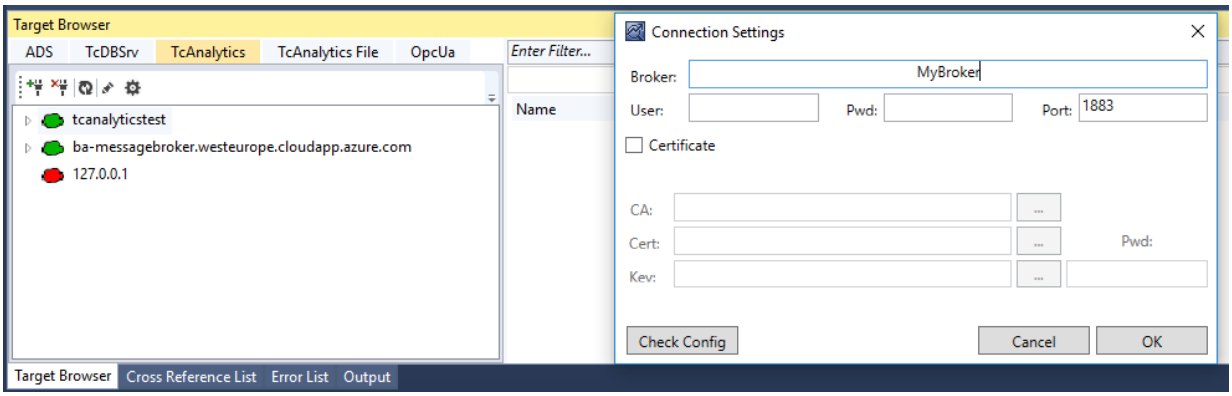


4. When you click on the network, an editor opens. Now you can drag and drop the desired algorithm into the editor interface.
5. After selecting the algorithm, you need to connect input variables to the modules (algorithm). To do this, open the **Target Browser**.
   TwinCAT > Target Browser > Target Browser

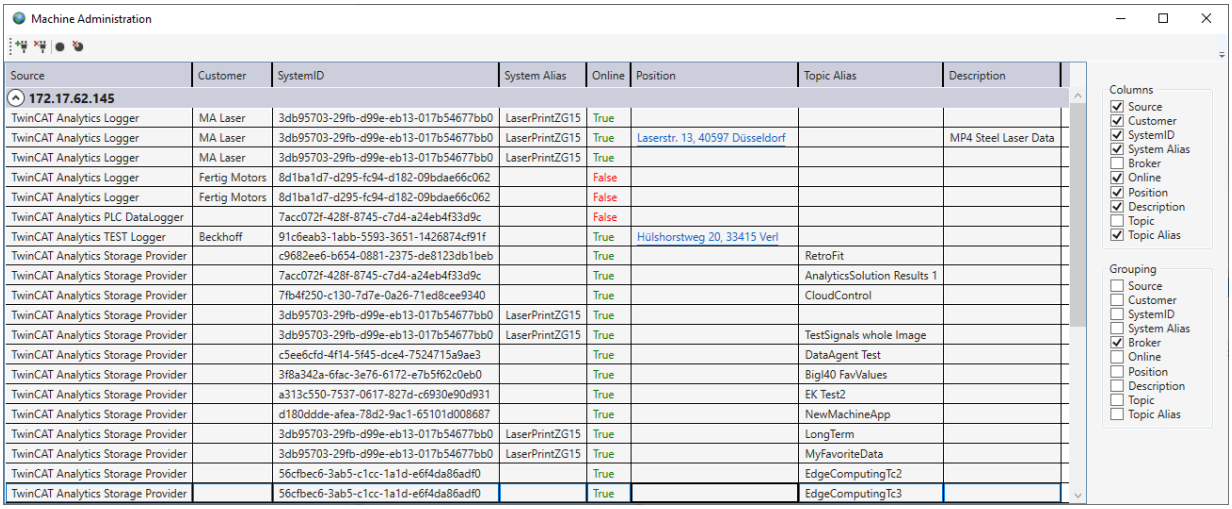


6. Now select the **TcAnalytics** or **TcAnalyticsFile** tab in the Target Browser. Continue with the tab **TcAnalytics** (MQTT).

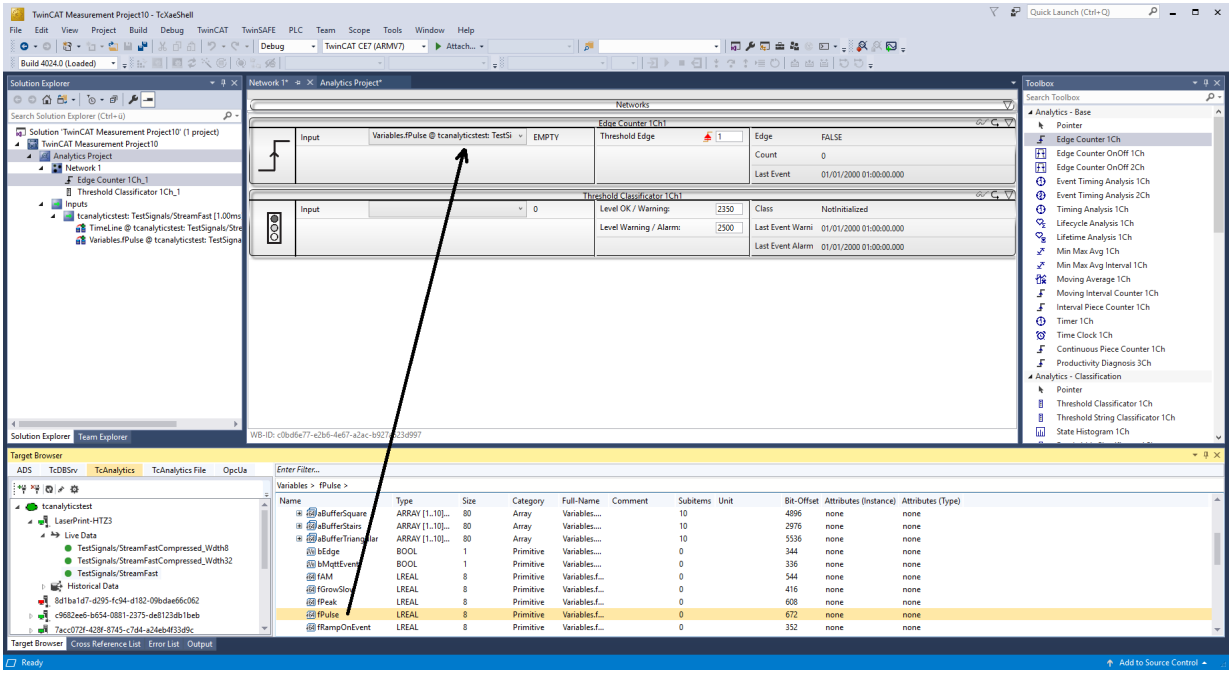
7. Click the icon highlighted in green in the toolbar of this Analytics extension. A window opens in which you can specify the connectivity data of your message broker.



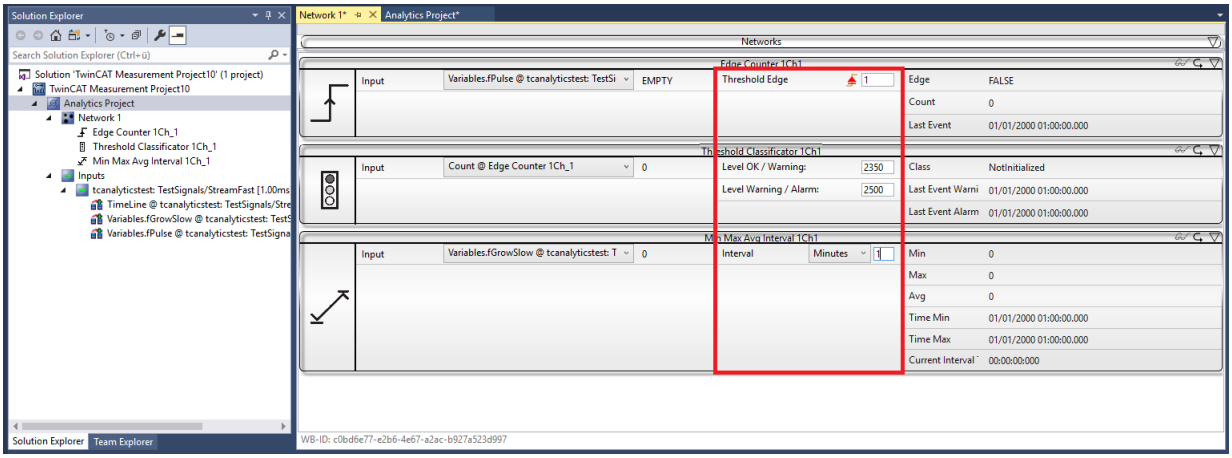
8. Select your MQTT Analytics client (TwinCAT Analytics Logger, TwinCAT IoT Data Agent or Beckhoff EK9160). There is a unique ID for each control. This ID is displayed in the Target Browser.
9. Clicking on the **gear icon**, you will get to the Machine Administration page. Here you can assign a system alias name that will be displayed in the Target Browser instead of the ID.



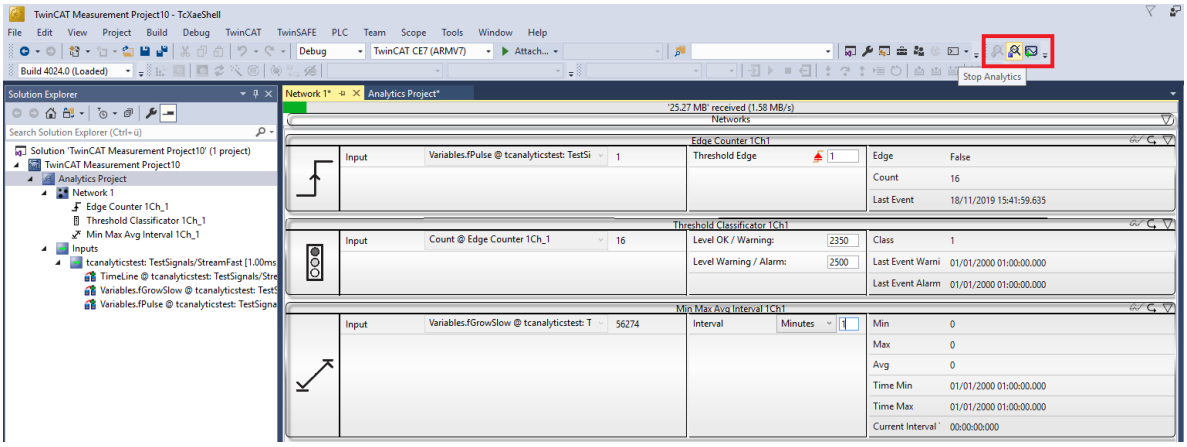
10. In the next step, you can choose between live data and historical data for each MQTT Analytics client. In this case, the historical data is provided by the TwinCAT Analytics Storage Provider.



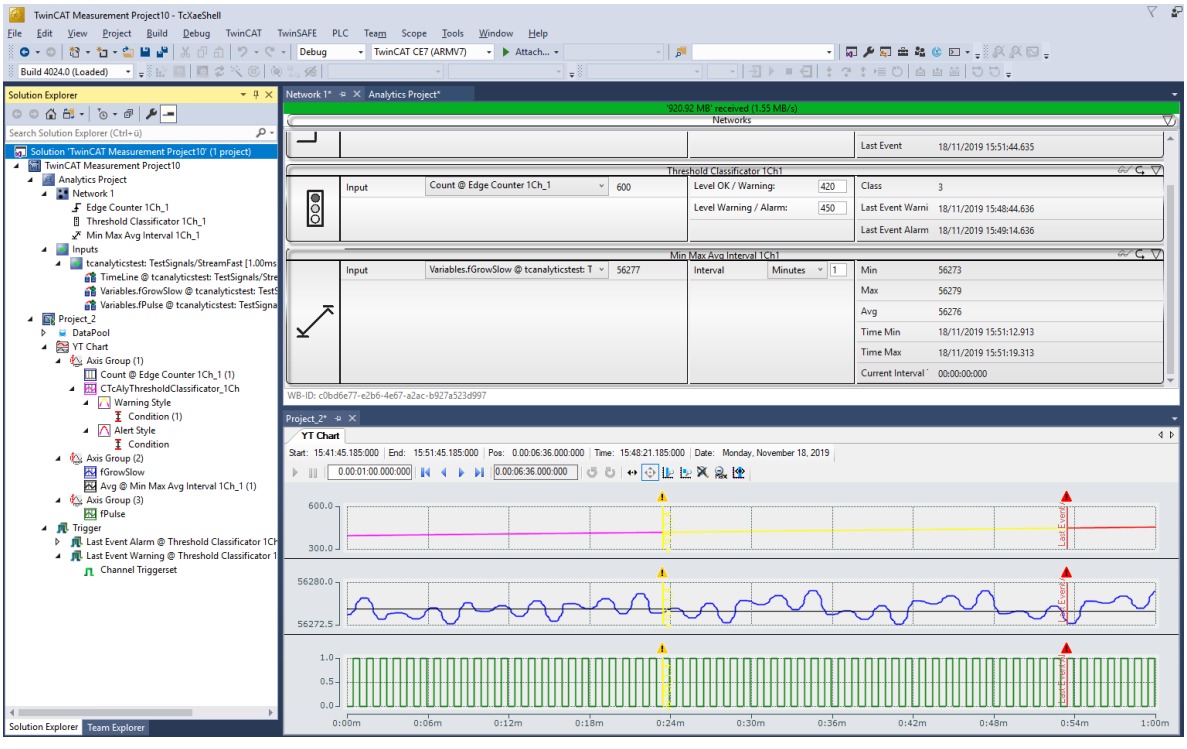
11. You can drag and drop the variables into the inputs of the specific algorithm. In most algorithms, conditions such as thresholds, time intervals, logical operators etc. can be specified. These settings are made in the middle of each module.



⇨ Finally, your first Analytics Project is complete. To start the analysis, click **Start Analytics**. To stop the analysis, click **Stop Analytics**.



⇨ Before starting the analysis or during runtime, you can click the **Add Reference Scope** button. This will automatically create a Scope configuration that matches your Analytics project.



⇨ The analysis results can be displayed in the Scope View graphs using drag-and-drop. For example, a mean value can be displayed as a new channel in the view. Timestamps as markers on the X-axes show significant values.

5 Technical introduction

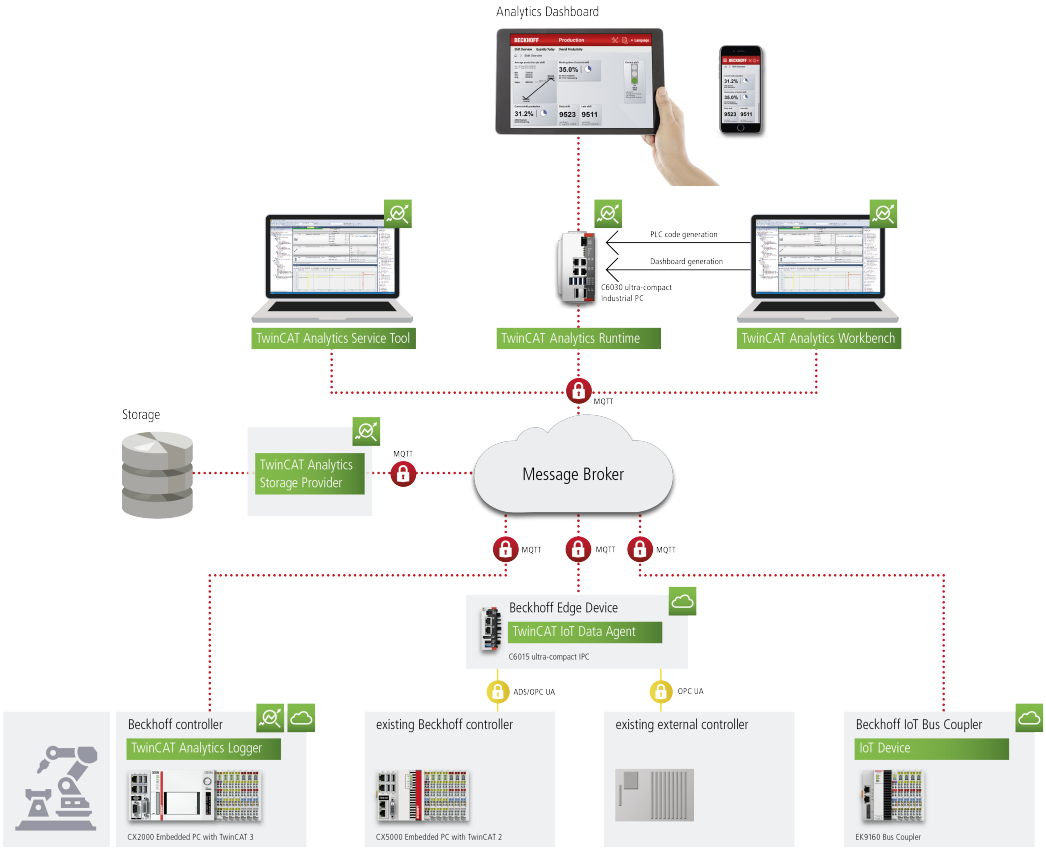
5.1 Basic concept

The following figure shows the basic concept of TwinCAT Analytics from the data source to the Analytics Dashboard based on TwinCAT 3 HMI. The communication in an Analytics scene is realized by the IoT communication protocol MQTT.

Data sources:

Currently there are three different data sources for TwinCAT Analytics. All these sources can communicate with the specific binary data format of TwinCAT Analytics. This format is necessary to achieve high performance.

- TwinCAT 3 control with TF3500 TwinCAT Analytics Logger
- TwinCAT 2, TwinCAT 3 and external control together with a gateway of the TF6720 TwinCAT IoT Data Agent
- All EK9160 IoT Bus Coupler



Storage:

With TwinCAT Analytics it is possible to analyze live and historical data. The TwinCAT Analytics Storage Provider is the interface between native MQTT Message Broker to different stores. As storage TwinCAT Analytics is supporting an Azure Blob store and a Microsoft SQL database. The configuration of the stores is done automatically by the Storage Provider. Thus, it is not necessary to use classic SQL commandos to implement the communication. The user also does not need to setup a special table structure.

Analysis:

For service technicians and machine commissioning

The TE3520 TwinCAT Analytics Service Tool is the perfect tool for experts who like to analyze TwinCAT Analytics data sources. It is integrated into the Microsoft Visual Studio®. The user is able to make his analytics configuration in a graphical configurator choosing from a wide pool of different algorithms. A parallel interaction with the Scope View is also possible. The user is able to find significant values easily by drag and drop from the configurator into the data stream of our Scope View.

For continues 24/7 machine analysis

The TE3500 TwinCAT Analytics Workbench has the same functionality as the Service Tool. In addition, it is possible to automatically generate a PLC code with associated HMI dashboard based on the realized analytics configuration in the configurator. The PLC code is ready to use, so you can start data analysis immediately as in the configurator. But now for 24 hours 7 days per week if necessary. The automatically generated code can be downloaded into the TF3550 TwinCAT Analytics Runtime. Alternatively, a download of the pure PLC project, when created without HMI dashboard, into the Analytics Runtime Base TF3551 (without HMI Server) is possible. Both runtime products are essentially license bundles and can run on a classic IPC or Embedded PC, but also in a virtual machine.

Products:

We have different single products in the TwinCAT Analytics Workflow. See therefore the following list with all products.

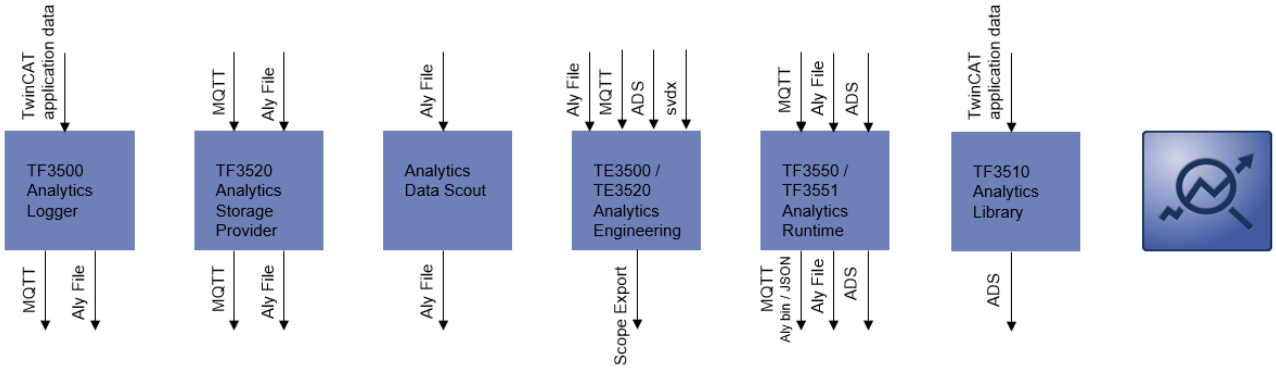
Product number	Product name
TE3500	Analytics Workbench
TE3520	Analytics Service Tool
TF3500	Analytics Logger
TF3510	Analytics Library
TF3520	Analytics Storage Provider
TF3550	Analytics Runtime - including HMI Server and Client Pack
TF3551	Analytics Runtime Base - without HMI
TF3560	Analytics Controller Pack 4
TF3561	Analytics Controller Pack 8
TF3562	Analytics Controller Pack 16
TF3563	Analytics Controller Pack 32
TF3564	Analytics Controller Pack 64
TF3565	Analytics Controller Pack 128
TF6720	IoT Data Agent
EK9160	IoT Coupler

The TwinCAT Analytics Service Tool can be used as a kind of Scope ++ via an ADS channel. This is automatically the most sensible minimum configuration from the basic concept shown. The Analytics Logger can also be used to decouple data collection and analysis. It can store data locally on the machine computer. The data can be evaluated via the service tool. If you want to organize data storage centrally rather than decentrally, you can use the Analytics Storage Provider via MQTT. The data sources available here are TF3500/TF6720/EK9160. Also possible is just to use the TF3510 Analytics Library in a TwinCAT system.

6 Configuration

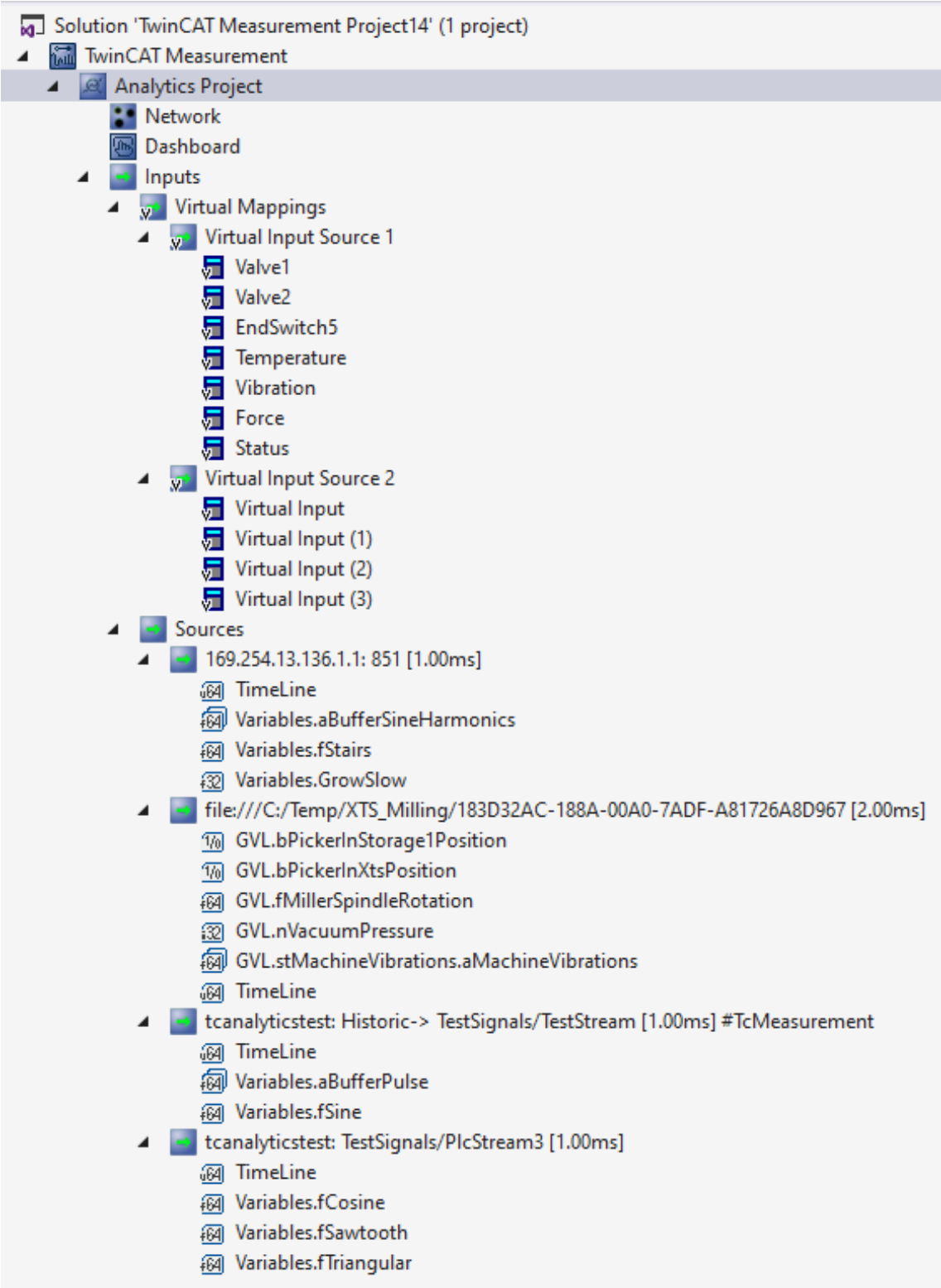
6.1 Data Source

There are many data sources for the TwinCAT Analytics products. Likewise, there are many output formats, thus ensuring a seamless transition from product to product and from tool to tool. The following graphic is intended to provide a general overview.



The most universal are the Analytics binary formats in the form of the Analytics File and an MQTT stream.

In Engineering itself, you can see the data sources in the Solution Explorer in the Inputs area divided into Virtual Input Source and Source. The Source embodies the actual data source as shown in the diagram above. For the engineering products TE3500 and TE3520 these are MQTT as live and historical stream, Analytics File and ADS.



The Virtual Input Source is an abstract mapping of the actual source. The Virtual Input Source is used for linking at the inputs of the algorithms. It is very convenient that through this mapping the actual data source can be exchanged very quickly and conveniently without having to re-link the algorithms.

Configurations for different sources can be created at the Virtual Input Sources. A configuration contains the mapping of the individual symbols of a source with a virtual input. It is also possible to create different configurations for a source, where the mapping of the symbols differs with a Virtual Input.

When adding sources and replacing them, you can optionally use the Source Wizard, which is explained on the following pages.

6.1.1 Wizard

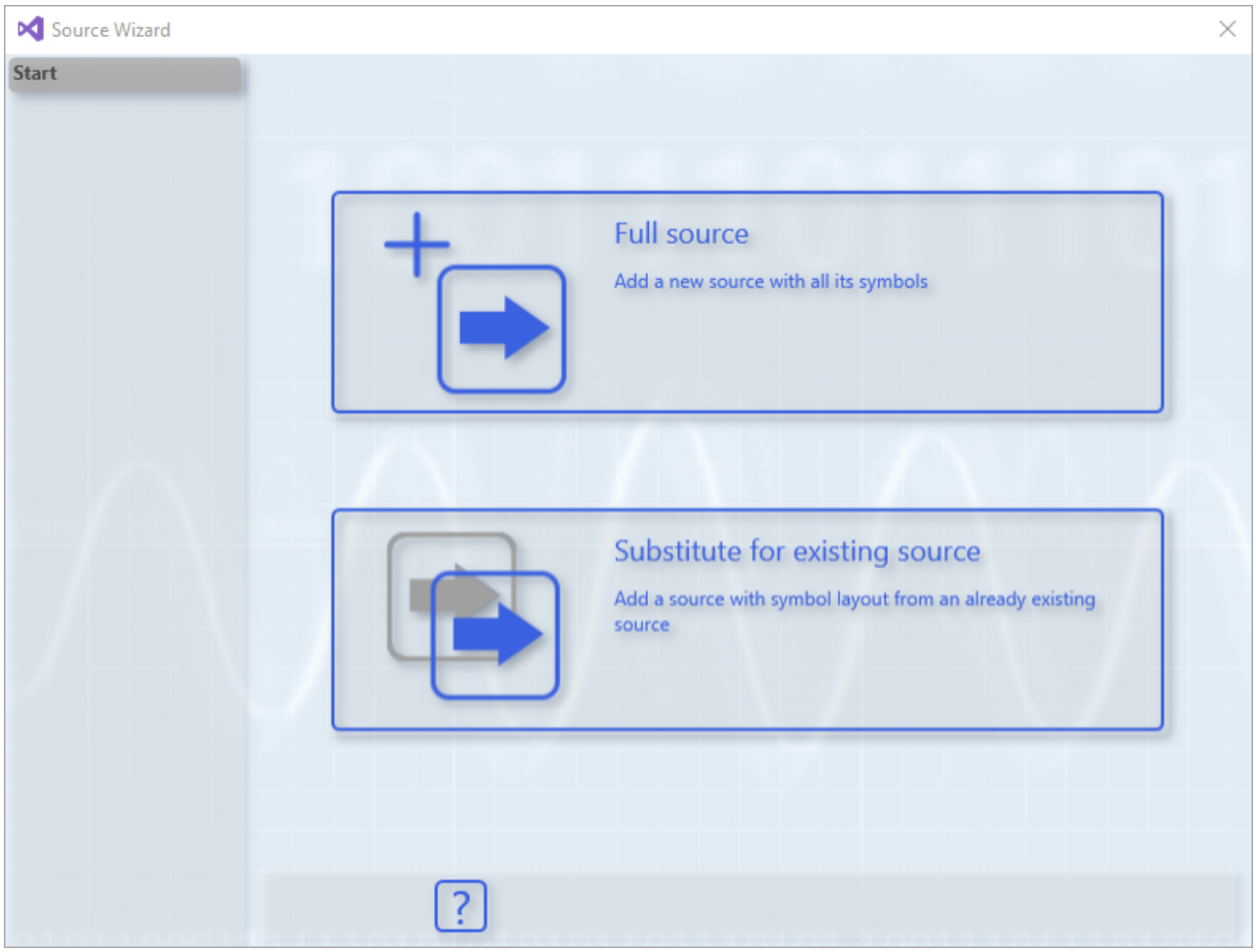
The Source Wizard can be used to add new data sources. This window provides the option to add to entire data sources or find a replacement for an existing data source.

The TwinCAT Target Browser must be installed in order to use the Source Wizard.

Pages:

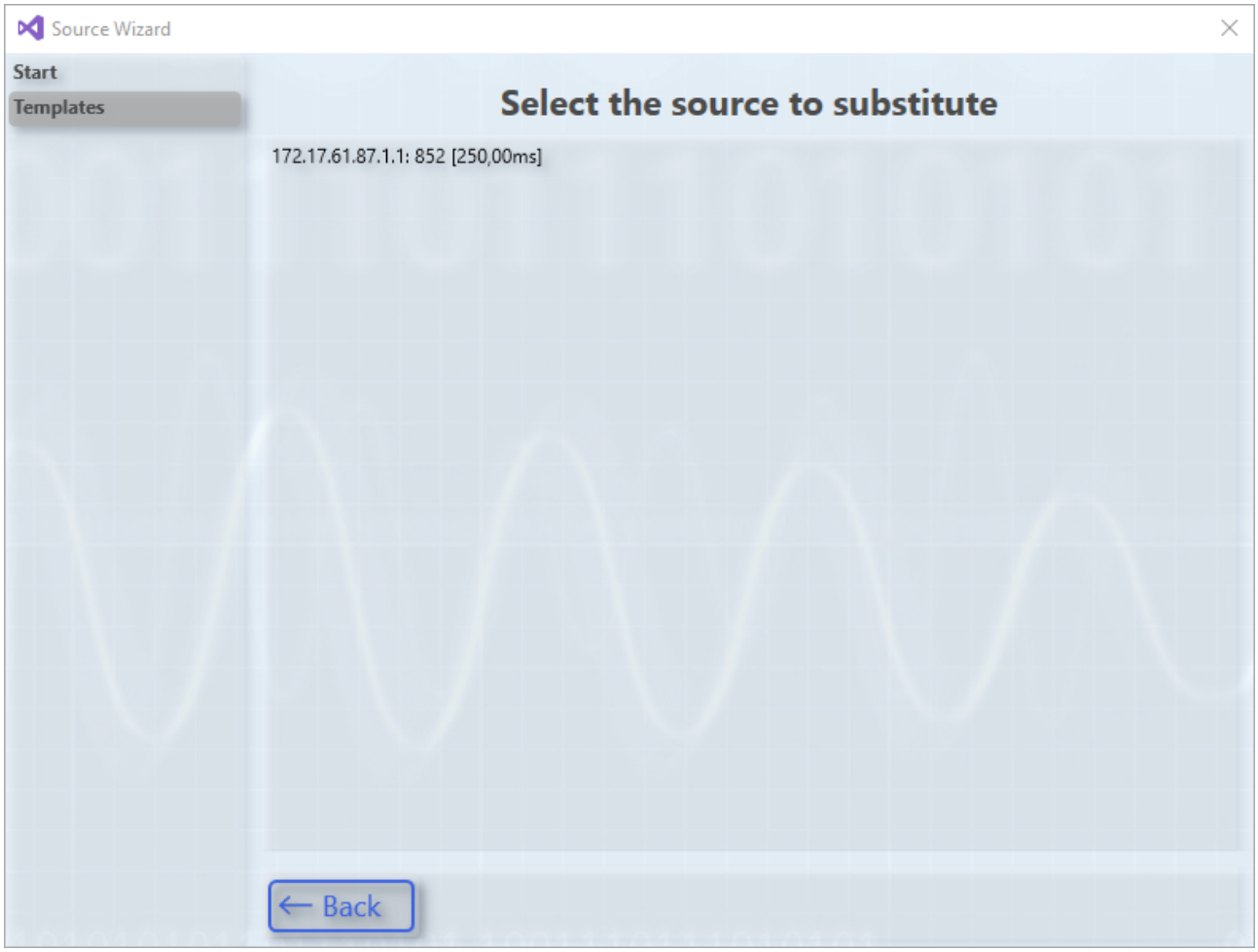
The Source Wizard is divided into several pages. The pages already shown are listed on the left-hand side. Previously opened pages can be displayed again via this list or the **Back** and **Next** buttons.

Start



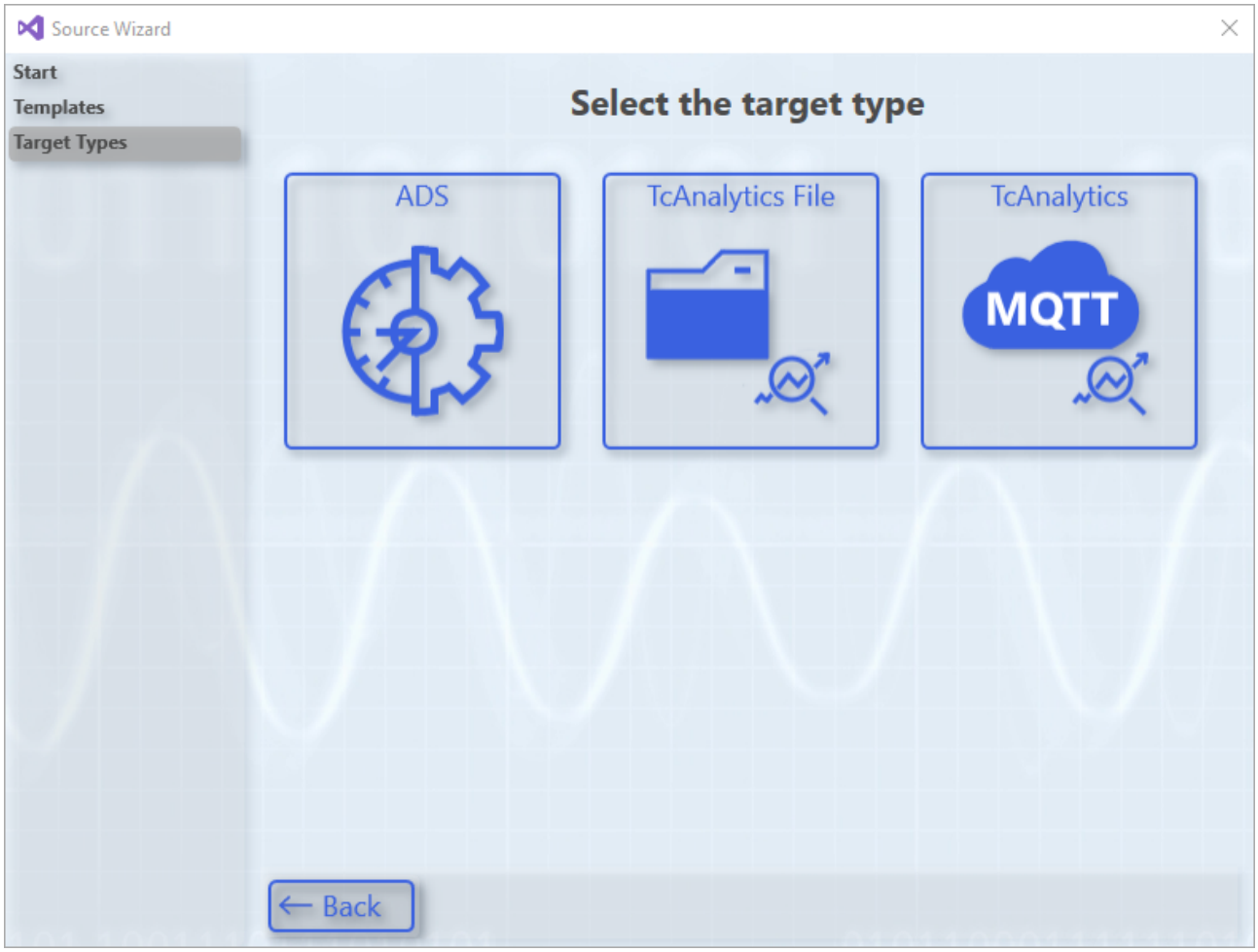
On this page, you can choose whether all symbols of a source are to be added or only symbols that were sought on the basis of an existing source. The option to replace a source can only be selected if a source already exists in the project.

Templates



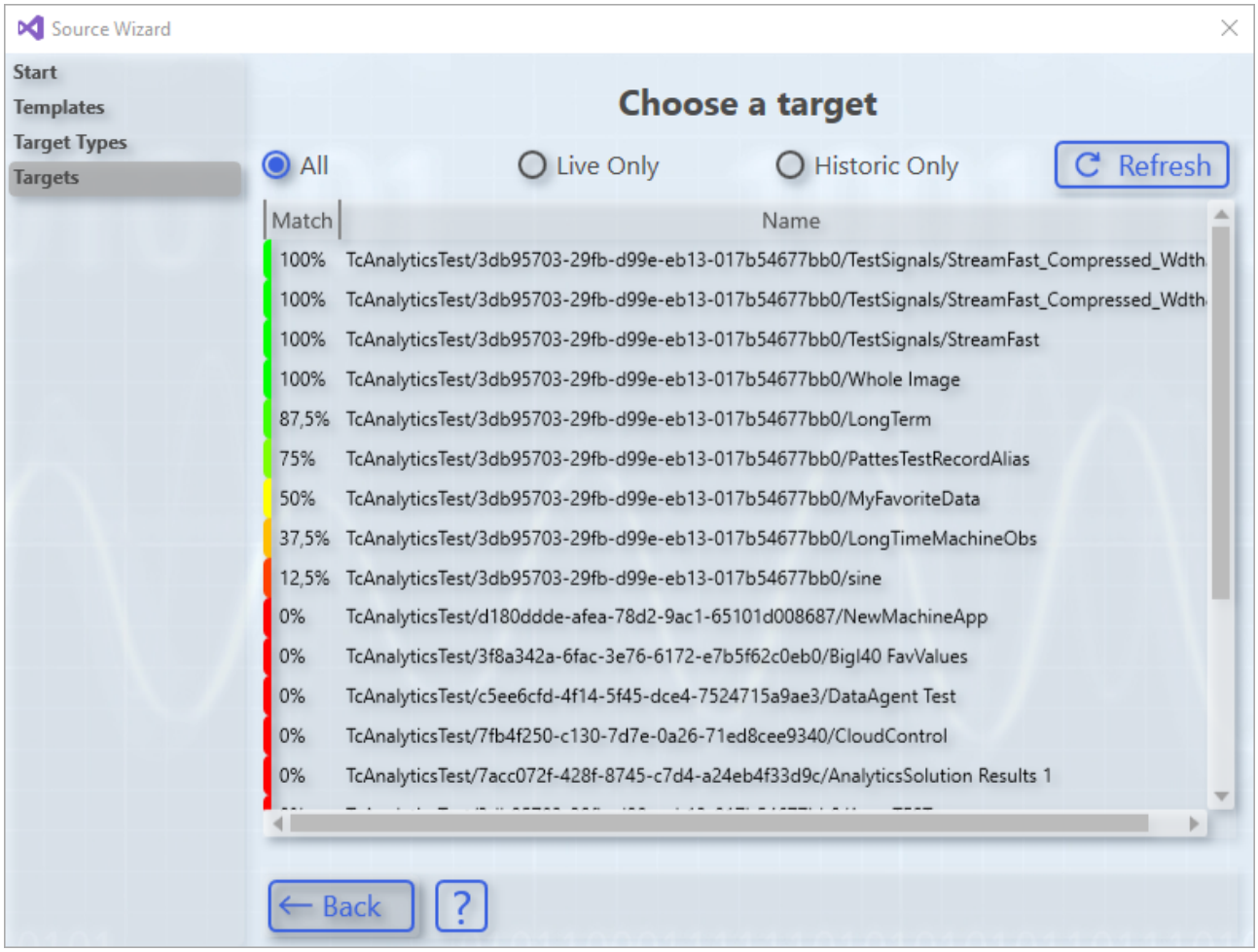
This page is only displayed if the Replace option has been selected on the Start page. Already existing sources of the project are displayed here. To continue, a source must be selected that will then serve as a template for the new one.

Source types



This page displays different source types, such as ADS or Analytics File. The number of types is different to the extensions installed in the Target Browser that are compatible with the Source Wizard.

Sources



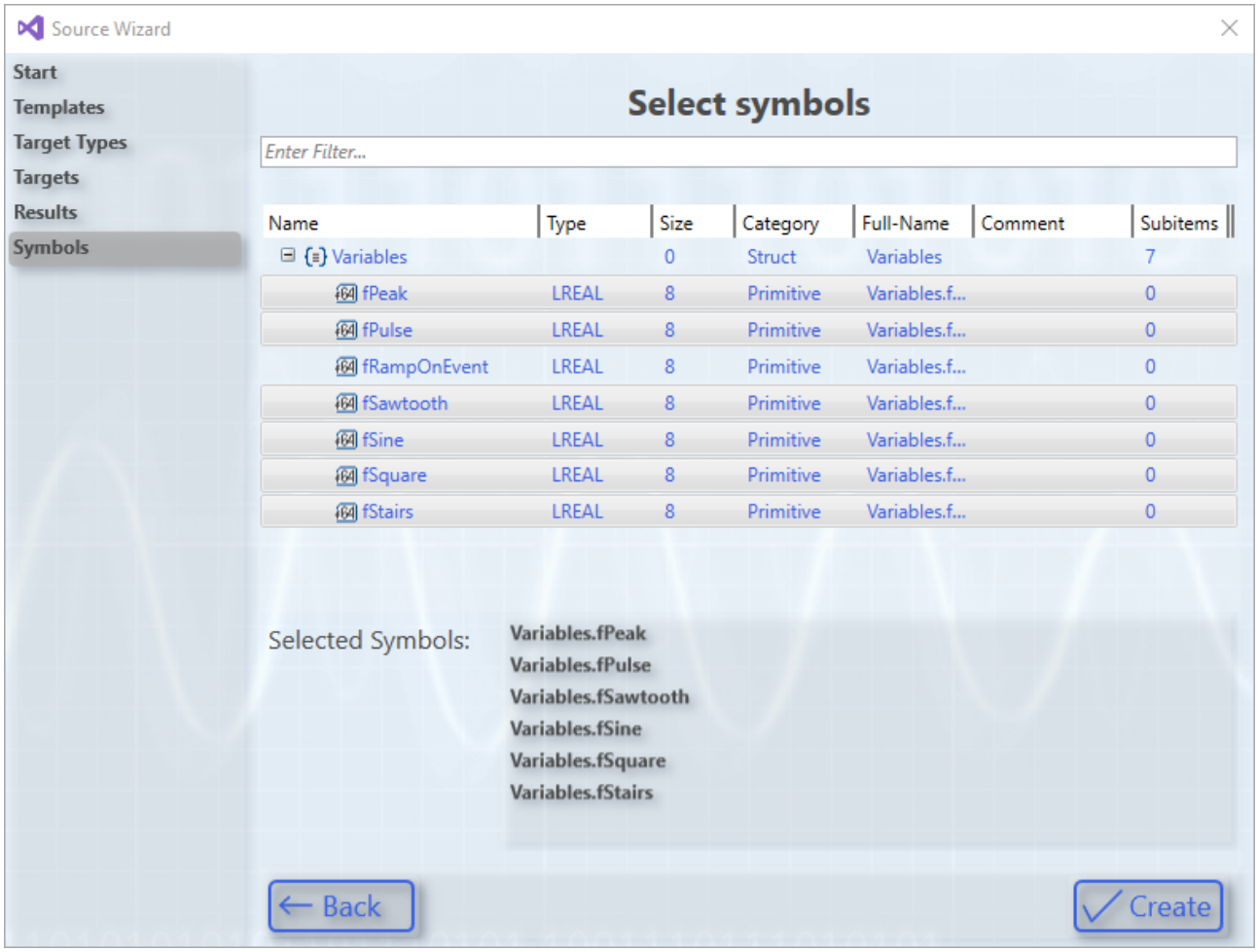
After selection of a type, its available sources are displayed. In order to find additional sources, new sources must be stored in the Target Browser. If the Replace option is selected on the Start page, the correlation with the template selected on the templates page is displayed in addition to the name of the sources. The correlation indicates how many of the symbols from the template have been found. In addition, the list can be filtered so that only live or only historical sources are displayed, unless there are only historical or only live sources.

Results

Result page when using Replace

The Result page lists all symbols that would be added. It also displays the icons of the templates that were not found (this list is omitted if the option to add whole sources has been selected on the Start page). Click **Create** to close the wizard. All symbols in the "Matches" list are added to the project. If the selected symbols do not meet the requirements, click "Select Symbols", which will open the Symbols page.

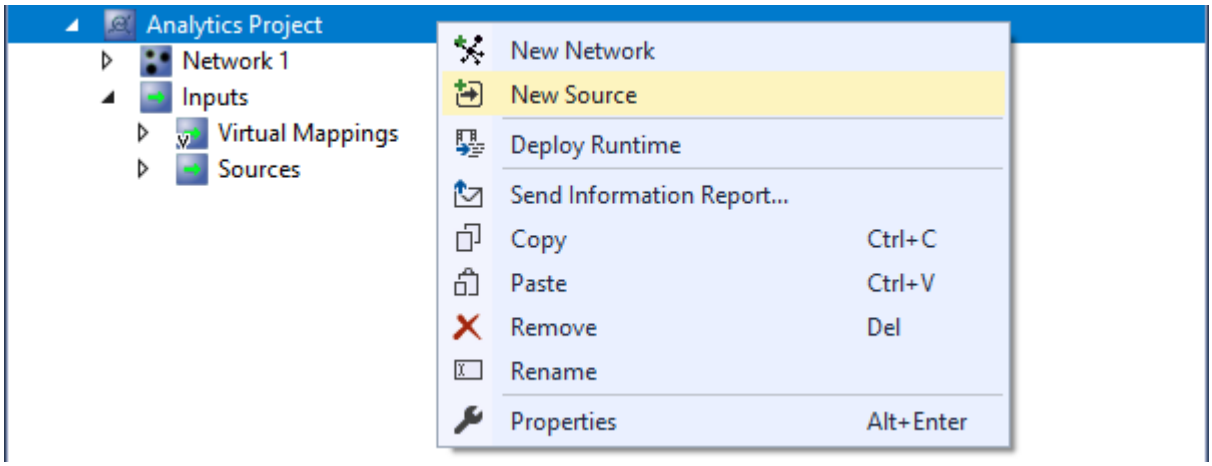
Symbols



If this page is opened, all symbols of the selected source are listed, with the symbols that are also listed as found on the Results page already highlighted. In addition, all currently highlighted symbols are listed below the Tree symbol. Here, you can now select the desired symbols or deselect unnecessary symbols. Clicking the “Create” button terminates the wizard and adds the selected symbols to the project.

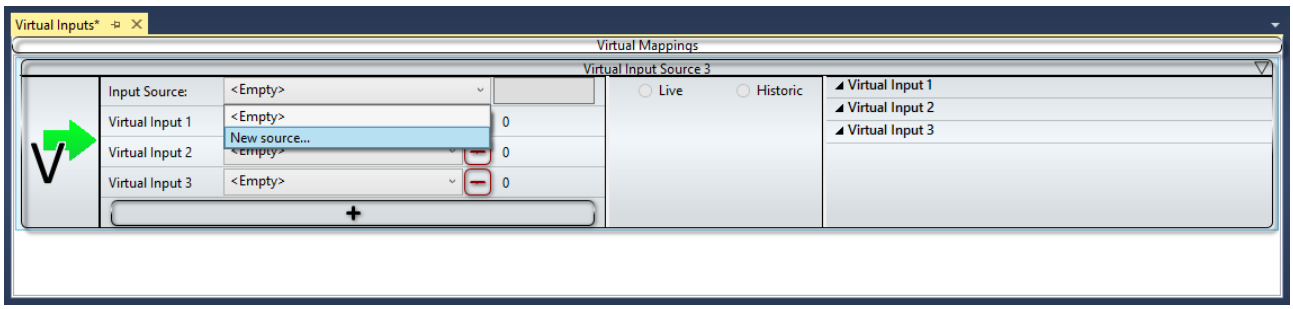
Call:

Context menu



Right-clicking the Analytics Project or a node below it in Solution Explorer opens the context menu, where the entry **New source** can be found. If this is selected, the wizard starts. Upon successful completion, the desired source is added to the project.

Virtual source



In a module for a virtual source, the entry **New source...** can be found in the **Input Source** combo box. If this is selected, the wizard starts. This way to open the wizard does not start at the Start page, but on the Templates page, where the currently selected template is the source of the virtual source. Upon successful completion, the new source is selected in the virtual source and the symbols can be assigned to the virtual inputs in selection mode.

Query

If a symbol is added to a function by drag and drop and this symbol does not belong to any source in the project, then you can choose between three options. A new virtual source can be added and, in order to use the source at the same time as the other sources, a source from an already existing virtual source can be changed or the source wizard can be opened. When the wizard is opened, there is a different action, depending on the selection on the Start page, which is performed after the wizard is completed. If an entire source has been added, a virtual source is created in addition to the source so that it can be used with others at the same time. If Replace has been selected, the new source is added and selected in a virtual source. The symbols can then be assigned to the virtual inputs in selection mode.

6.2 Data Scout

The TwinCAT Analytics Data Scout is a very important tool in the TwinCAT Analytics processing flow. It is used for data viewing and can manipulate data or its recordings. For example, different recordings can be merged into one large recording or unneeded data sections can be removed. The artificially created new image can be used as a data source for the actual analysis. The Data Scout is only responsible for processing data that has already been recorded. The input format is Analytics File, which is also the output format. The engineering tool is available with TE3500 Analytics Workbench and TE3520 Analytics Service Tool.

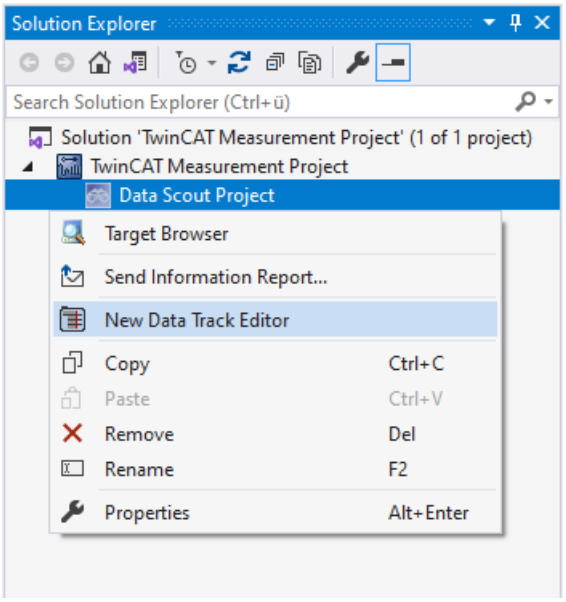
The Data Scout project can be created in the Engineering environment below the Measurement project and is explained in more detail on the following pages.

6.2.1 Data Track Editor

The Data Track Editor is part of the Analytics Data Scout. It can be used to view data, merge data from different sources and to cut out parts of the data or whole symbols.

Create

A new track editor can be added to an Analytics Data Scout by clicking New Data Track Editor in the context menu of a Data Scout project.



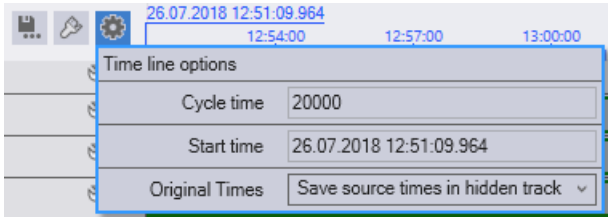
Structure

1. Tracks [▶ 46] represent the symbolism and data in the Data Track Editor.
2. The Chart [▶ 52] is used to visualize the data lying in the tracks as graphs.
3. The toolbar [▶ 53] contains functions, for example, to change the displayed area or to eliminate areas.

To export [▶ 54] the  button must be clicked.

Timeline

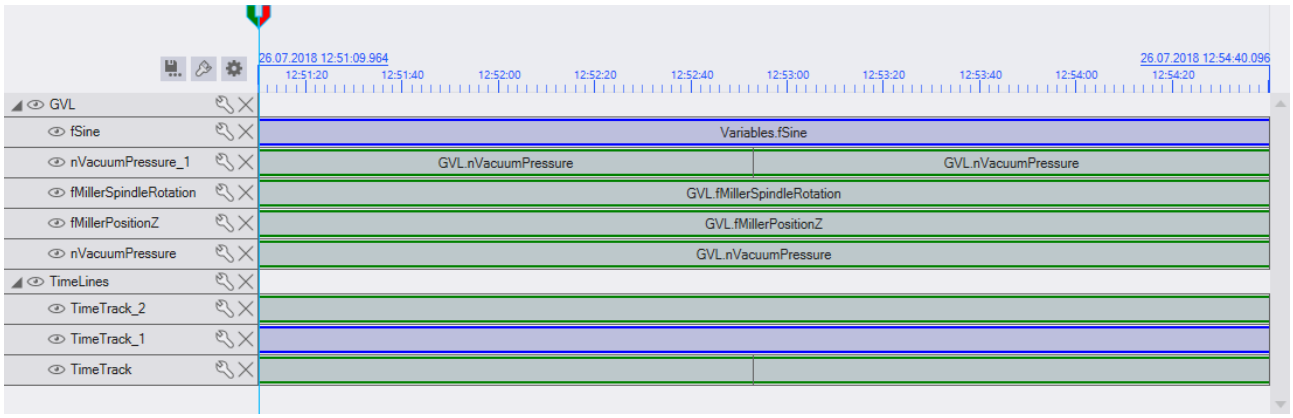
As different sources can be added in the Track Editor, new times are assigned. The start time and the cycle time can be set in the Timeline Options. These times are used in the export and in the chart. The start and cycle time of the first added source will be taken over if they have not been set yet.



The end time is calculated automatically, using the track with the fewest samples.

Time Tracks

In order not to lose the original times of the sources, Time Tracks can be generated. How these are generated can be set in the Timeline Options. There are three variants for generating Time Tracks. The Time Tracks can be generated as visible tracks, and all of them are combined in one container. The second variant is to generate the Time Tracks in the same way, but they and the container they are in are not displayed in the editor. The last variant is not to create any Time Tracks and to discard the original times when exporting.



The Time Track generation works via the data elements of the tracks. For each combination of data items there is a Time Track. Only the time range and the source are considered, but not the symbols of the data elements.

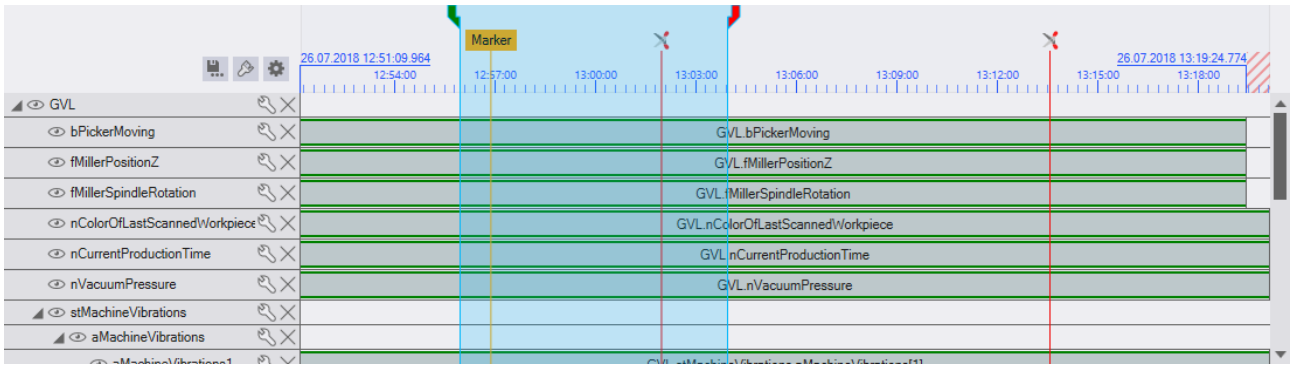
In the example display, the Time Tracks were generated visibly. Three were automatically created because three combinations were found in the tracks above. On the one hand the track with one blue data element, on the other hand a track with two green data elements and three more tracks with one green data element.

Time Tracks are structured like Single Tracks. They can contain data items, but these data items have no symbols, only a time range and a source.

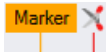
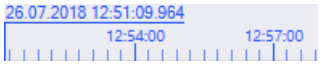
6.2.1.1 Tracks

Tracks are elements of the Data Tracks Editor, which contain information about the contained data and the structure of the symbolism.

Presentation



The area provided for tracks in the Data Track Editor has the following elements:

	This button opens a new Track Editor Export window.
	Clicking on it opens a popup where settings for the display of the tracks can be made.
	Clicking on it opens a popup where settings for the timeline and Time Tracks can be made.
	The blue area lies above the tracks and represents which data is displayed in the chart. Also, the selected area is used for some toolbar functions. The red and green areas can be moved to increase or decrease the selection. If clicked between the red and green area, the selected area can be moved. On the left and right edges, the selection can snap. This happens near marker and track ends. To ignore the snap function, the Alt key can be pressed while moving. The selection cannot be smaller than 10 values.
	Markers are displayed in the track as well as in the chart. In the tracks, markers can be renamed by clicking on the text and markers can be removed by right-clicking. The markers that have no text but scissors are special markers that can be used for elimination.
	The timeline has the start time marked at the left end. Markers then follow at even intervals, indicating the time present there.
	At the right end of the timeline is a marker for the end time. The end time will be equal to the start time plus the number of samples from the track with the fewest samples times the cycle time. If the number of samples in the tracks differs, a red dashed area is displayed behind the end time marker. This area is not exported, because there would not be data for all tracks.
Tracks Container	In the lower part the tracks are displayed, where the display of the data from the tracks starts at the start time and before that the new symbolism is displayed.

Data Tracks

Currently there are two types of tracks that can contain data. The first is the Single Track. This represents a single symbol. The second is the Multi Track. This can group multiple symbols together, keeping the presentation simple. However, in Multi Tracks e.g. the data types or names of the contained symbols cannot be customized.

Tracks that contain data are displayed in a similar way.

On the left side are the following functions:

	Here you can define whether the track should be visible as a graph in the chart or not.
Track Name	The name of the track is used in the chart for the graph and in the export as the symbol name. The name must be unique below a container and cannot contain spaces.
	Clicking on it will open a popup. This shows the settings of the track (see the Settings section for more details).
	This button removes the track and all the tracks under it

On the right side the data elements of the track are displayed. Data Tracks can contain multiple data elements that match the type of track (e.g. a Single Track cannot contain an array).

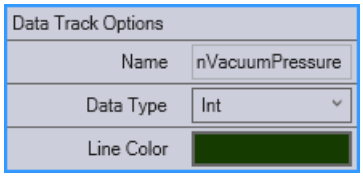
Container Tracks

Container Tracks are structured similarly to Data Tracks. However, no data elements can be in Container Tracks, but they can contain tracks. Included tracks are displayed below the Container Track and can be expanded or collapsed using the buttons. When a Container Track is made visible or invisible, all sub tracks become visible or invisible as well.

Settings

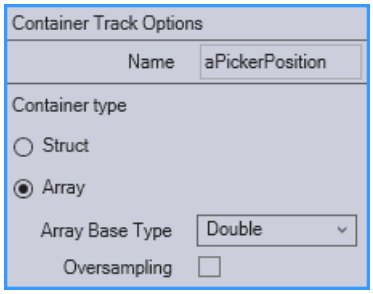
For all types of tracks there are different settings in the popup of the button.

For Single Tracks there are these:



Data type	Here you can define the data type of the Single Track. The displayed as well as the exported data will be converted to the types specified here. For string data types, another text field is displayed, which determines the length of the string. If the data type could lead to problems with the conversion, because one or more data elements of the track have a larger data type, a warning will be issued, but the values will still be converted.
Lines color	Sets the color of the graph in the chart. When you click on it, a color picker is displayed in which you can set the color.

For Container Track there are these settings:

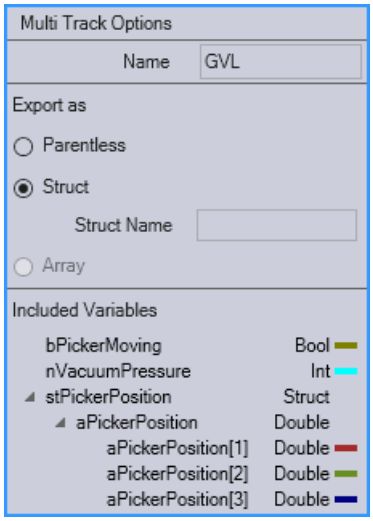


Container type	A Container Track can be a structure or an array. To be created as an array, however, all sub tracks must be compatible to form an array. If only Single Tracks are available, a Container Track can be made into an array with a primitive data type. If there are only Container Tracks that have the same structure, the Container Track can be made into an array with primitive array or structure as the base data type.

Each container type has its own properties:

Structure	Structure Name	Specifies the desired name for the data type in the export.
Array	Base data type	Here you can set the data type for the array elements. Setting the data type here changes the array base data type of Sub Container Track and the data type of Sub Single Tracks. For string data types, the length of the string can be set.
	Oversampling	This option is only available if the Container Track has only Single Tracks as sub tracks. When active, the sub tracks are not displayed as individual graphs in the chart, but the Container Track is displayed as one graph. For this purpose, the values of the sub tracks are loaded and all values at the same time are placed one after the other.

For Multi Tracks there are these settings:



Export as	Similar to the container type of the Container Track, it can be set how the Multi Track should be handled in the export. For Structure, the Multi Track is exported as a structure data type with the contained variables as elements. The desired name for the structure can be specified. The option to export as array is available only if all contained variables have the same data type. The data type can then also not be changed. The Multi Track becomes an array during export with the contained variables as array elements. With Parentless, the Multi Track is not taken into account in the export and it is as if the contained variables were on the Multi Track level.
Variables included	The symbolism present in the Multi Track is displayed here. All data elements of the Multi Track must have the same symbolism.

Data elements

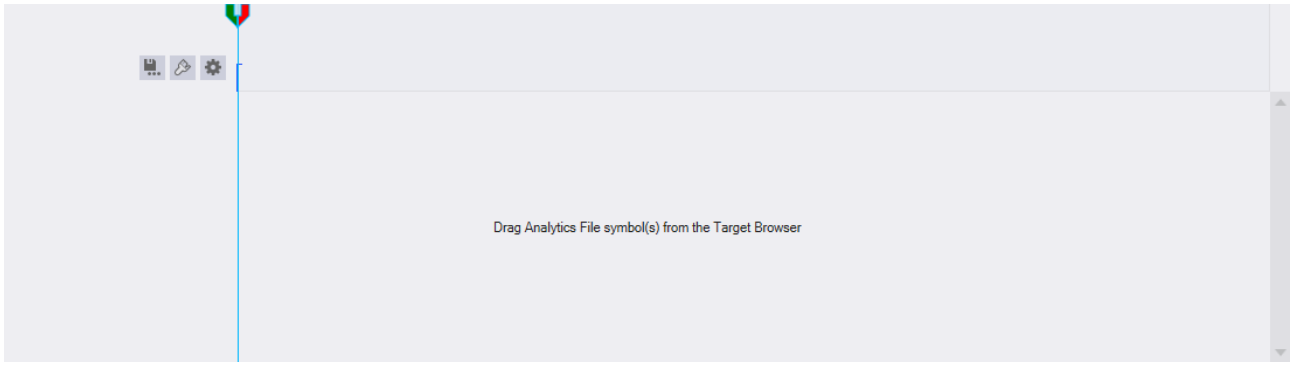
Data elements can be located in Tracks. These contain one or more variables from a source, and a time range from the source. Using the time range and the cycle time of the source, it is calculated how many samples are in the data element. The length of a data item is determined by looking at which track contains the most samples and how much space is available in total. The length then depends on the number of samples and how many pixels correspond to one sample.

A data item is represented as a block. The color of the data item depends on the source it comes from, so all data items from the same source have the same color. There is a dividing line at the end of the data element. The original name of the symbol from the source is displayed in the data element. If there are several symbols, only the number of symbols is displayed.

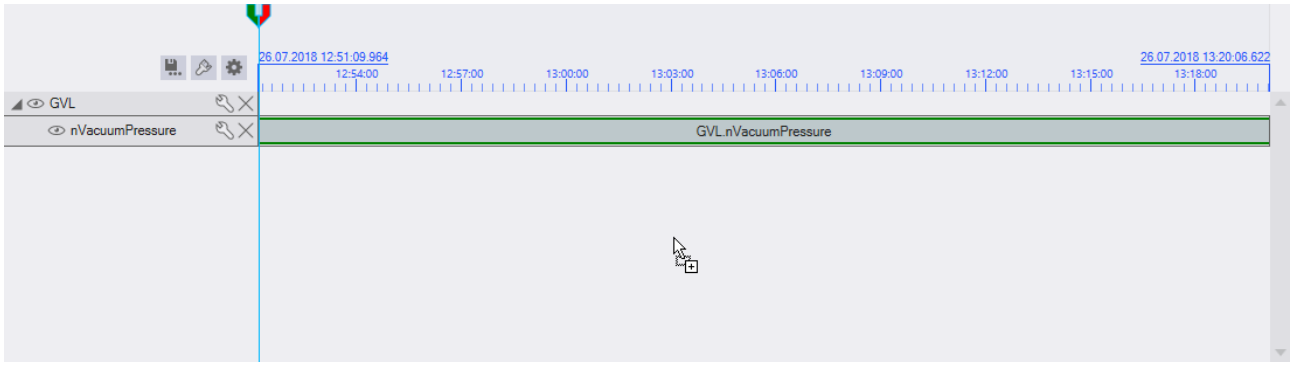
By right-clicking on a data element, the context menu can be opened. The data element can also be removed completely without using the functions from the toolbar.

Add via drag and drop

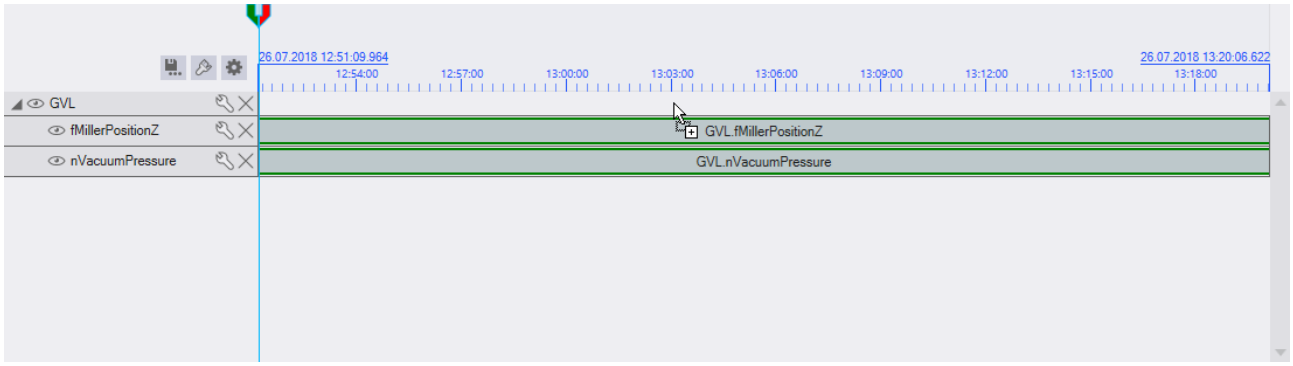
The tracks of a Data Track Editor are displayed in a separate area. This is empty at the beginning and can be filled using the Target Browser.



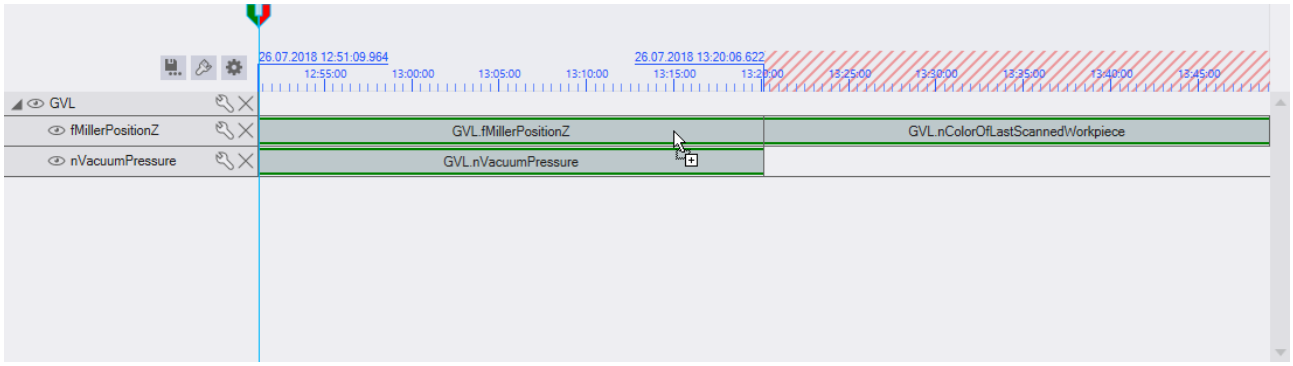
When one or more symbols are selected from the Target Browser and dragged into the empty area of the Track Editor, a new Container Track is created. To the new Container Track all selected symbols will be added as tracks.



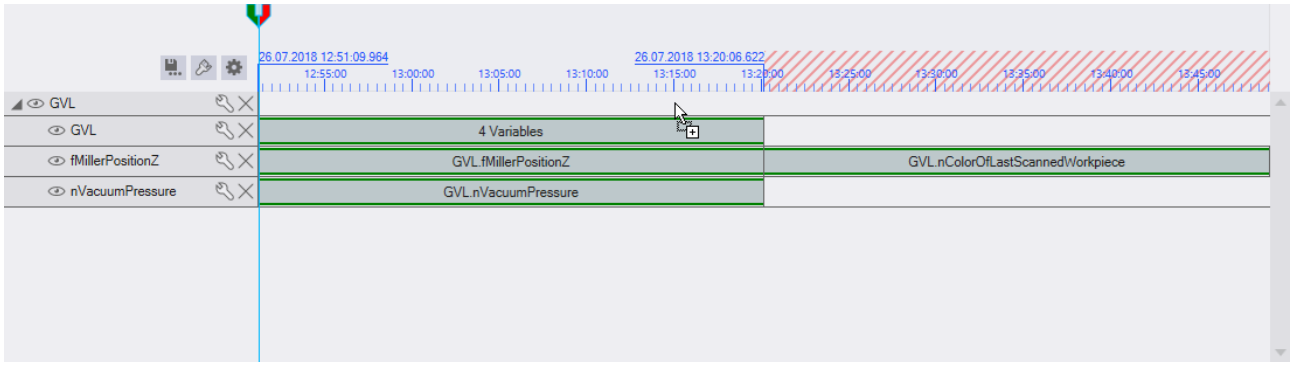
If a symbol is dragged onto the data area of a Container Track, a new track is created below this container.



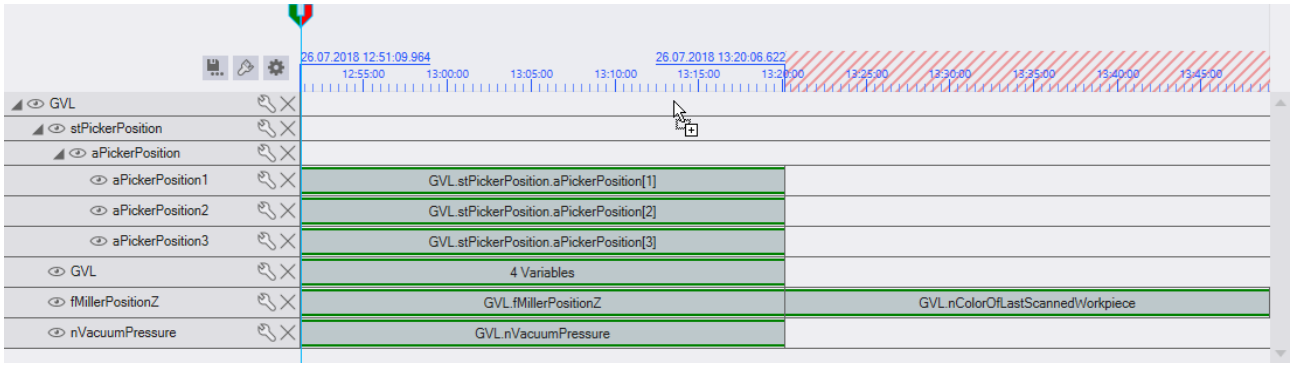
When dragging to a Data Track, a new data element can be added to that track. However, this is only possible if the symbol to be added is compatible with the track. No structure can be drawn on a track with primitive data types.



To create new Multi Tracks you can drag to a Container Track or to the empty area while the [Ctrl] key is pressed.

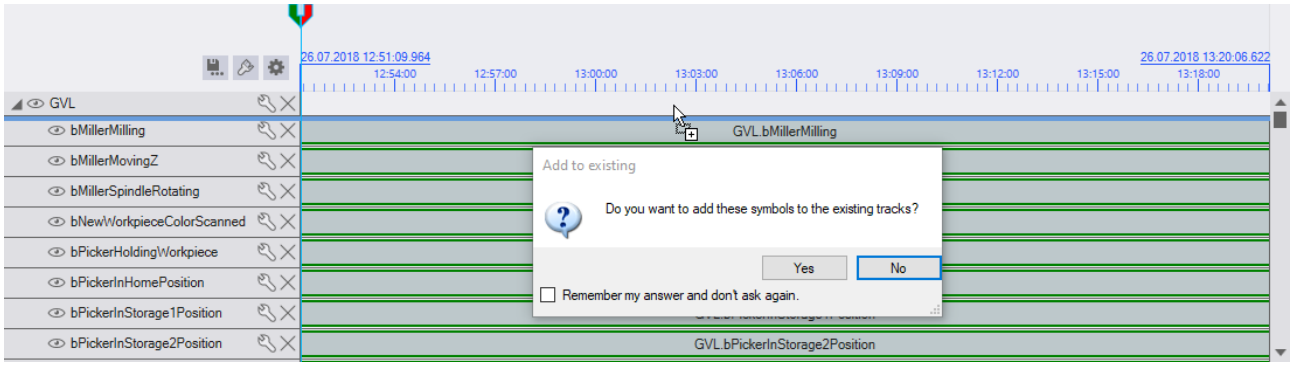


If structures or arrays are to be added, this can happen as with symbols with primitive data type. However, Container Tracks are created for the upper elements of structures and arrays.

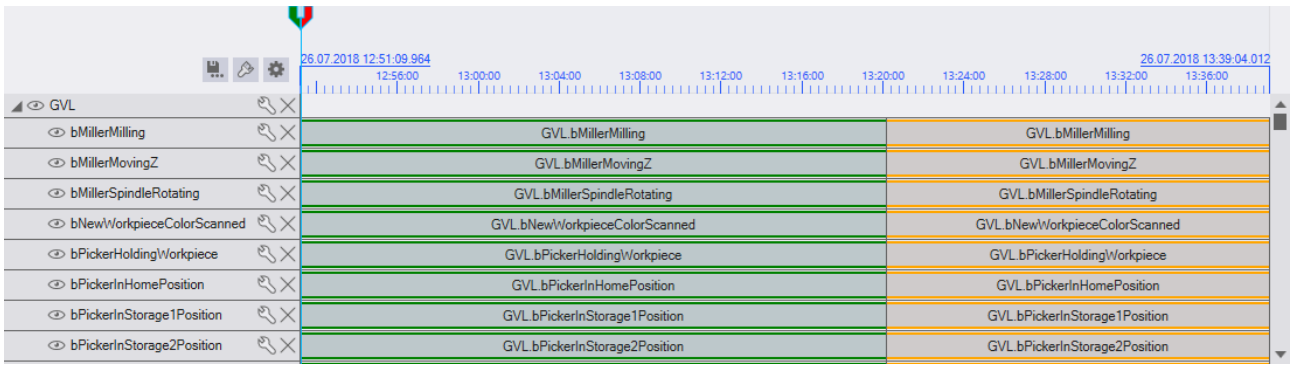


Automatic merging

To more easily merge symbols with the same structure, such as in an Analytics File with multiple records, when dragging multiple symbols onto a container or the empty area of the tracks, the system checks if the symbolism is the same. As soon as this is detected, a message appears that it is possible to add the symbols to the already existing tracks.



If this query is answered with Yes, the added symbols will be appended as a data element to the matching existing track. If the query is answered with No, new tracks are added.



The same result can be achieved by dragging each symbol individually onto the already existing track.

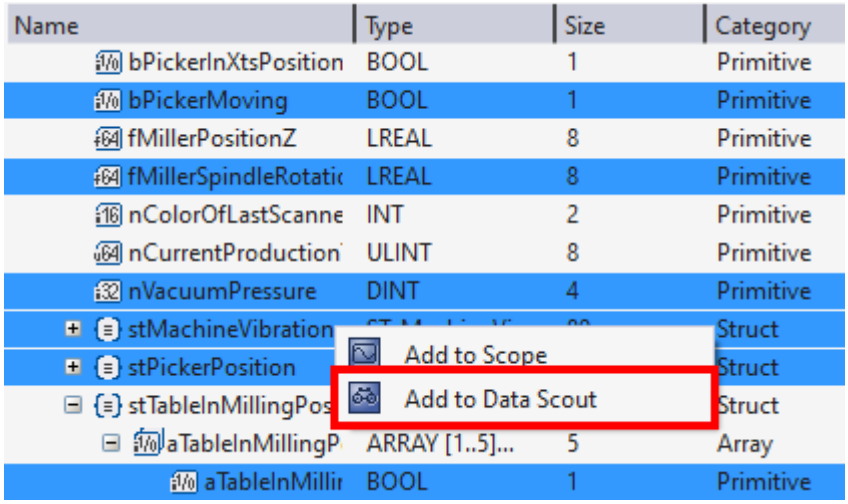
Change order

If the order of existing tracks or data elements is to be changed, this can be done with drag and drop. Data elements can be moved on the whole surface and tracks on the left border  . To do this, the area must be clicked with the left mouse button and moved with the mouse button pressed.

Data elements can be rearranged in the same track or moved to another track. When data elements are dragged onto container tracks or into the track editor, a new track is created. Also, the data element must be compatible with the track it is moved to e.g. a data element with multiple symbols from a multi track cannot be moved to a single track.

Tracks can be rearranged below the container they are in or moved to another container. It is not possible to move tracks into containers that are inside the track to be moved or to move tracks that cannot be changed by their configuration or the configuration of the container above them, e.g. structure arrays.

Add via Target Browser Command

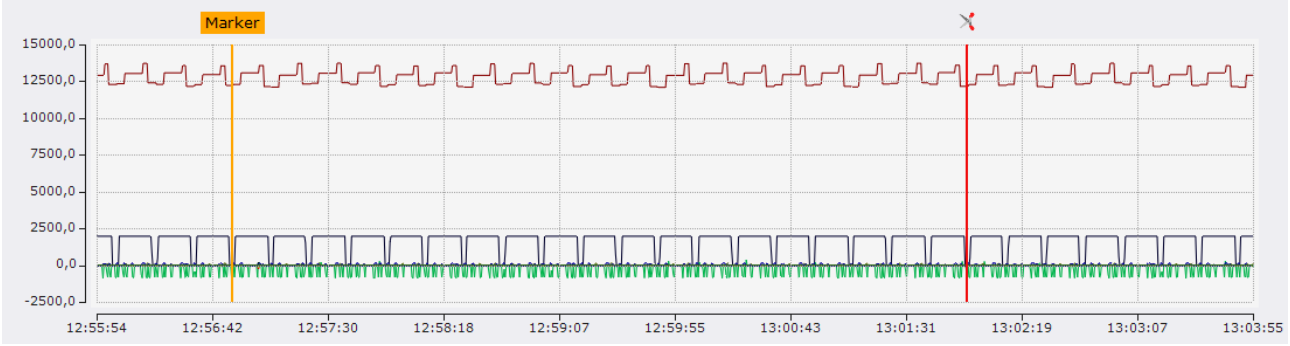


It is possible to add symbols from the Target Browser to a Track Editor by calling the "Add to Data Scout" command in the context menu. The command is displayed when symbols compatible with the Track Editor are selected in the Target Browser and right-clicked. Tracks are then created for each symbol and the sub-symbols of arrays and structures.

If no Data Scout Project exists, an attempt is made to create a new one. If there is already a Data Scout Project in the current Solution, the symbols will be added there. If a Track Editor is selected in the Solution Explorer, the symbols are inserted into it, otherwise a new one is created.

6.2.1.2 Chart

The Data Track Editor uses a chart similar to the YT chart of the Scope View to visualize the data of the tracks.



The chart has a Y axis on the left and an X axis at the bottom, on which the times are displayed. On these axes, each visible track containing data is represented as a graph.

The time values are reassigned for the chart and the time values from the original sources are not used to represent multiple sources in one chart. The start time as well as the cycle time can be set in the Track Editor Timeline options.

A blue area is displayed in the Track Editor above the timeline and track data elements. This area indicates which area is displayed in the chart. If you zoom in the chart using the toolbar functions or the mouse wheel, the selected area in the tracks also changes. It works the same the other way around, so in the tracks the selected range can be adjusted and the range displayed in the chart changes.

Loading data

As soon as a track with data is in the Data Track Editor, this data is loaded. Loading happens automatically in the background. When loading, the tracks whose composition of data elements is the same, i.e. belong to the same Time Track, are loaded simultaneously. Each time the selected area changes or a new track is inserted, the data for the selected area is updated.

If too much data would be displayed in the chart with a high display width, the data will only be partially displayed.

Load Analytics Files

Analytics Files must be prepared for fast display. In the Target Browser in the Analytics File Extension, a green arrow is displayed at the folder for Analytics Files that are not prepared. Clicking on this arrow opens a new window and the Analytics File is prepared for quick display.

Marker

Markers can be displayed in the chart. These remain at a certain position and can be subsequently moved by clicking and dragging on the X axis.

There are two types of markers. The standard marker has a text above the marker line. The text is highlighted with the same color as the marker line. The second type of marker is intended for cutting out data. There can be only two of these markers in a Data Track editor, if a third is inserted the elimination marker closest to the new one will be removed. Cutout markers have no text above their line, but scissors and the line is always red.

When a marker is clicked with the left mouse button, a tooltip opens showing the time and values at the marker's position. If a marker is clicked with the right mouse button, a context menu is opened. In this there is an option to remove the marker from the Data Track Editor.

6.2.1.3 Toolbar

The Data Track Editor toolbar provides various functions to manipulate the display in the chart and edit the data in the tracks.

These functions are used to reset the X and/or Y axis.

	Sets the display width of the chart to the maximum possible width and resets the scaling of the Y axis.
	Resets the scaling of the Y axis.

Of these functions, only one is active at a time. They influence how a left click into the chart behaves.

	When this function is activated, it is possible to click in the chart to select a range on the X axis. For this purpose it is necessary to click into the chart, keep the mouse button pressed and then move the mouse to the left or right, which will lead to a preview of the selection. When the mouse button is released, the display width is reduced to this area.
	When this function is activated, it is possible to click in the chart to select a range on the Y axis. When you press the mouse button, a preview of the selected area is displayed. When the mouse button is released, the scaling on the Y axis is reduced to this range.
	When this function is activated, it is possible to click in the chart to reduce the displayed area on the X and Y axes. When you press the mouse button on the chart, a preview of the area that will be displayed is shown. When the mouse button is released, the scaling of the Y axis and the display width is reduced to this area.
	When this function is activated, it is possible to click in the chart to insert a new marker. The marker is also displayed in the tracks and can be renamed there.
	When this function is enabled, it is possible to click in the chart to insert a special marker to cut out an area.

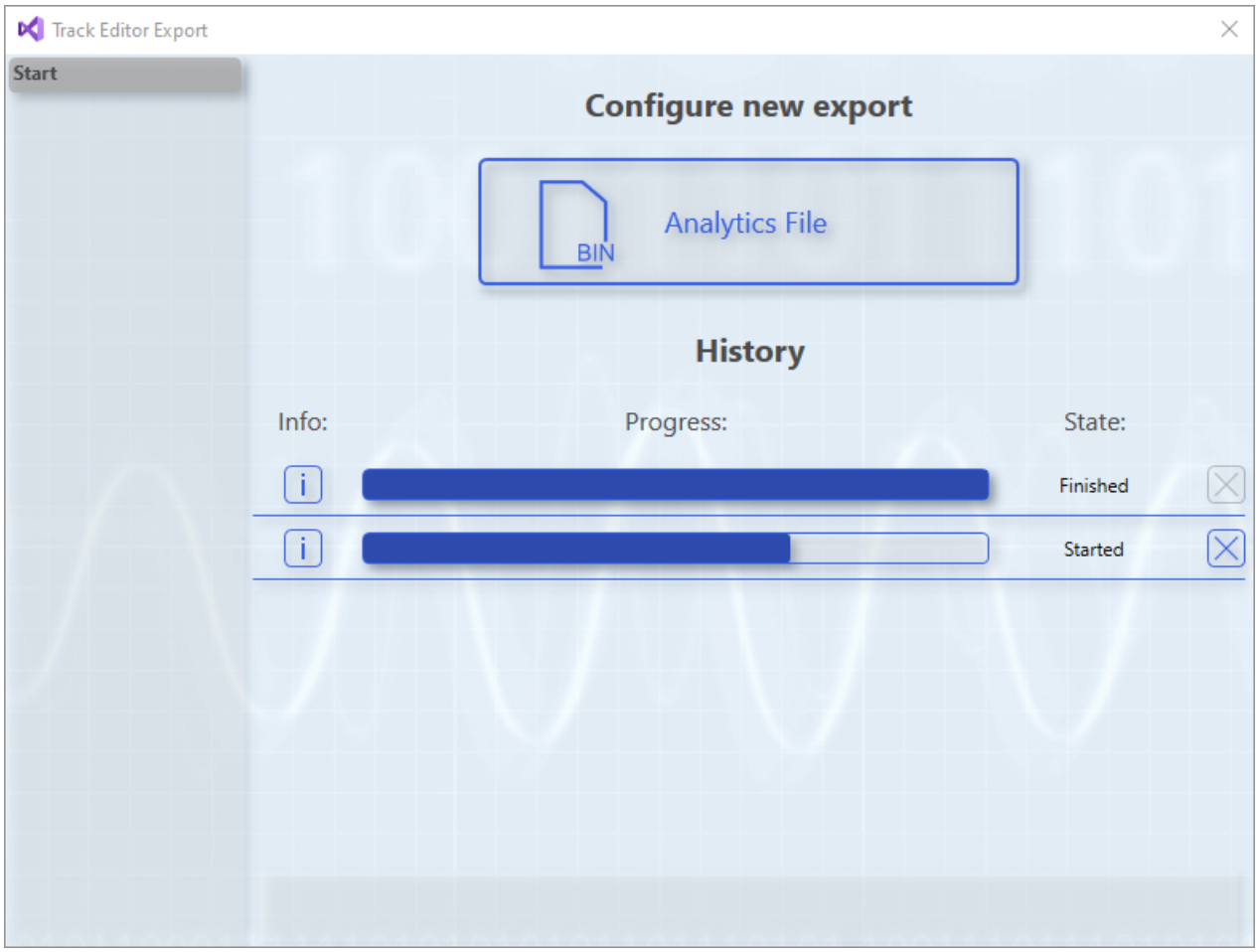
These functions can be used to edit the data of the tracks.

	Determines whether only visible tracks or also invisible tracks are processed by the following functions that can remove data.
	This function can only be used if two markers have been inserted for cutting. These can be inserted with  or by right-clicking on a graph. When running, the area between the two markers is removed.
	This function can be used to remove the area selected in the tracks. The selected area will then be reduced in size as the area previously displayed on the chart has been removed.
	This function removes everything except the selected area.
	This function can only be used if not all tracks have the same amount of data. This difference is also shown dashed red in the tracks timeline. When running, the tracks that contain more data will have as much data removed at the end so that the number matches the track that contains the least data.
	Here you can open the Curve Creator in a new window. This contains the data of the selected area, from the visible tracks. After closing the Curve Creator, the graphs are saved in an Analytics file and replaced in the tracks. If the function to change the start or endpoint has been used in the Curve Creator, the range from which the data originates will be replaced, otherwise the previously selected range. This function is only available if the selected range is smaller than 100,000 values.

6.2.1.4 Export

A Data Track Editor can be exported using the Track Editor Export Wizard. To call the wizard you should click this button  , which is next to the tracks timeline.

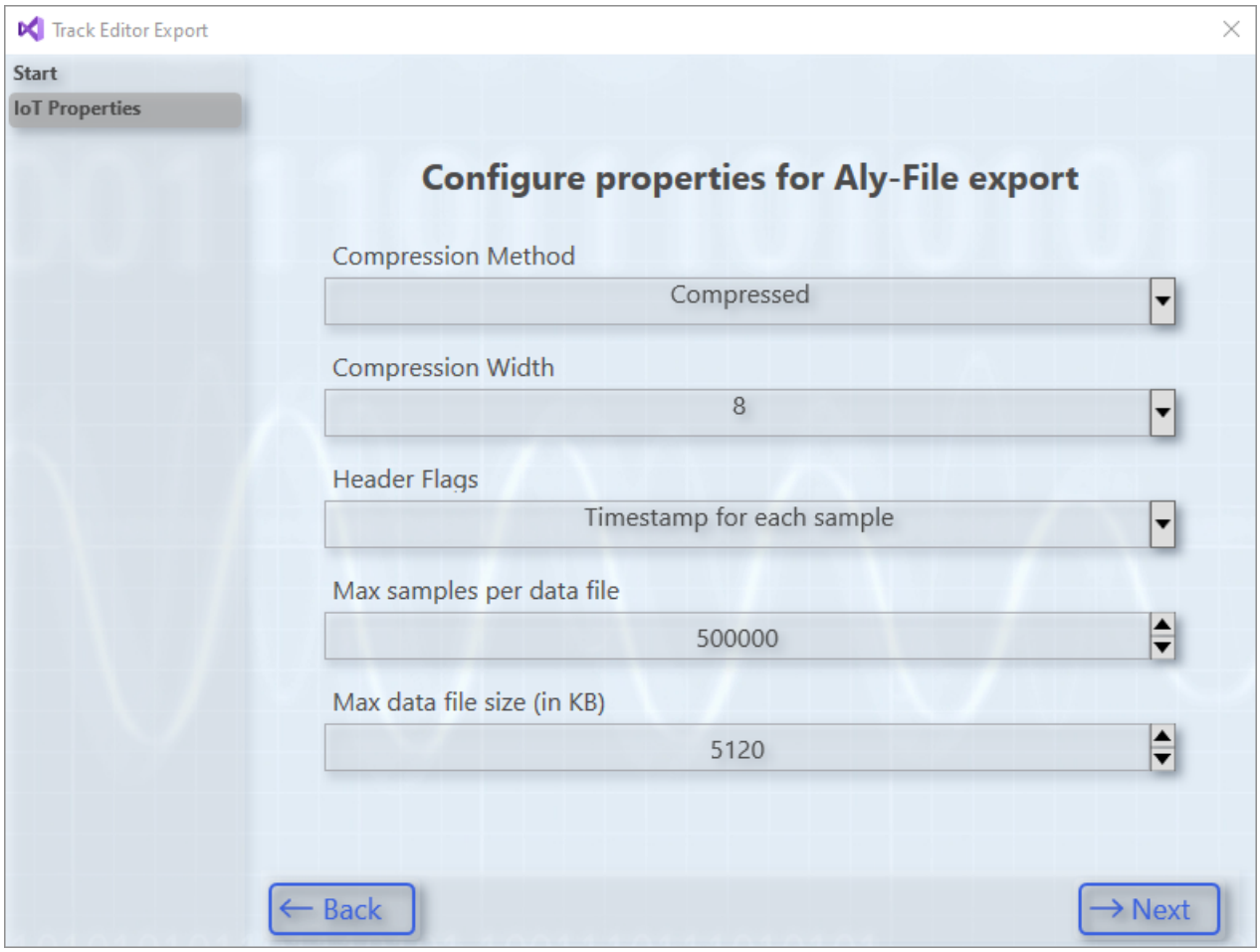
Start page



On the start page of the wizard, new exports can be started in the upper part. All available export formats are displayed as a separate button. In the lower part is a history of exports that were started while the project was open. Each entry in the history has a button on the left side with info about the export. The info contains the start time of the export, important properties depending on the format and if errors occur, the error message. To the right of this is a progress bar showing how many of the samples to be exported have already been exported. After that, the status of the export is output. The button on the far right can be used to cancel running exports.

When clicking one of the buttons to start a new export, one or more new pages will appear in the wizard where the settings for the format can be made.

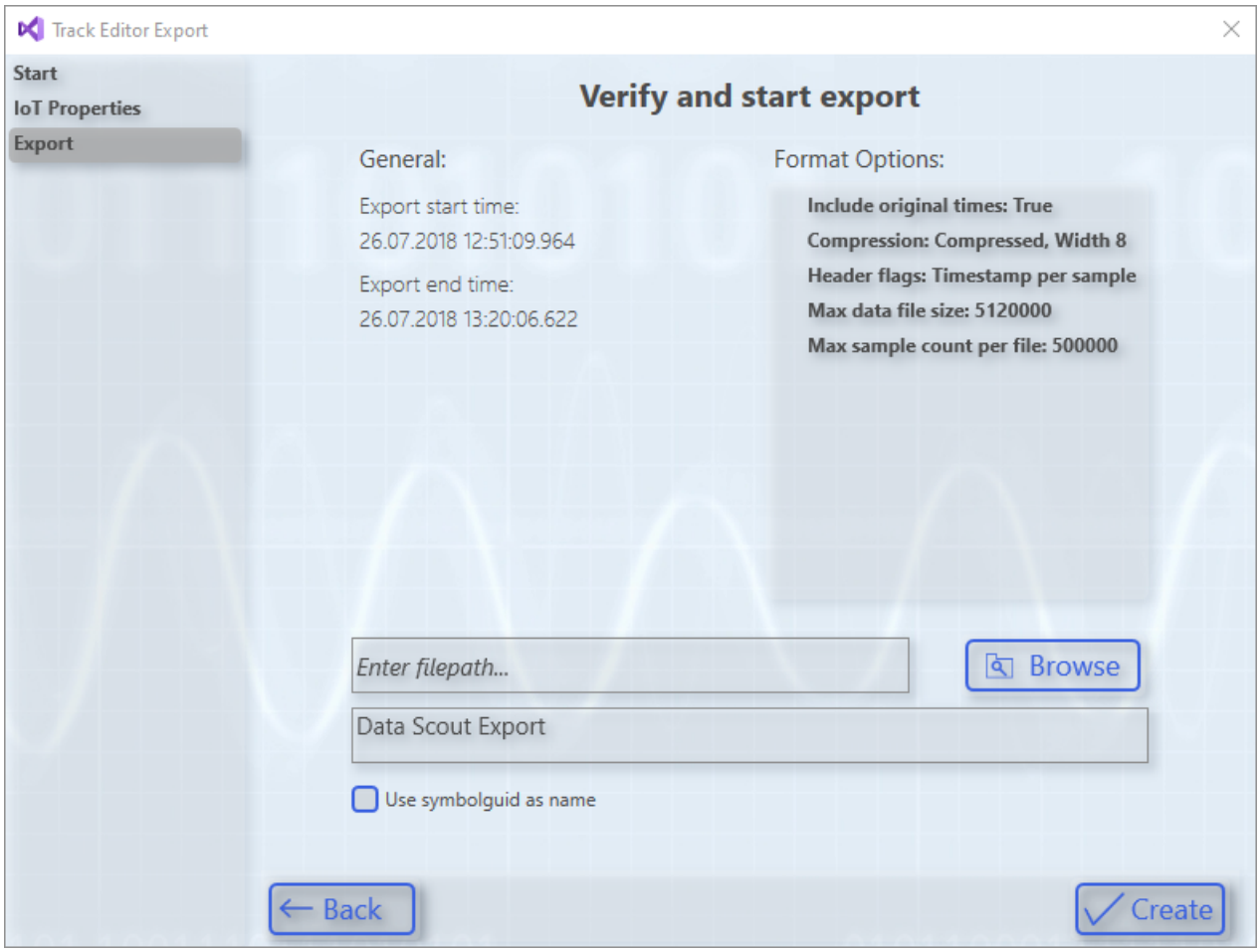
Analytics File Configuration Page



This Analytics File Format configuration page specifies the following settings:

Compression method	Analytics File can be saved compressed or uncompressed. (Data Compression)
Compression width	This option is only available if the Analytics File is to be saved in compressed form. It gives the maximum width to compare for compression.
Header flags	In the flags you can specify whether each .tay file should have only one time value or whether each sample in the .tay file should have its own time value.
Maximum samples and file size	With these two properties you can set how big the .tay files of the Analytics File can become. On the one hand you can set the maximum number of samples to be stored in a file and on the other hand you can set the size of a file in kilobytes in the file system. If one of these values is exceeded, a new .tay file is created.

Export page



The Export page is displayed after going through the configuration pages for the format. Here all settings are listed again and a storage location can be specified if necessary for the format. The upper text field indicates the folder and the lower one the file name.

For the Analytics File Format, a checkbox can also be checked instead of specifying a file name, then the ID of the symbolism will be used as the folder name.

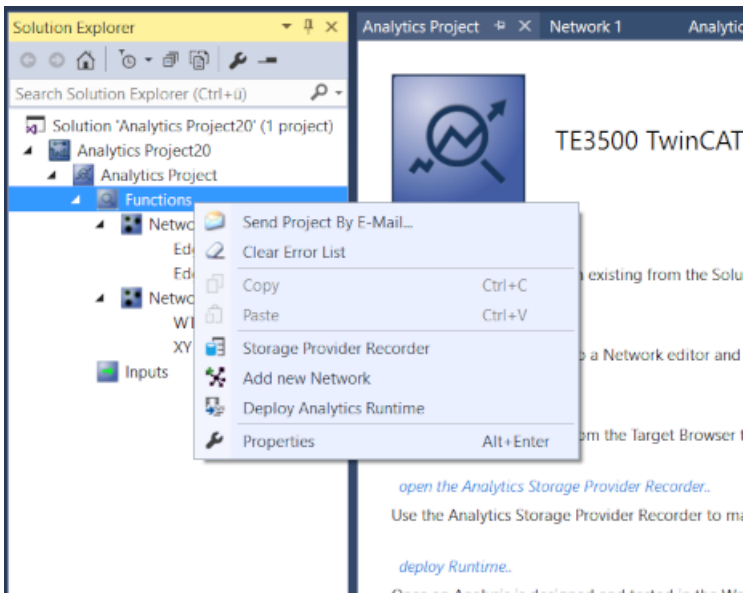
When the Create button is clicked, the export starts in the background. Then this page closes and the wizard jumps back to the start page. The new export is then also listed in the history and the progress can be tracked.

6.3 Editor Basics

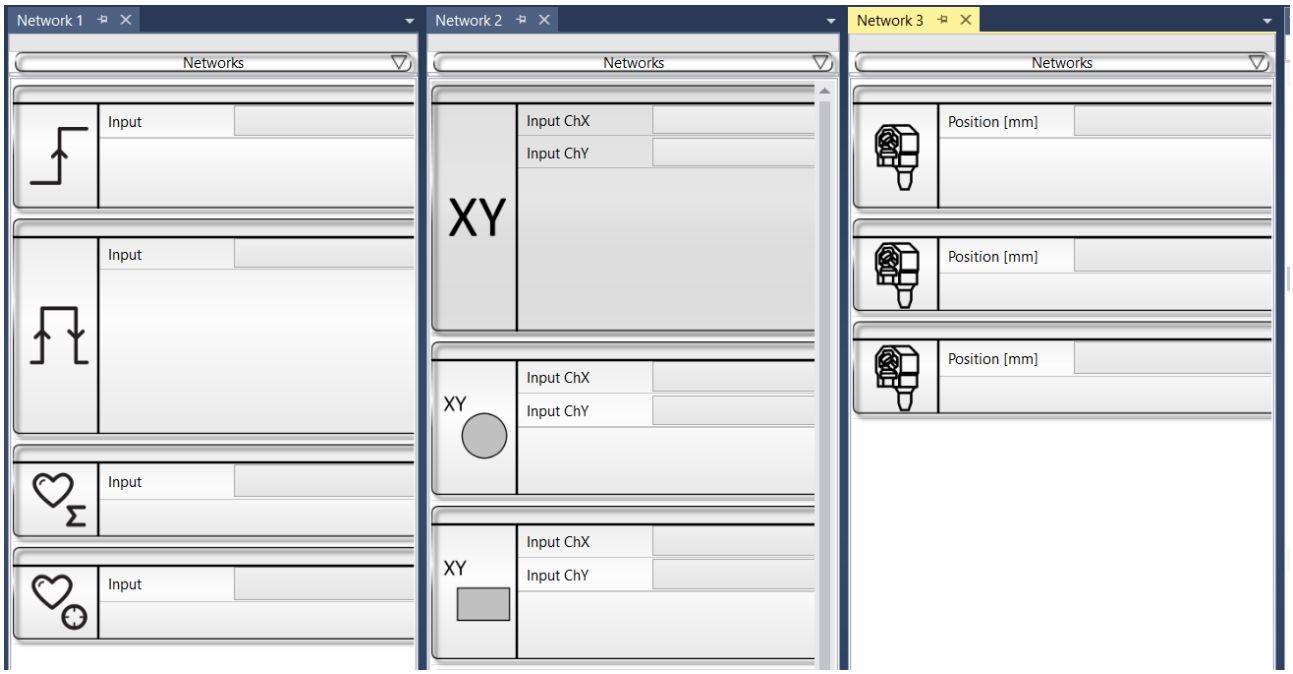
6.3.1 Networks

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.

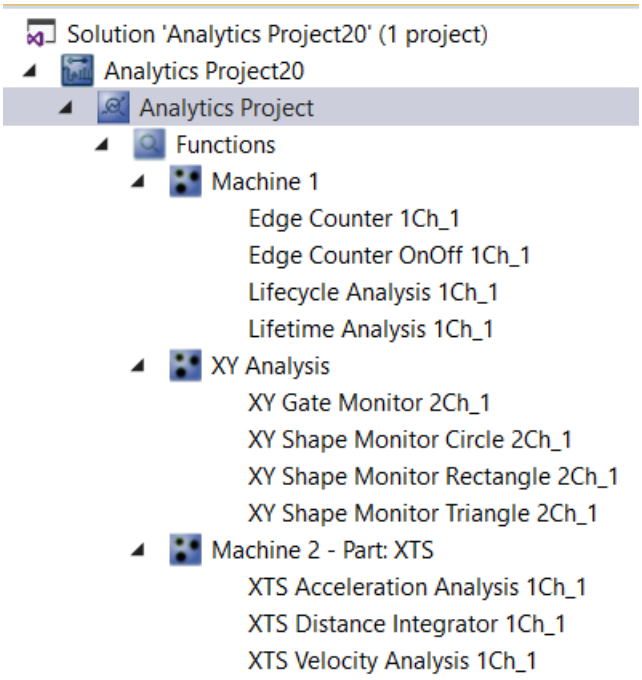
A network can be added directly from the Analytics Project home page or from the context menu.



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.



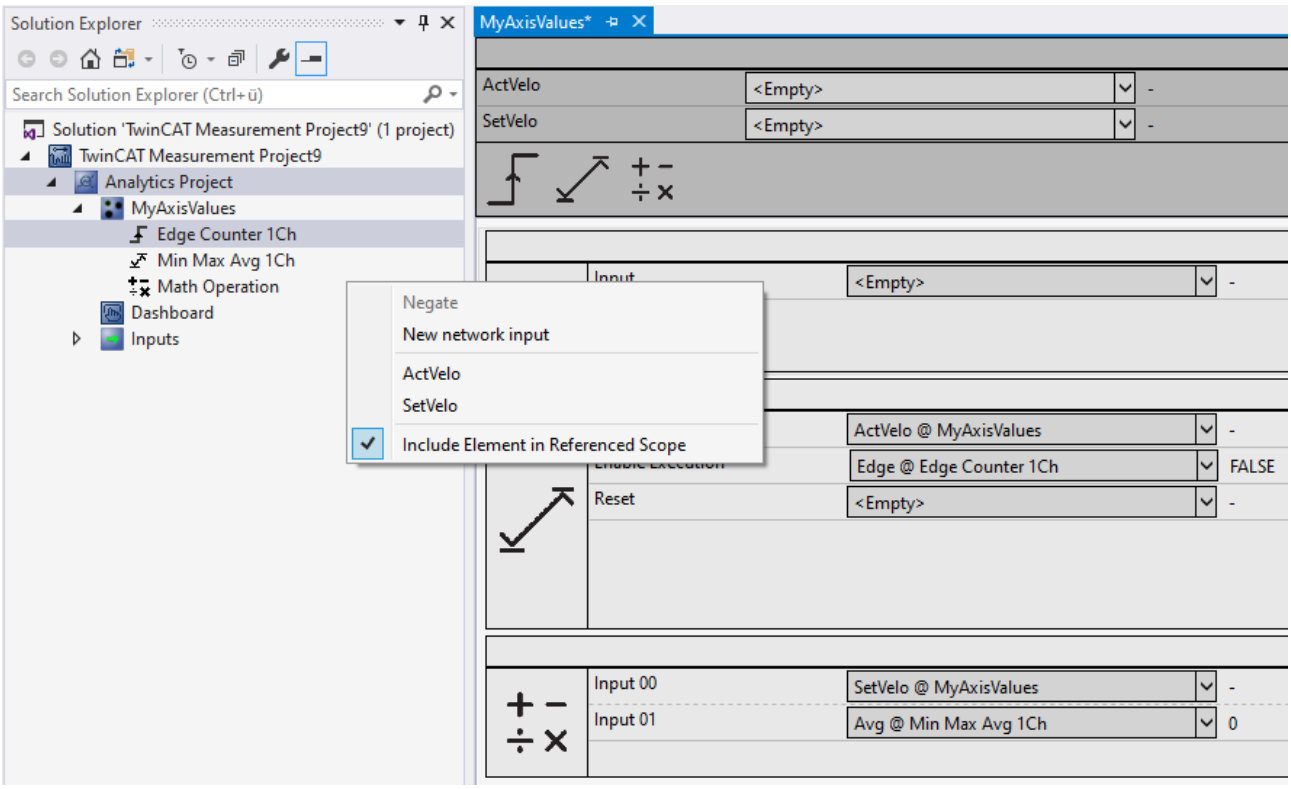
Furthermore, you have the option to rename the networks (F2 on the selected network element in the Solution Explorer) to create networks for different machines, machine parts or other content-related connections, for example.



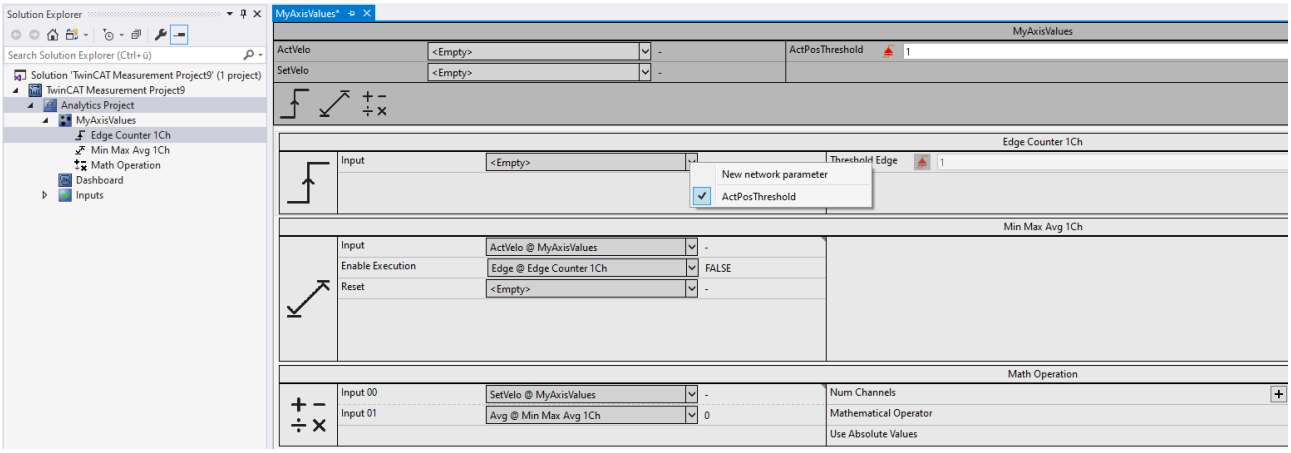
6.3.1.1 Networks as template

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.

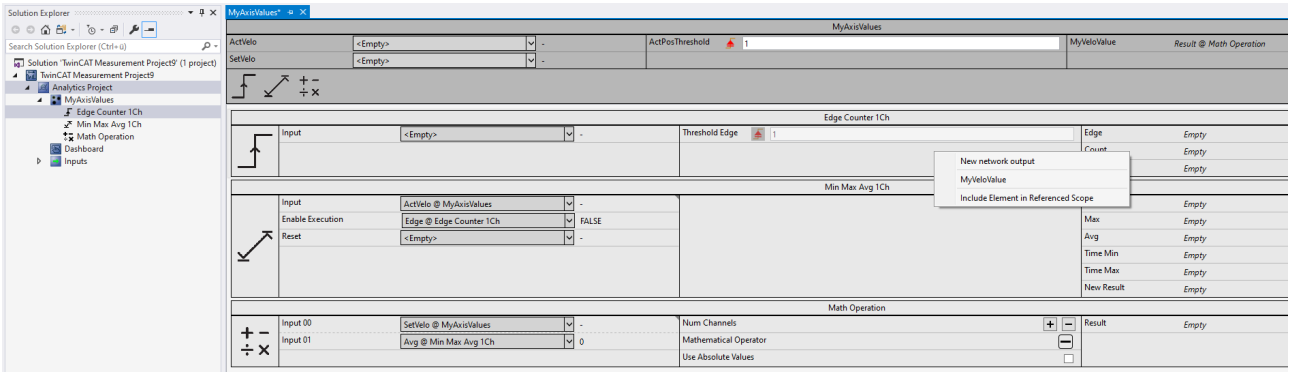
Once you have created an analysis, you must first select which of the available inputs should be visible to the outside of the network. To do this, select the desired input and switch to the context menu by right-clicking. There you can link the input to an existing network input or alternatively define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



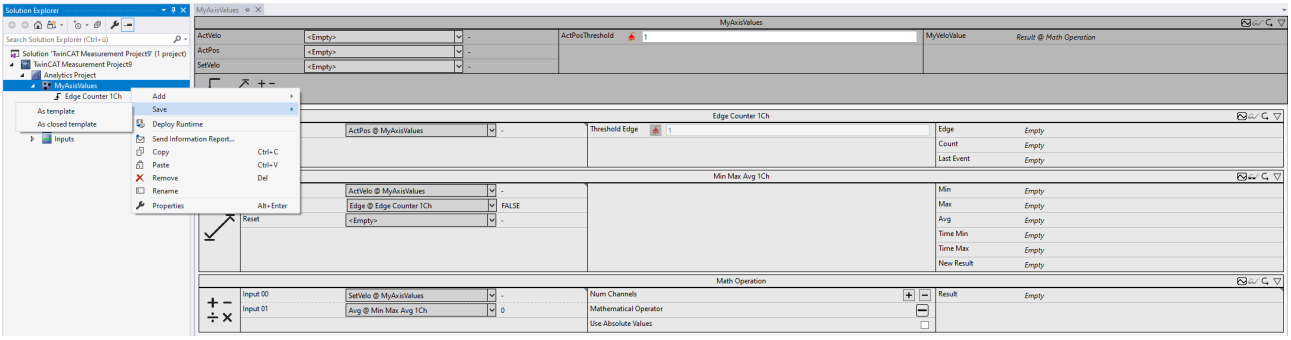
The same procedure is available for parameters and outputs.



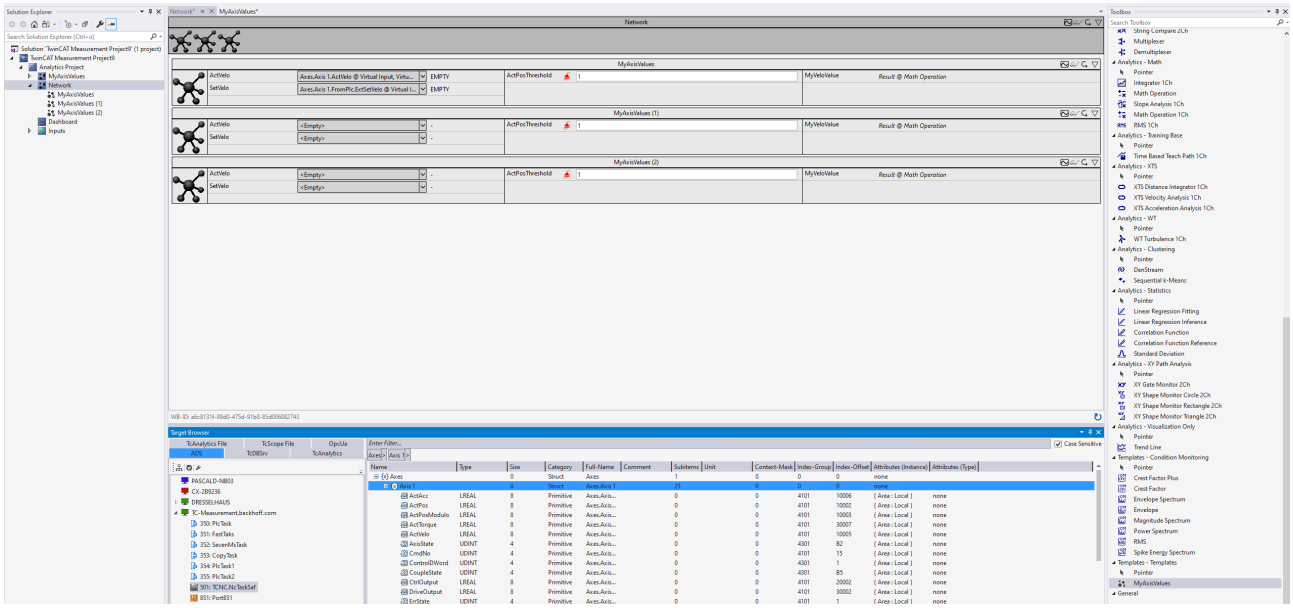
In addition to these inputs and outputs, it is also possible to specify dynamic inputs and outputs. These are automatically offered to you when one of the selected algorithms supports this function. The background to this is the option of increasing the number of inputs or outputs on an algorithm via 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 [▶ 303], in order to automatically obtain the same Scope configuration when a network is used again.

6.3.2 Enable Disable

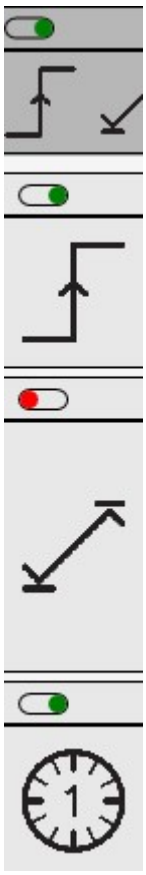
The TwinCAT Analytics Workbench offers the functionality to switch individual algorithms or even complex networks on or off in a data analysis. This document explains the use and advantages of this function.

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.

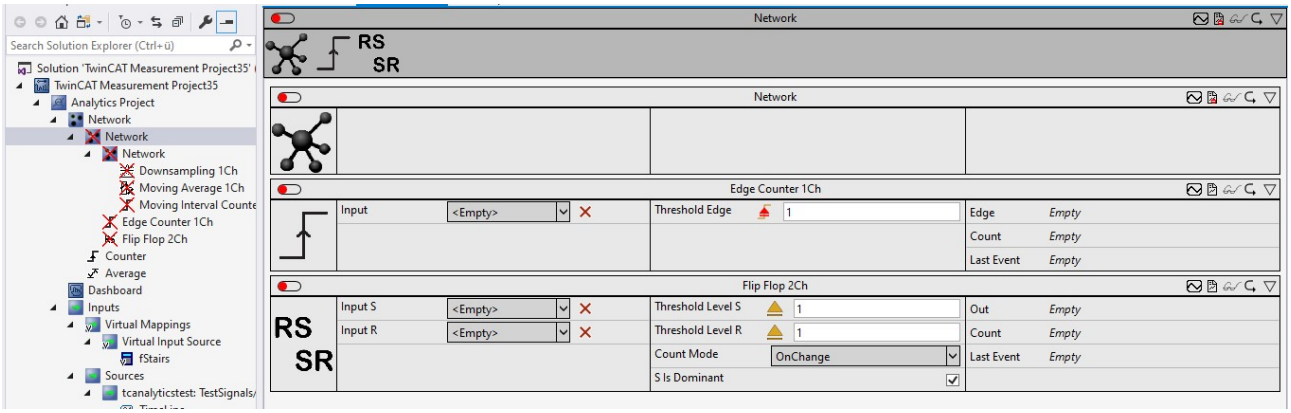
Switching modules on or off is controlled by the control elements in the upper left corner of the respective module.



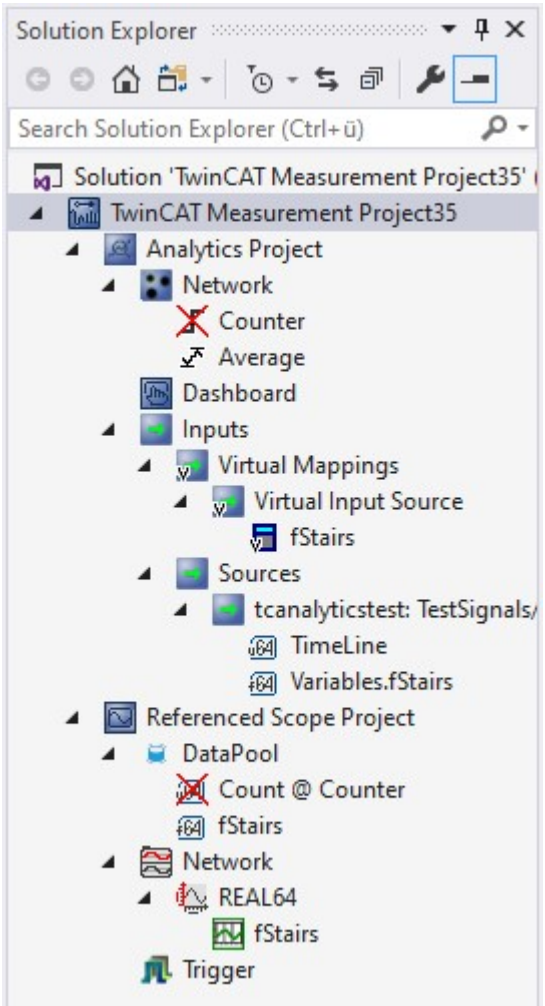
Alignment	Color	Status
Right	Green	Enabled
Left	Red	Disabled

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:

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 [▶ 298]), linked elements of the Scope are also affected by the deactivation.
Deactivating the algorithm also deactivates the associated Scope acquisition:

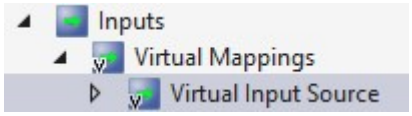


6.3.3 Static Values

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.

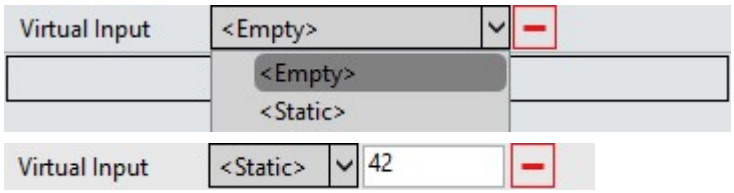
The use of static values in analysis algorithms offers several advantages. First, the performance of the algorithm can be improved by avoiding computations that otherwise, would have to be performed during each execution of the algorithm. Instead, the algorithm can access the static value set during 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.

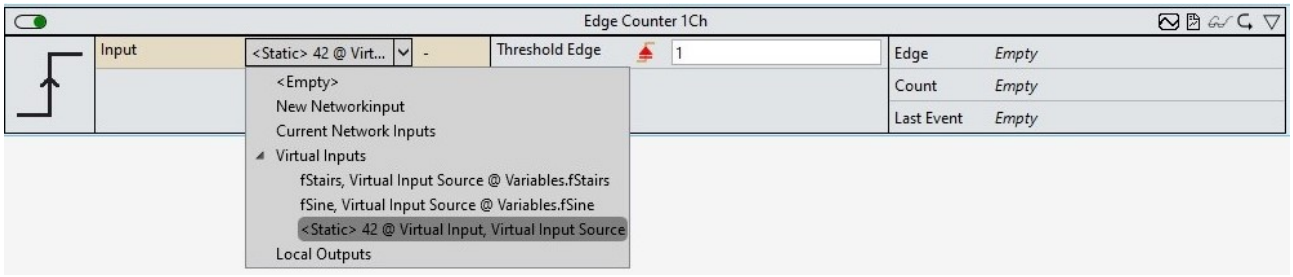


In the editor of the Virtual Input Source you can create a new Virtual Input or edit an existing one by using **+**.

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.



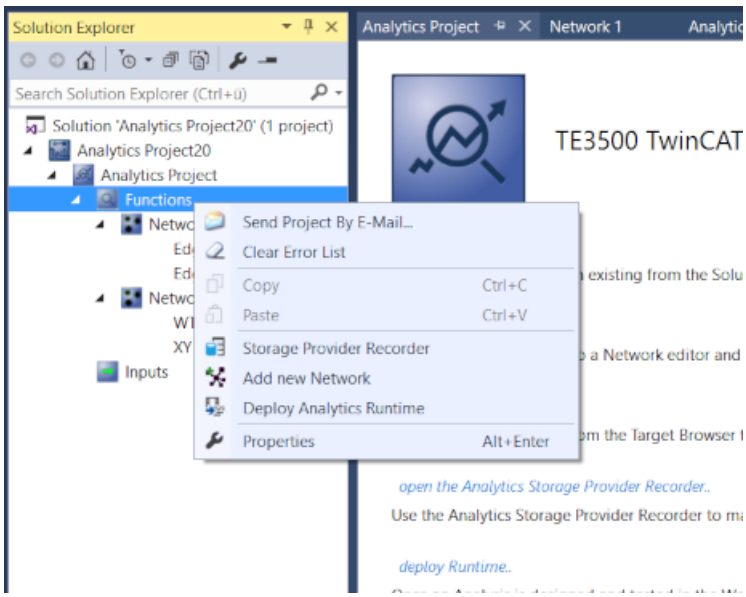
Static values can also be used on network inputs. Here the linking with the dependent inputs of the individual modules would run exactly the same as the behavior with incoming data inputs.

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.

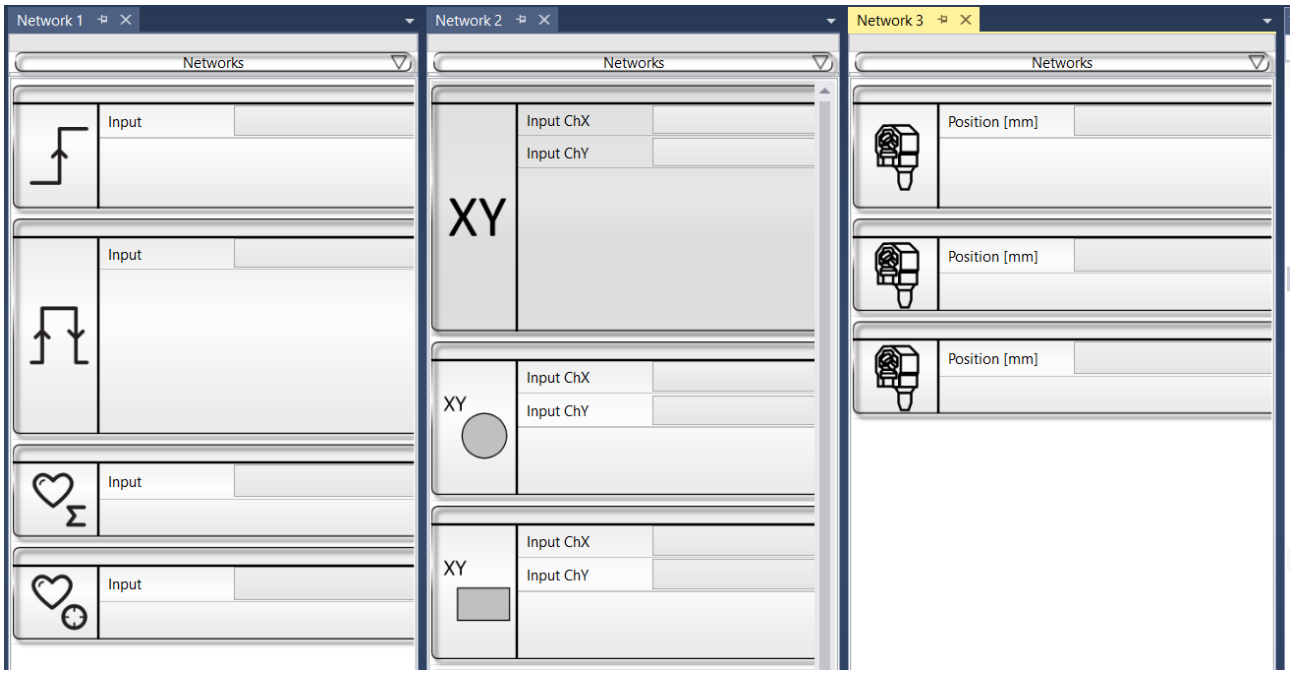
6.4 Networks

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.

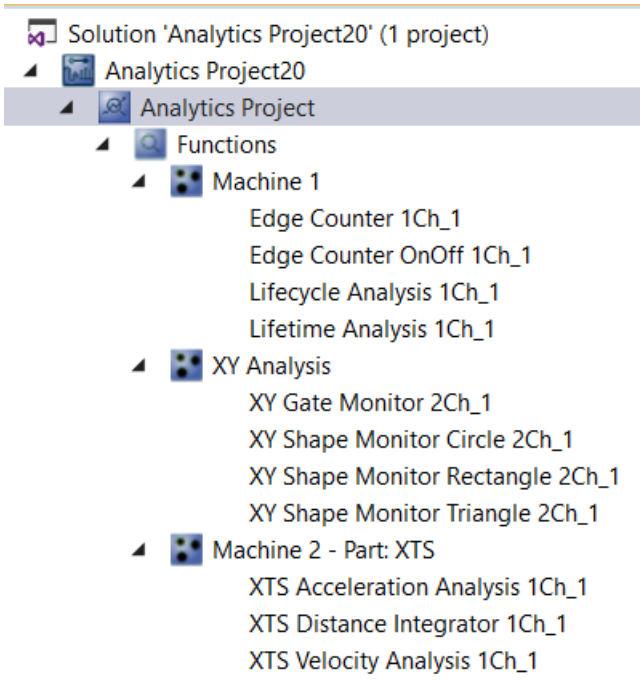
A network can be added directly from the Analytics Project home page or from the context menu.



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.



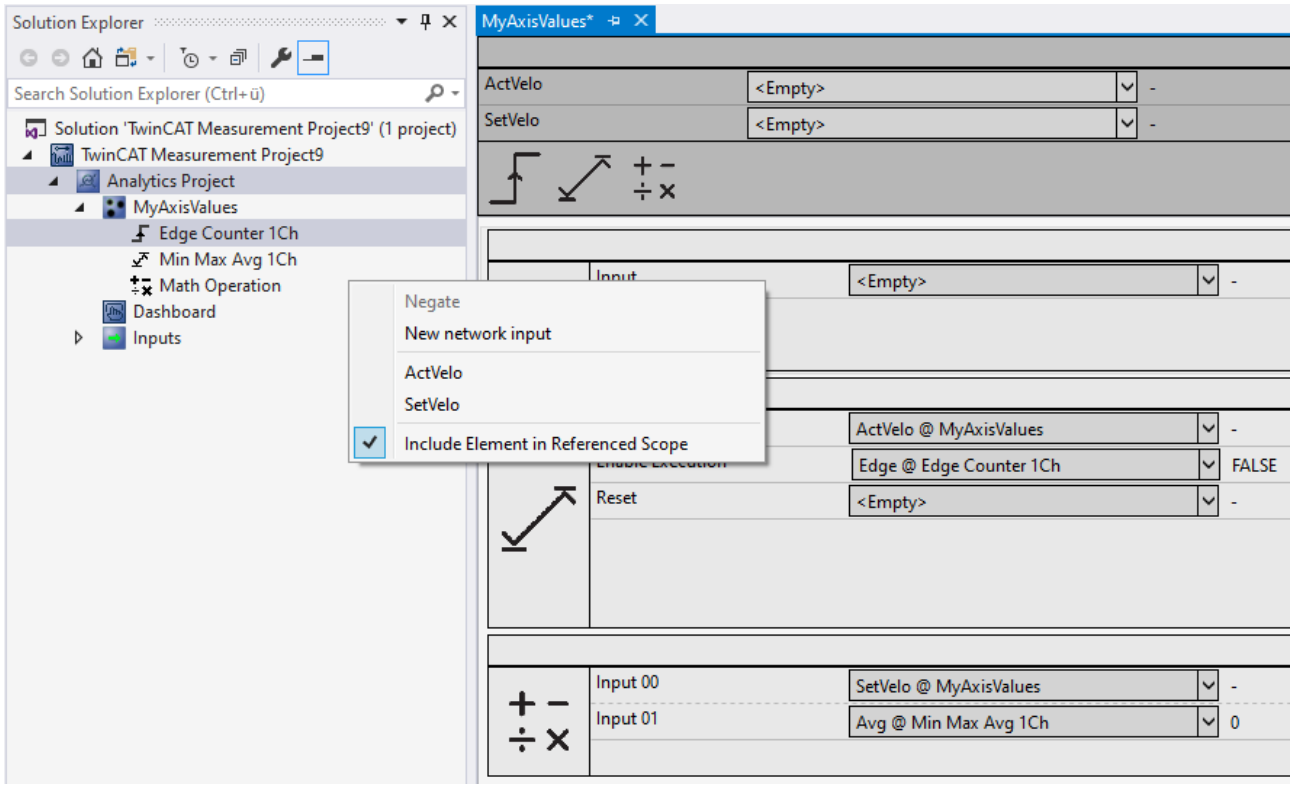
Furthermore, you have the option to rename the networks (F2 on the selected network element in the Solution Explorer) to create networks for different machines, machine parts or other content-related connections, for example.



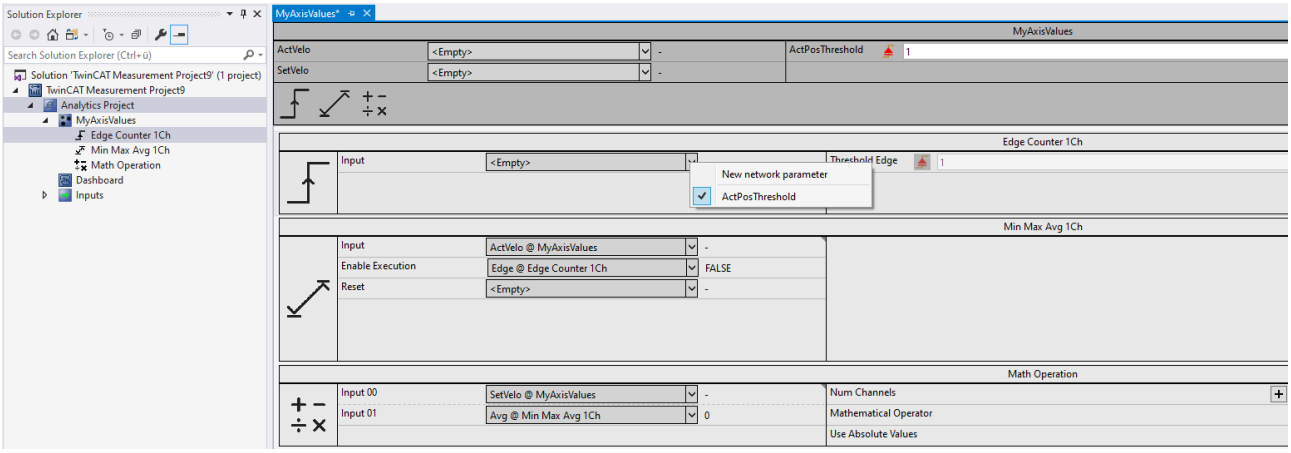
6.4.1 Networks as template

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.

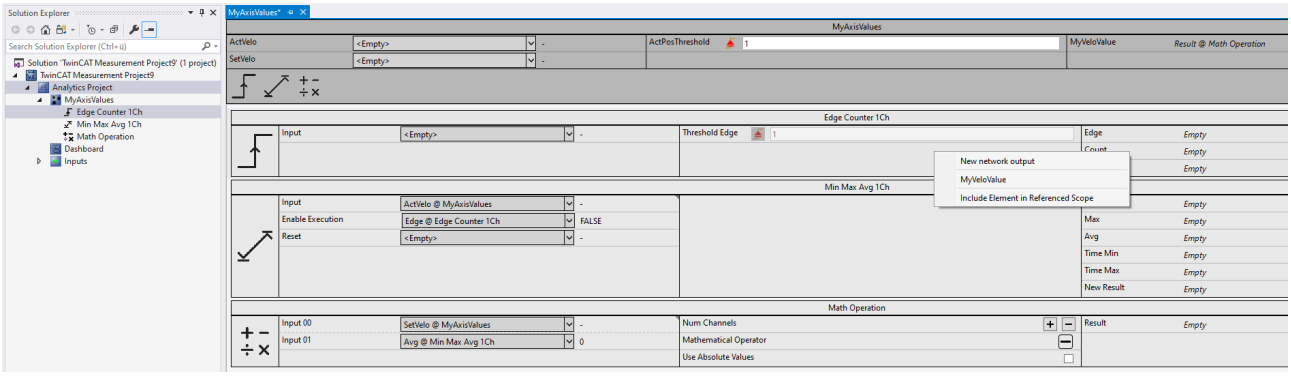
Once you have created an analysis, you must first select which of the available inputs should be visible to the outside of the network. To do this, select the desired input and switch to the context menu by right-clicking. There you can link the input to an existing network input or alternatively define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



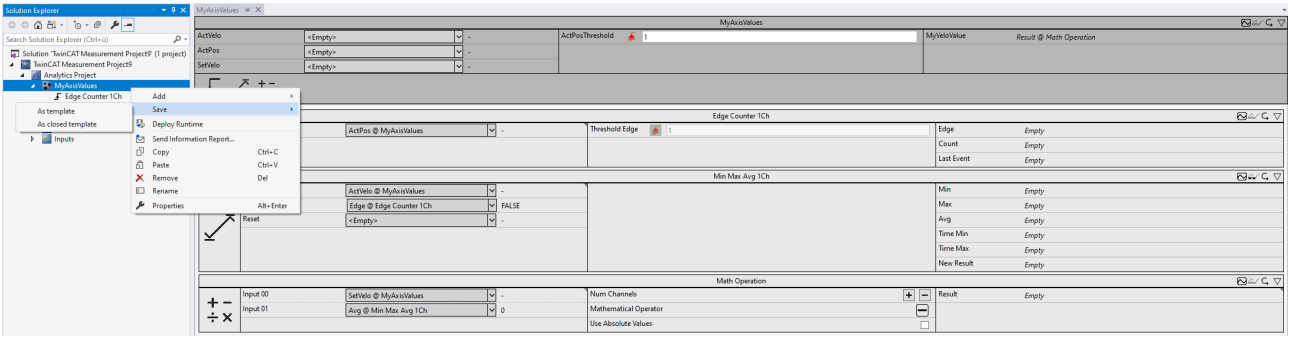
The same procedure is available for parameters and outputs.



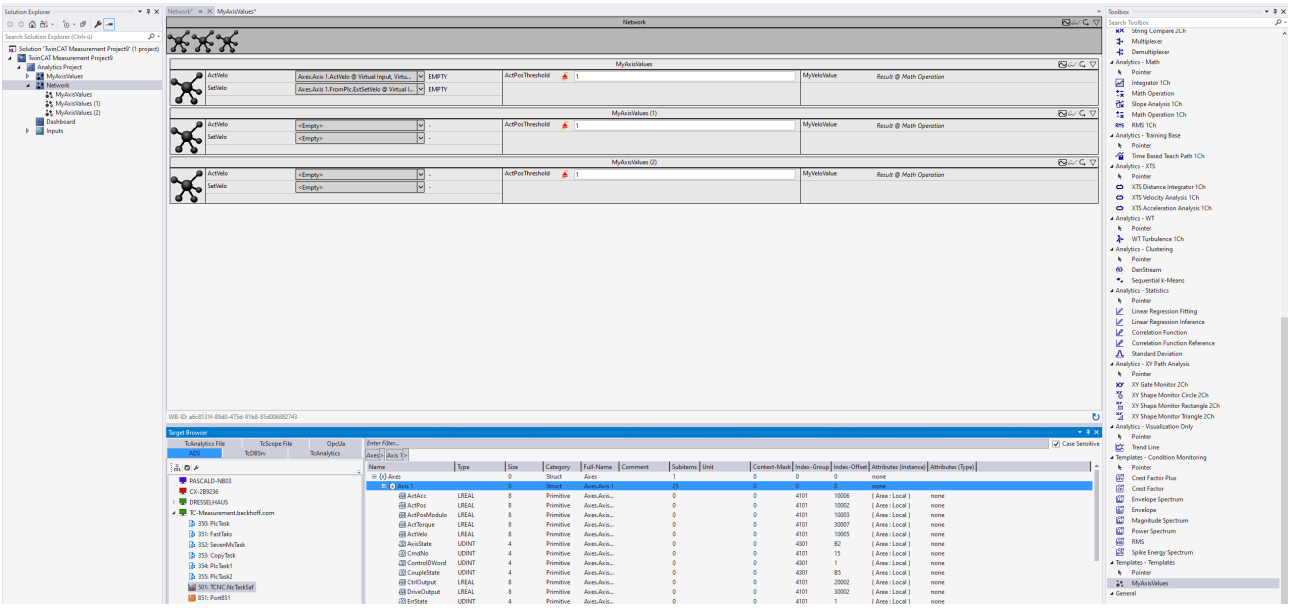
In addition to these inputs and outputs, it is also possible to specify dynamic inputs and outputs. These are automatically offered to you when one of the selected algorithms supports this function. The background to this is the option of increasing the number of inputs or outputs on an algorithm via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



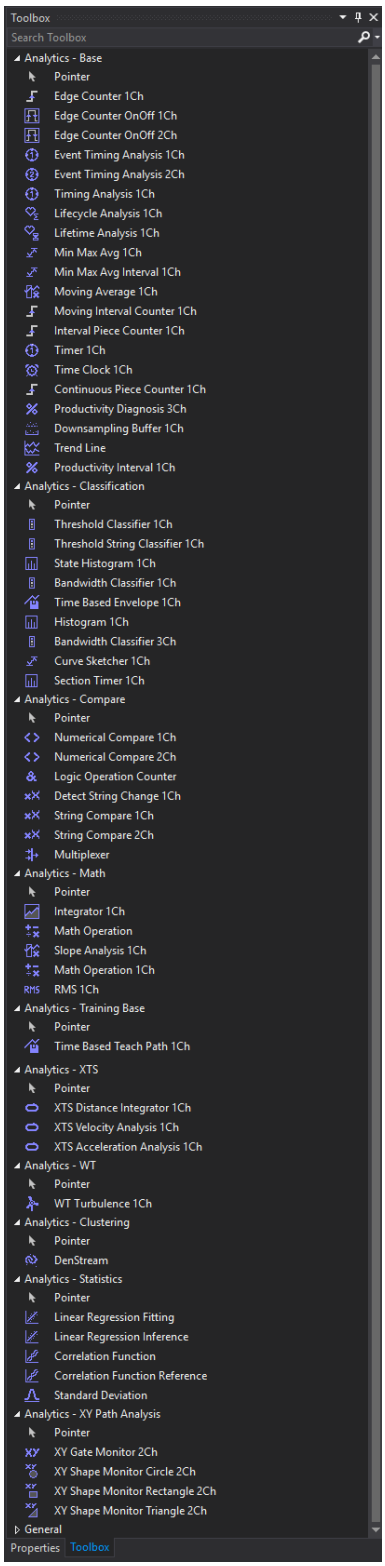
The Target Browser offers a special function.
If the names have been chosen for the network in such a way that they correspond exactly to variables of a structure, you can drag a complete structure such as an axes structure directly onto one of the x-inputs of a network. All matching names of all network 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 [▶ 303], in order to automatically obtain the same Scope configuration when a network is used again.

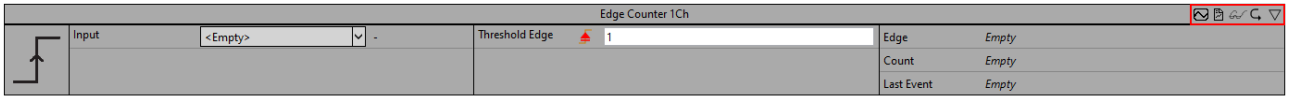
6.5 Algorithms

The TwinCAT Analytics Workbench configurator and the TwinCAT Analytics Service Tool include various analysis algorithms that can be found in the toolbox. If the more than 50 algorithms do not meet the requirements, user-specific algorithms can be developed using the Analytics Lambda functions. If the toolbox is empty, select the Analytics project to see the algorithms.



Currently, there are eleven different groups of algorithms: Analytics-Base, Analytics-Classification, Analytics-Compare, Analytics-Math, Analytics-Training Base, Analytics-XTS, Analytics-WT, Analytics-XY Path Analysis, Analytics-Clustering, Analytics-Statistics and Analytics-C++ Lambda Functions. User-specific algorithms can be developed using the C++ lambda functions (see C++ lambda functions [▶ 234]). In addition, it is possible to use algorithms from other libraries that are implemented in the Analytics Toolbox (see Algorithms from other TwinCAT libraries [▶ 264]). The Analytics Custom Toolbox Cleaner [▶ 263] can be used to clean up the toolbox with regard to user-specific elements, such as the templates or the C++ lambda functions.

Each algorithm has the same five icons in the upper right corner:



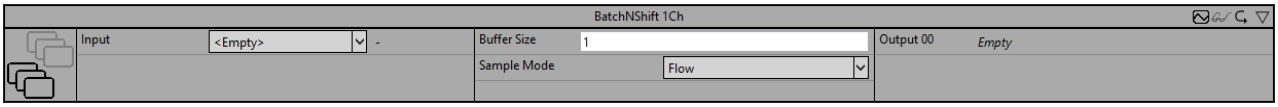
- **Include in Referenced Scope:** if you click on the *Included in Referenced Scope* icon, the algorithm outputs are added to a referenced Scope project.
- **Include in Report:** If you click on the *Include in Report* icon, the algorithm with the configured visualization will be integrated into a manual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

The different algorithm groups are described below.

6.5.1 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.

6.5.1.1 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.

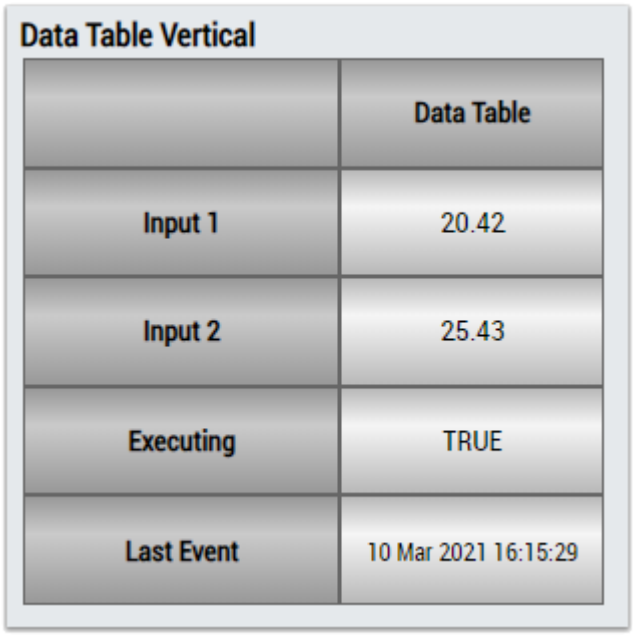
Output values

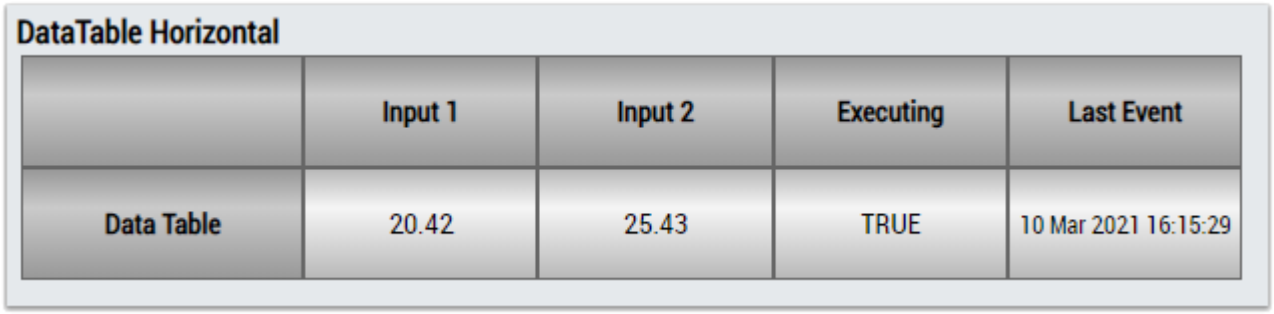
- **Output Value 00..n**: results of the BatchNShift buffer according to the Sample Mode. Each output channel represents a buffer storage space.

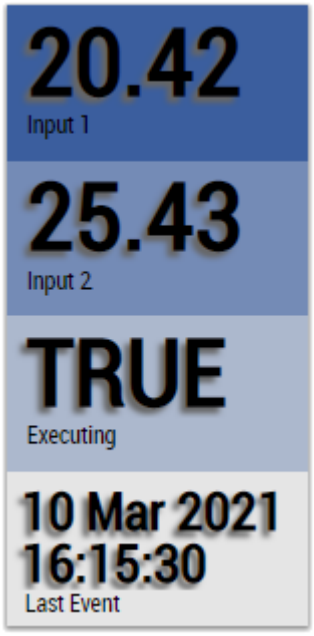
Standard HMI Controls

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*

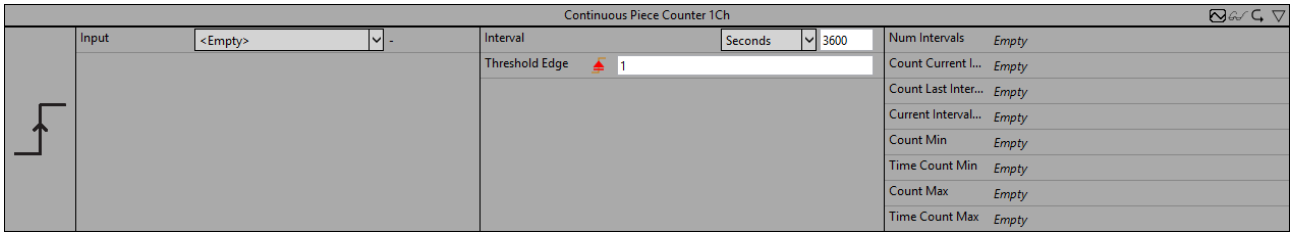






Alternatively, customer-specific HMI controls can be mapped in the *BatchNShift 1Ch* algorithm using the Mapping Wizard.

6.5.1.2 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.

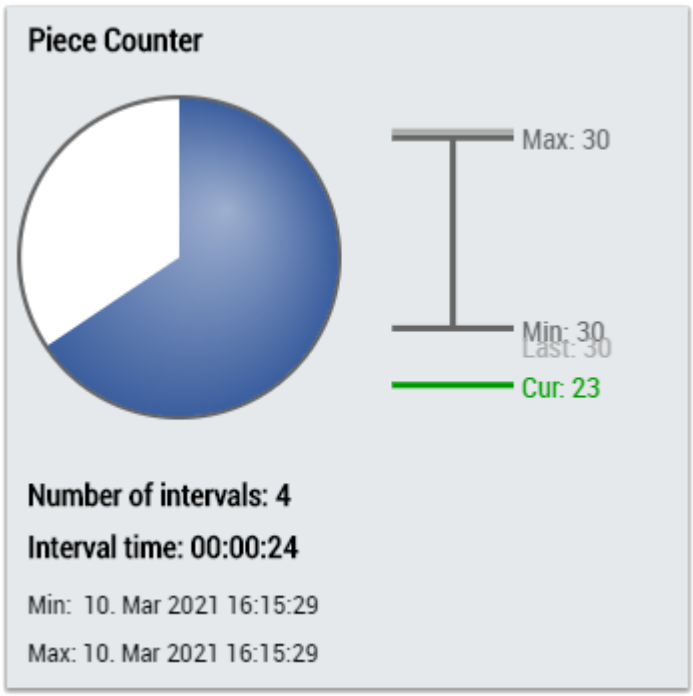
Output Values

- **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.

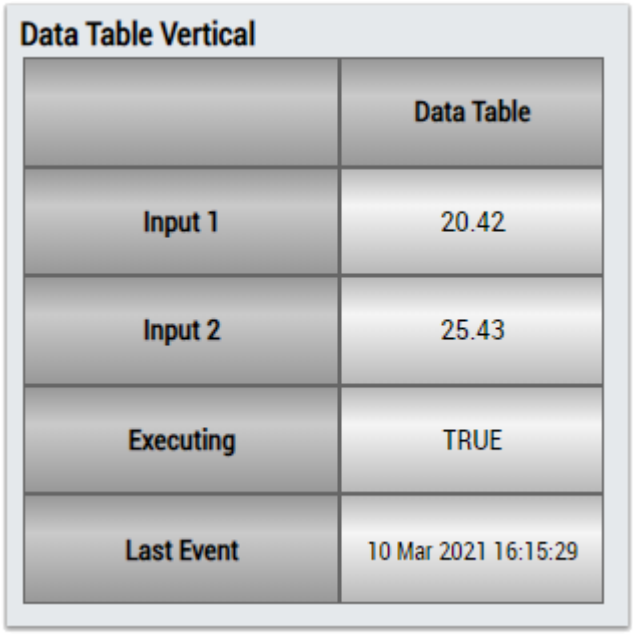
Standard HMI Controls

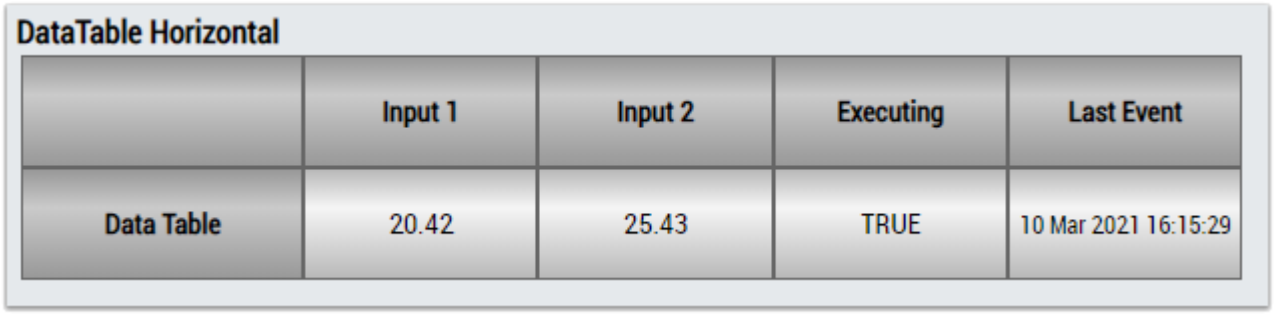
For the Continuous Piece Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

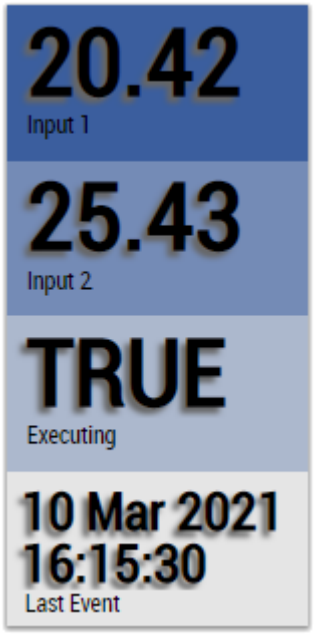
1. The PieceCounter control visualizes the output values Num Intervals, Count Last Interval, Count Current Interval, Count Min, Count Max, Time Count Min, Time Count Max and Time Current Interval.



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.

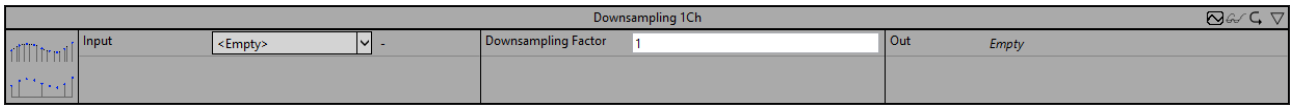






Alternatively, customer-specific HMI controls can be mapped in the Continuous Piece Counter 1Ch algorithm using the Mapping Wizard.

6.5.1.3 Downsampling 1Ch



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.

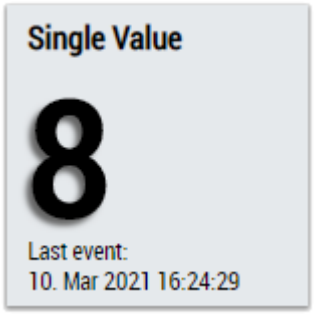
Output values

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

Standard HMI Controls

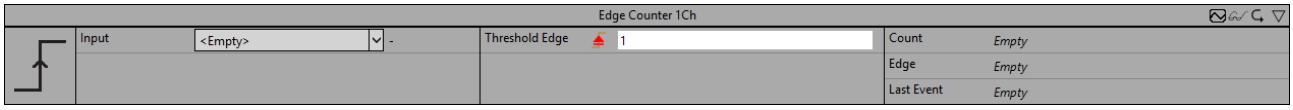
For the *Downsampling 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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.

6.5.1.4 Edge Counter 1Ch



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.

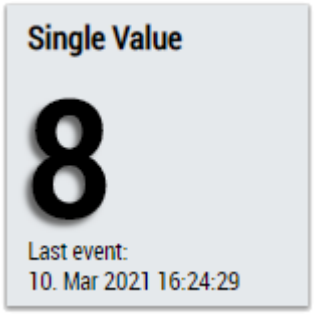
Output values

- **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.

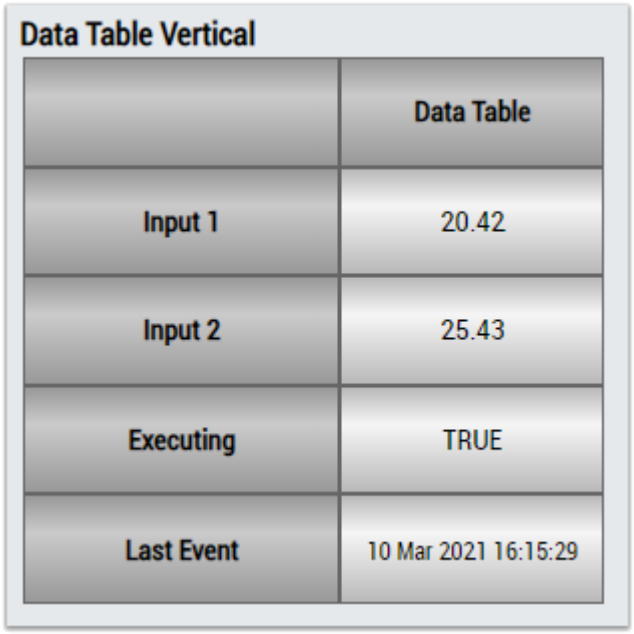
Standard HMI Controls

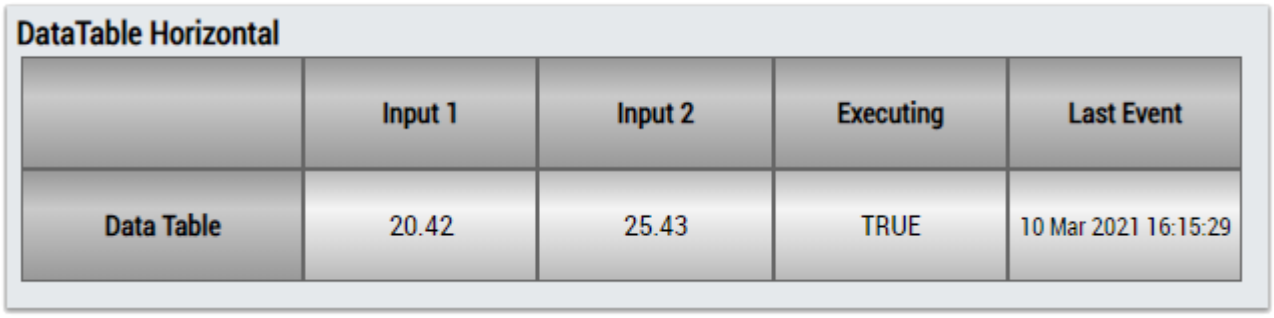
For the Edge Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

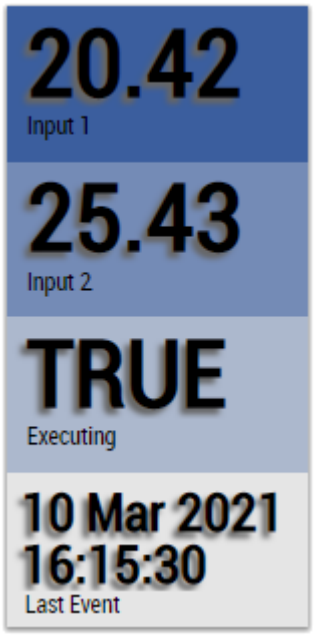
1. The SingleValue control visualizes the output values Count and Last Event.



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

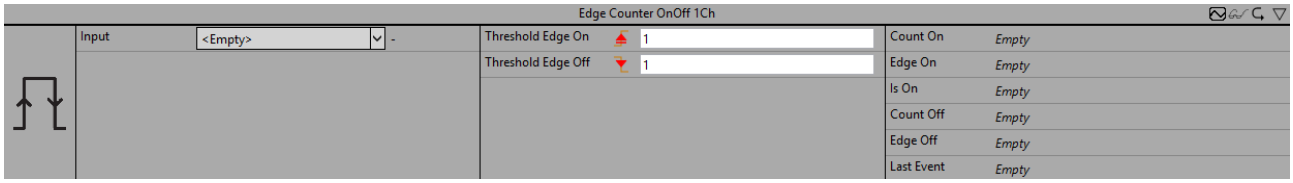






Alternatively, custom HMI controls can be mapped in the Edge Counter 1Ch algorithm using the Mapping Wizard.

6.5.1.5 Edge Counter On Off 1Ch



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

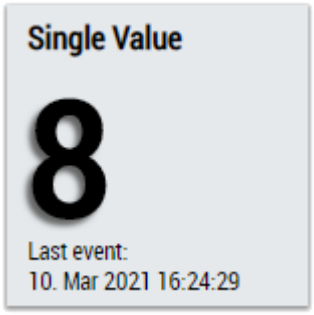
- **Is On:** Indicates *TRUE* within the timespan between the On event and the Off result, otherwise *FALSE*.
- **Edge On:** Indicates TRUE on a rising edge.
- **Edge Off:** Indicates TRUE on a falling edge.
- **Count On:** Counts the number of triggered On events.
- **Count Off:** Counts the number 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

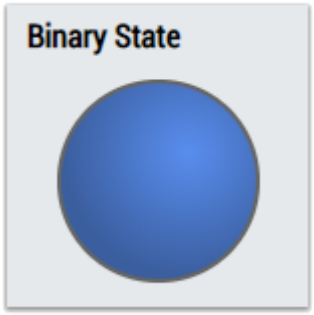
For the Edge Counter On Off 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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.



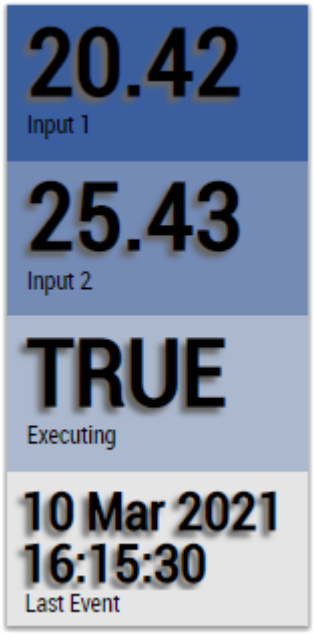
4. The Table Control or Multivalue Control visualizes all output values: Flanks (Is On, Edge On, Edge Off), Count On, Count Off, Last Event.

Data Table Vertical

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

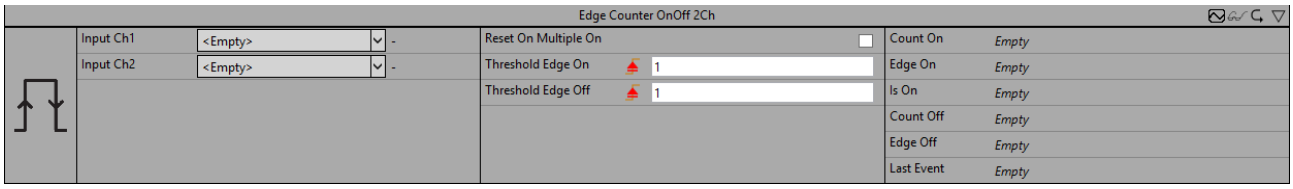
DataTable Horizontal

	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.

6.5.1.6 Edge Counter On Off 2Ch



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.

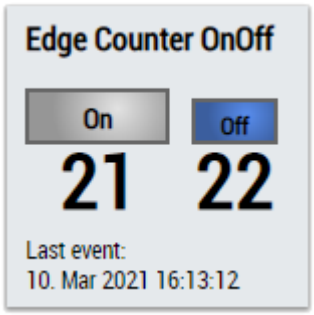
Output values

- **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.

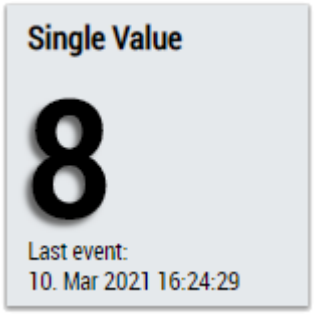
Standard HMI Controls

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

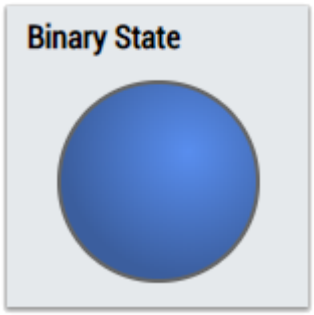
1. The EdgeCounterOnOff control visualizes the output values Is On, Count On, Count Off and Last Event.



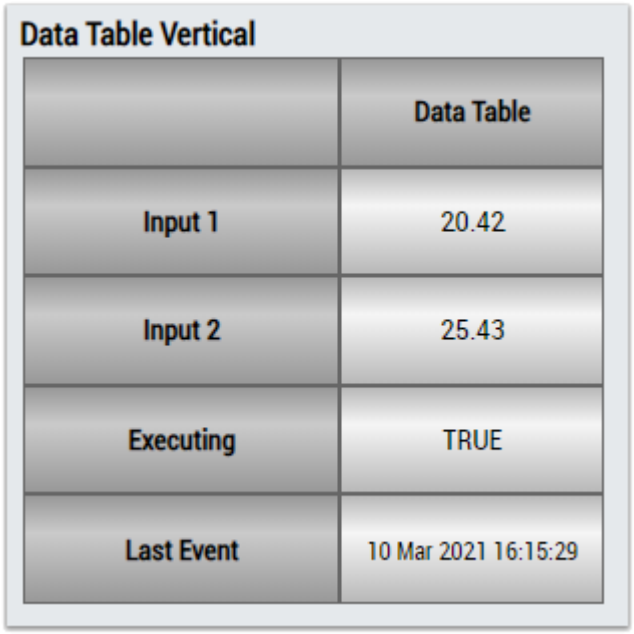
2. The SingleValue control visualizes the output values Count On and Last Event.

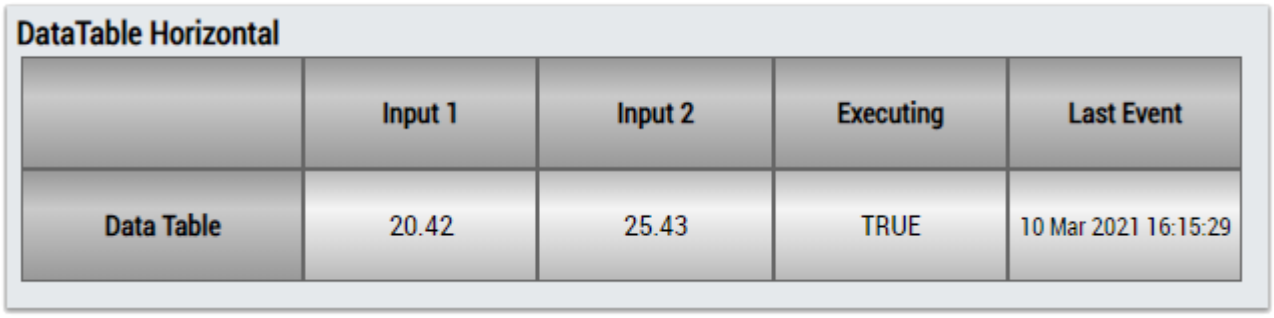


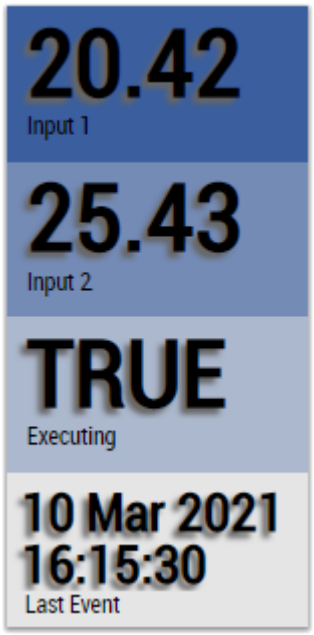
3. The BinaryState control visualizes the output value Is On.



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

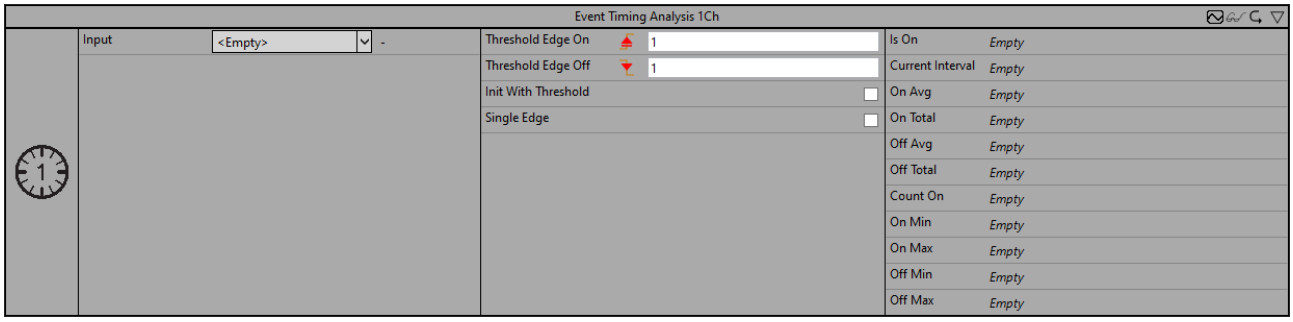






Alternatively, custom HMI controls can be mapped in the Edge Counter On Off 2Ch algorithm using the Mapping Wizard.

6.5.1.7 Event Timing Analysis 1Ch



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.

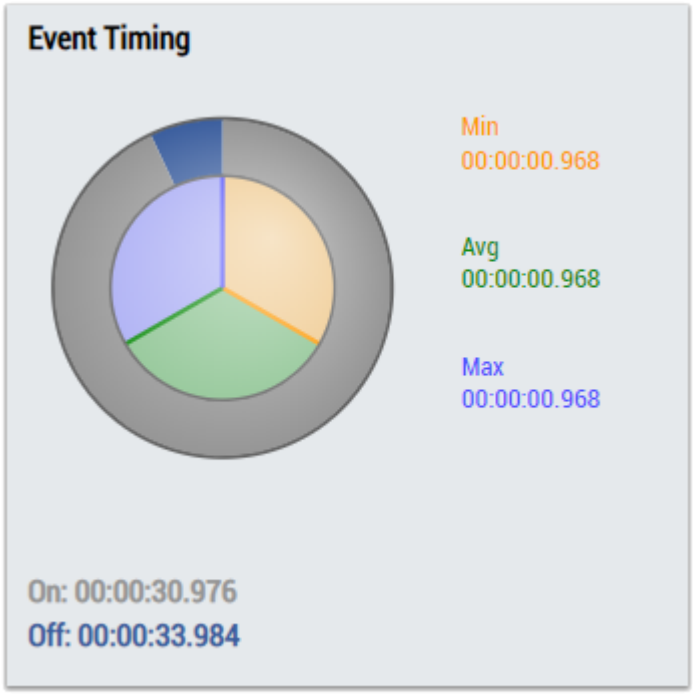
Output Values

- **Is On:** Shows *TRUE* within the time range between on-event and off-event, otherwise *FALSE*.
- **Current Interval:** Shows the time of the current interval.
- **On Min:** Shows the minimal time the "Is On"-value is *TRUE*.
- **On Max:** Shows the maximal time the "Is On"-value is *TRUE*.
- **On Avg:** Shows the average time the "Is On"-value is *TRUE*.
- **On Total:** Shows the total time the "Is On"-value is *TRUE*.
- **Off Min:** Shows the minimal time the "Is On"-value is *FALSE*.
- **Off Max:** Shows the maximal time the "Is On"-value is *FALSE*.
- **Off Avg:** Shows the average time the "Is On"-value is *FALSE*.
- **Off Total:** Shows the total time the "Is On"-value is *FALSE*.
- **Count On:** Counts the amount of raised on-events.

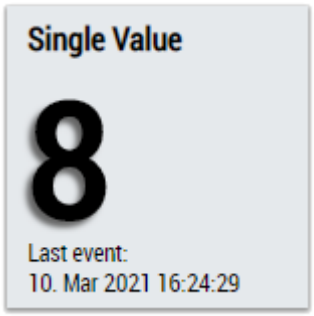
Standard HMI Controls

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

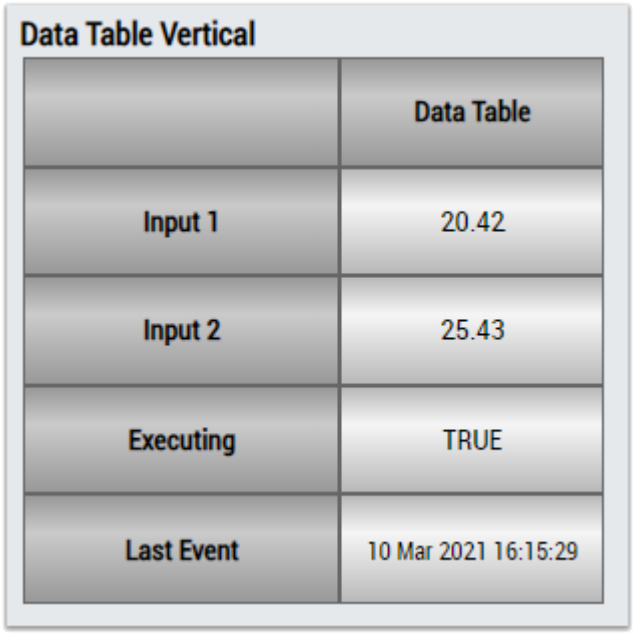
1. The EventTiming control visualizes the output values Is On, Count On, Current Interval, On Min, On Max, On Avg, On Total, Off Min, Off Max, Off Avg and Off Total.



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

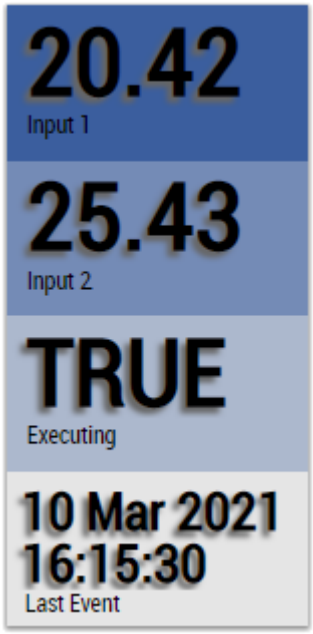


3. The Table Control or Multivalue Control visualizes all output values: Is On, Count On, Current Interval, On Min, On Max, On Avg, On Total, Off Min, Off Max, Off Avg, Off Total.



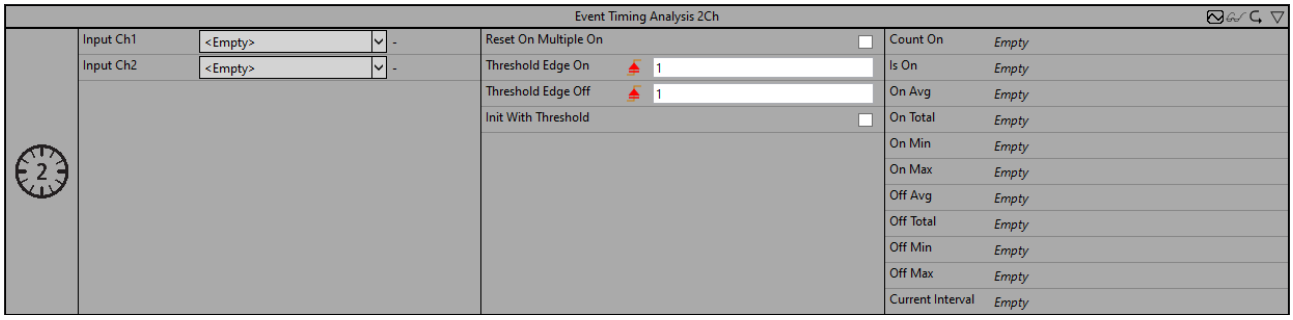
DataTable Horizontal

	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 1Ch algorithm using the Mapping Wizard.

6.5.1.8 Event Timing Analysis 2Ch



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.

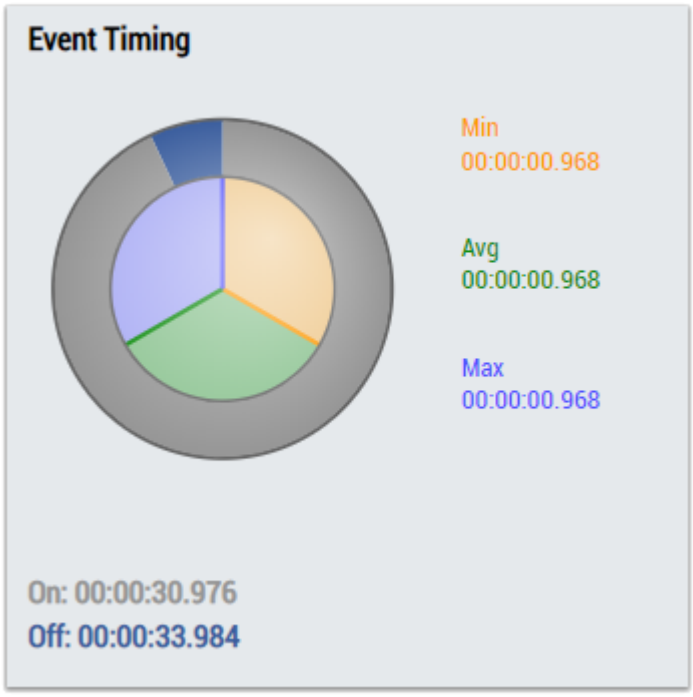
Output Values

- **Is On:** Shows *TRUE* within the time range between on-event and off-event, otherwise *FALSE*.
- **Current Interval:** Shows the time of the current interval.
- **On Min:** Shows the minimal time the "Is On"-value is *TRUE*.
- **On Max:** Shows the maximal time the "Is On"-value is *TRUE*.
- **On Avg:** Shows the average time the "Is On"-value is *TRUE*.
- **On Total:** Shows the total time the "Is On"-value is *TRUE*.
- **Off Min:** Shows the minimal time the "Is On"-value is *FALSE*.
- **Off Max:** Shows the maximal time the "Is On"-value is *FALSE*.
- **Off Avg:** Shows the average time the "Is On"-value is *FALSE*.
- **Off Total:** Shows the total time the "Is On"-value is *FALSE*.
- **Count On:** Counts the amount of raised on-events.

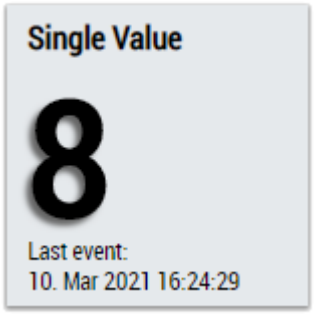
Standard HMI Controls

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

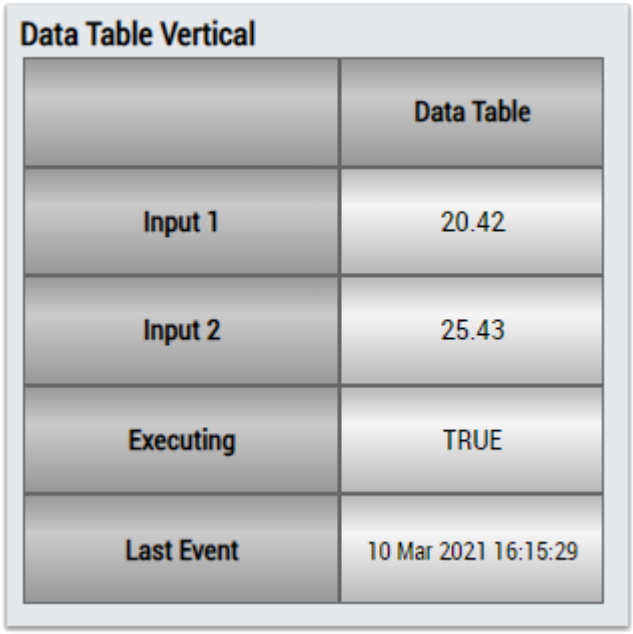
1. The EventTiming control visualizes the output values Is On, Count On, Current Interval, On Min, On Max, On Avg, On Total, Off Min, Off Max, Off Avg and Off Total.

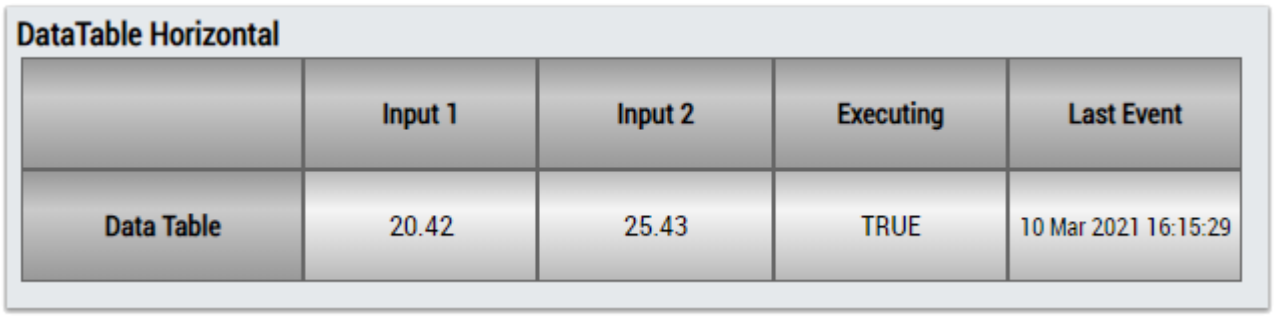


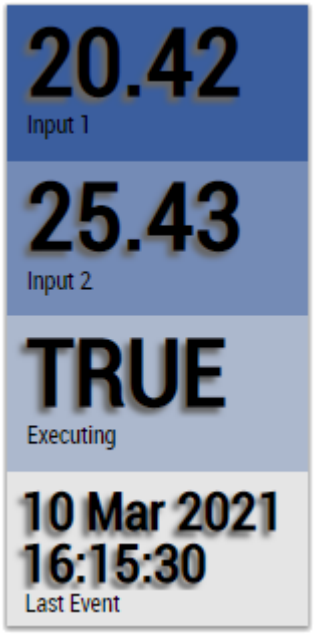
2. The SingleValue control visualizes the output value Count On.



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.

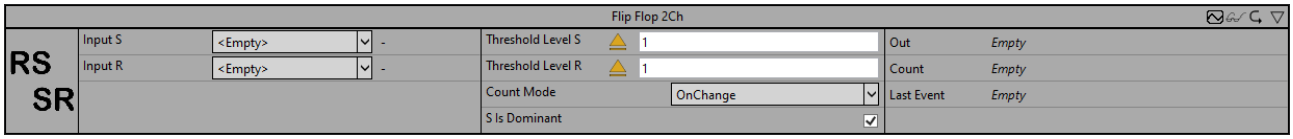






Alternatively, customer-specific HMI controls can be mapped in the Event Timing Analysis 2Ch algorithm using the Mapping Wizard.

6.5.1.9 Flip Flop 2Ch



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.

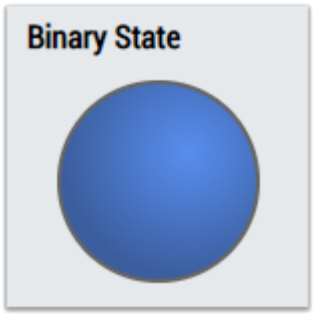
Output values

- **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*.



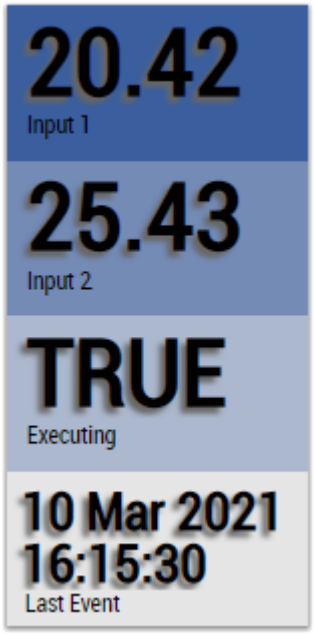
2. The Table Control and Multivalue control visualize all output values: *Out*, *Count*, *Last Event*.

Data Table Vertical

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

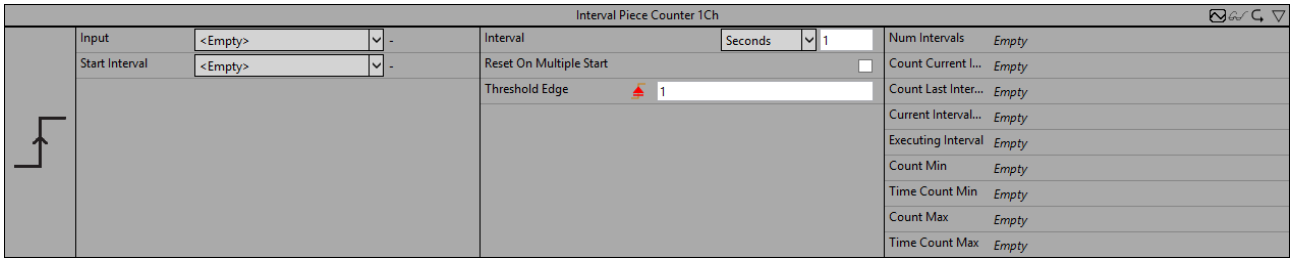
DataTable Horizontal

	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.

6.5.1.10 Interval Piece Counter 1Ch



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

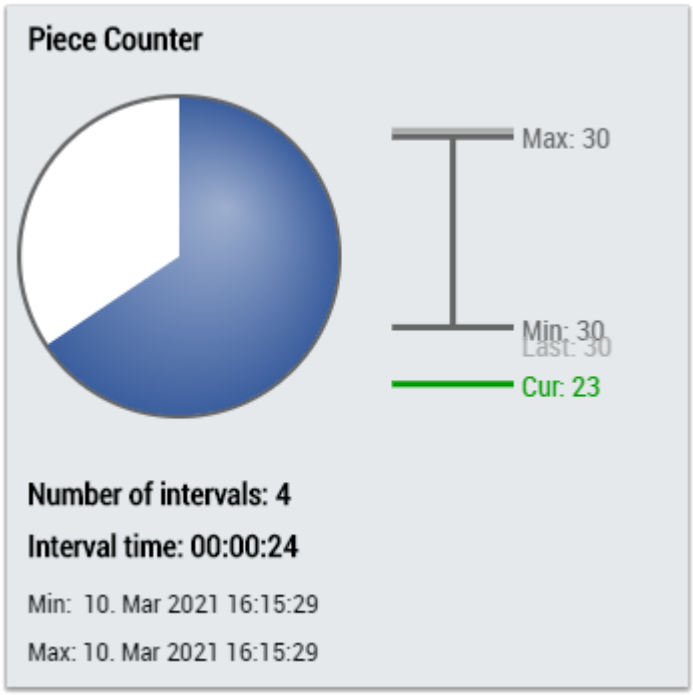
- **Executing Interval:** Indicates *True* if the calculation is active and the interval is running, otherwise *False*.
- **Num Intervals:** Indicates the number of intervals.
- **Count Last Interval:** Indicates the number of triggered events in the last interval.
- **Count Current Interval:** Indicates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

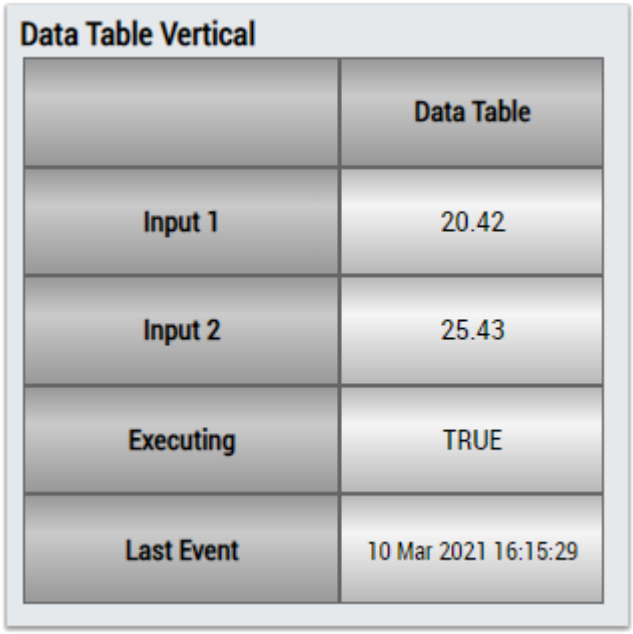
Standard HMI Controls

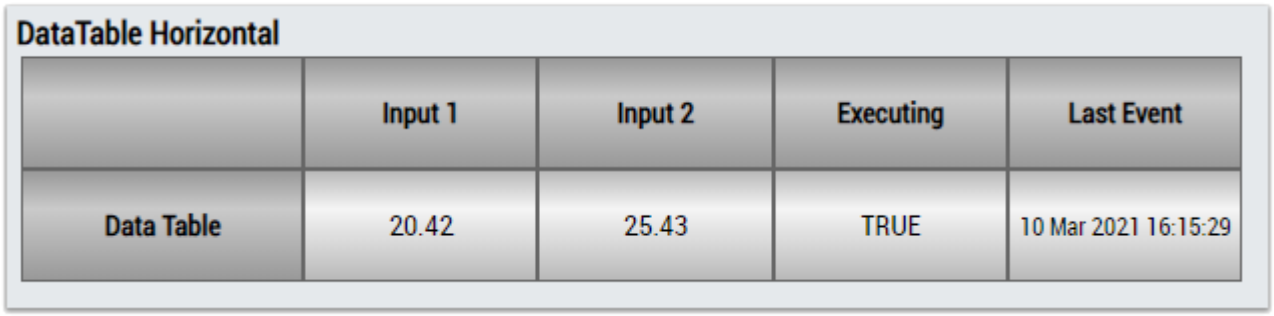
For the Interval Piece Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

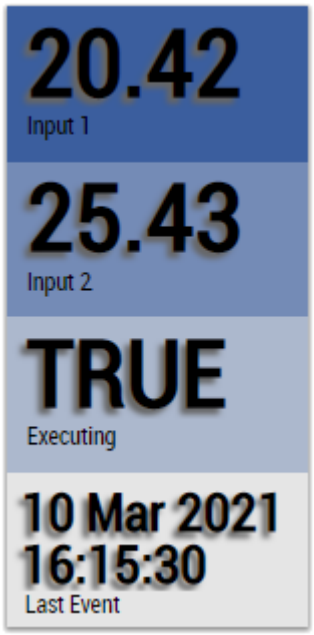
1. The PieceCounter control visualizes the output values Num Intervals, Count Last Interval, Count Current Interval, Count Min, Count Max, Time Count Min, Time Count Max and Time Current Interval.



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.

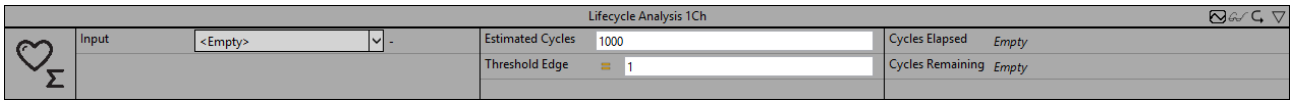






Alternatively, customer-specific HMI controls can be mapped in the Interval Piece Counter 1Ch algorithm using the Mapping Wizard.

6.5.1.11 Lifecycle Analysis 1Ch



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.

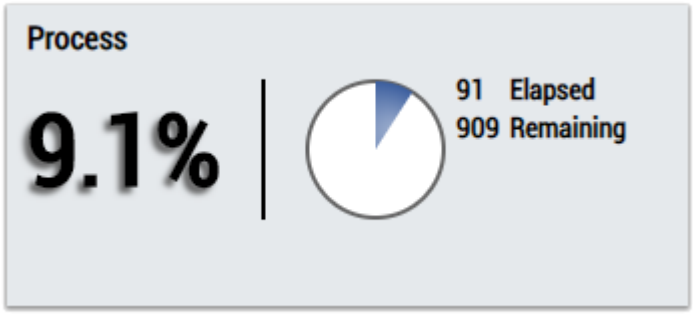
Output Values

- **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.

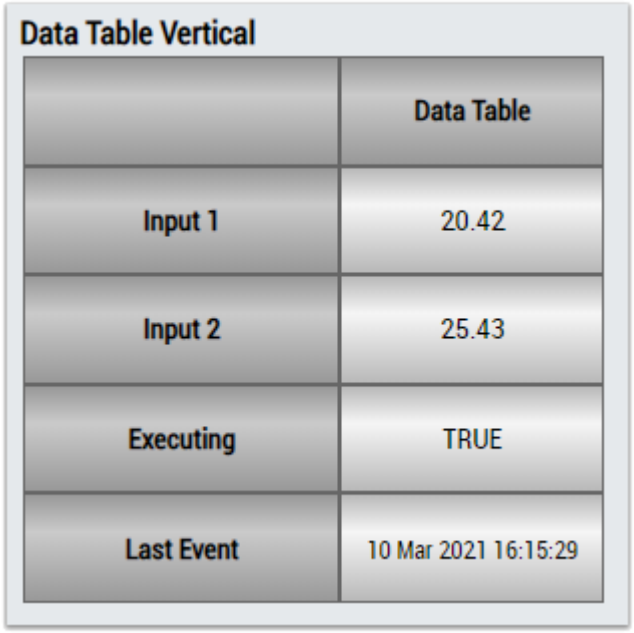
Standard HMI Controls

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.

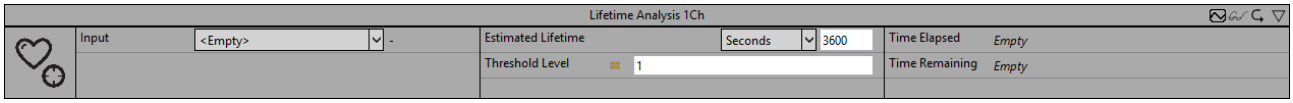


DataTable Horizontal

	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 Lifecycle Analysis 1Ch algorithm using the Mapping Wizard.

6.5.1.12 Lifetime Analysis 1Ch



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.

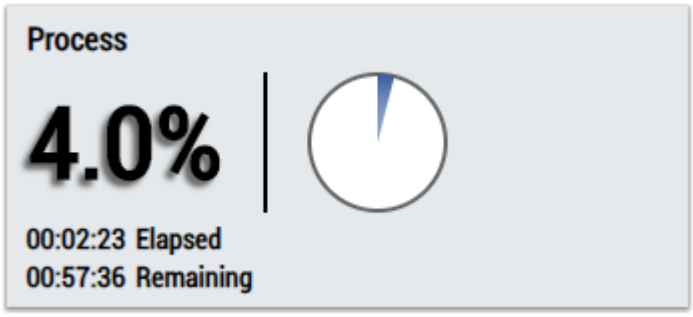
Output Values

- **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.

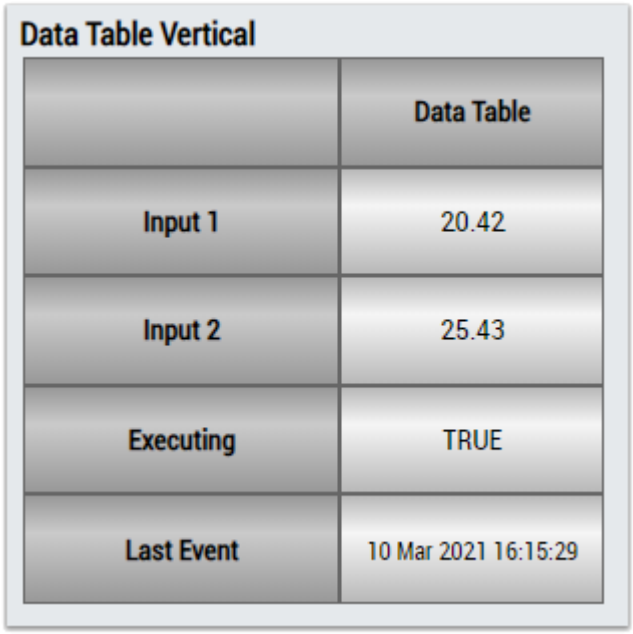
Standard HMI Controls

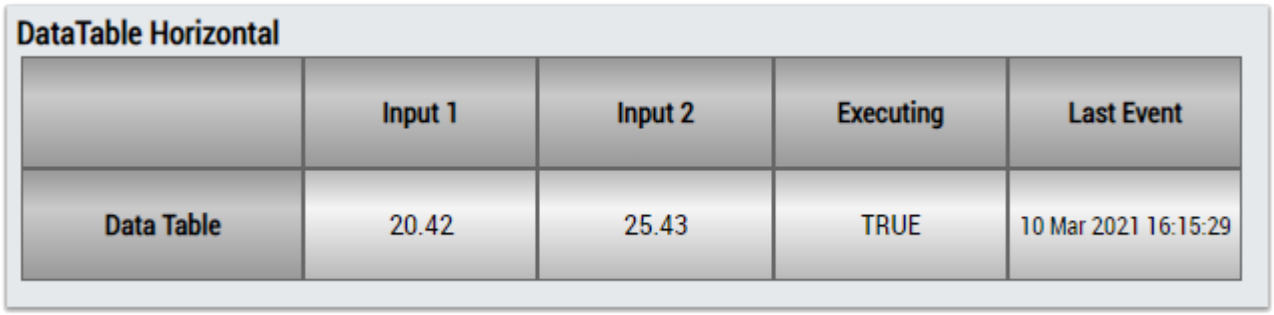
For the Lifetime Analysis 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

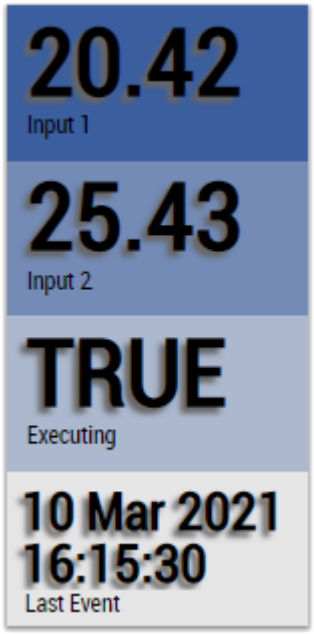
1. The Process control visualizes the output values Elapsed Lifetime and Remaining Lifetime.



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

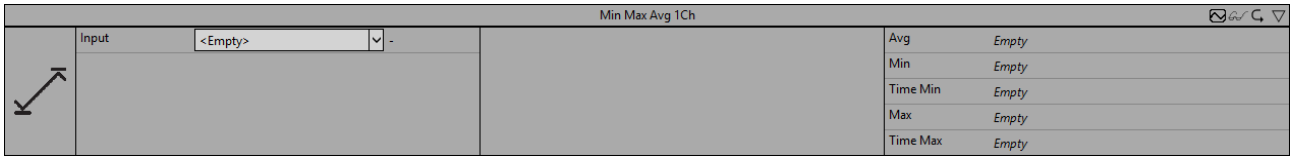






Alternatively, customer-specific HMI controls can be mapped in the Lifetime Analysis 1Ch algorithm using the Mapping Wizard.

6.5.1.13 Min Max Avg 1Ch



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*.

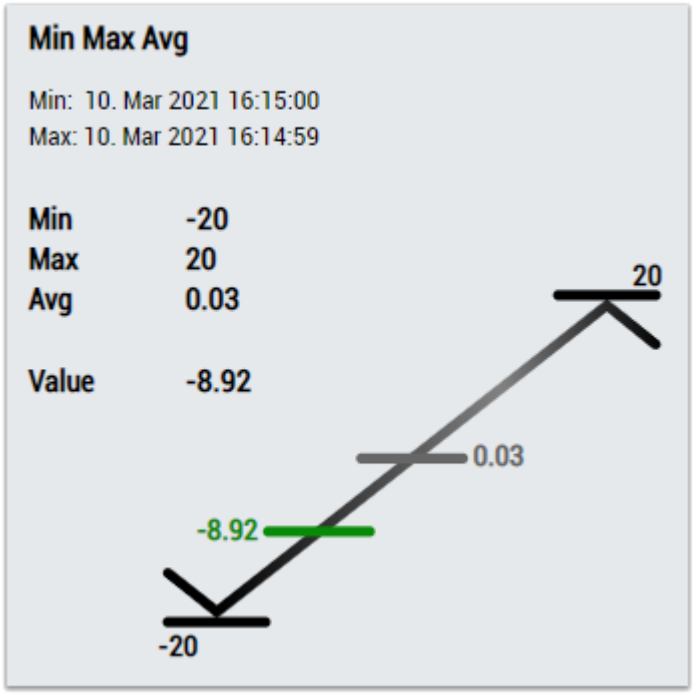
Output values

- **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.

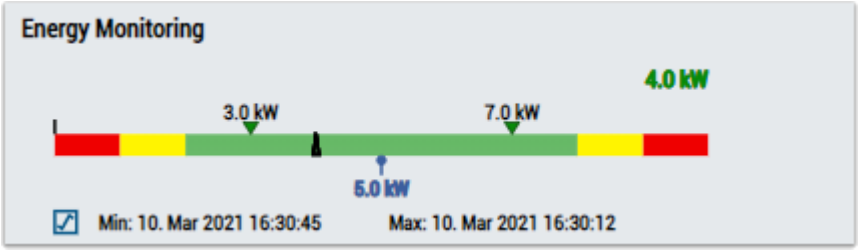
Standard HMI Controls

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

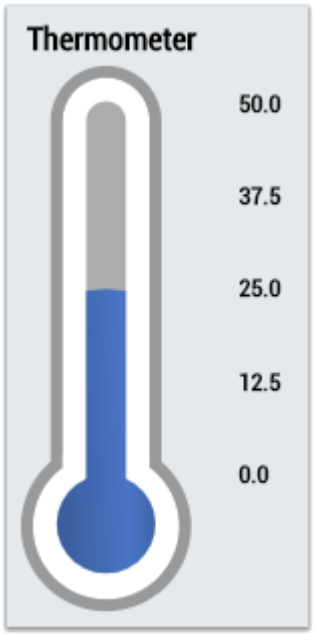
1. The MinMaxAvg control visualizes the output values Min, Max, Avg, Time Min and Time Max as well as the input value of the data.



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

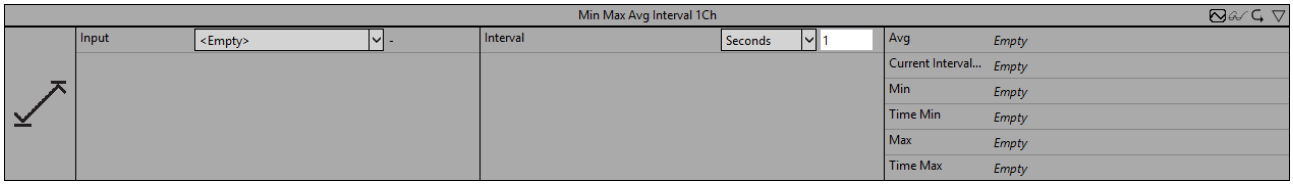
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

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

Alternatively, customer-specific HMI controls can be mapped in the Min Max Avg 1Ch algorithm using the Mapping Wizard.

6.5.1.14 Min Max Avg Interval 1Ch



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.

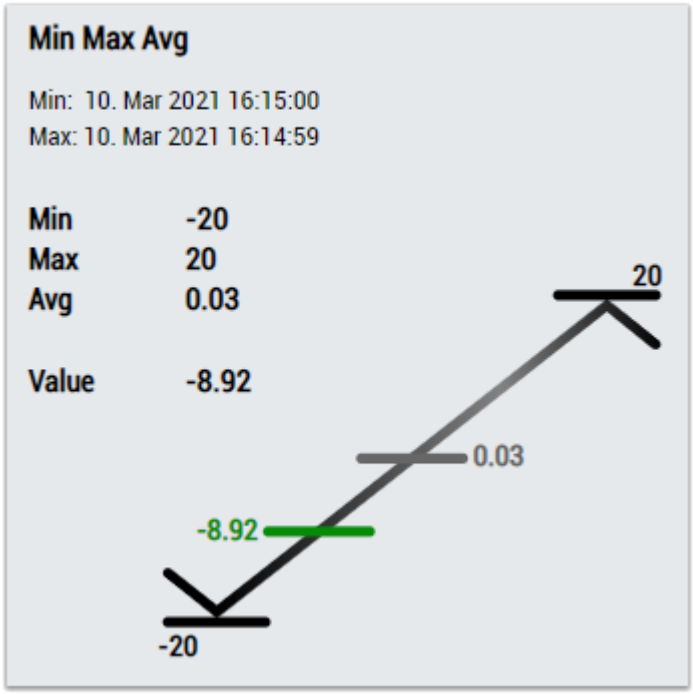
Output values

- **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.

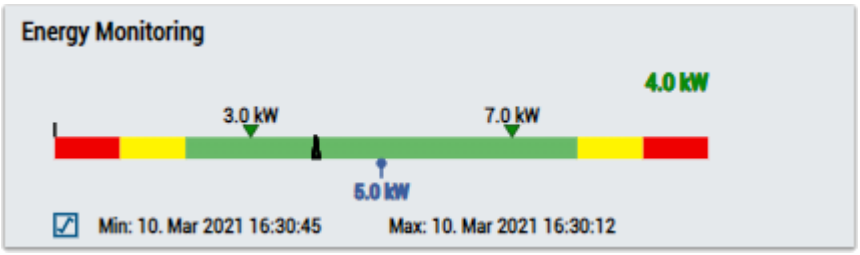
Standard HMI Controls

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

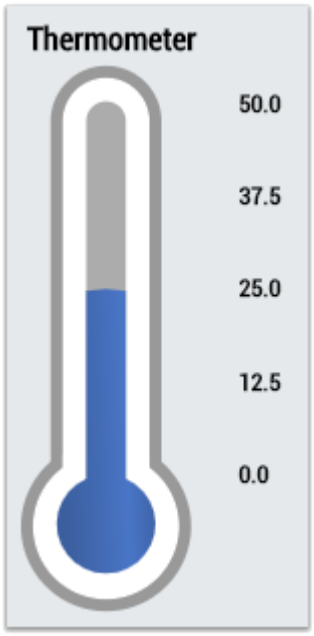
1. The MinMaxAvg control visualizes the output values Min, Max, Avg, Time Min and Time Max as well as the input value of the data.



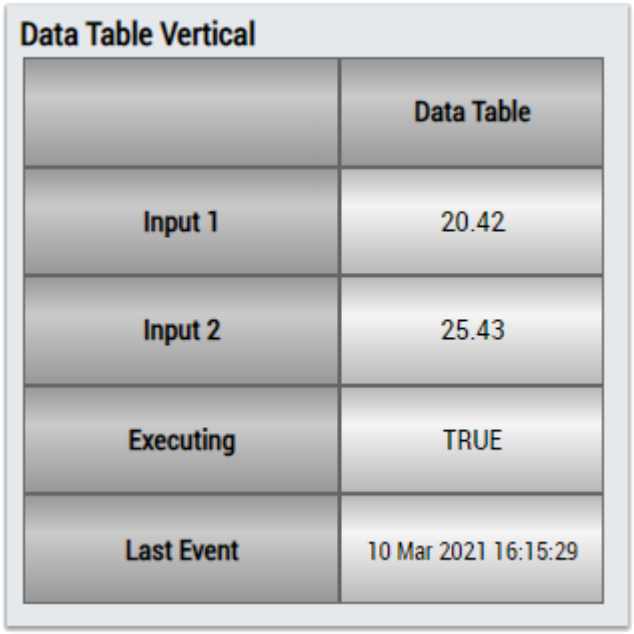
2. The EnergyMonitoring control visualizes the output values: Min, Max, Avg, Time Min, Time Max and the current input value.

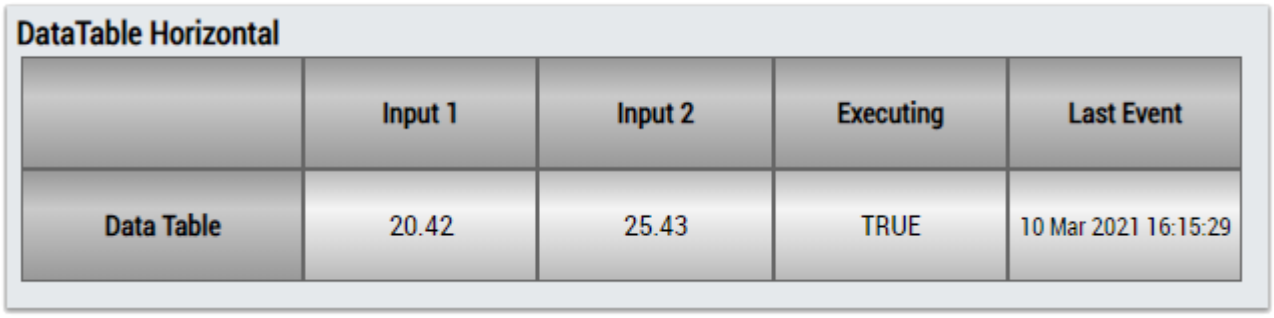


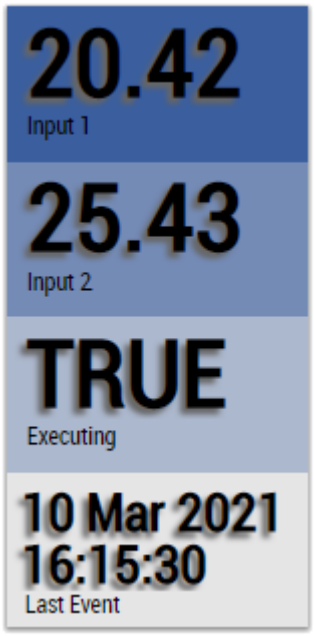
3. The Thermometer control visualizes the average (Avg) temperature.



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

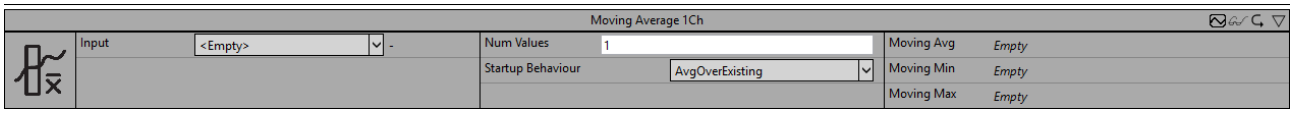






Alternatively, customer-specific HMI controls can be mapped in the Min Max Avg Interval 1Ch algorithm using the Mapping Wizard.

6.5.1.15 Moving Average 1Ch



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*.

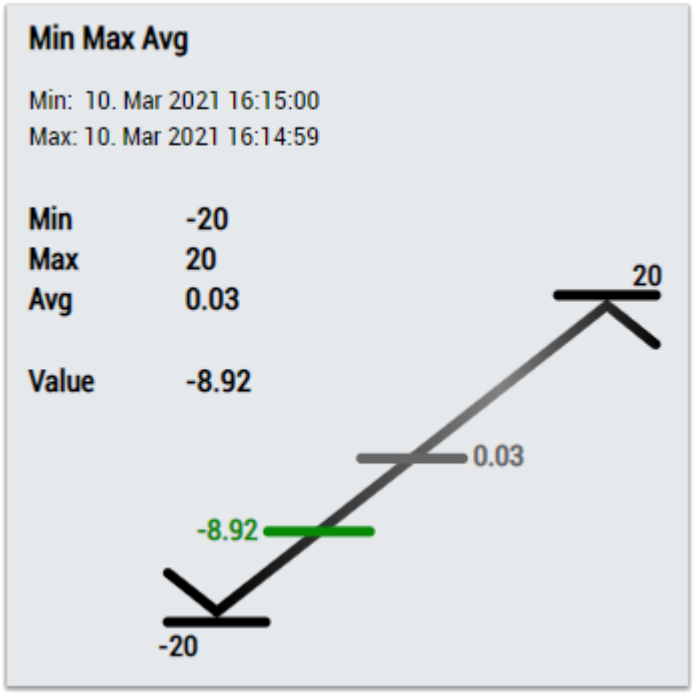
Output Values

- **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.

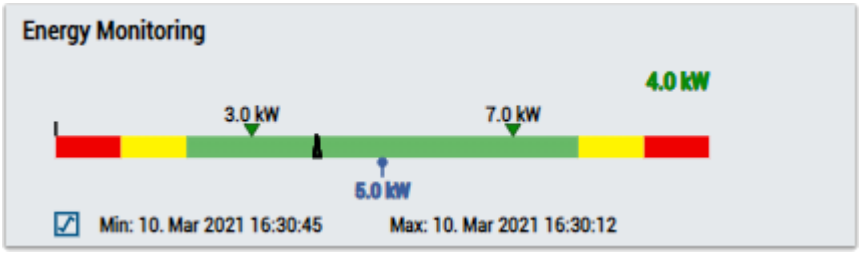
Standard HMI Controls

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

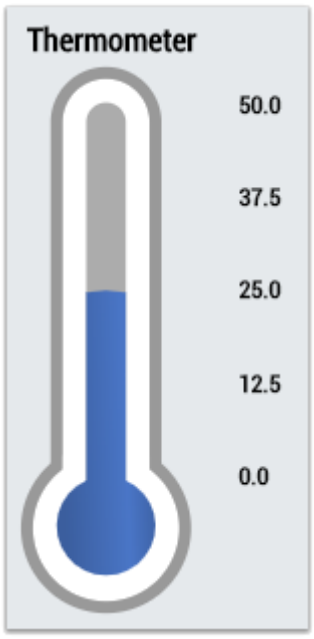
1. The MinMaxAvg control visualizes the output values Moving Min, Moving Max and Moving Avg as well as the input value of the data.



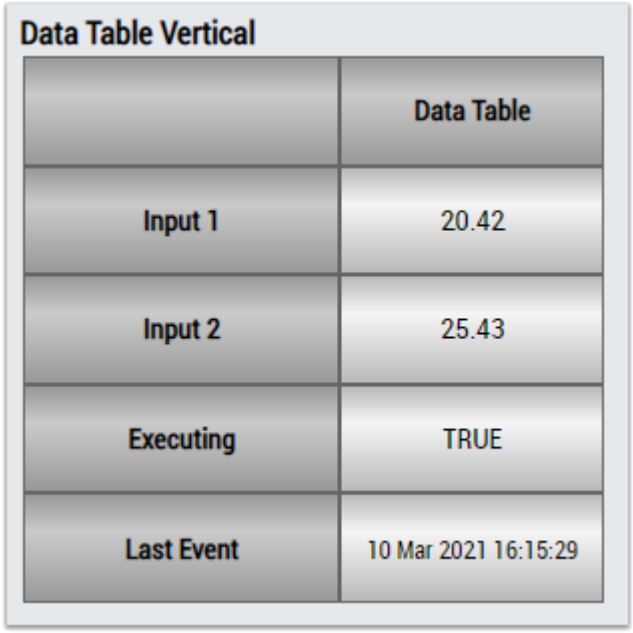
2. The EnergyMonitoring control visualizes the output values Moving Min, Moving Max and Moving Avg, and the input value of the data.



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

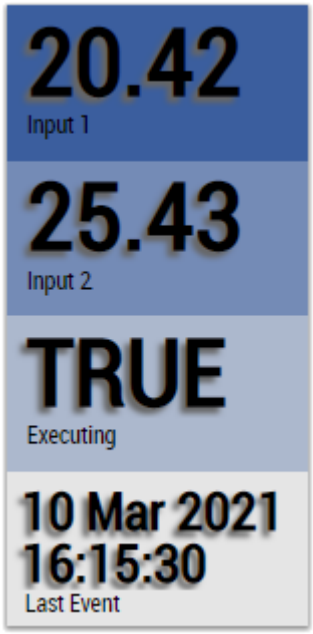


4. The Table Control or Multivalue Control visualizes all output values: Moving Min, Moving Max, Moving Avg.



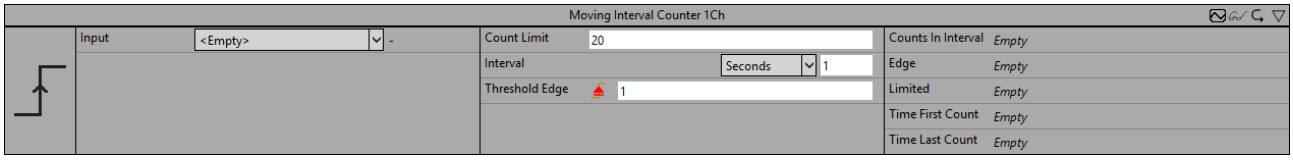
DataTable Horizontal

	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 Average 1Ch algorithm using the Mapping Wizard.

6.5.1.16 Moving Interval Counter 1Ch



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

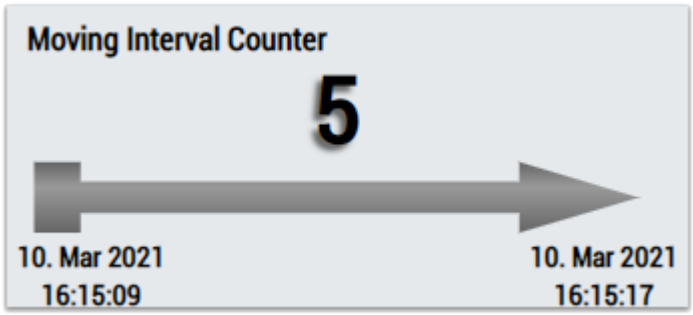
- **Edge:** Indicates *TRUE* at the time the event is triggered, otherwise *FALSE*.

- **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.

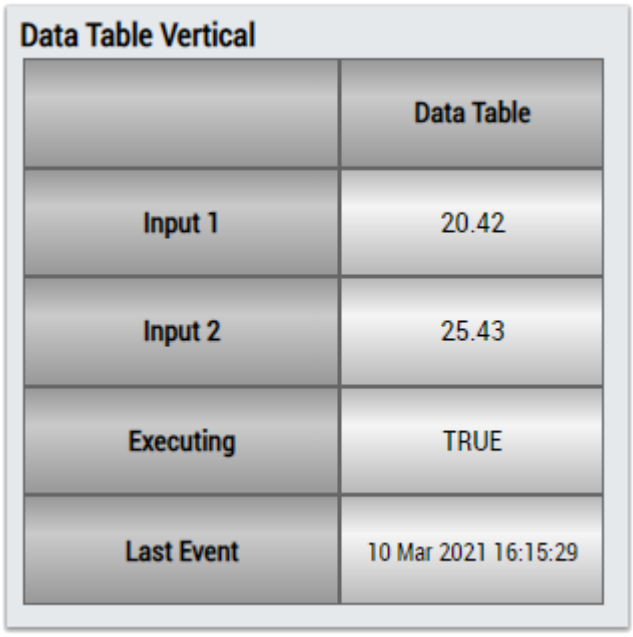
Standard HMI Controls

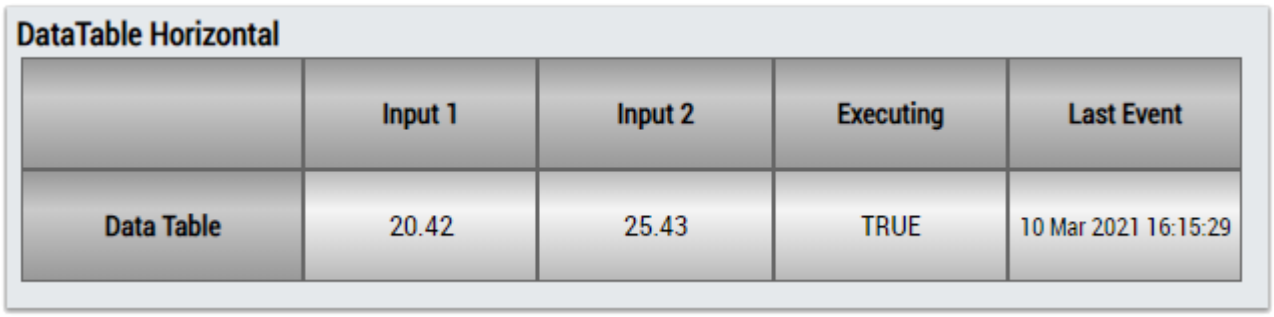
For the Moving Interval Counter 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

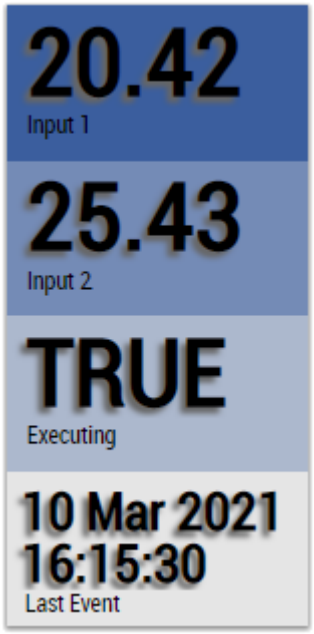
1. The MovingIntervalCounter control visualizes the output values Counts in Interval, Time First Count and Time Last Count.



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

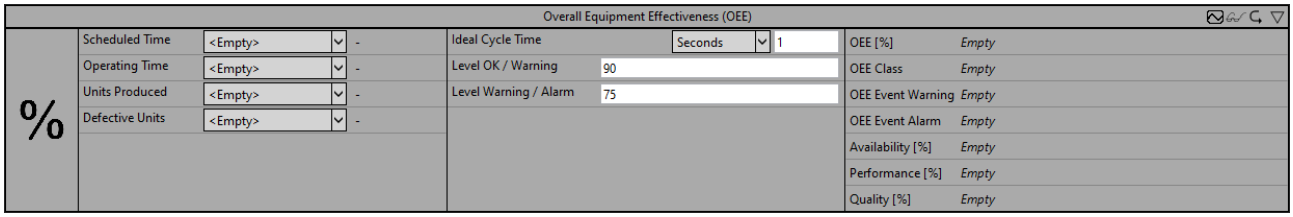






Alternatively, customer-specific HMI controls can be mapped in the Moving Interval Counter 1Ch algorithm using the Mapping Wizard.

6.5.1.17 Overall Equipment Effectiveness (OEE)



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

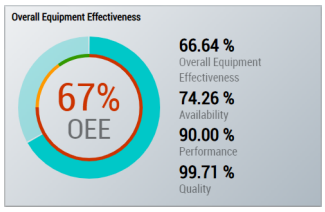
- **OEE:** Overall equipment effectiveness in percent. It is calculated by multiplying the availability factor, the performance factor and the quality factor.

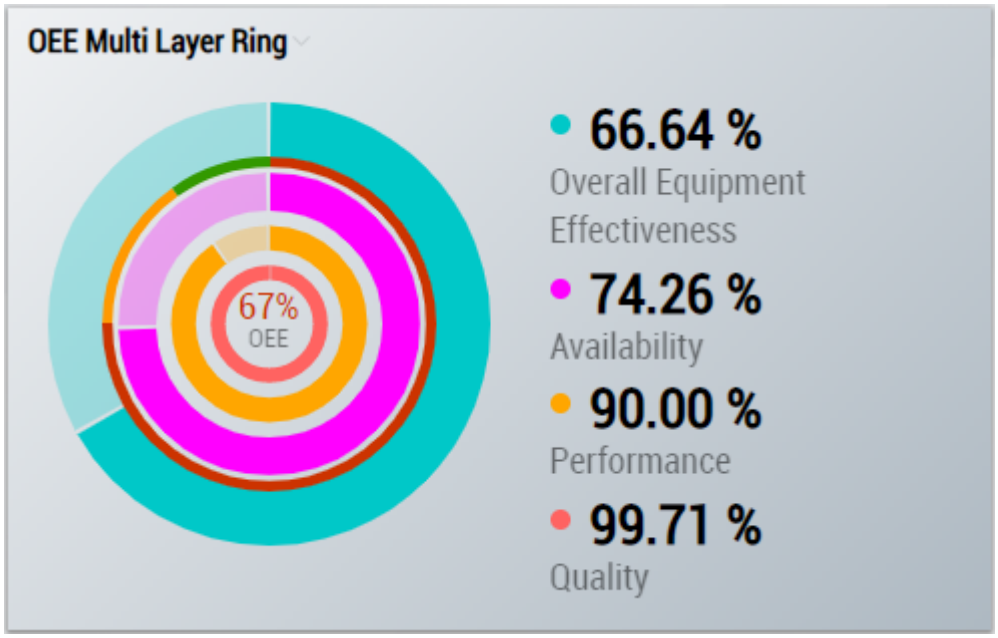
- **OEE Class:** Classification of overall equipment effectiveness.
- **OEE Event Warning:** Indicates the time of the last triggered warning event → the event can be dragged and dropped into the scope chart to display it as a trigger-event.
- **OEE Event Alarm:** Indicates the time of the last triggered 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*.





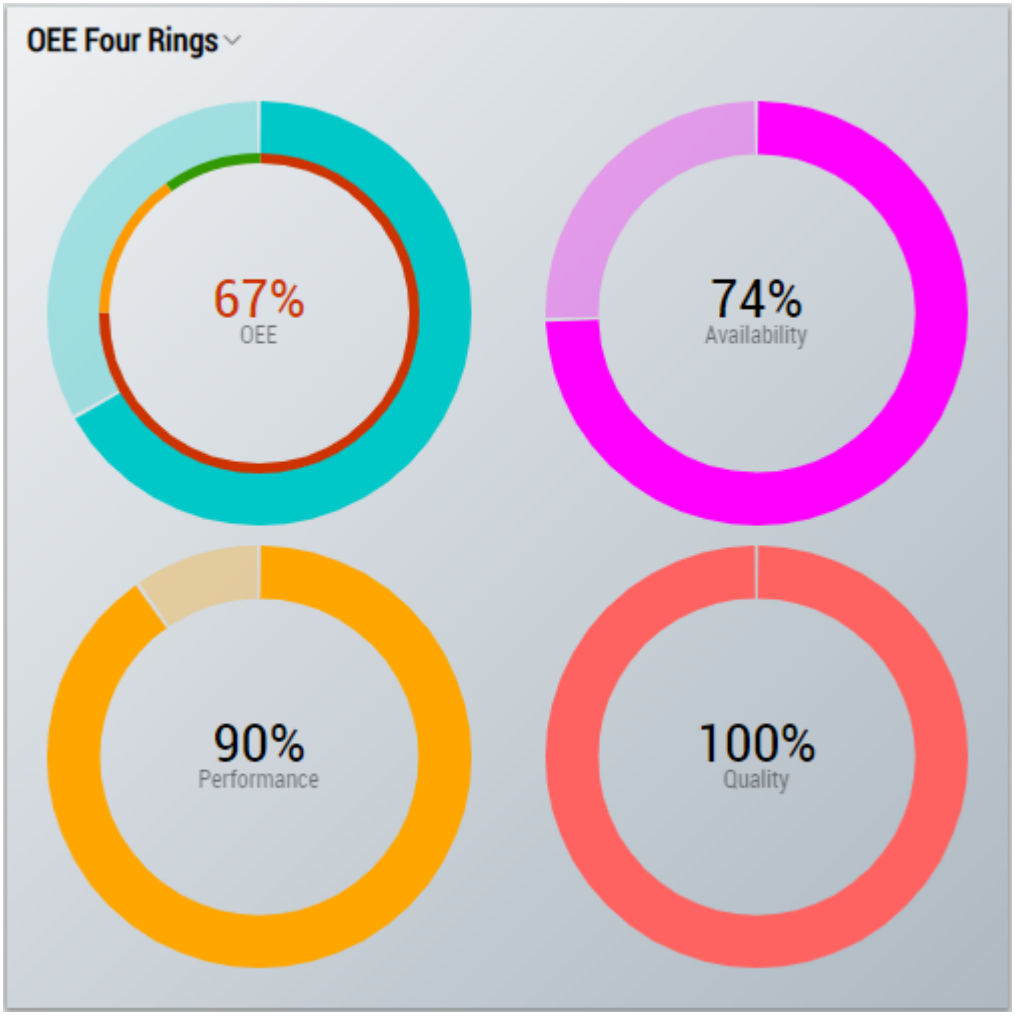
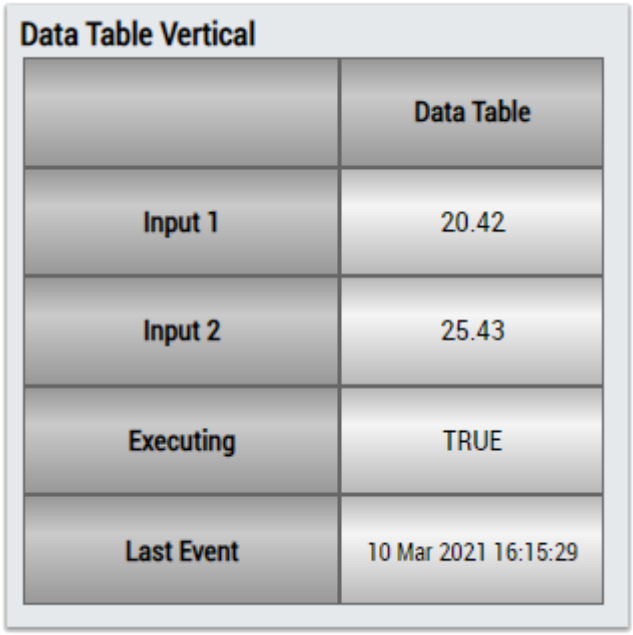


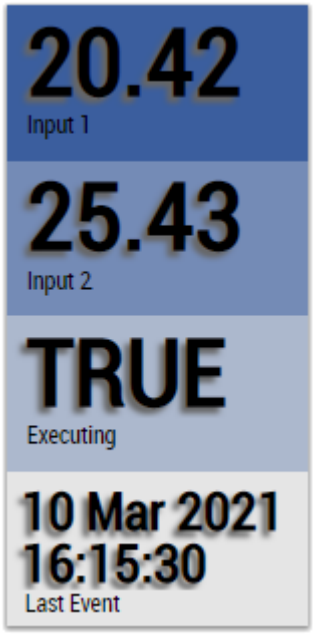
Fig. 1:

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



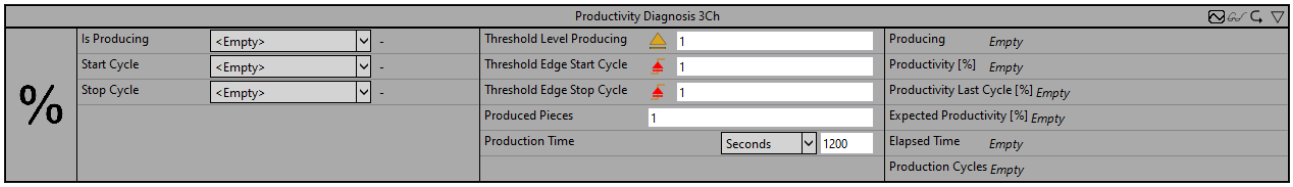
DataTable Horizontal

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

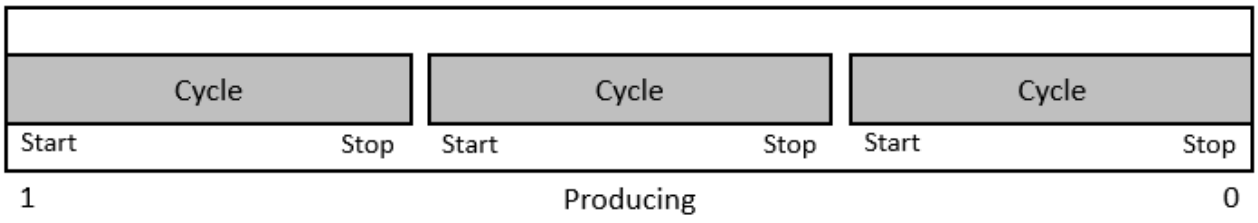


Alternatively, customer-specific HMI controls can be mapped in the *Overall Equipment Effectiveness (OEE)* algorithm using the Mapping Wizard.

6.5.1.18 Productivity Diagnosis 3Ch



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.

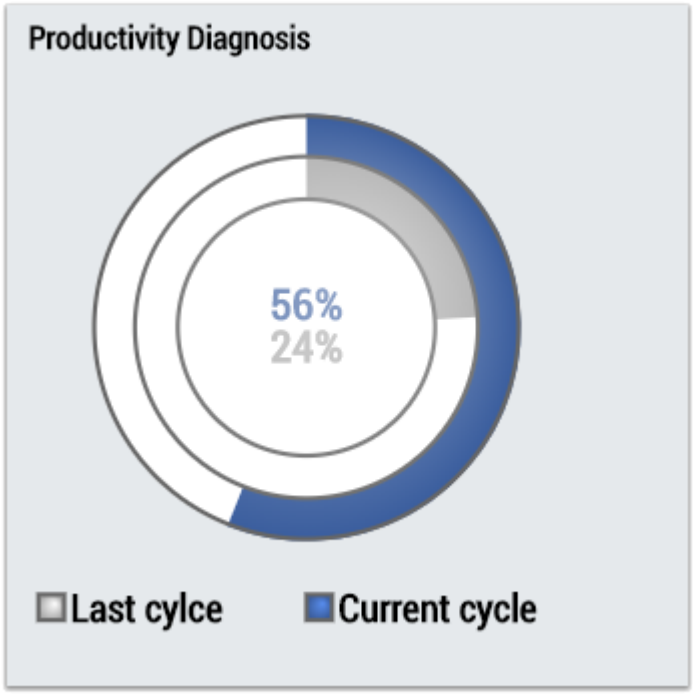
Output values

- **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.



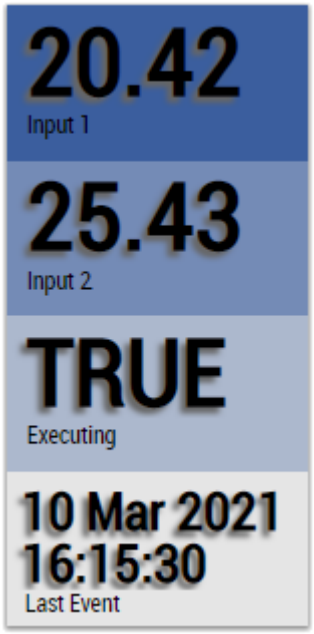
2. The Table Control or Multivalue Control visualizes all output values: New Result, Producing, Cycle Finished, Productivity, Productivity Last Cycle.

Data Table Vertical

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

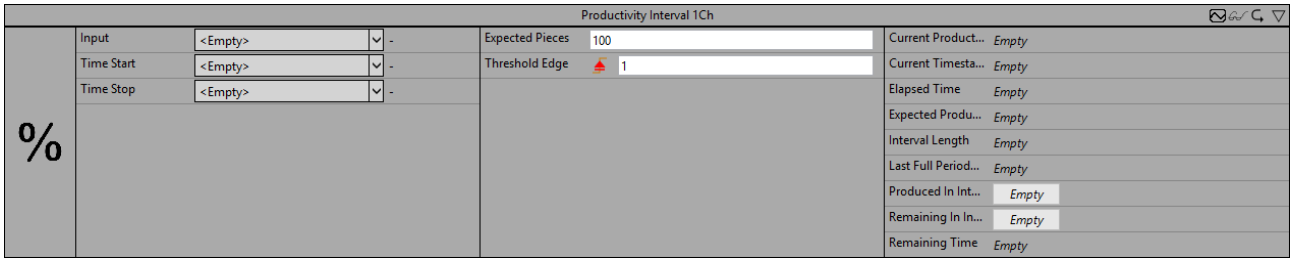
DataTable Horizontal

	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.

6.5.1.19 Productivity Interval 1Ch



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.

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.

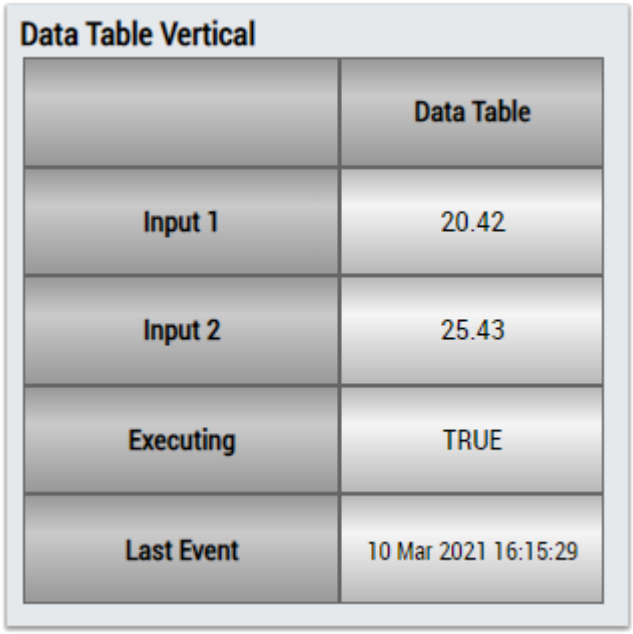
- **Current Productivity:** Current productivity of the interval in percent. Takes into account the length of the interval, the time already elapsed, the pieces to be produced and the pieces already produced. The output is in percent.
- **Expected Productivity:** Expected productivity of the interval 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.

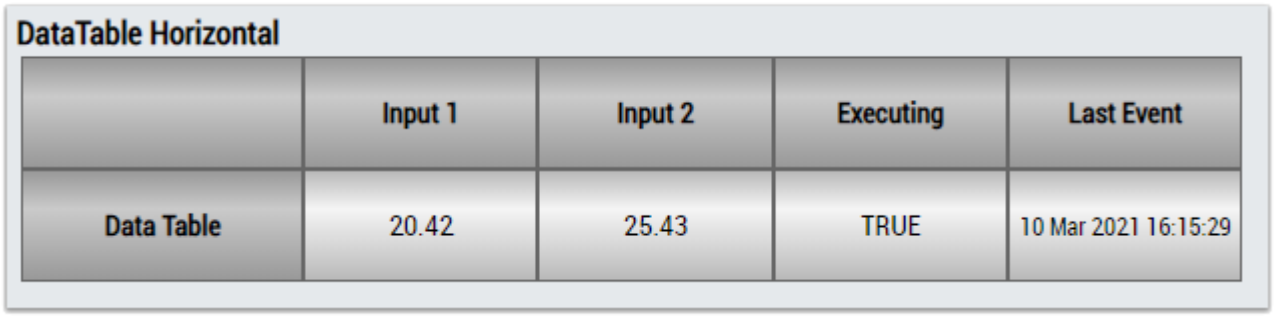
Standard HMI Controls

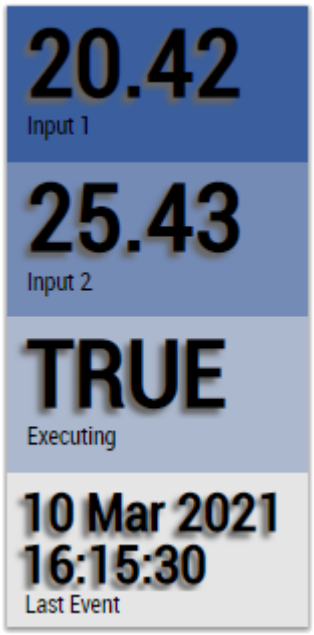
For the Productivity Interval 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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.

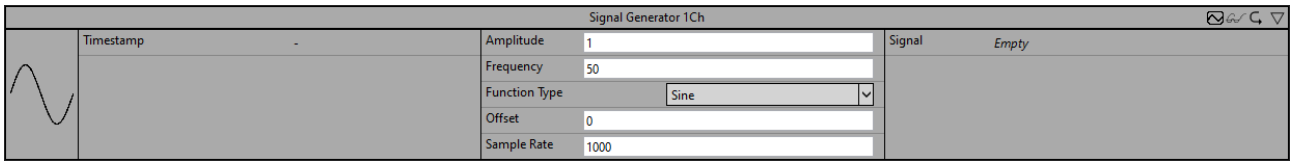






Alternatively, customer-specific HMI controls can be mapped in the Productivity Interval 1Ch algorithm using the Mapping Wizard.

6.5.1.20 Signal Generator 1Ch



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.

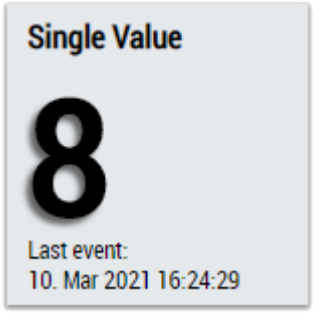
Output values

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

Standard HMI Controls

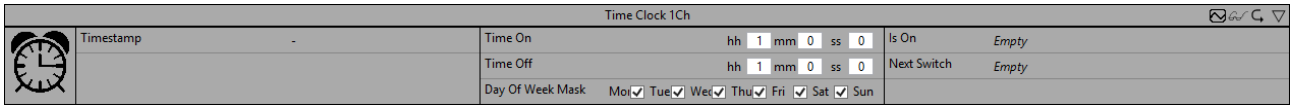
For the Signal Generator 1Ch, the following HMI controls are available for generating an Analytics Dashboard:

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.

6.5.1.21 Time Clock 1Ch



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

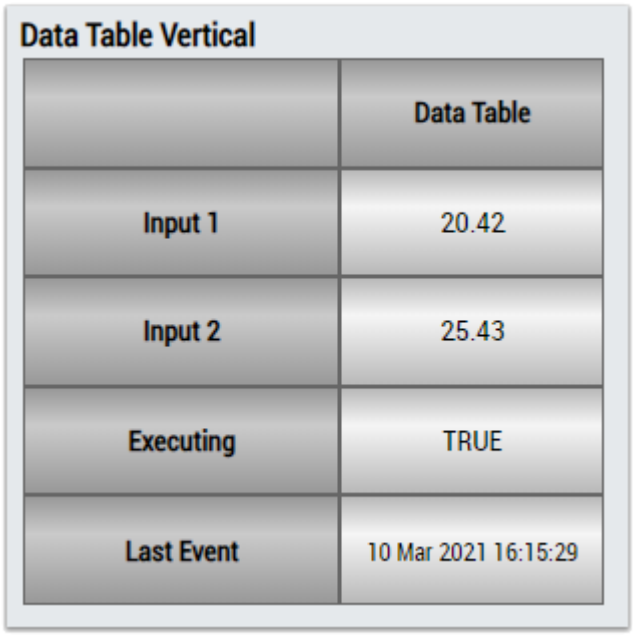
- **Is On:** Shows *TRUE*, if the time switch is on, otherwise *FALSE*.
- **Next Switch:** Shows the remaining time up to the next switch.

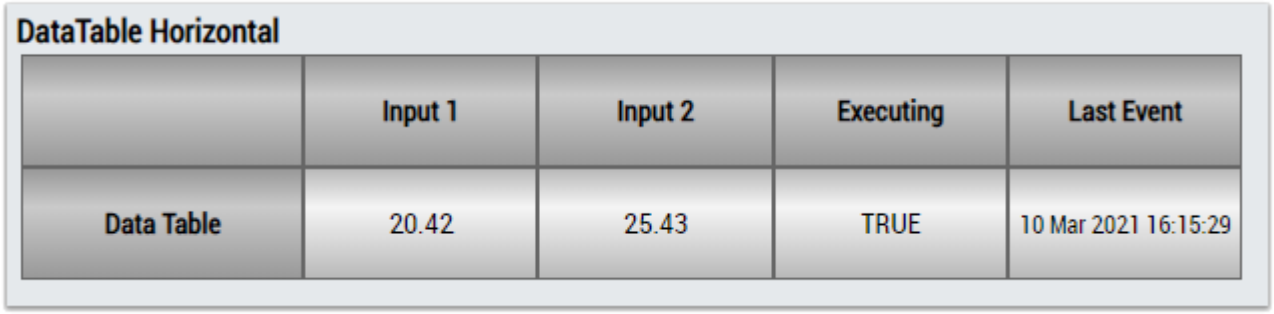
Standard HMI Controls

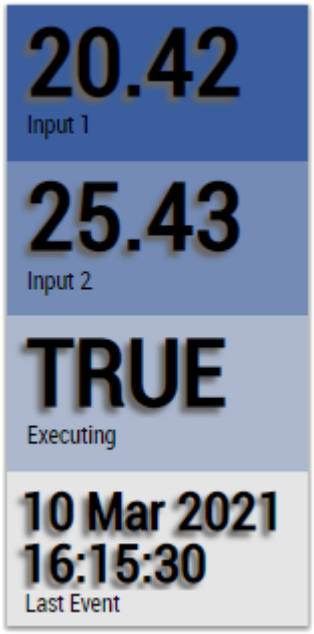
For the Timer Clock 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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

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

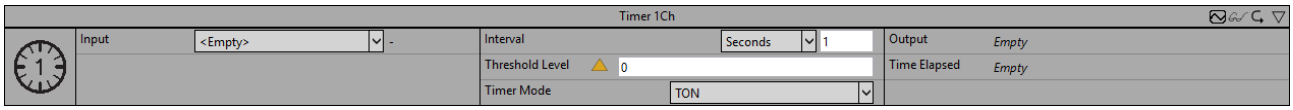






Alternatively, customer-specific HMI controls can be mapped in the Timer Clock 1Ch algorithm using the Mapping Wizard.

6.5.1.22 Timer 1Ch



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.

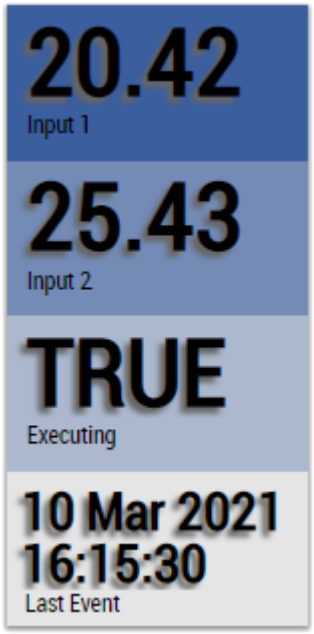
2. The Table Control or Multivalue Control visualizes all output values: Output, Elapsed Time.

Data Table Vertical

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

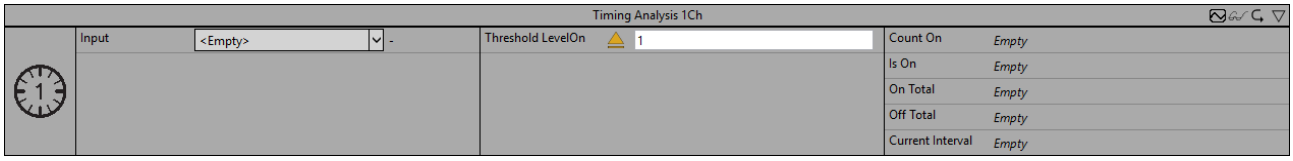
DataTable Horizontal

	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.

6.5.1.23 Timing Analysis 1Ch



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.

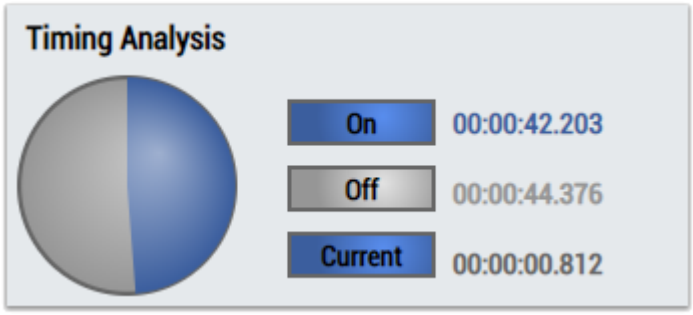
Output Values

- **Is On:** Shows *TRUE* within the time range of the on-period, otherwise *FALSE*.
- **Current Interval:** Shows the time of the current interval.
- **On Total:** Shows the total time the "Is On"-value is *TRUE*.
- **Off Total:** Shows the total time the "Is On"-value is *FALSE*.
- **Count On:** Counts the amount of raised on-events.

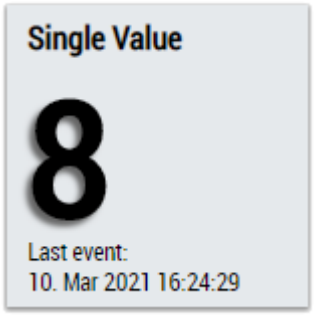
Standard HMI Controls

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

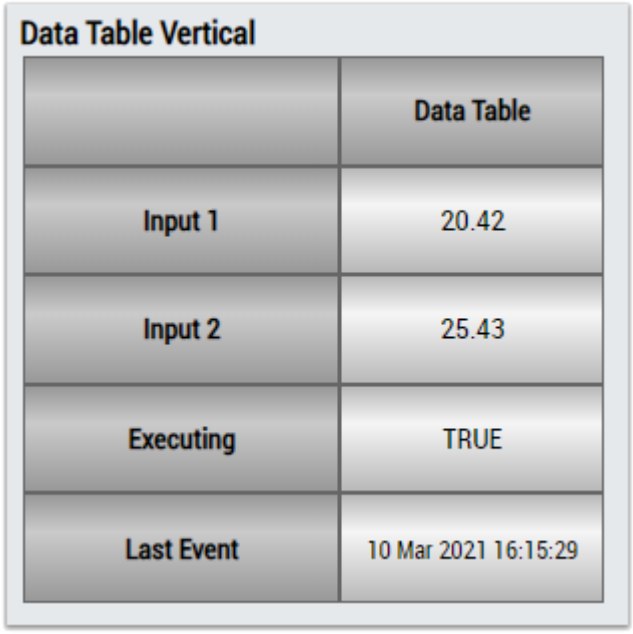
1. The TimingAnalysis Control visualizes the output values Is On, On Total, Off Total and Current Interval.

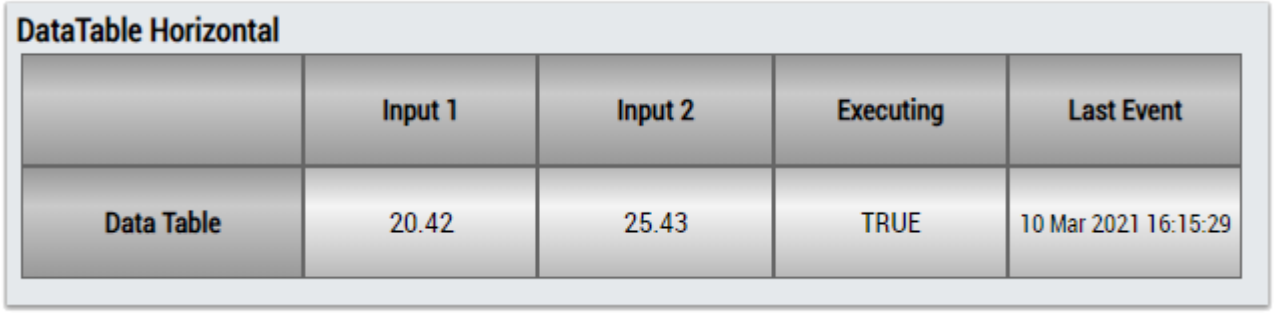


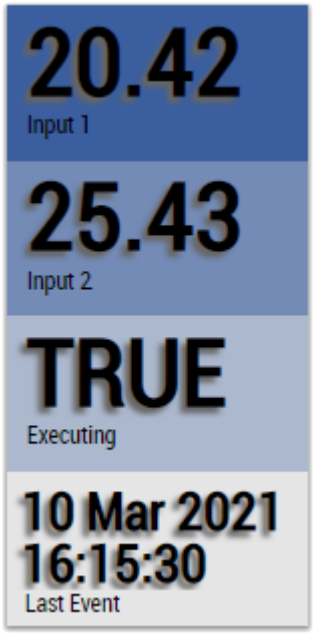
2. The SingleValue control visualizes the output value Count On.



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

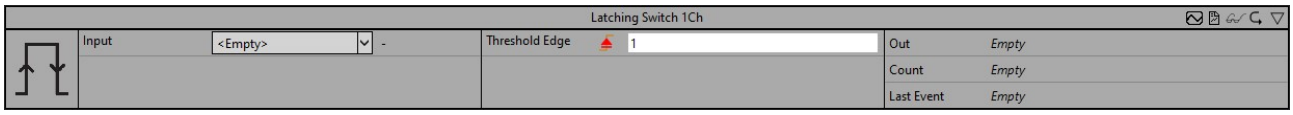






Alternatively, customer-specific HMI controls can be mapped in the Timing Analysis 1Ch algorithm using the Mapping Wizard.

6.5.1.24 Latching Switch 1Ch



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.

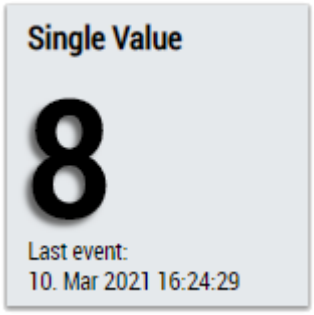
Output values

- **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.

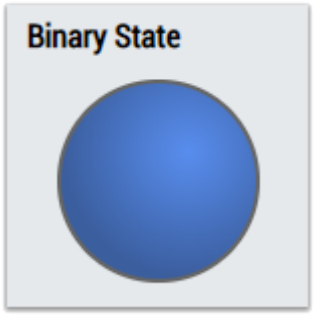
Standard HMI Controls

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

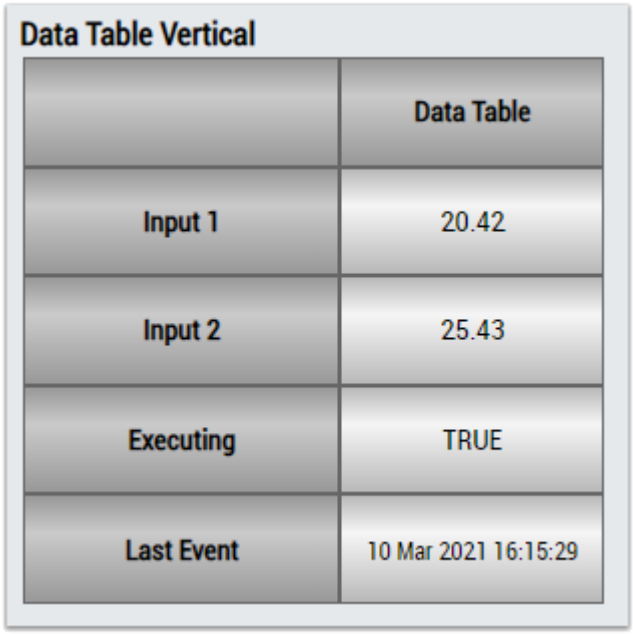
1. The SingleValue control visualizes the output values Count and Last Event.

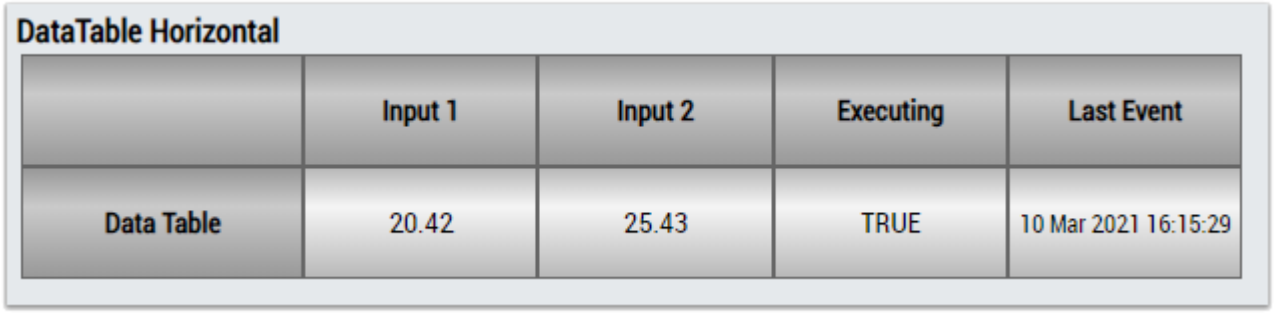


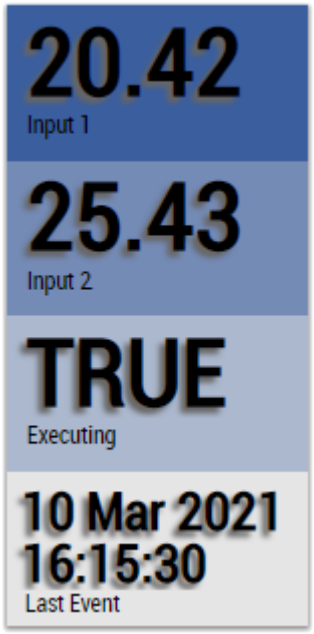
2. The BinaryState control visualizes the output value *Out*.



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





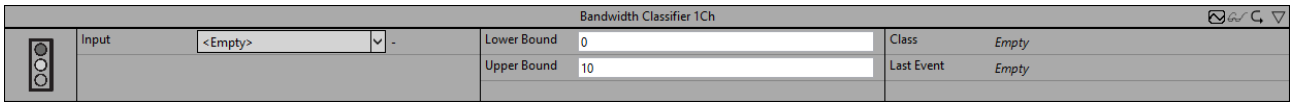


Alternatively, customer-specific HMI controls can be mapped using the Mapping Wizard.

6.5.2 Analytics - Classification

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

6.5.2.1 Bandwidth Classifier 1Ch



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.

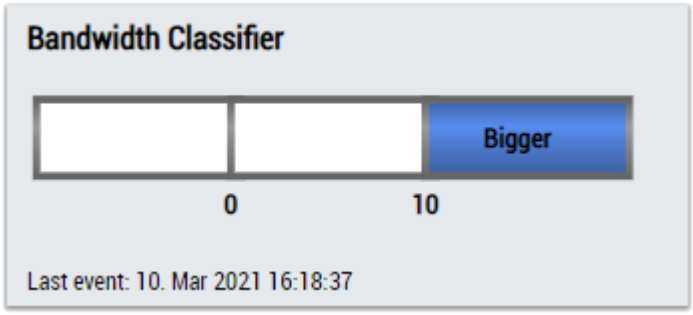
Output values

- **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.

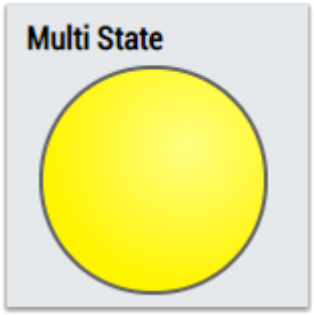
Standard HMI Controls

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

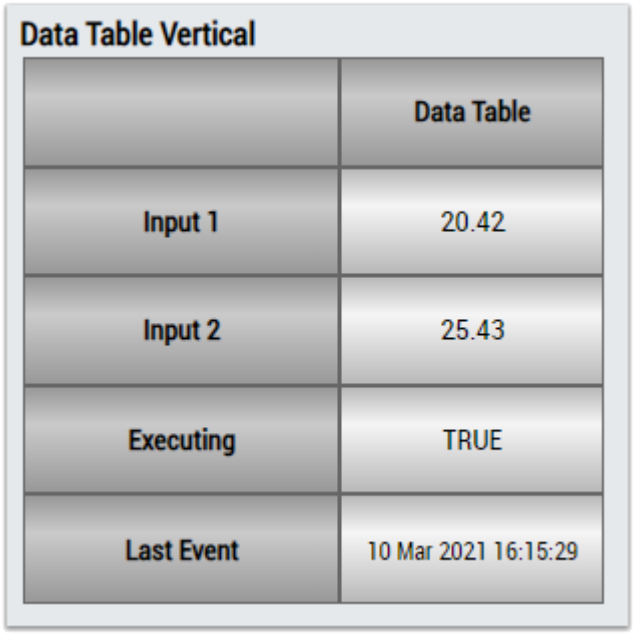
1. The BandwidthClassifier control visualizes the output values Class and Last Event and the configuration options Lower Bound and Upper Bound.

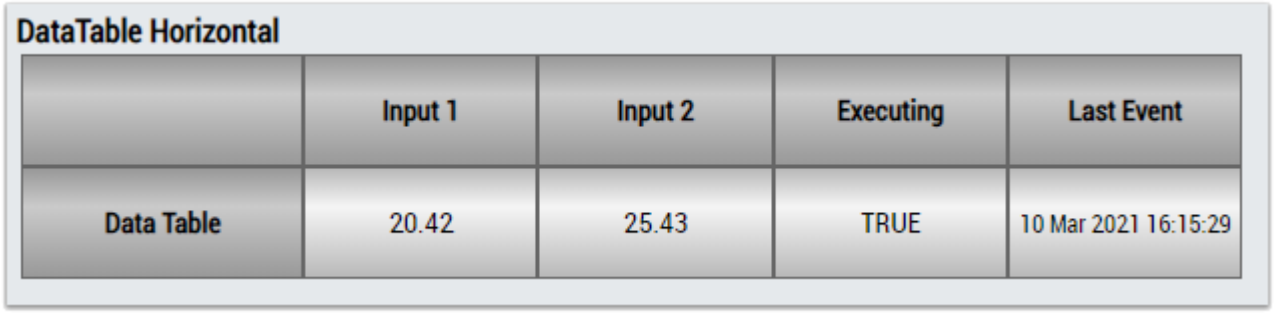


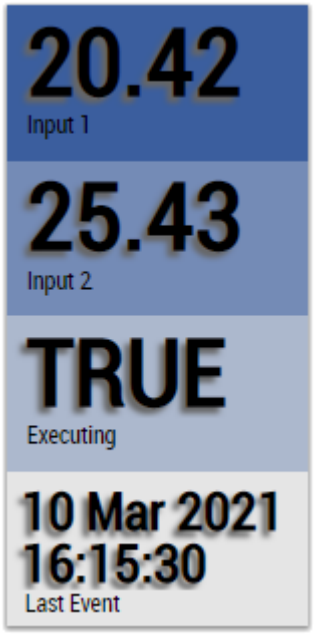
2. The MultiState control visualizes the output value Class.



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

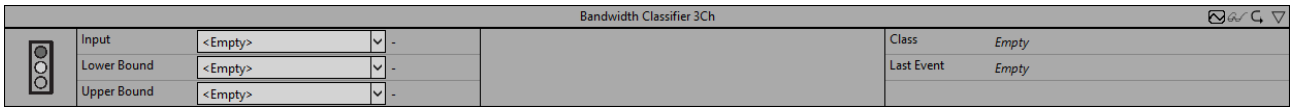






Alternatively, customer-specific HMI controls can be mapped in the Bandwidth Classifier 1Ch algorithm using the Mapping Wizard.

6.5.2.2 Bandwidth Classifier 3 Ch



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*.

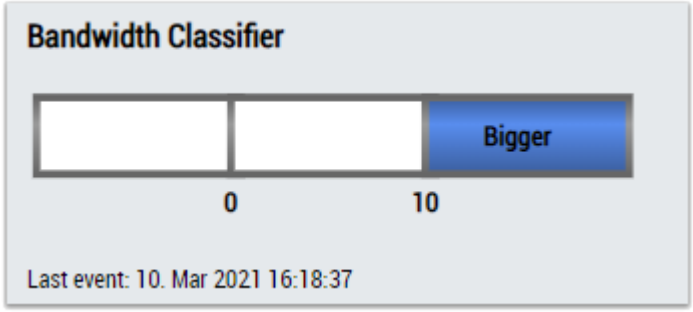
Output values

- **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.

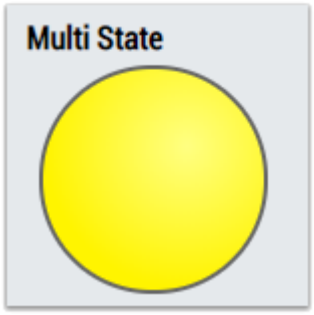
Standard HMI Controls

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

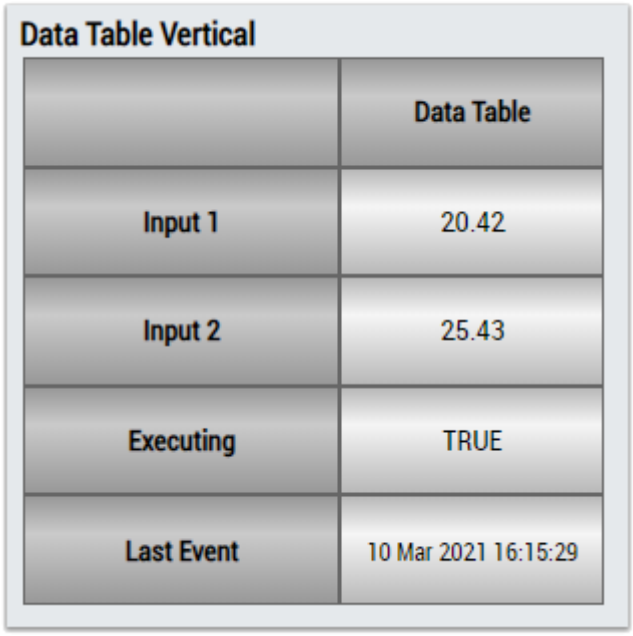
1. The BandwidthClassifier control visualizes the output values Class and Last Event and Input Lower Bound and Input Upper Bound.

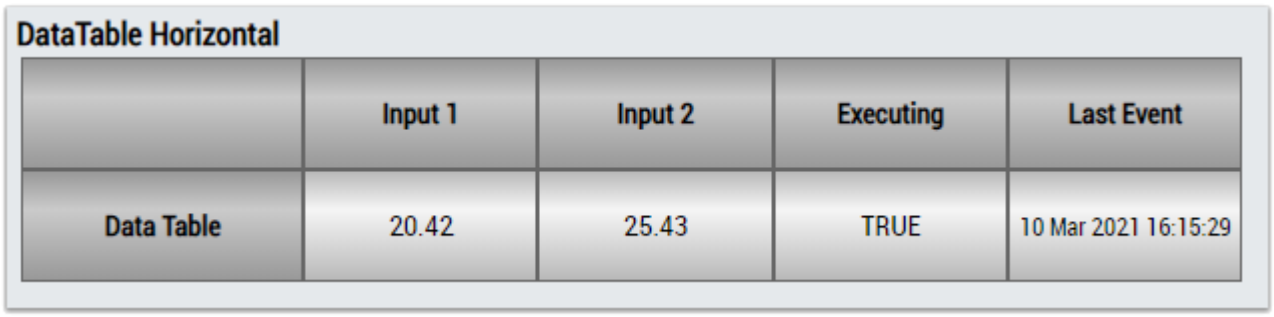


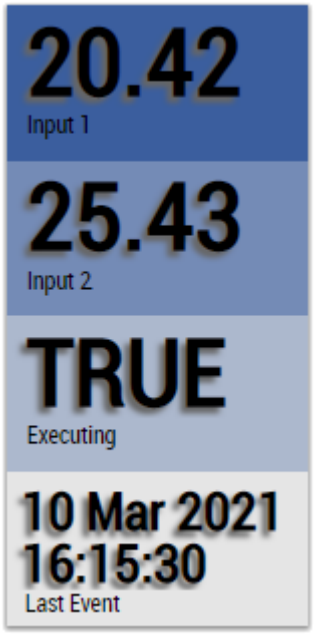
2. The MultiState control visualizes the output value Class.



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

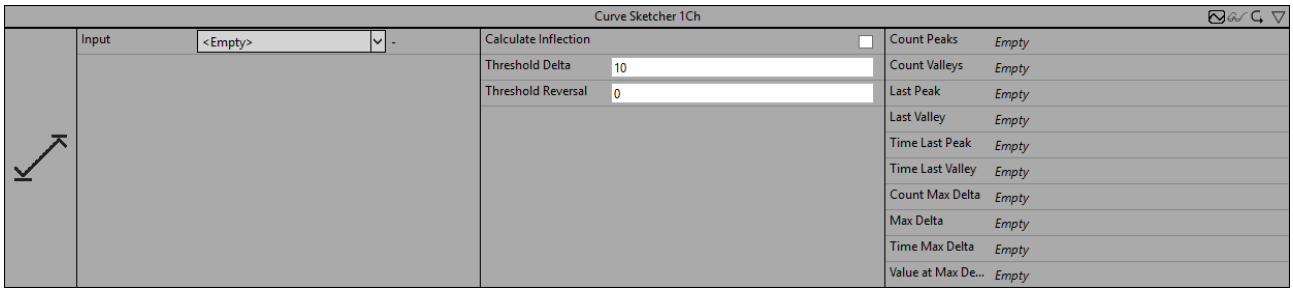






Alternatively, customer-specific HMI controls can be mapped in the Bandwidth Classifier 3Ch algorithm using the Mapping Wizard.

6.5.2.3 Curve Sketcher 1Ch

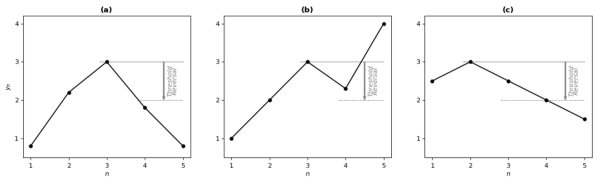


Curve Sketcher 1Ch identifies inversions (*peaks* and *valleys*) in an input data stream. Furthermore, local maxima of the absolute difference between two consecutive values (referred to as *Delta*) can be identified. Analogous to a continuous curve, the identified peaks and valleys correspond to local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

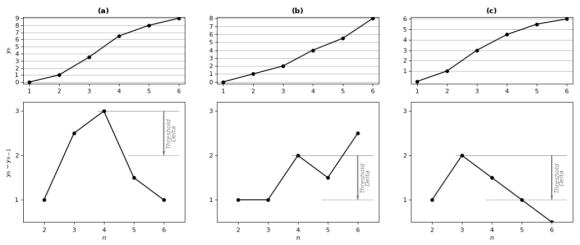
Optionally, a Boolean signal can be selected for the *Enable Execution* input so that the algorithm is only active if the value of the selected signal is *TRUE*.

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*.
  (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.

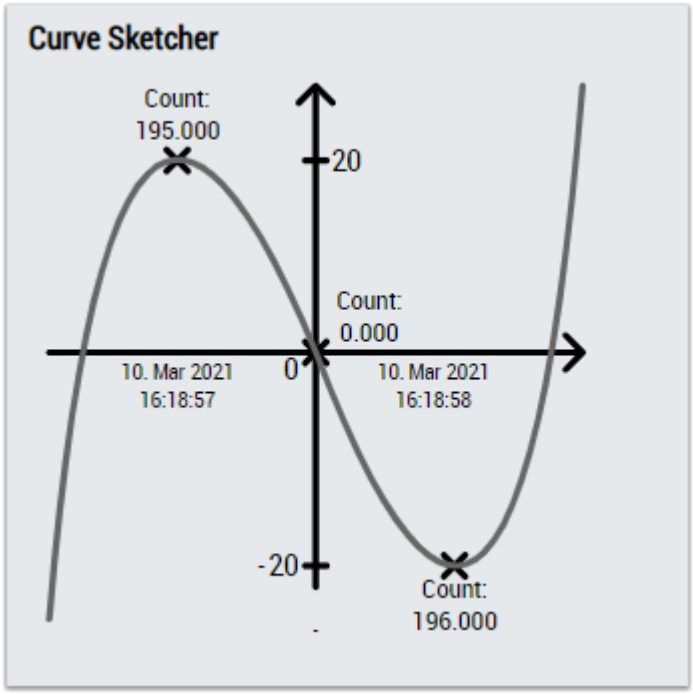
Output Values

- **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.



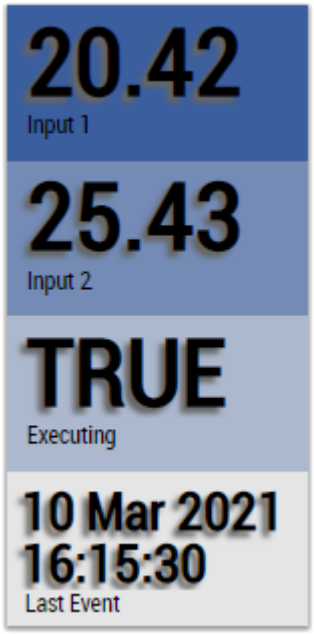
2. The Table Control visualizes all output values: Last Peak, Time Last Peak, Count Peaks, Last Valley, Time Last Valley, Count Valley, Value at Max Delta, Last Delta, Time Last Delta and Count Delta.

Data Table Vertical

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

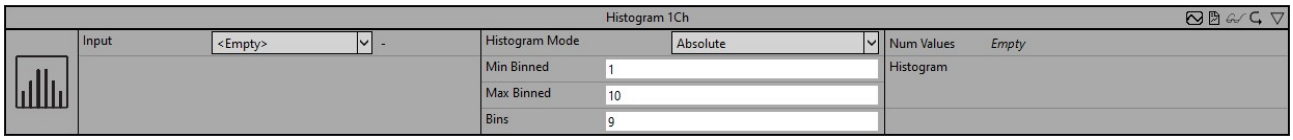
DataTable Horizontal

	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.

6.5.2.4 Histogram 1Ch



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.

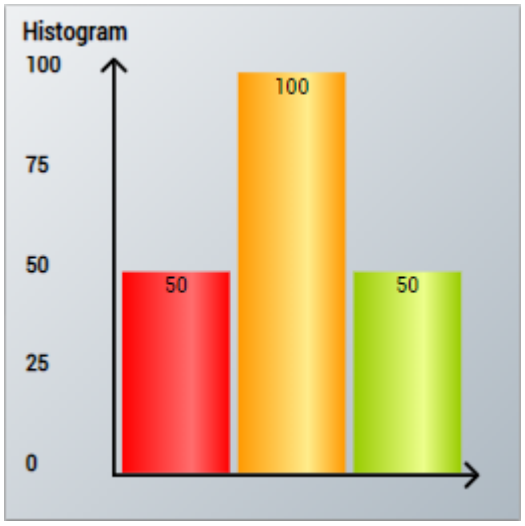
Output values

- **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.

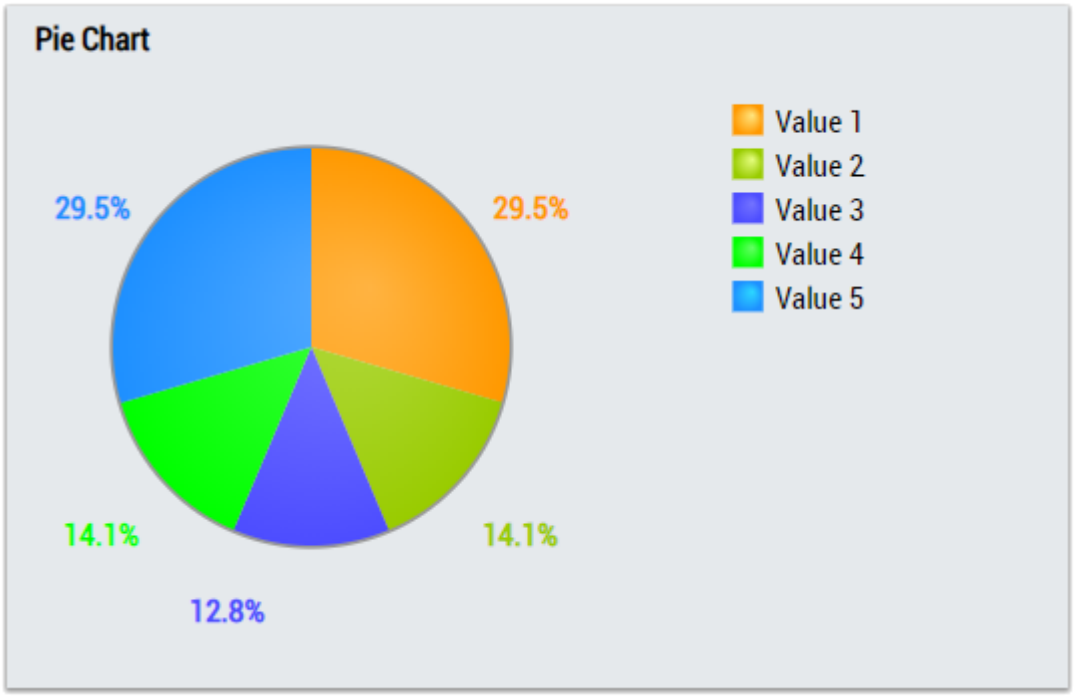
Standard HMI Controls

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

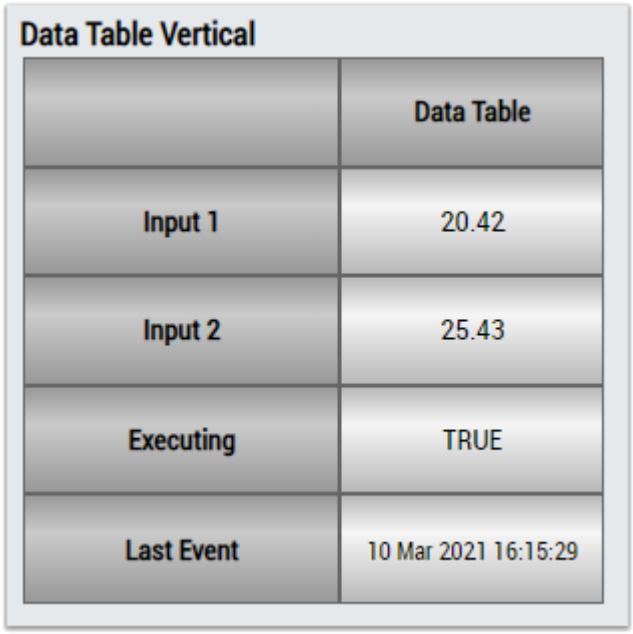
1. The Histogram control visualizes the output value Num Values.

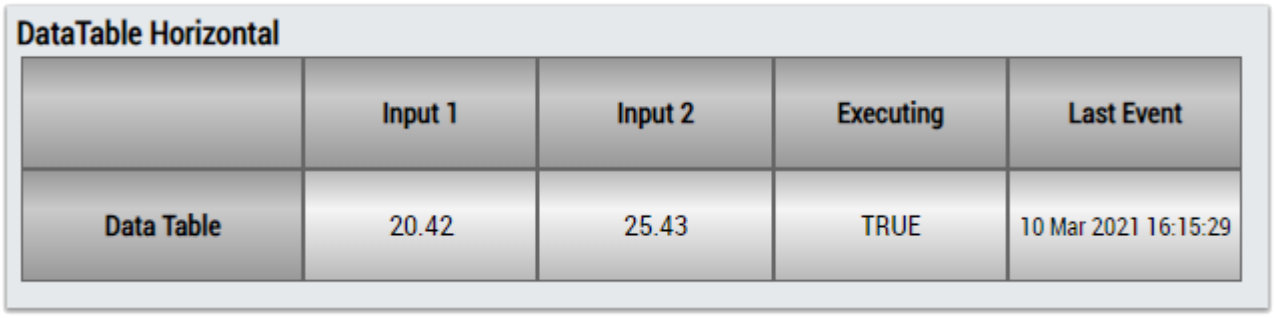


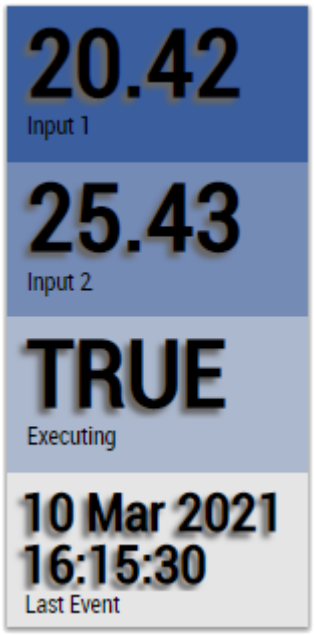
2. The PieChart control visualizes the output value Num Values.



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

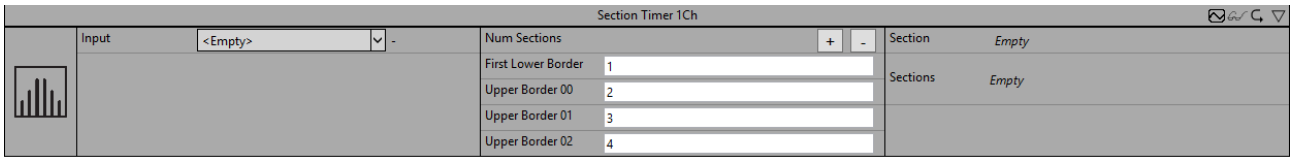






Alternatively, customer-specific HMI controls can be mapped in the Histogram 1Ch algorithm using the Mapping Wizard.

6.5.2.5 Section Timer 1 Ch



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 n0**: 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.

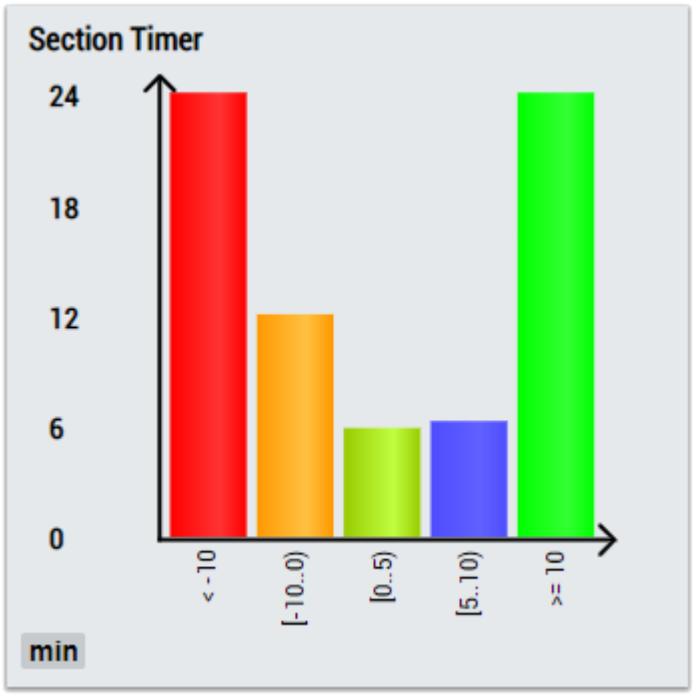
Output values

- **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.

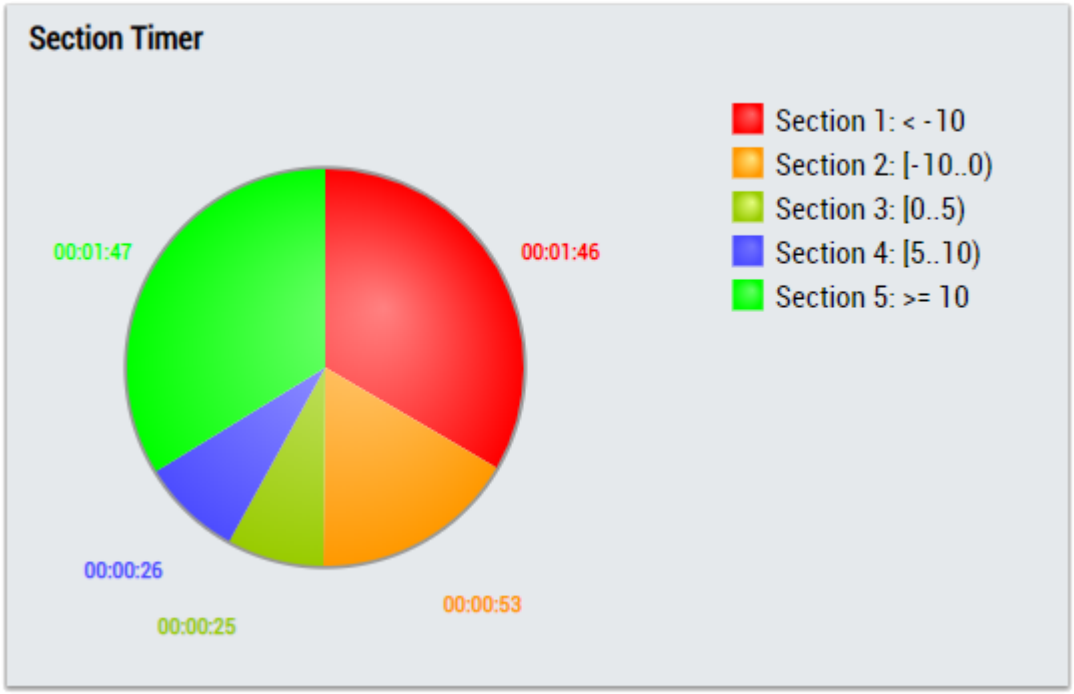
Standard HMI Controls

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.

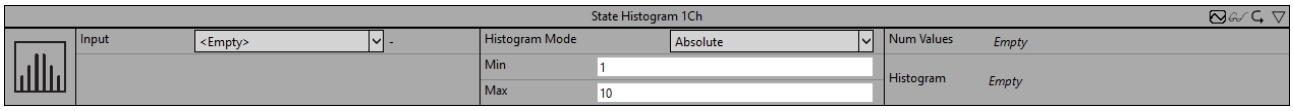


2. The PieChart control visualizes the input value Array of Timespans and the configuration options First Lower Border and Upper Border.



Alternatively, customer-specific HMI controls can be mapped in the Section Timer 1Ch algorithm using the Mapping Wizard.

6.5.2.6 State Histogram 1Ch



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.

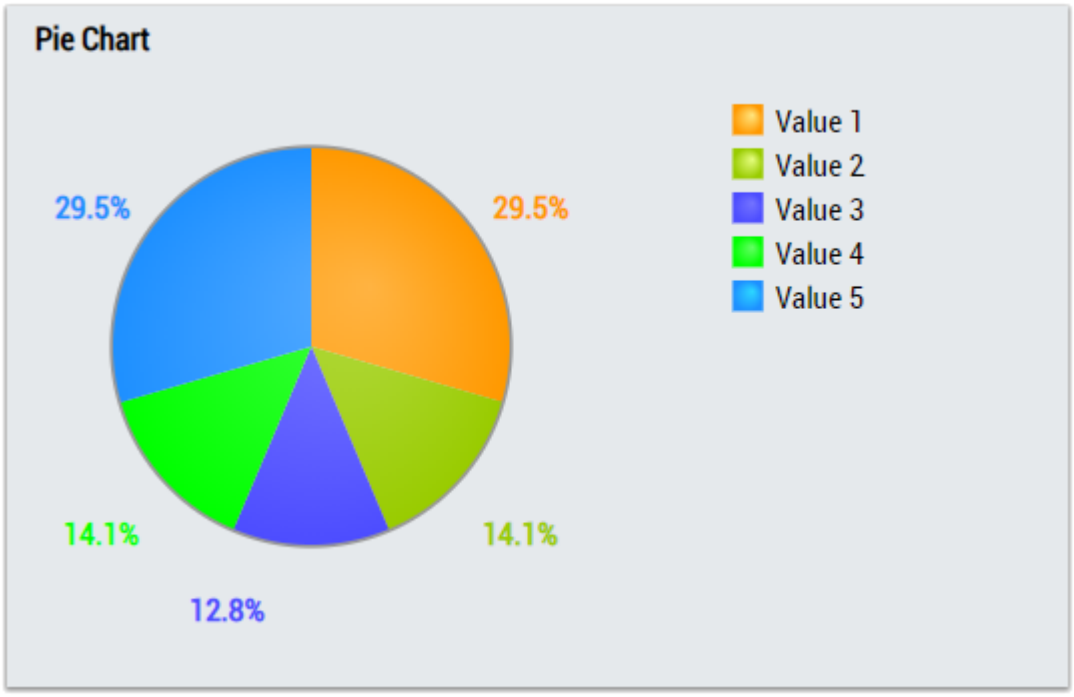
Output values

- **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.

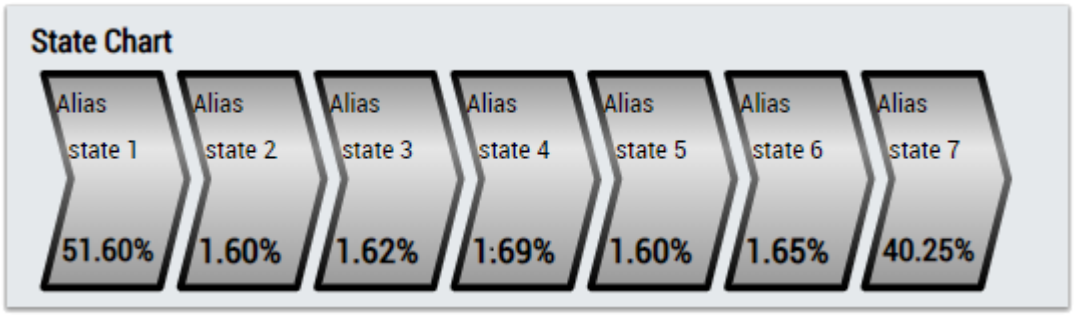
Standard HMI Controls

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

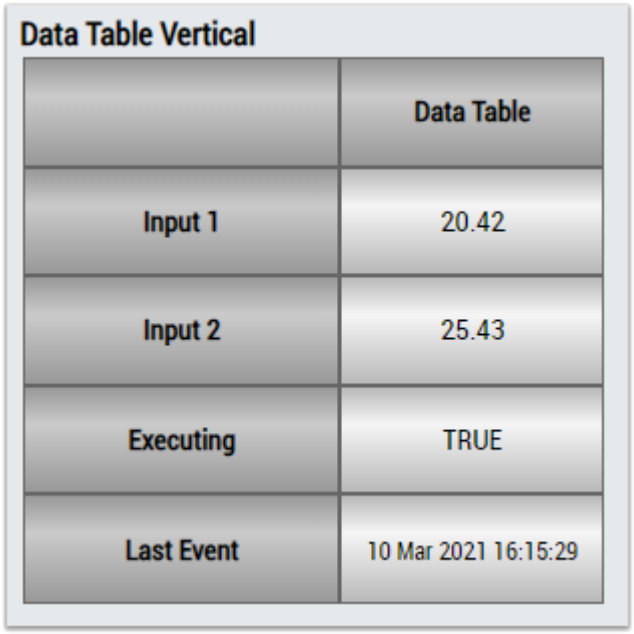
1. The PieChart control visualizes the output value histogram.

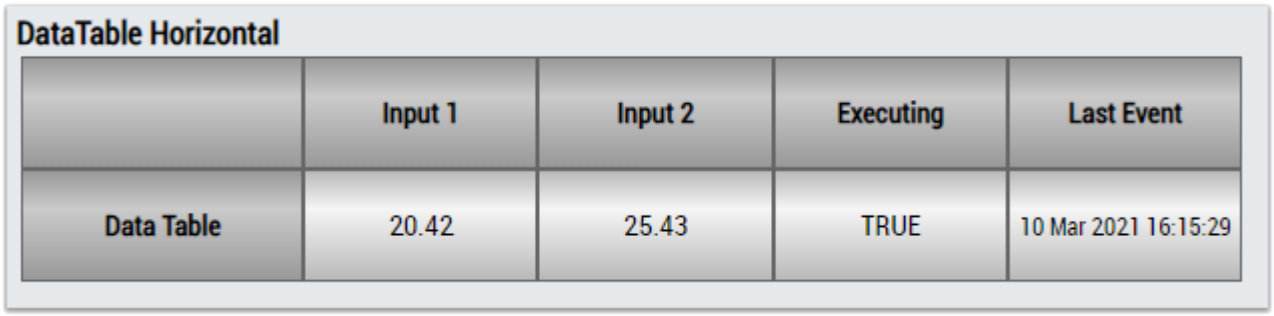


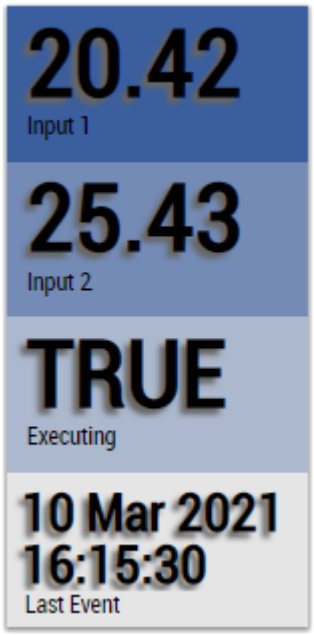
2. The StateHistogram control visualizes the output value histogram.



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

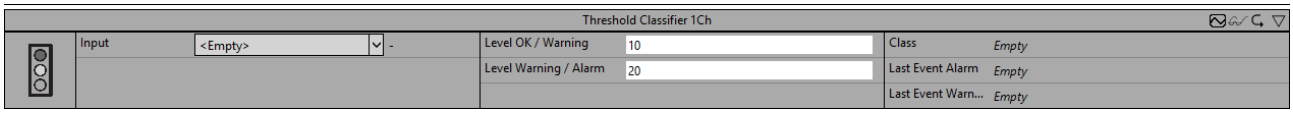






Alternatively, customer-specific HMI controls can be mapped in the State Histogram 1Ch algorithm using the Mapping Wizard.

6.5.2.7 Threshold Classifier 1Ch



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.

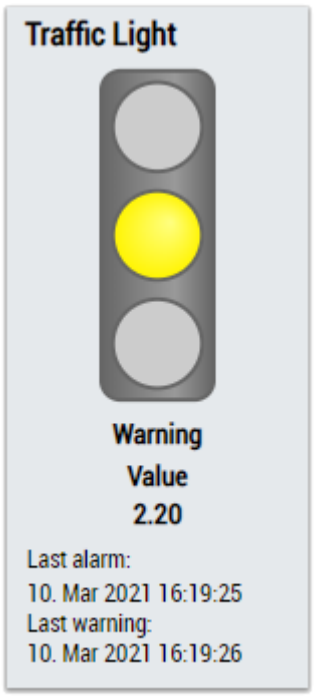
Output values

- **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.

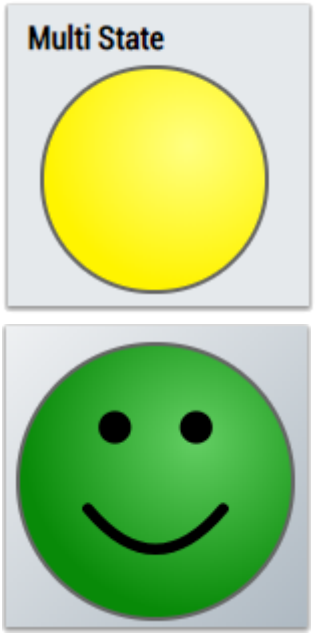
Standard HMI Controls

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

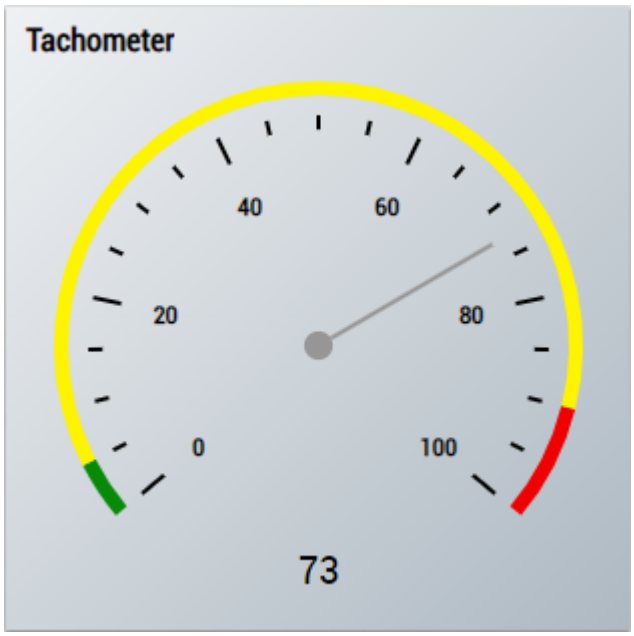
1. The TrafficLight Control visualizes the output values Class, Last Event Warning and Last Event Alarm as well as the input value of the data.



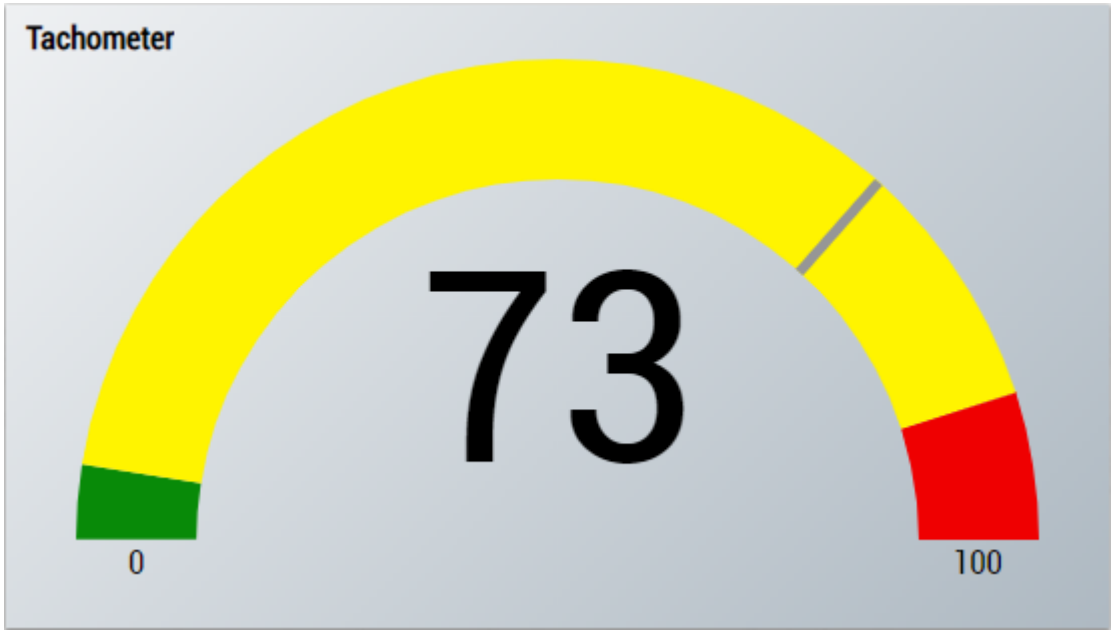
2. The MultiState Control visualizes the Class output value, optionally with a smiley.



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.



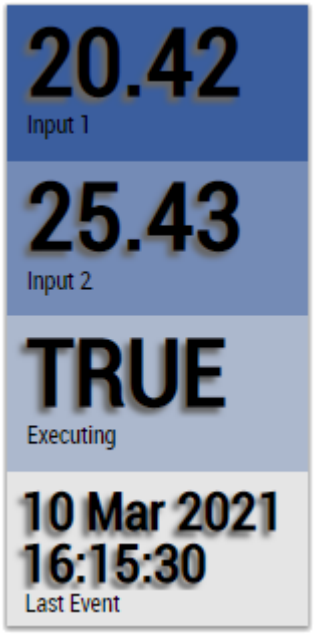
5. The Table Control or Multivalue Control visualizes all output values: Class, Last Event Warning Last Event Alarm.

Data Table Vertical

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

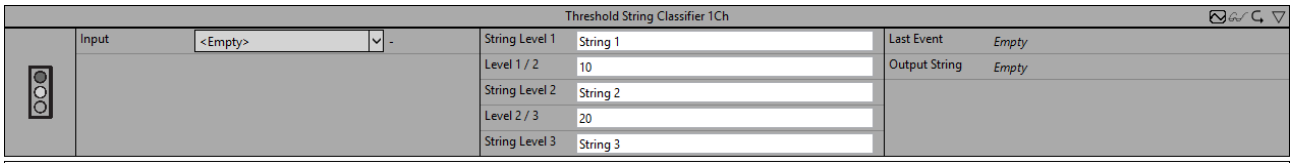
DataTable Horizontal

	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.

6.5.2.8 Threshold String Classifier 1Ch



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.

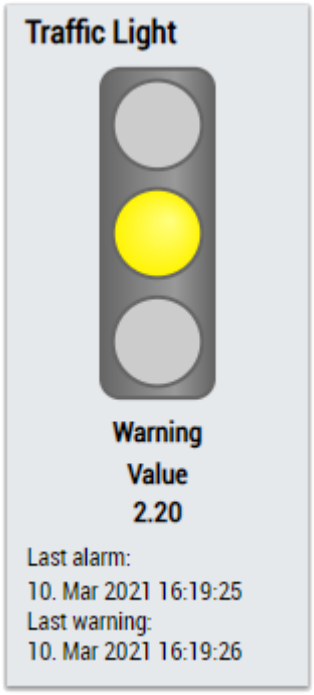
Output values

- **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.

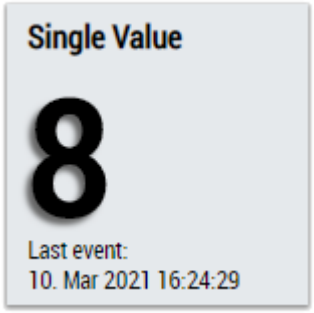
Standard HMI Controls

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

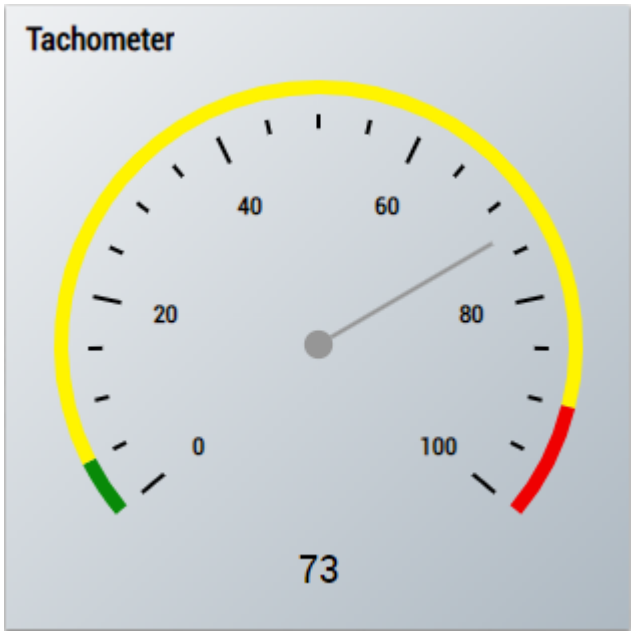
1. The TrafficLight control visualizes the output values Output String and Last Event as well as the input value of the data.



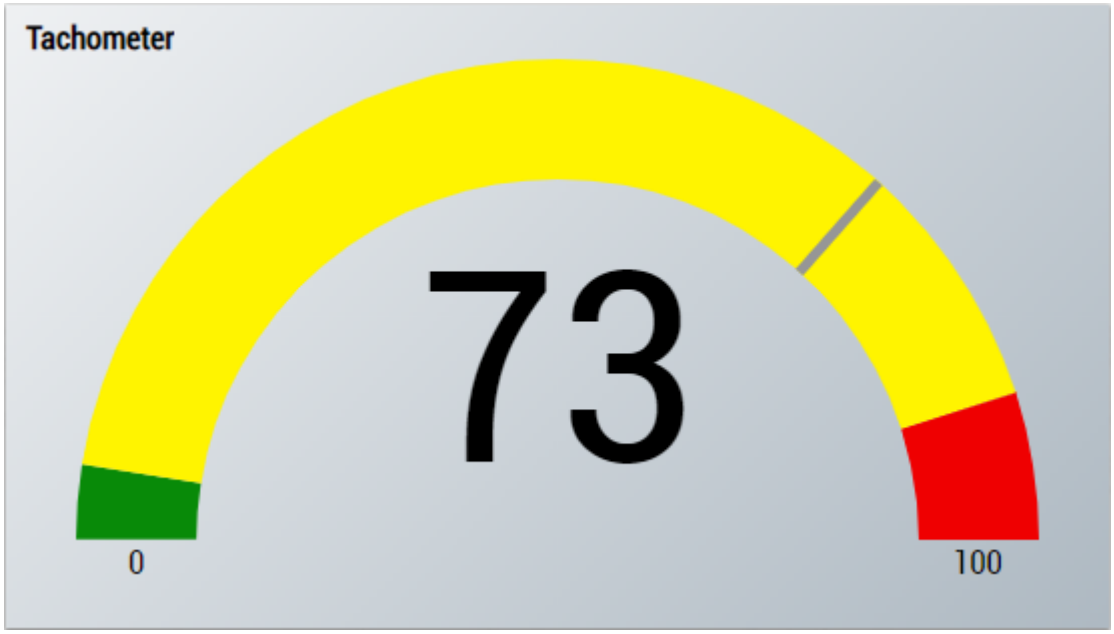
2. The SingleValue control visualizes the output value Output String.



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.



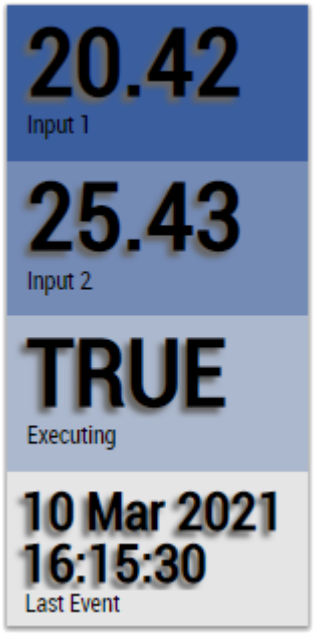
5. The Table Control or Multivalue Control visualizes all output values: Output String, Last Event.

Data Table Vertical

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

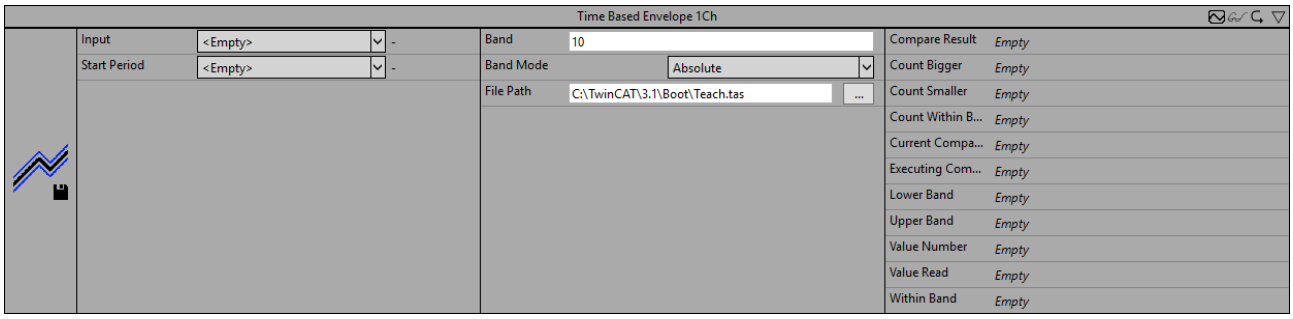
DataTable Horizontal

	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.

6.5.2.9 Time Based Envelope 1Ch



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 [▶ 189], 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 [▶ 189] due to concurrent file access. Instead, a reference signal should first be taught in with Time Based Teach Path 1Ch [▶ 189] 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 teached data file.
- **Band Mode:** Mode of the band operation (use absolute or relative values).
- **Band:** Bandwidth of the band operation.

Output values

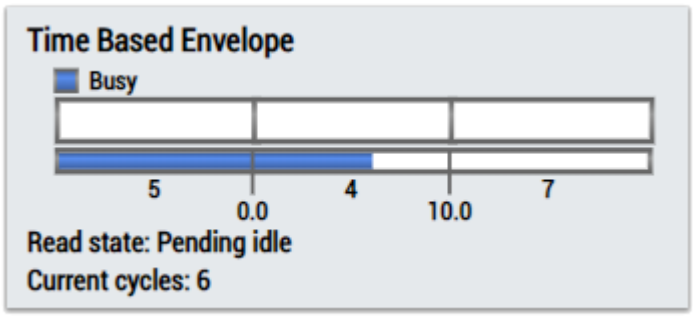
- **Executing Compare:** displays *TRUE* if the algorithm is processing the envelope, otherwise *FALSE*. The envelope process begins when the Start Period flag is *TRUE*.
- **Lower Band:** displays the value of the lower band depending on the band mode.
- **Upper Band:** displays the value of the upper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

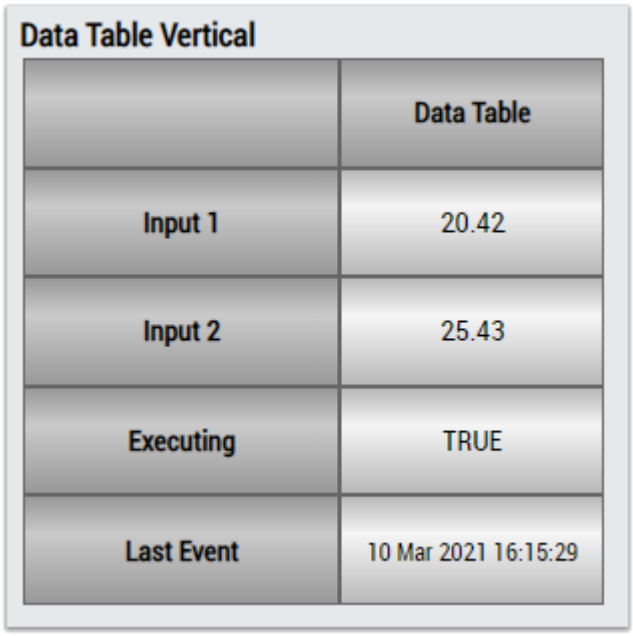
Standard HMI Controls

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.

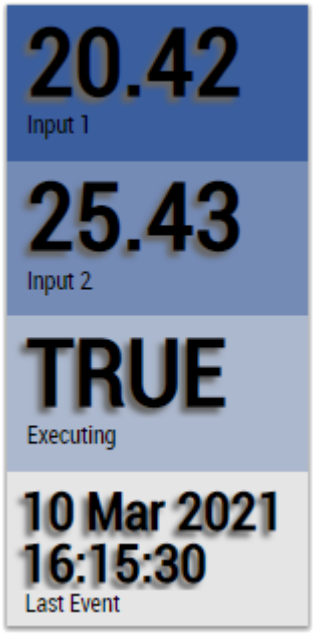


2. The Table Control or Multivalue Control visualizes all output values: Executing Comparison, State, Lower Band, Upper Band, Within Band, Compare Result, Current Compared Cycles, Count Within Band, Count Smaller, Count Bigger, Value Number.



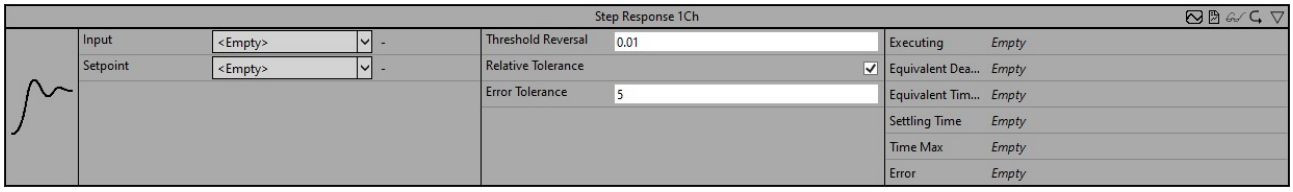
DataTable Horizontal

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

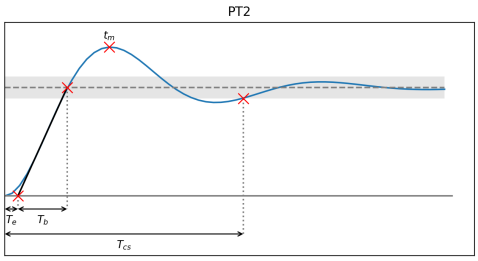


Alternatively, customer-specific HMI controls can be mapped in the Time Based Envelope algorithm using the Mapping Wizard.

6.5.2.10 Step Response 1Ch



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.



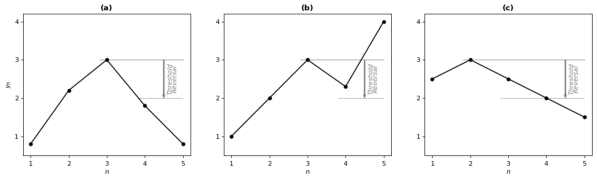
To detect whether the track is steady, a local minimum or maximum is searched for. If this is within the tolerance band (marked in gray), it is assumed that the track is steady. Only then is the settling time set.

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

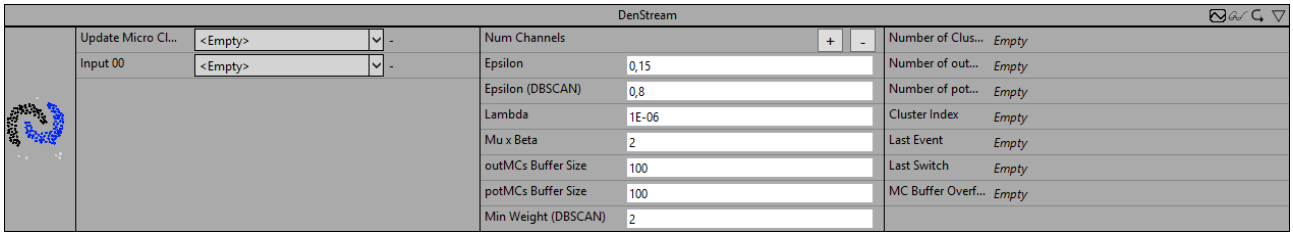
Fig. 2:

Alternatively, customer-specific HMI controls can be mapped using the Mapping Wizard.

6.5.3 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*.

6.5.3.1 DenStream



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 (#1, 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 x 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 x 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 x Beta* and *Lambda*), it is deleted from the memory.

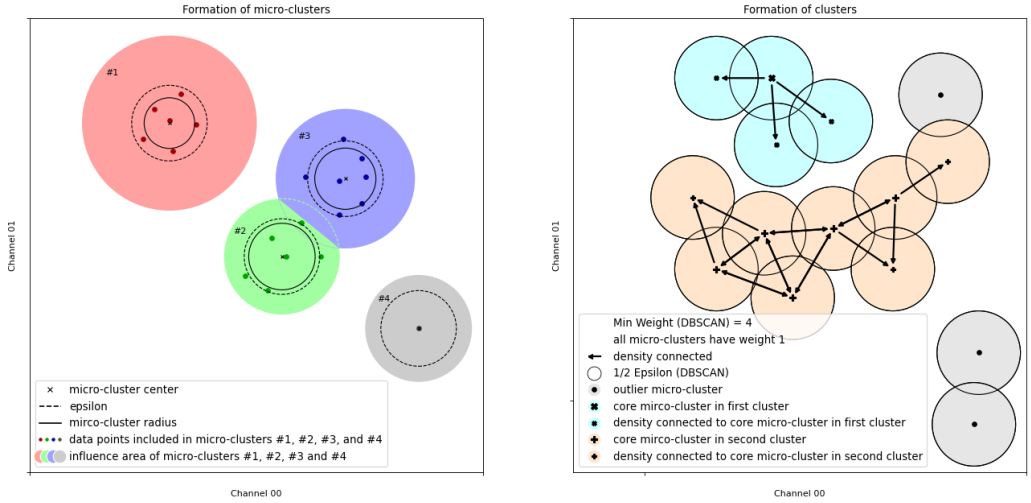
The parameters *Epsilon (DBSCAN)* and *Min Weight (DBSCAN)* affect the second phase. These parameters were adopted from the DBSCAN algorithm [3].

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^(-*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.

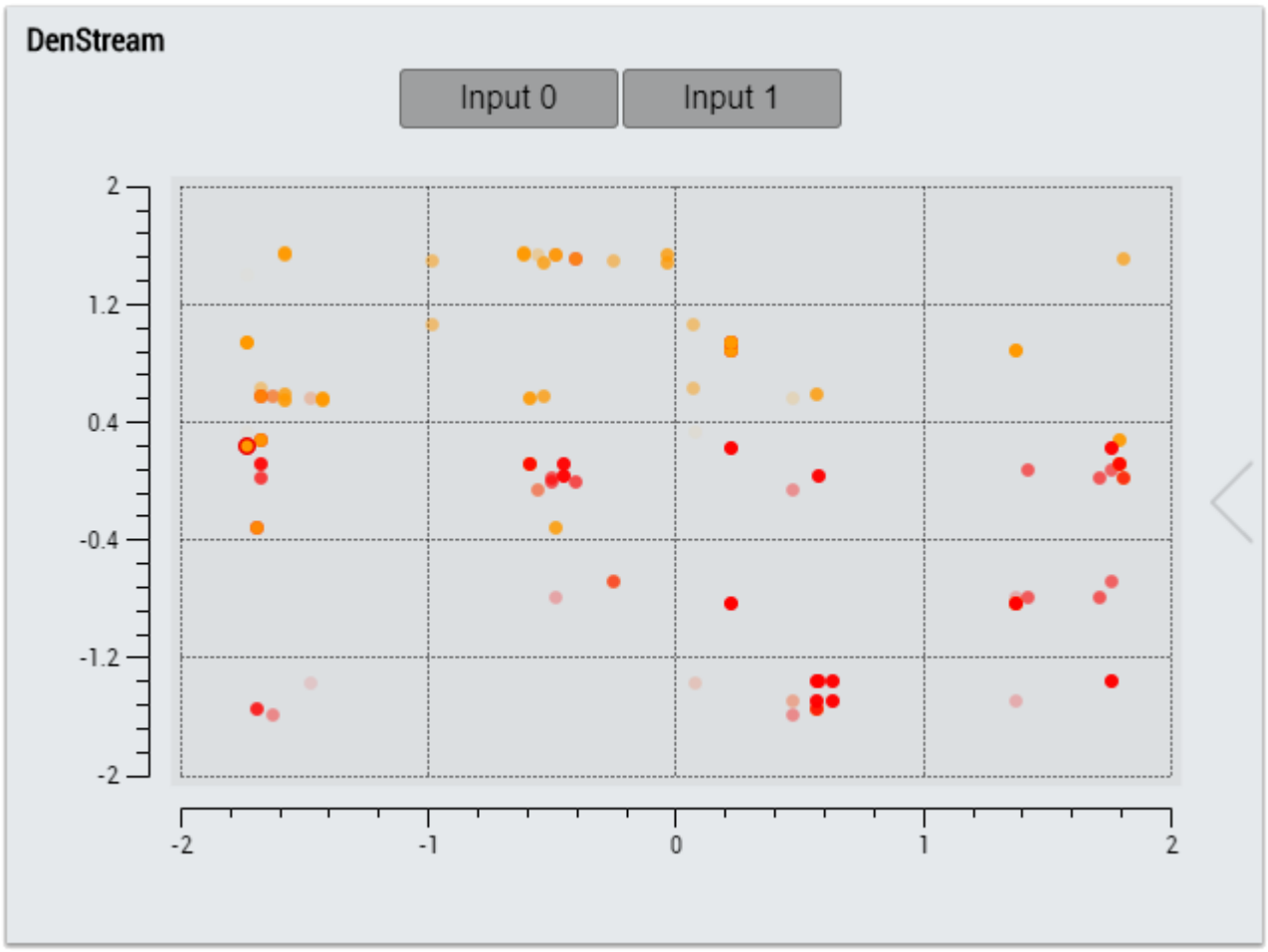
Output values

- **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.

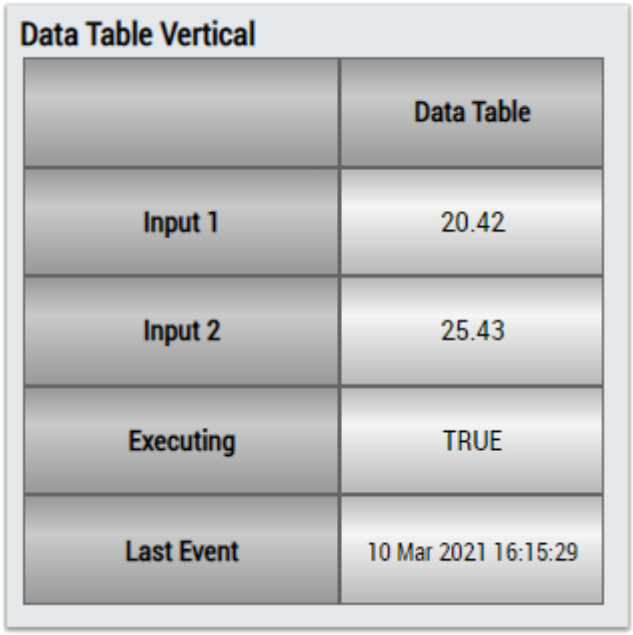
Standard HMI Controls

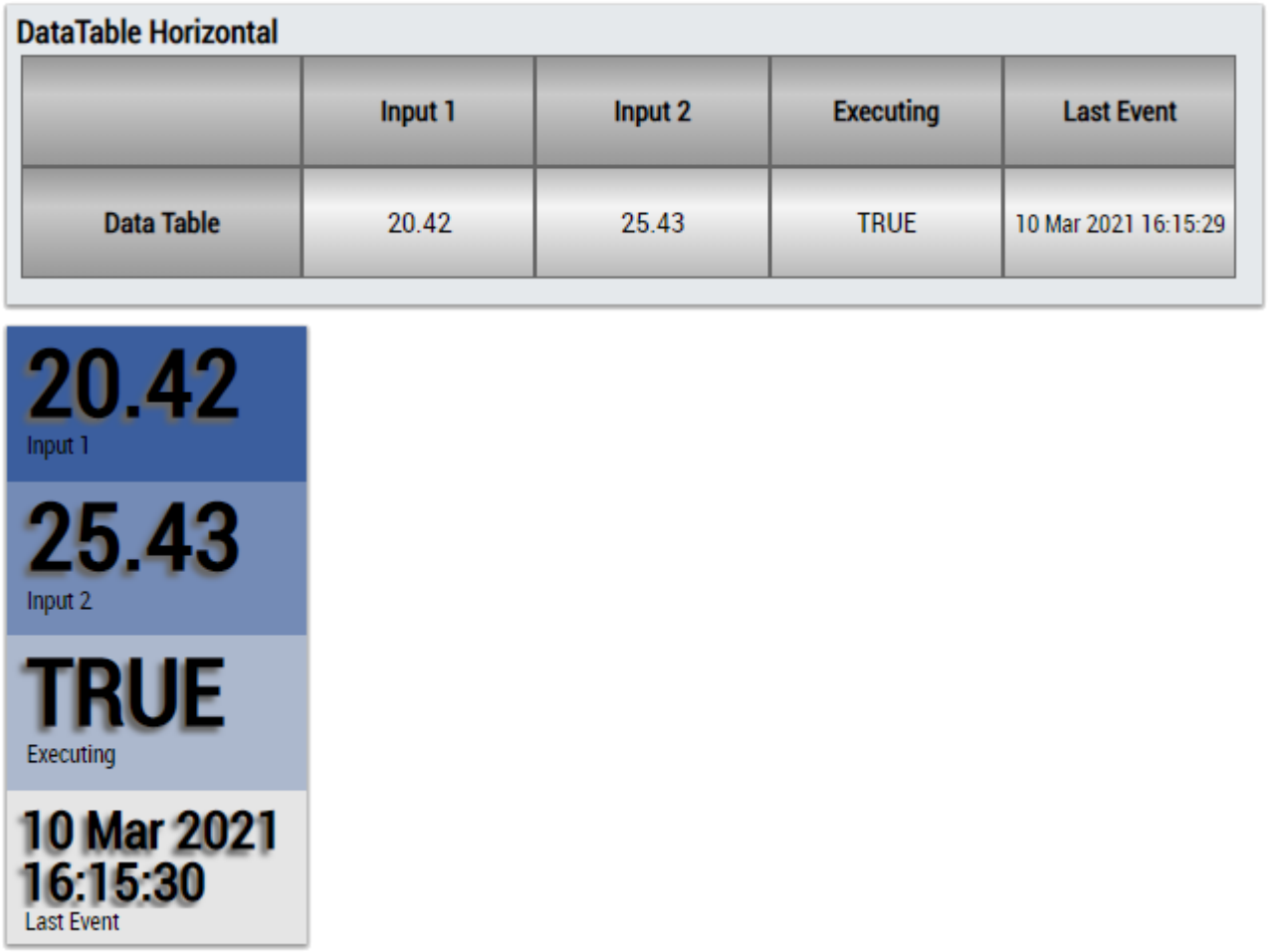
For the DenStream algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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.



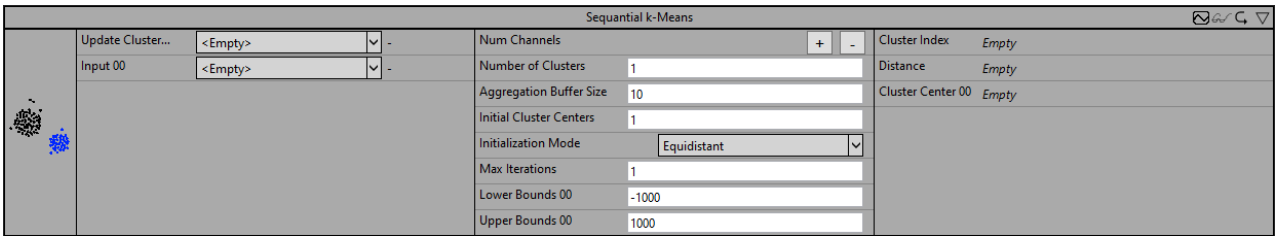
2. The Table Control or Multivalue Control visualizes all output values: cluster count, number of potential micro-clusters, number of outlier micro-clusters, cluster index, last event, last switch.





Alternatively, customer-specific HMI controls can be mapped in the DenStream algorithm using the Mapping Wizard.

6.5.3.2 Sequential k-Means



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:

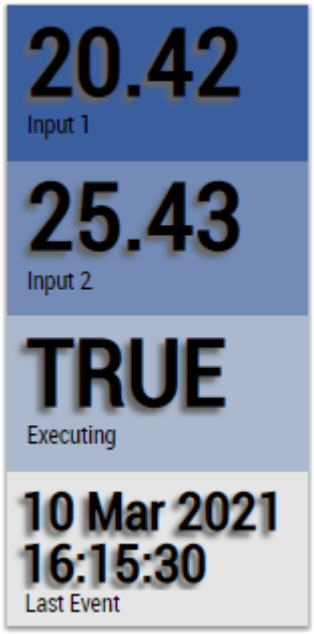
1. The Table Control or Multivalue Control visualizes the output values: Lower Bounds, Upper Bounds and Initial Centers.

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

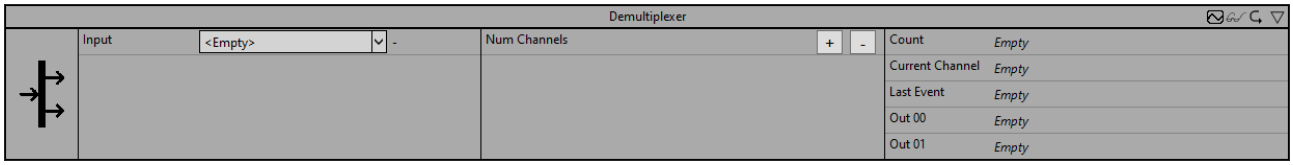


Alternatively, customer-specific HMI controls can be mapped in the *Sequential k-Means* algorithm using the Mapping Wizard.

6.5.4 Analytics - Compare

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

6.5.4.1 Demultiplexer



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.

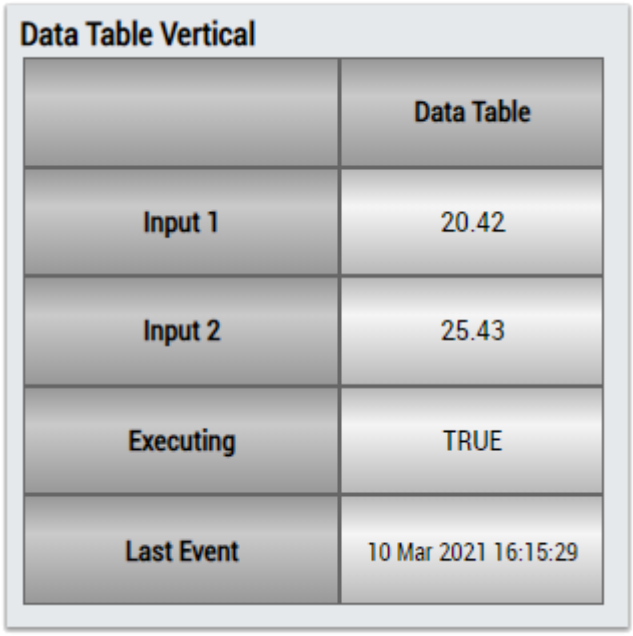
Output values

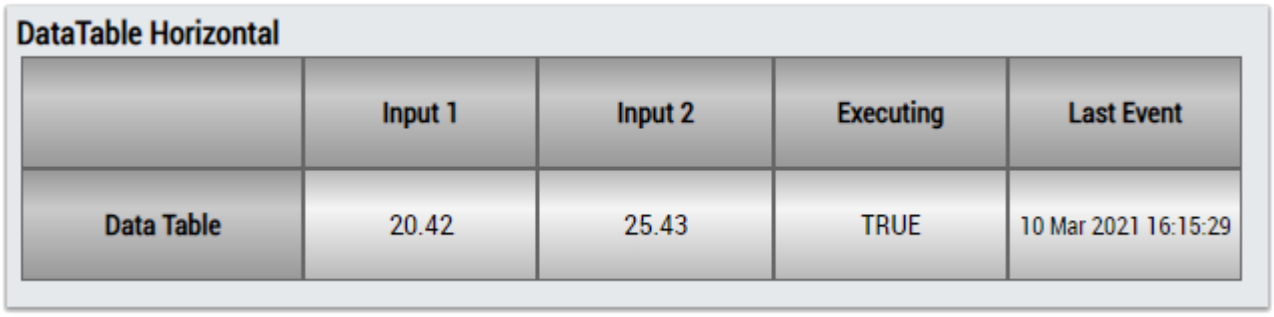
- **Current Channel:** Indicates the number of the selected channel. The value is 0 if the selected channel is outside the configured channels.
- **Count:** Starts with 1 for the channel selected at the start of the analysis and increments each time another channel is selected.
- **Last Event:** Timestamp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

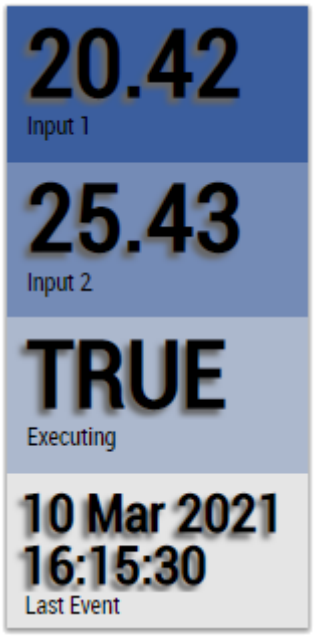
Standard HMI Controls

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.

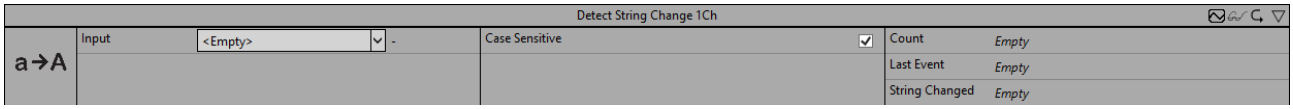






Alternatively, customer-specific HMI controls can be mapped in the Demultiplexer algorithm using the Mapping Wizard.

6.5.4.2 Detect String Change 1Ch



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.

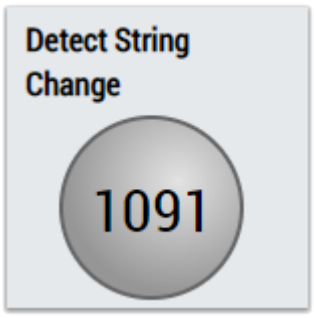
Output values

- **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

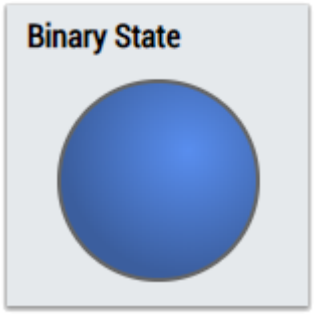
For the Detect String Change 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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.



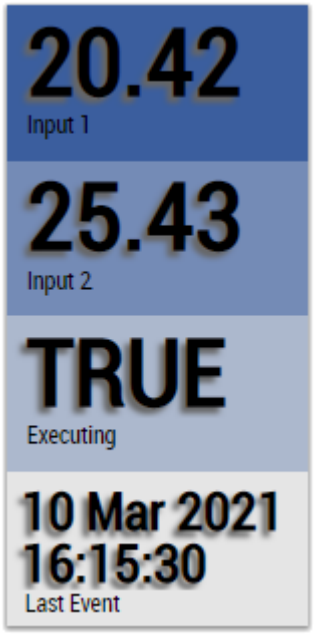
4. The Table Control or Multivalue Control visualizes all output values: Operation Out, Count, Last Event.

Data Table Vertical

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

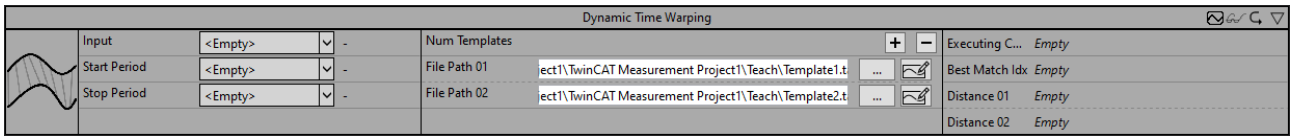
DataTable Horizontal

	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.

6.5.4.3 Dynamic Time Warping



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 [▶ 189] due to concurrent file access. Instead, a reference signal should first be taught in with the Time Based Teach Path 1Ch [▶ 189] 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 [▶ 189]. As a rule, the templates are a few hundred supporting points. Therefore, a reduction of the data with the function block Downsampling 1Ch [▶ 74] 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

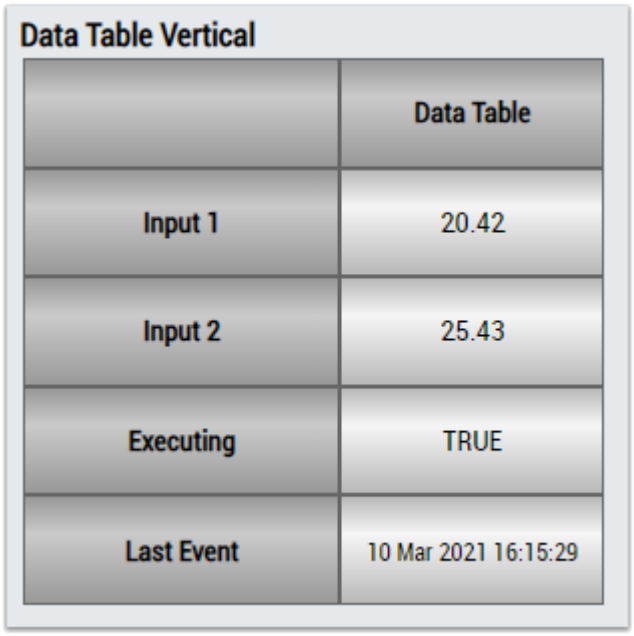
- **Executing Compare:** displays *TRUE* if the algorithm is processing the incoming values, otherwise *FALSE*. The processing begins when the Start Period flag is *TRUE* and ends when Stop Period is TRUE.
- **Best Match Idx:** outputs the index of the template with the smallest distance to the input channel.

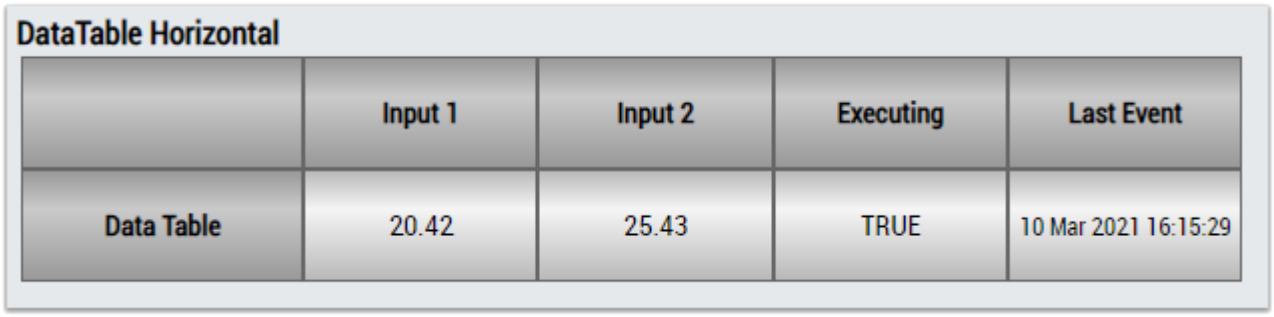
- **Distance 01 ... n:** specifies the distance between the input signal and the respective template. The smaller the distance, the more equal the compared signals are.

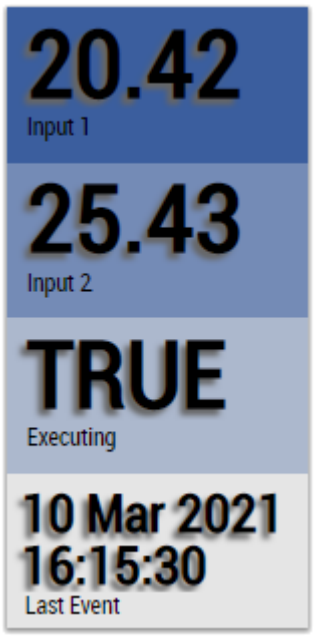
Standard HMI Controls

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*.

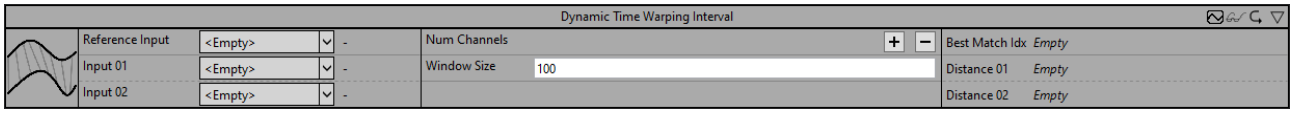






Alternatively, customer-specific HMI controls can be mapped in the *Dynamic Time Warping* algorithm using the Mapping Wizard.

6.5.4.4 Dynamic Time Warping Interval



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.

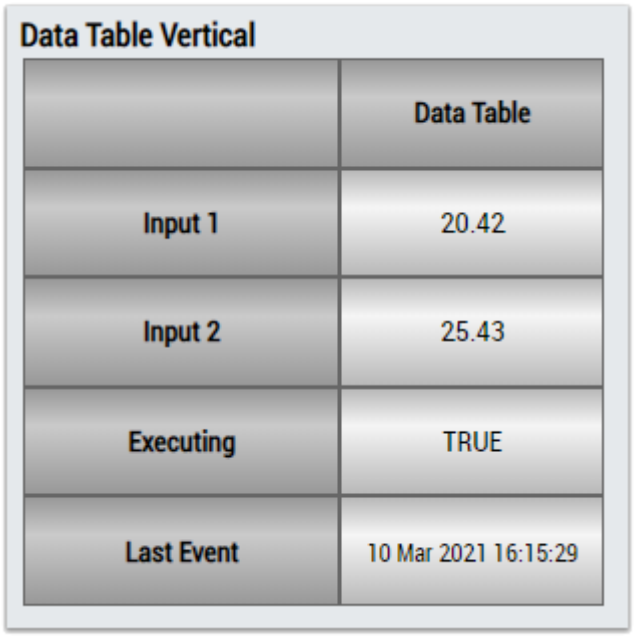
Output values

- **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*.

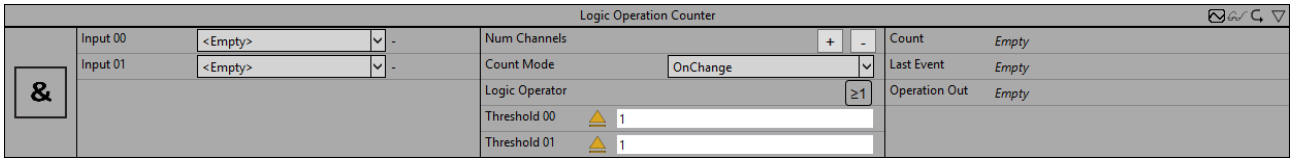


DataTable Horizontal

	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.

6.5.4.5 Logic Operation Counter



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.

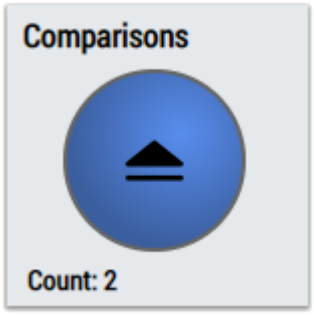
Output values

- **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

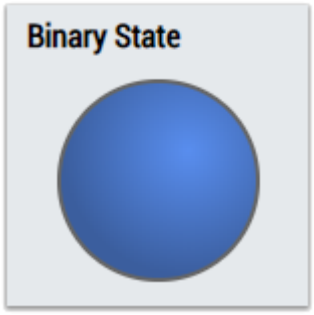
For the Logic Operation Counter algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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.



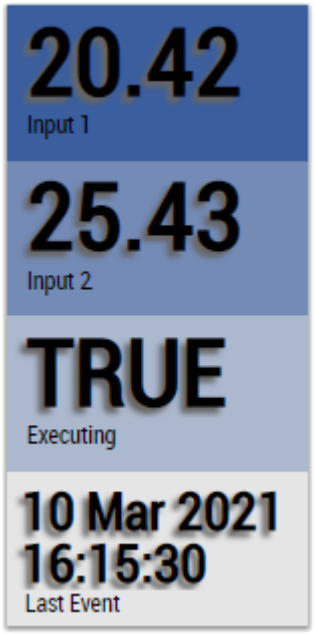
4. The Table Control or Multivalue Control visualizes all output values: Operation Out, Count, Last Event.

Data Table Vertical

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

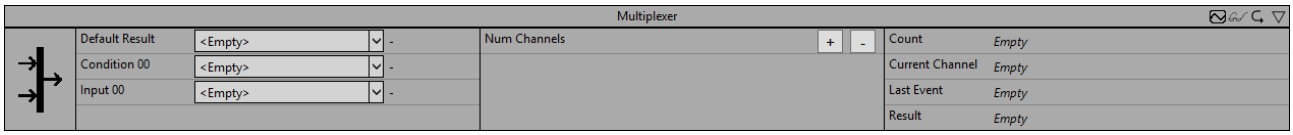
DataTable Horizontal

	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.

6.5.4.6 Multiplexer



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

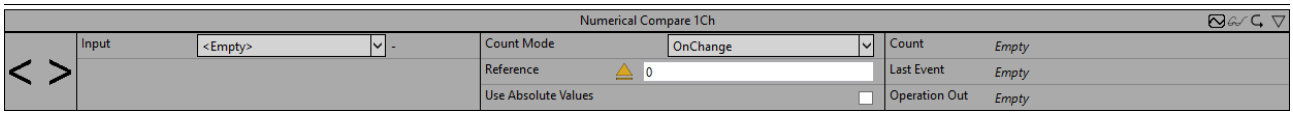
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

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

Alternatively, customer-specific HMI controls can be mapped in the Multiplexer algorithm using the Mapping Wizard.

6.5.4.7 Numerical Compare 1Ch



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.

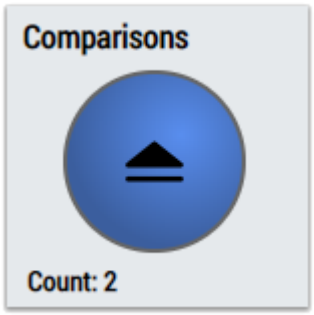
Output values

- **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

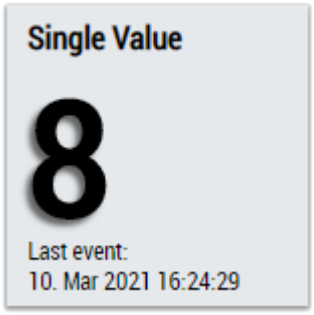
Standard HMI Controls

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

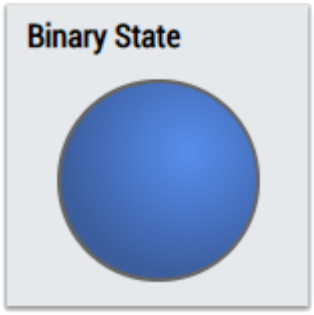
1. The Comparison Control or Multivalue Control visualizes the output values Operation Out and Count as well as the configuration option Operator.



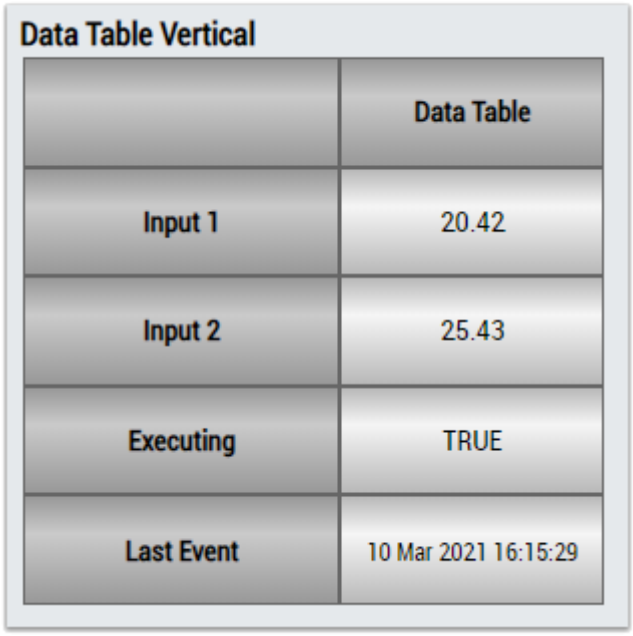
2. The SingleValue control visualizes the output values Count and Last Event.

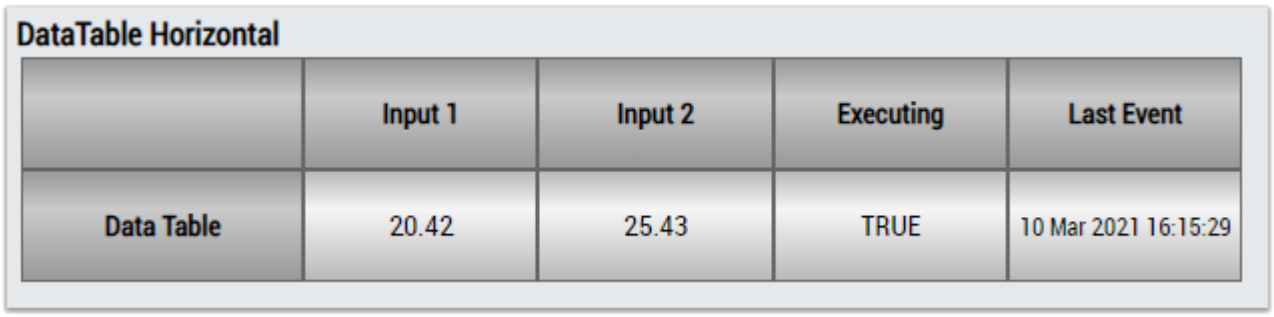


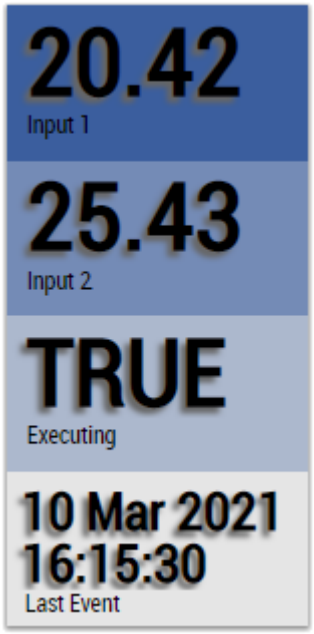
3. The BinaryState control visualizes the output value Operation Out.



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

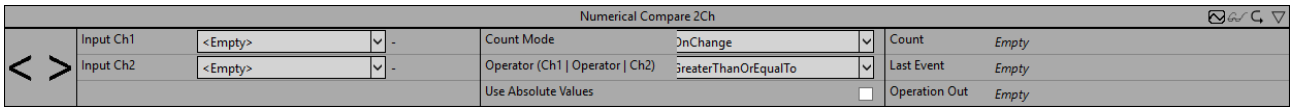






Alternatively, customer-specific HMI controls can be mapped in the *Numerical Compare 1Ch* algorithm using the Mapping Wizard.

6.5.4.8 Numerical Compare 2Ch



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.

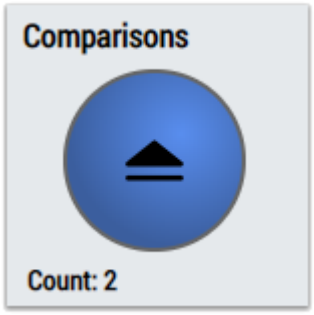
Output values

- **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

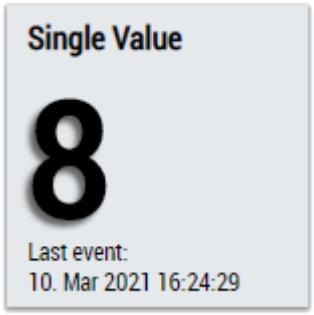
Standard HMI Controls

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

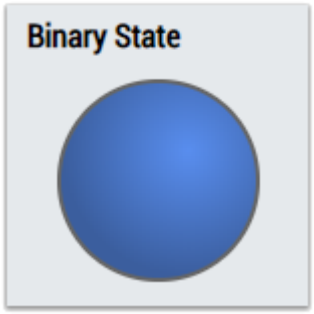
1. The Comparison Control or Multivalue Control visualizes the output values Operation Out and Count as well as the configuration option Operator.



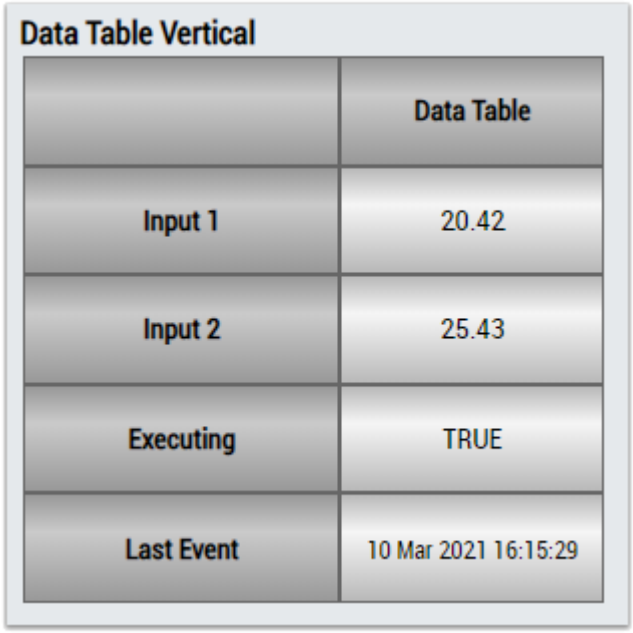
2. The SingleValue control visualizes the output values Count and Last Event.



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



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

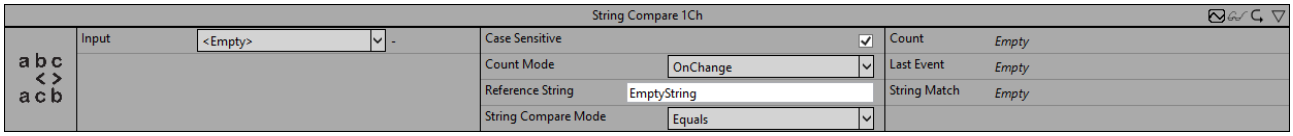


DataTable Horizontal

	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.

6.5.4.9 String Compare 1Ch



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*.

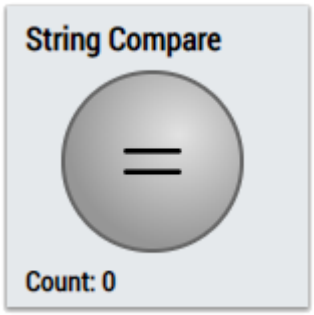
Output values

- **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

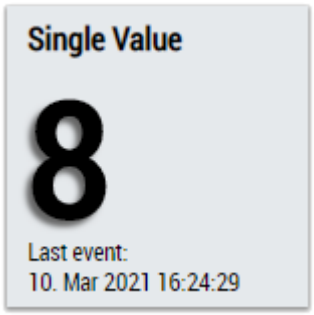
Standard HMI Controls

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

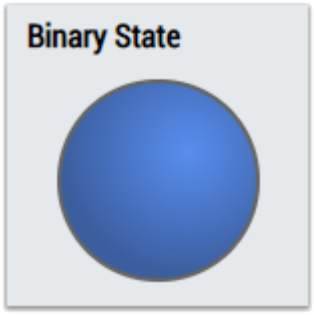
1. The Comparison control visualizes the output values String Match and Count as well as the configuration option String Compare Mode.



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

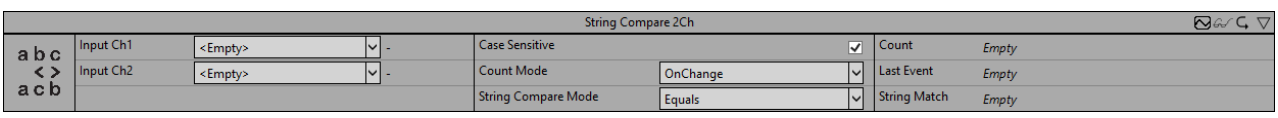
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

DataTable Horizontal

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

Alternatively, customer-specific HMI controls can be mapped in the String Compare 1Ch algorithm using the Mapping Wizard.

6.5.4.10 String Compare 2Ch



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*.

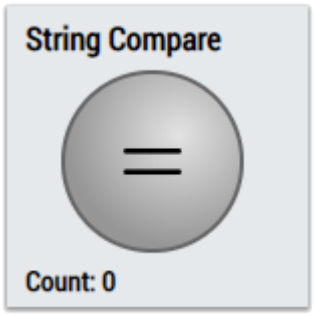
Output values

- **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

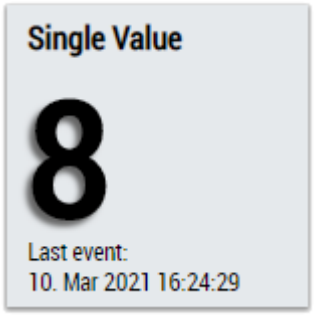
Standard HMI Controls

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

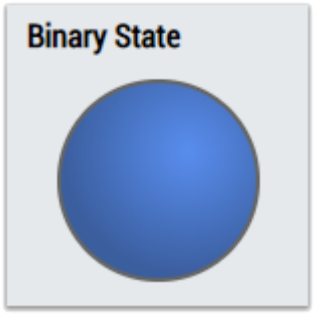
1. The Comparison control visualizes the output values String Match and Count as well as the configuration option String Compare Mode.



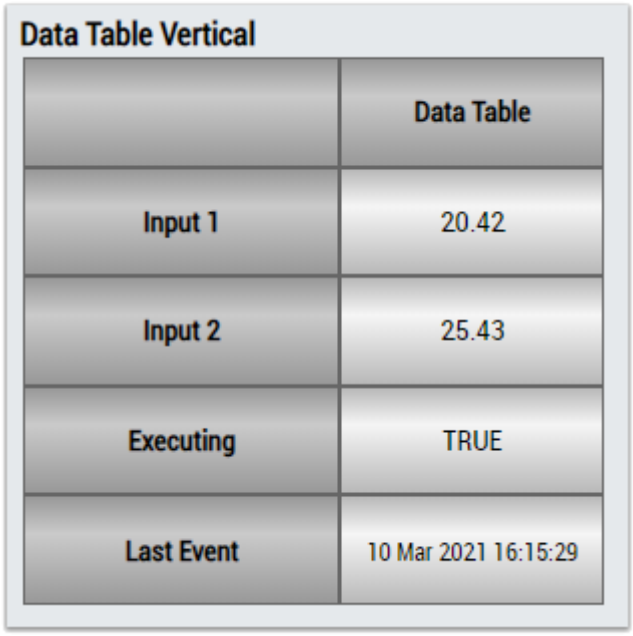
2. The SingleValue control visualizes the output values Count and Last Event.

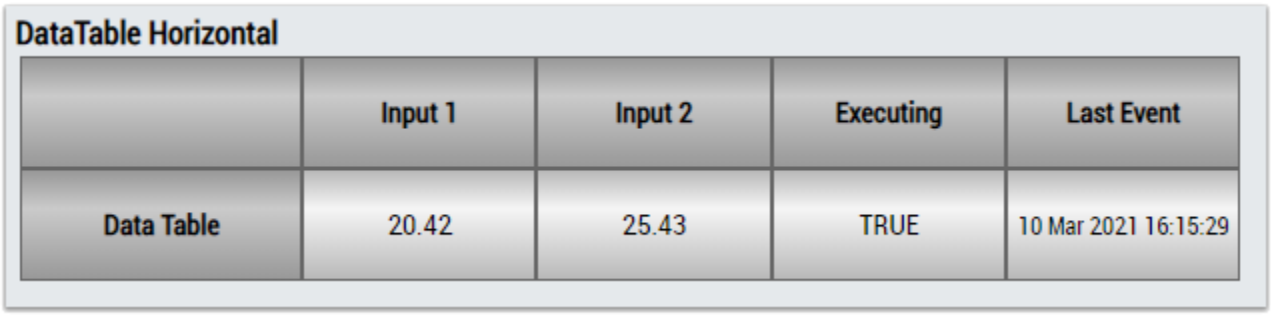


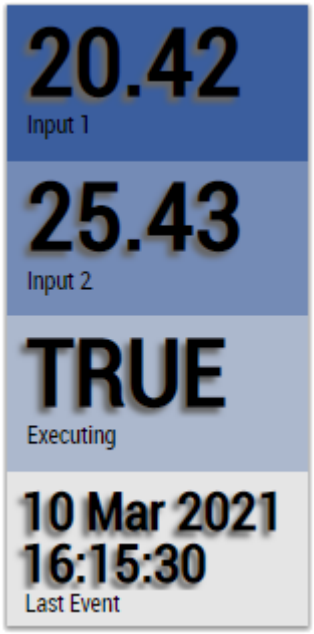
3. The BinaryState Control visualizes the output value String Match.



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

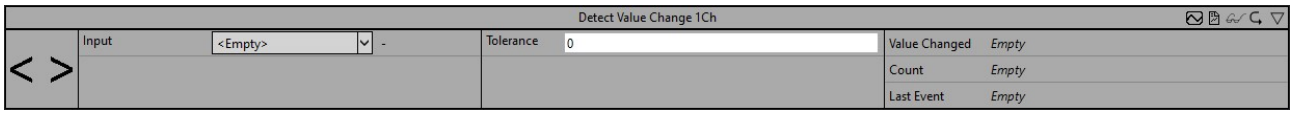






Alternatively, customer-specific HMI controls can be mapped in the String Compare 2Ch algorithm using the Mapping Wizard.

6.5.4.11 Detect Value Change 1Ch



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.

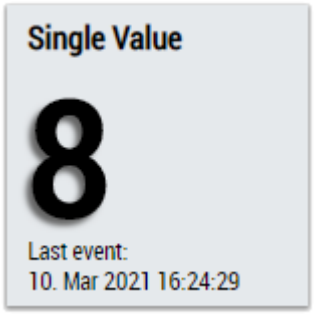
Output values

- **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

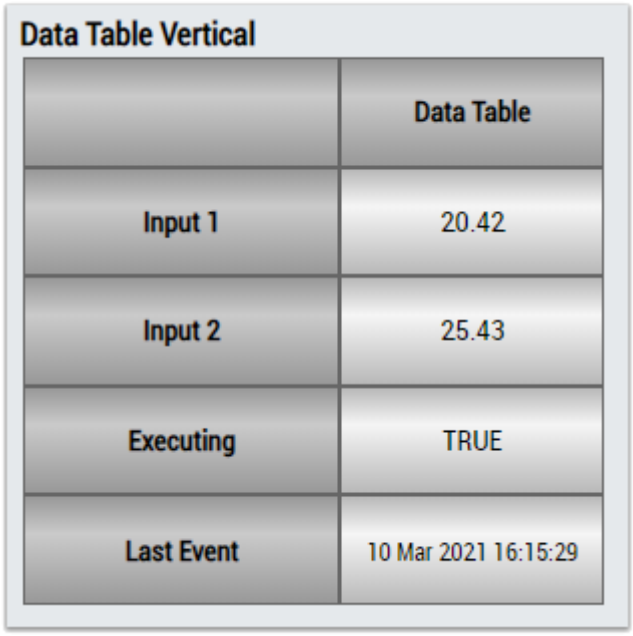
Standard HMI Controls

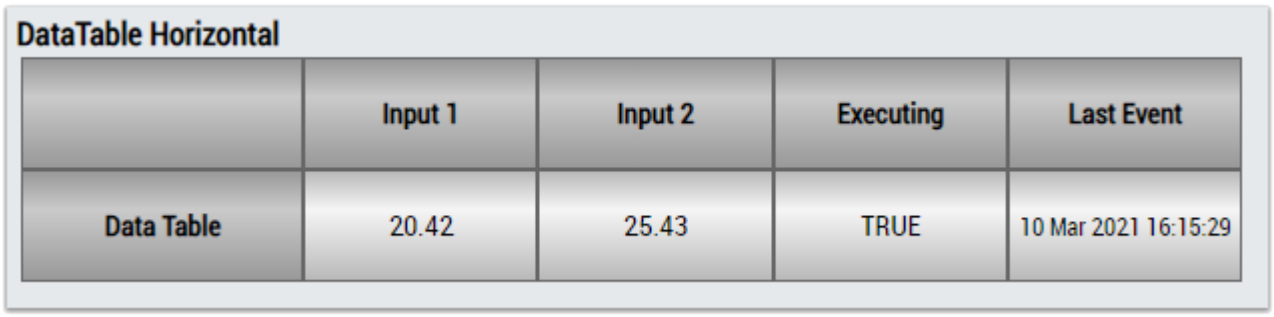
For the *Detect Value Change 1Ch* algorithm, the following HMI controls are available for generating an Analytics Dashboard:

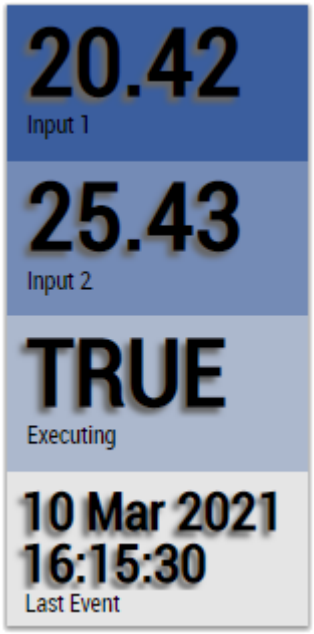
1. The SingleValue control visualizes the output values Count and Last Event.



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





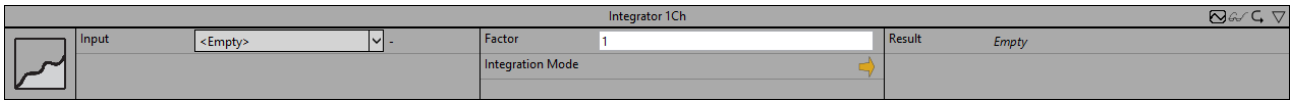


Alternatively, customer-specific HMI controls can be mapped using the Mapping Wizard.

6.5.5 Analytics - Math

The algorithms of the category *Analytics-Math* provide functionalities for mathematical operations such as basic arithmetic operation, integration or slope analysis.

6.5.5.1 Integrator 1Ch



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.

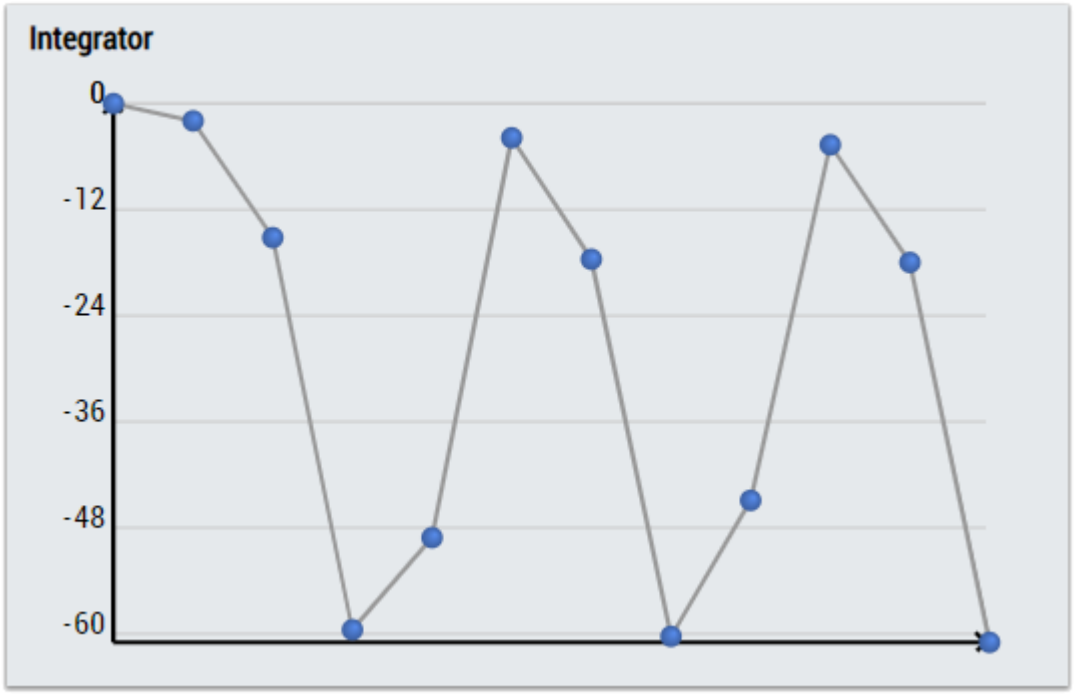
Output Values

- **Result:** Shows the result of the integration operation.

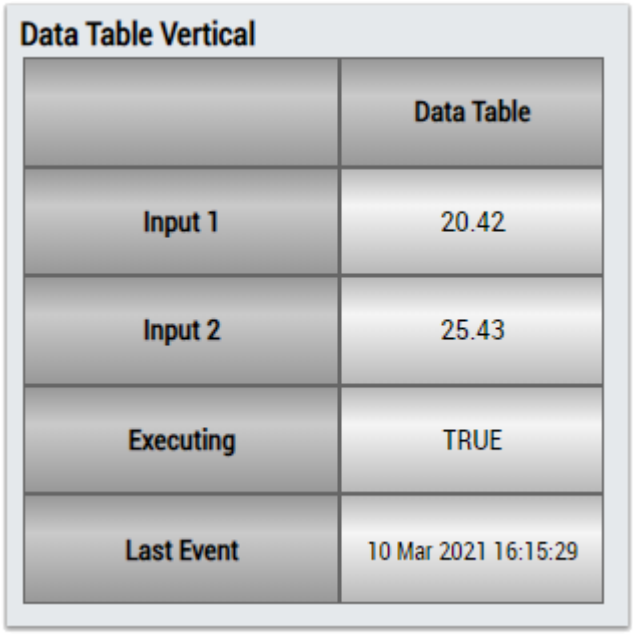
Standard HMI Controls

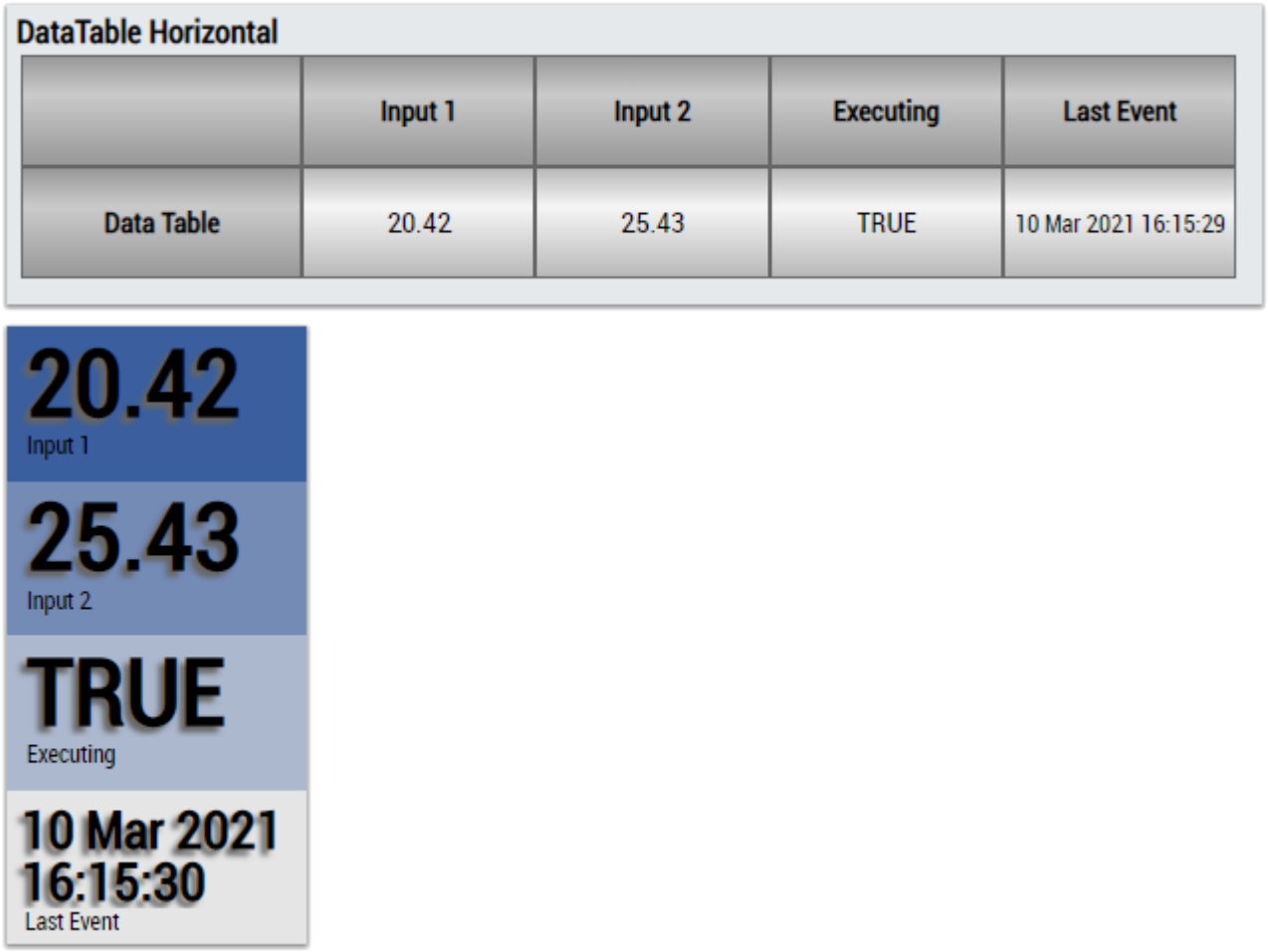
For the Integrator 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

1. The Integrator Control visualizes the last x output values Result.



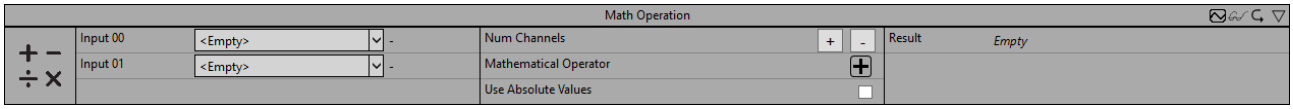
2. The Table Control or Multivalue Control visualizes all output values: Result.





Alternatively, customer-specific HMI controls can be mapped in the Integrator 1Ch algorithm using the Mapping Wizard.

6.5.5.2 Math Operation



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 and 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:** The number of operands that are inputs to the function.
- **Mathematical Operator:** Mathematical operator of the operation ("+", "-", "x", "/", "x^y", "%").
- **Use Absolute Values:** If the checkbox is checked, the absolute values of the input signal are used.

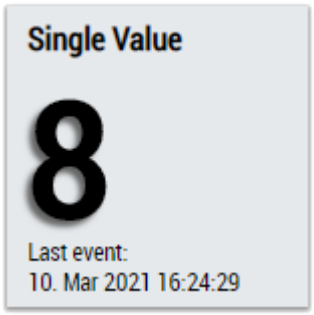
Output Values

- **Result:** Result of the mathematical operation.

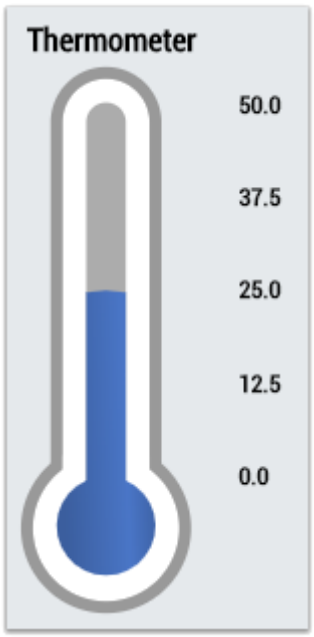
Standard HMI Controls

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

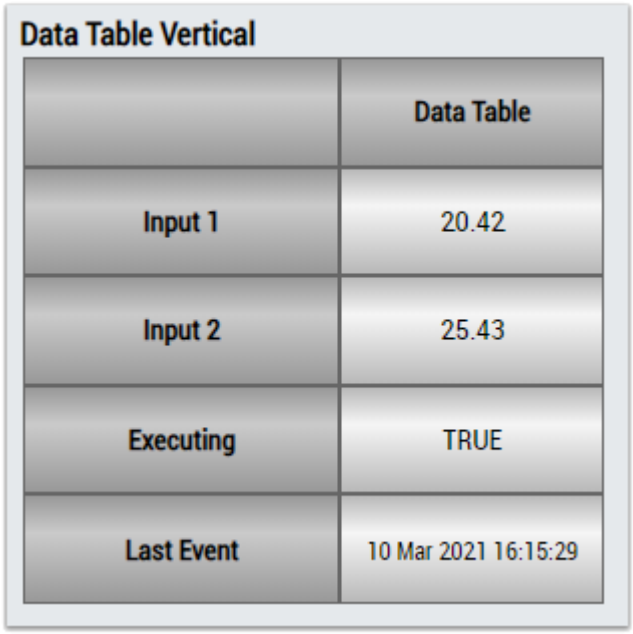
1. The SingleValue control visualizes the output value Result.

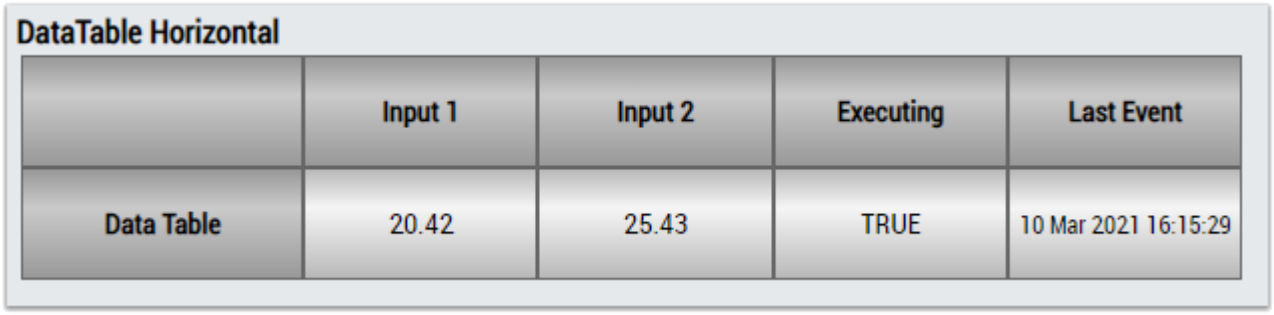


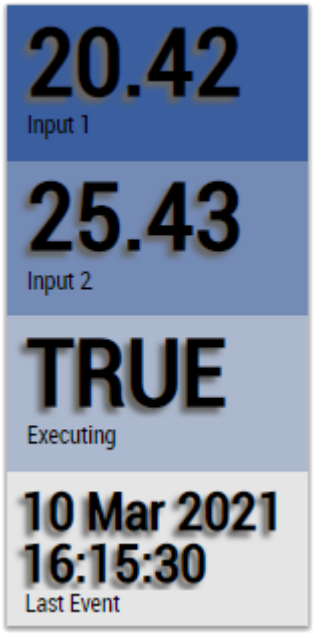
2. The Thermometer control visualizes the Result output value and can be used for temperature displays.



3. The Table Control or Multivalue Control visualizes all output values: Result.

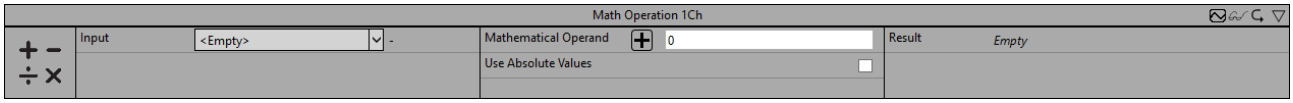






Alternatively, customer-specific HMI controls can be mapped in the Math Operation algorithm using the Mapping Wizard.

6.5.5.3 Math Operation 1Ch



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 ("+", "-", "x", "/", "x^y", "%").
- **Mathematical Operand:** Mathematical operand for the operation.
- **Use Absolute Values:** If the checkbox is checked, the absolute value of the input signal is used.

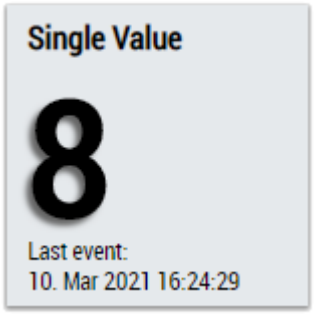
Output Values

- **Result:** Result of the mathematical operation.

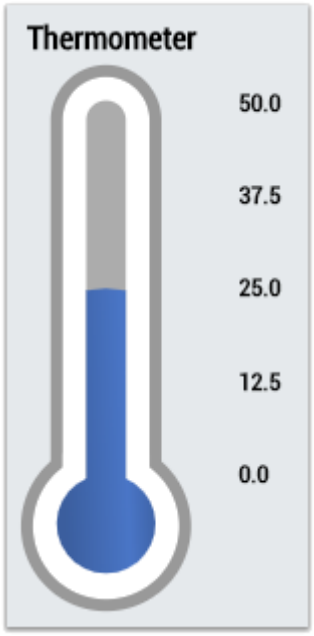
Standard HMI Controls

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

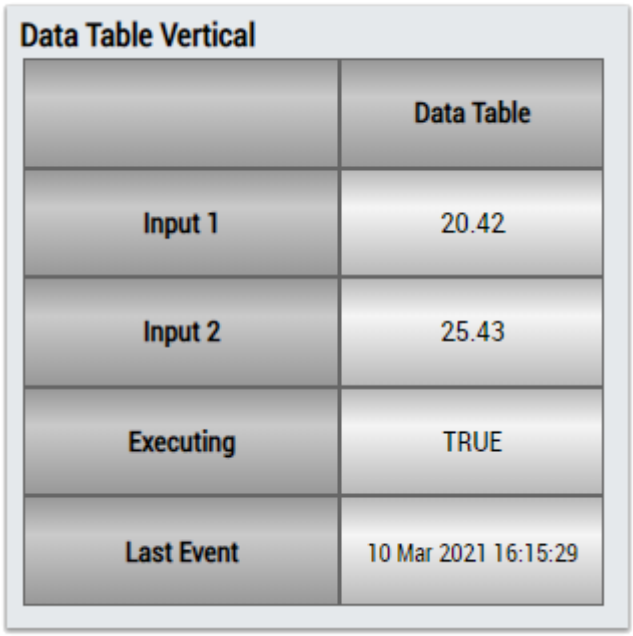
1. The SingleValue control visualizes the output value Result.



2. The Thermometer control visualizes the Result output value and can be used for temperature displays.

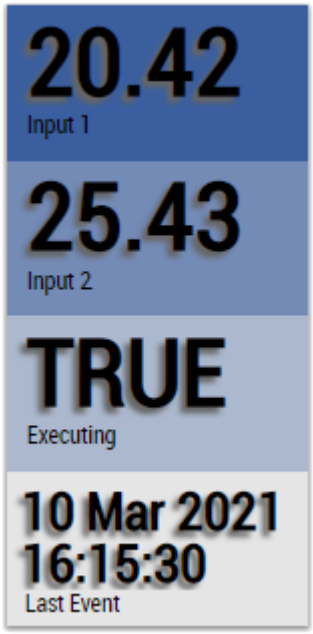


3. The Table Control or Multivalue Control visualizes all output values: Result.



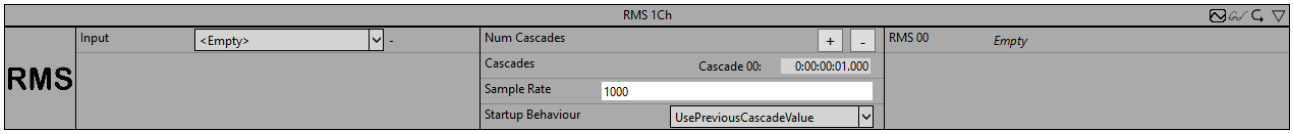
DataTable Horizontal

	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.

6.5.5.4 RMS 1Ch



RMS 1Ch calculates the root mean square over the input values according to the formula

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x[n]^2}$$

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:

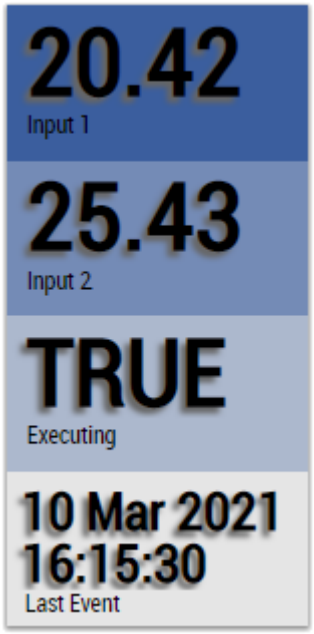
1. The Table Control or Multivalue Control visualizes all output values: RMS Results.

Data Table Vertical

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

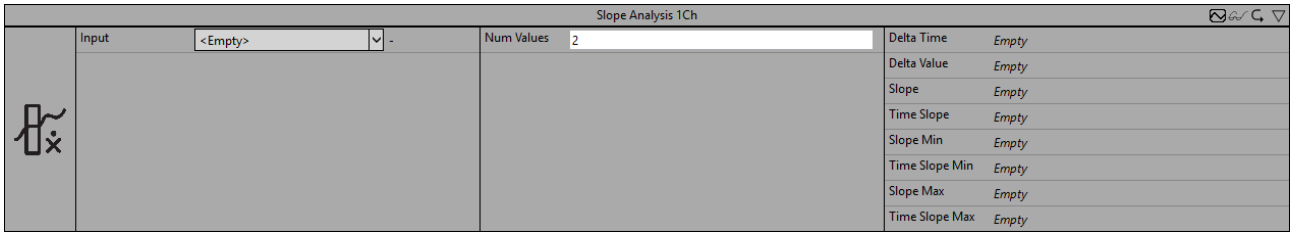
DataTable Horizontal

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



Alternatively, customer-specific HMI controls can be mapped in the RMS algorithm using the Mapping Wizard.

6.5.5.5 Slope Analysis 1Ch



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

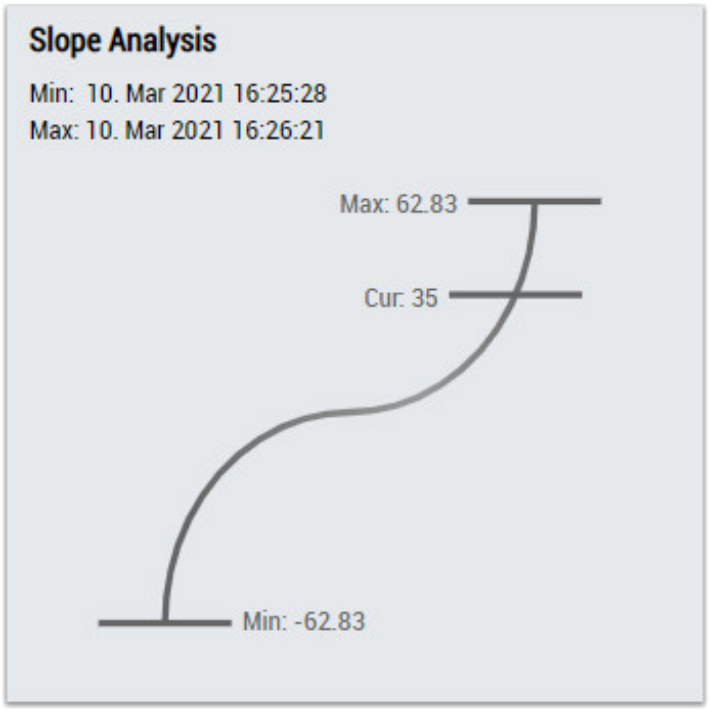
- **Slope:** Indicates the current slope value.

- **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

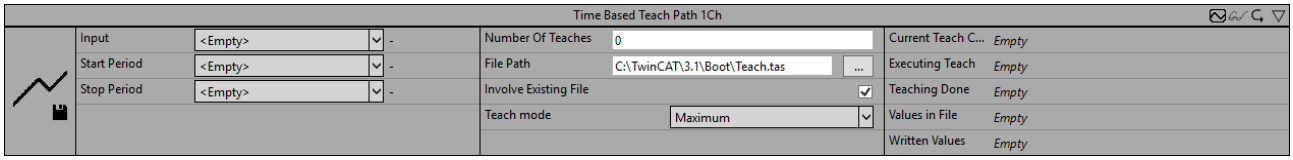
	Input 1	Input 2	Executing	Last Event
Data Table	20.42	25.43	TRUE	10 Mar 2021 16:15:29

Alternatively, customer-specific HMI controls can be mapped in the algorithm using the Mapping Wizard.

6.5.6 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.

6.5.6.1 Time Based Teach Path 1Ch



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 [▶ 142] 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 [▶ 142] due to competing file access. Instead, the reference signal should be learned first and only then should an evaluation with Time Based Envelope 1Ch [▶ 142] 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.

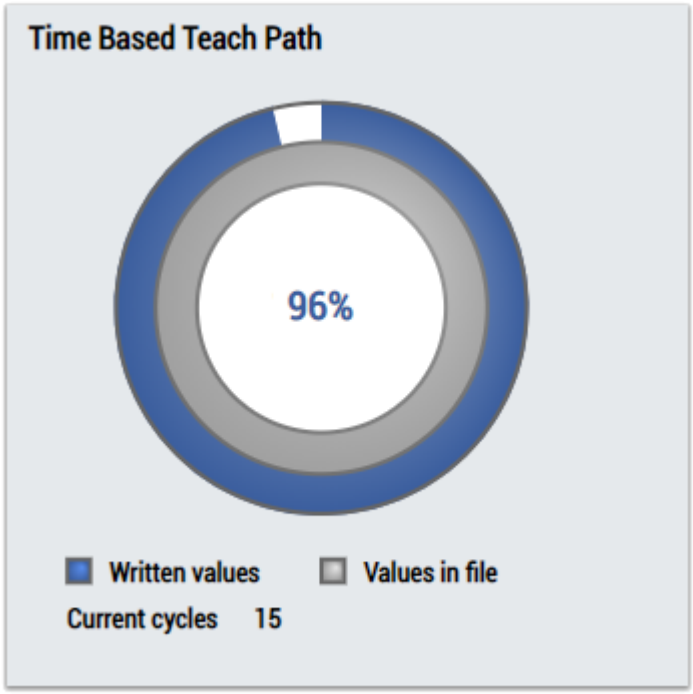
Output Values

- **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.



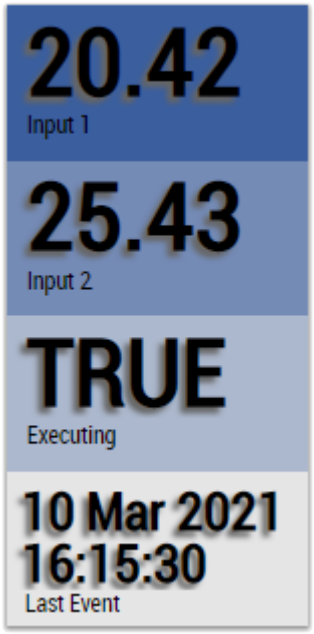
2. The Table Control or Multivalue Control visualizes all output values: Written Values, Values in File, Current Teach Cycles.

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

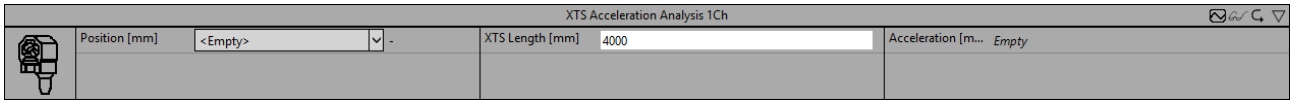


Alternatively, customer-specific HMI controls can be mapped in the Time Based Teach Path 1Ch algorithm using the Mapping Wizard.

6.5.7 Analytics - XTS

The algorithms of the category *Analytics-XTS* provide special functionalities for the Beckhoff XTS system. For example analysis of distance, velocity and acceleration.

6.5.7.1 XTS Acceleration Analysis 1Ch



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.

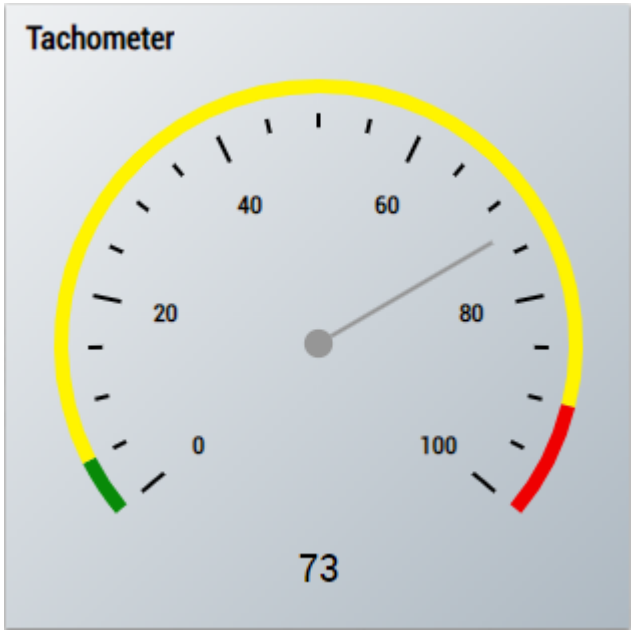
Output Values

- **Acceleration:** Current acceleration of the XTS mover.

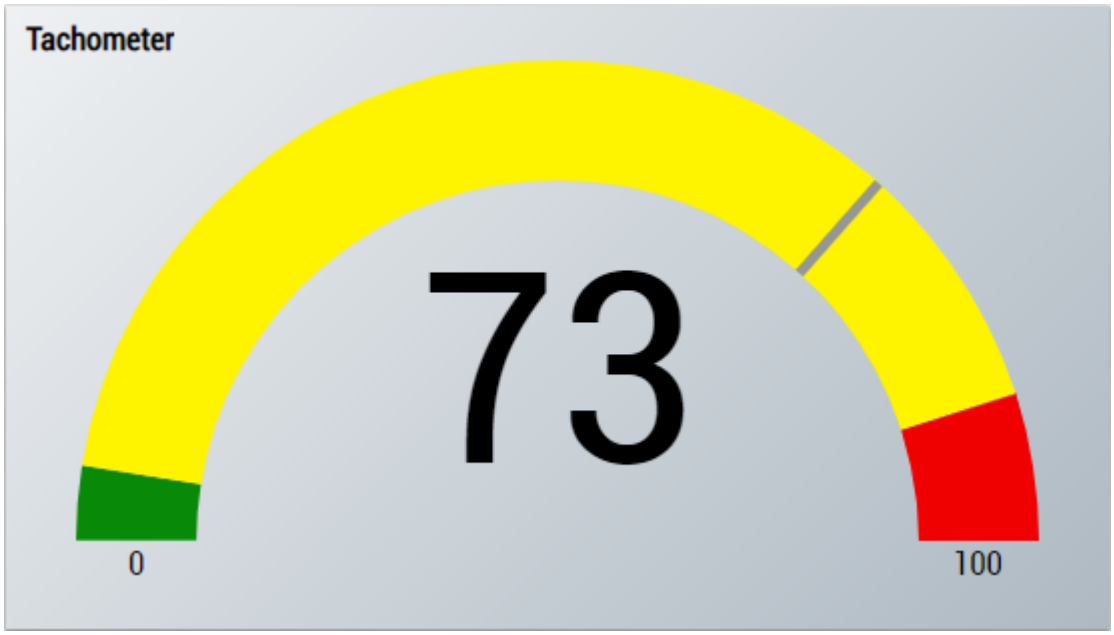
Standard HMI Controls

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

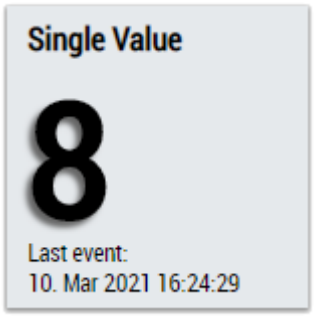
1. The Tachometer control visualizes the output value Acceleration.



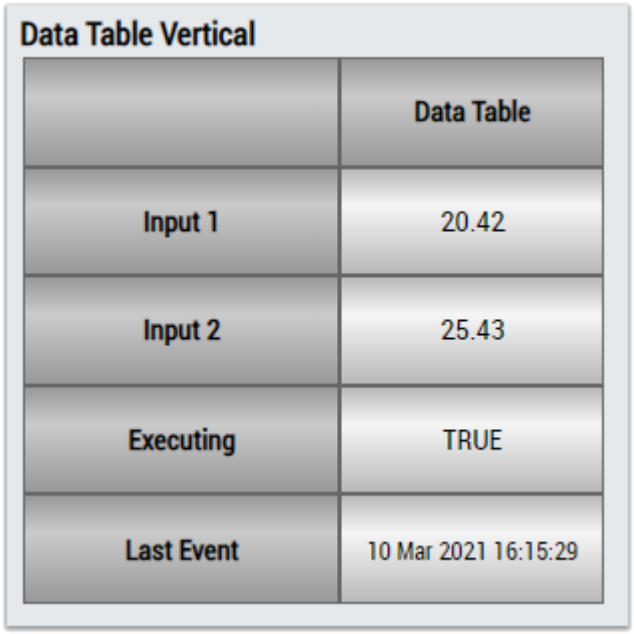
2. The Radial Gauge control visualizes the output value Acceleration.

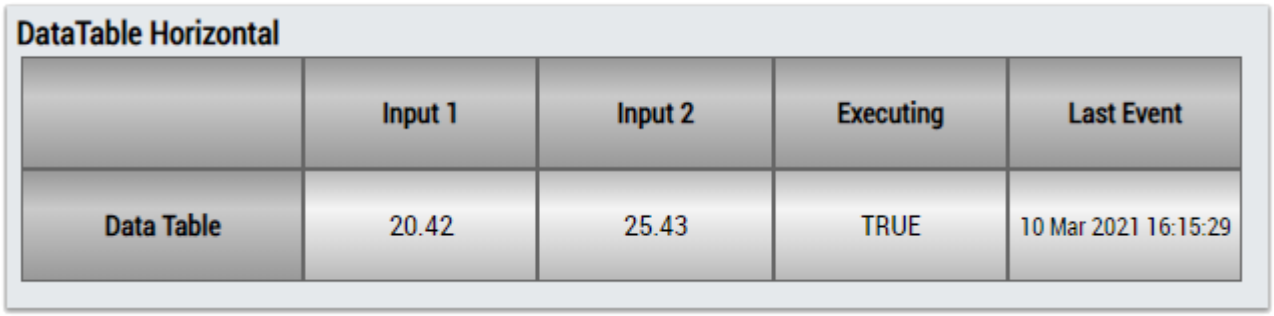


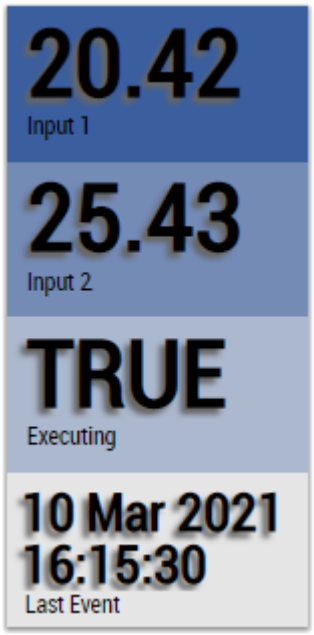
3. The SingleValue control visualizes the output value Acceleration.



4. The Table Control or Multivalue Control visualizes all output values: Acceleration.

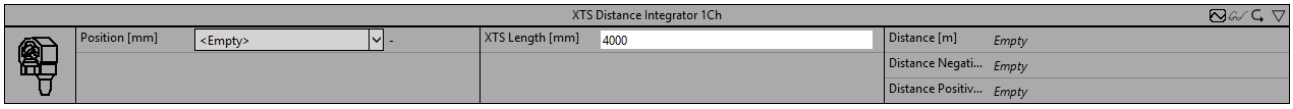






Alternatively, customer-specific HMI controls can be mapped in the XTS Acceleration Analysis 1Ch algorithm using the Mapping Wizard.

6.5.7.2 XTS Distance Integrator 1Ch



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.

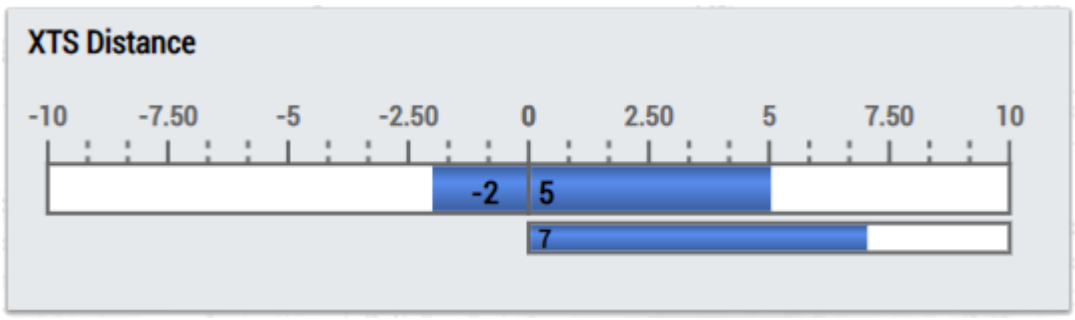
Output Values

- **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).

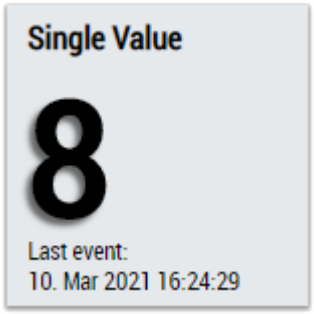
Standard HMI Controls

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

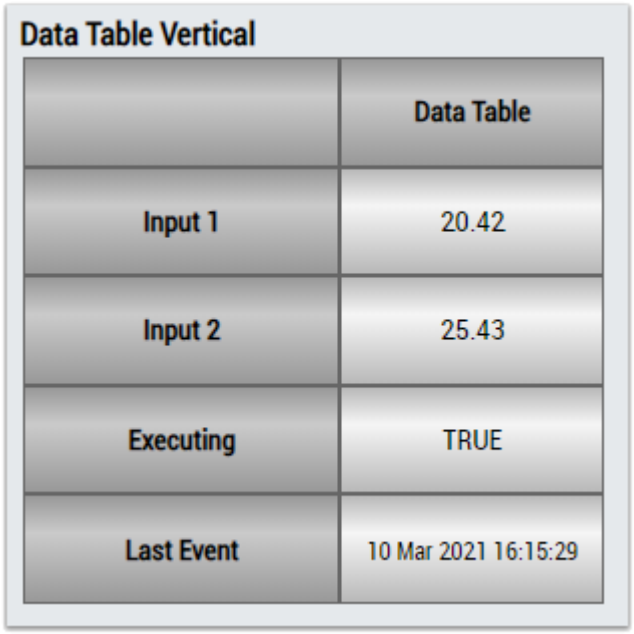
1. The XTSDistance control visualizes the output values Distance, Distance Positive and Distance Negative.

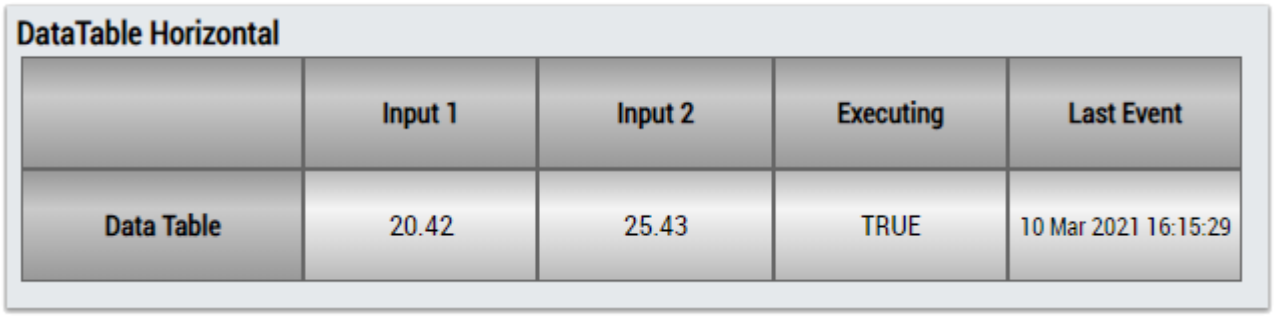


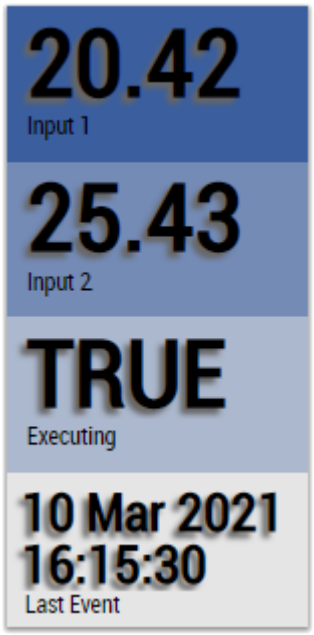
2. The SingleValue control visualizes the output value Distance.



3. The Table Control or Multivalue Control visualizes all output values: Output values Distance, Distance Positive, Distance Negative.

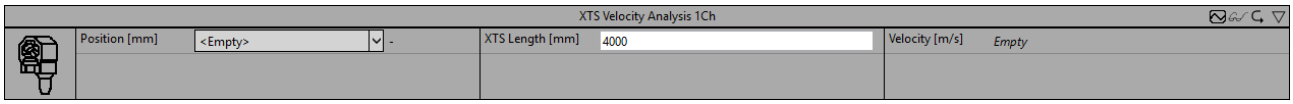






Alternatively, customer-specific HMI controls can be mapped in the XTS Distance Integrator 1Ch algorithm using the Mapping Wizard.

6.5.7.3 XTS Velocity Analysis 1Ch



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.

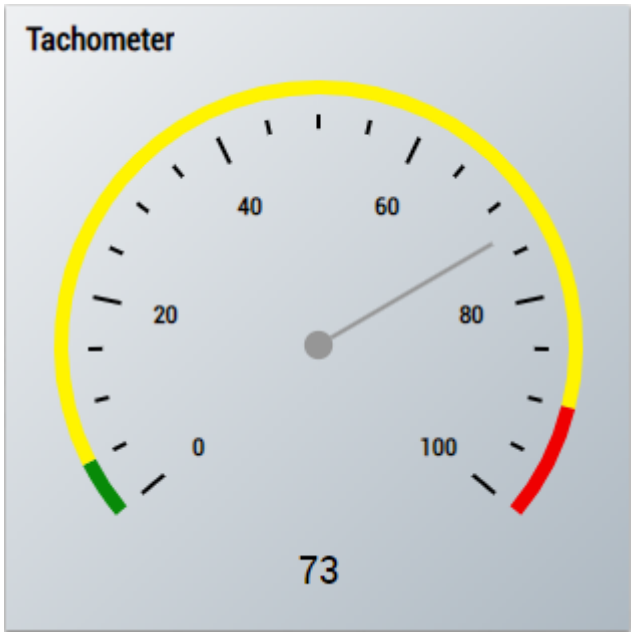
Output Values

- **Velocity:** Current velocity of the XTS mover.

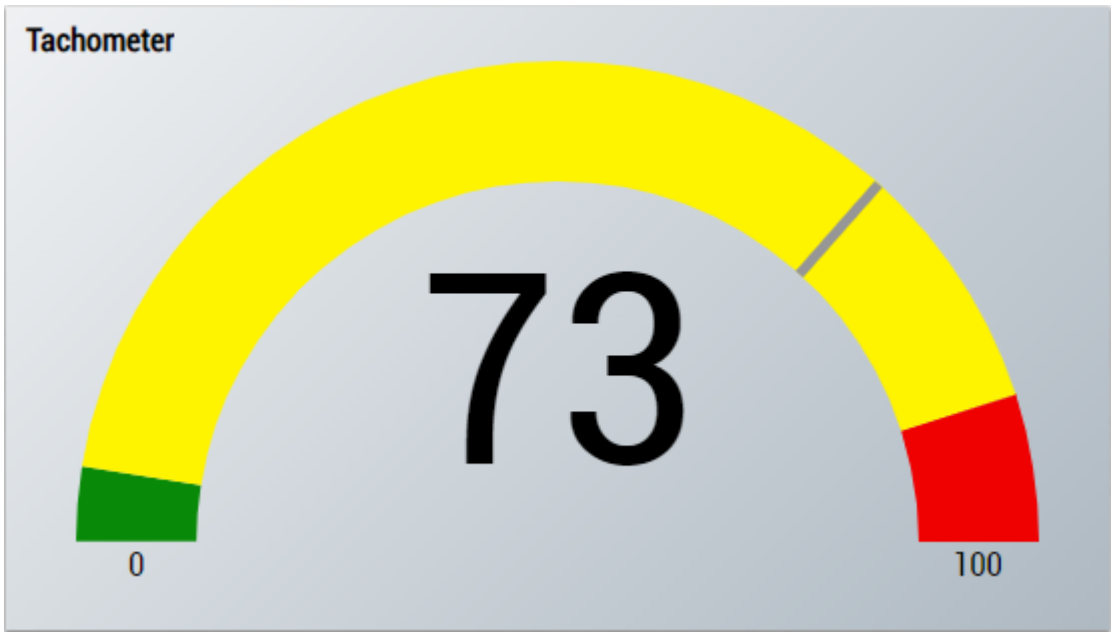
Standard HMI Controls

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

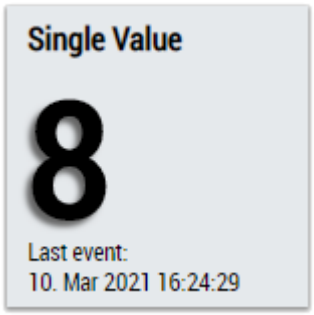
1. The Tachometer control visualizes the output value Velocity.



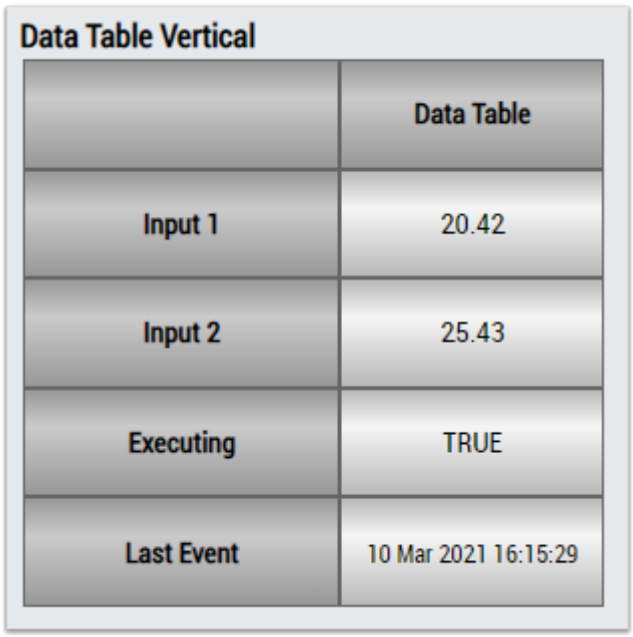
2. The Radial Gauge control visualizes the output value Velocity.

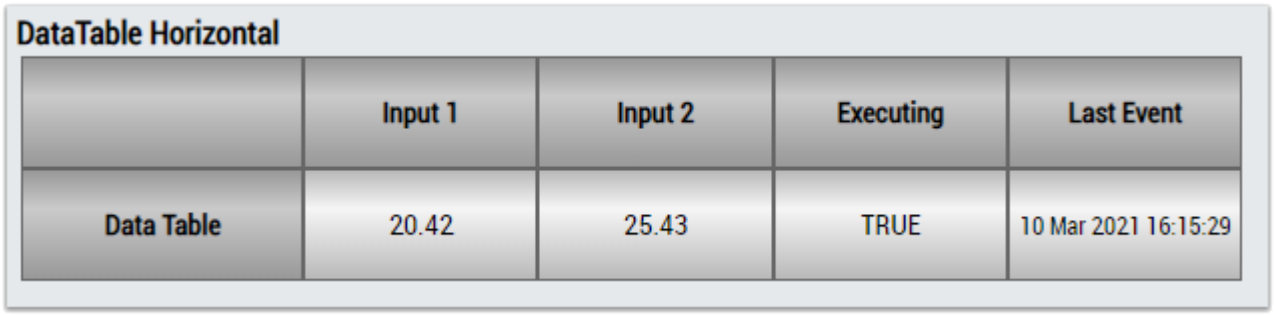


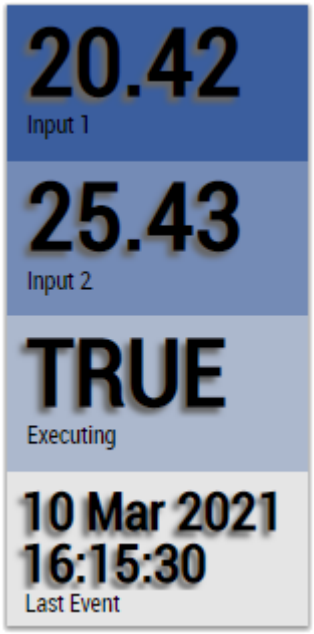
3. The SingleValue control visualizes the output value Velocity.



4. The Table Control or Multivalue Control visualizes all output values: Velocity.





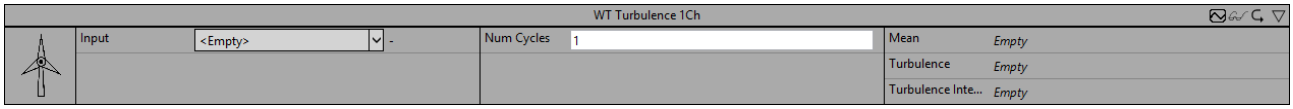


Alternatively, customer-specific HMI controls can be mapped in the XTS Velocity Analysis 1Ch algorithm using the Mapping Wizard.

6.5.8 Analytics - WT

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.

6.5.8.1 WT Turbulence 1Ch



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.

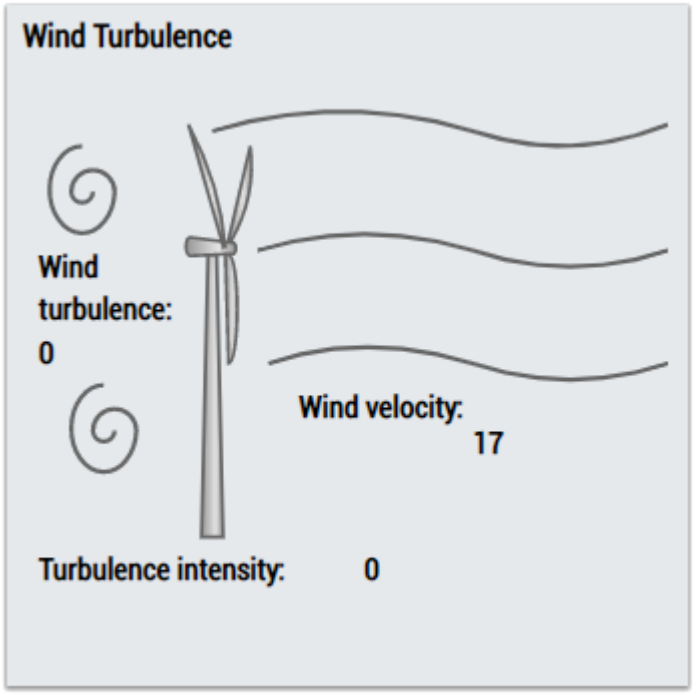
Output Values

- **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.



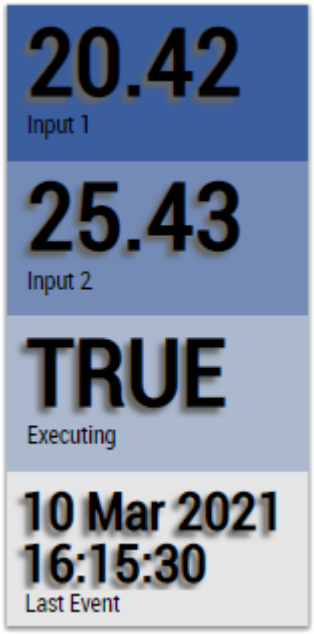
2. The Table Control or Multivalue Control visualizes all output values: Mean, Turbulence, Turbulence Intensity.

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

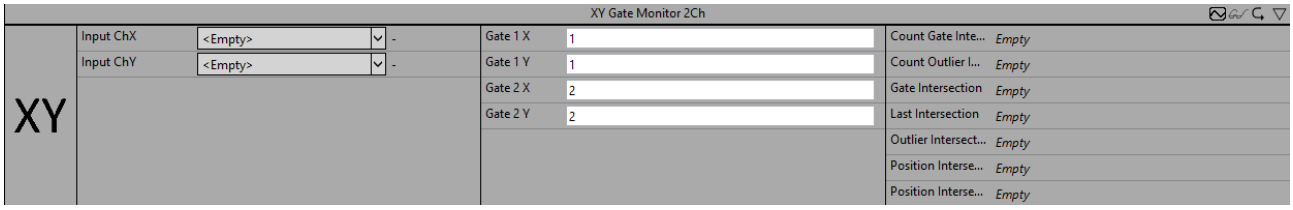


Alternatively, customer-specific HMI controls can be mapped in the WT Turbulence 1Ch algorithm using the Mapping Wizard.

6.5.9 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.

6.5.9.1 XY Gate Monitor 2Ch

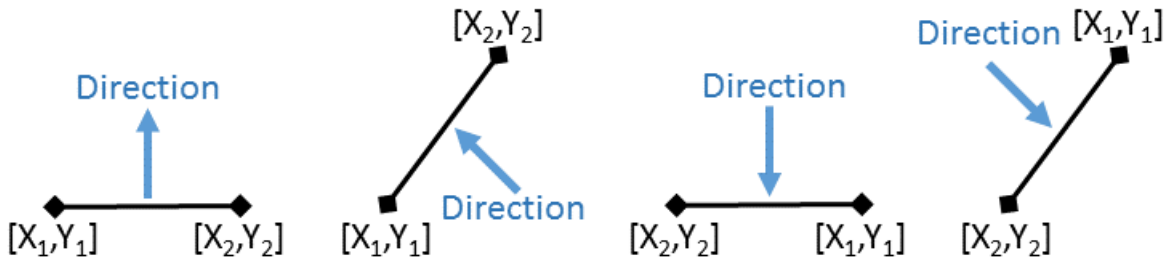


The *XY Gate Monitor 2Ch* counts the amount of intersections of an XY input with a specified gate or its projection (straight line between the gate points) depending on the configured Gate Mode. The analysis period can be started with the inputs *Start* and *Stop*. The algorithm is direction sensitive, which means that just intersection in the right direction are counted. The direction interpretation depends on the order of the gate points (X1/Y1) and (X2/Y2). 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 (X1/Y1) and (X2/Y2). The direction of the intersection points is counted when the signal rotates counterclockwise around the first gate point (X1/Y1).



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.

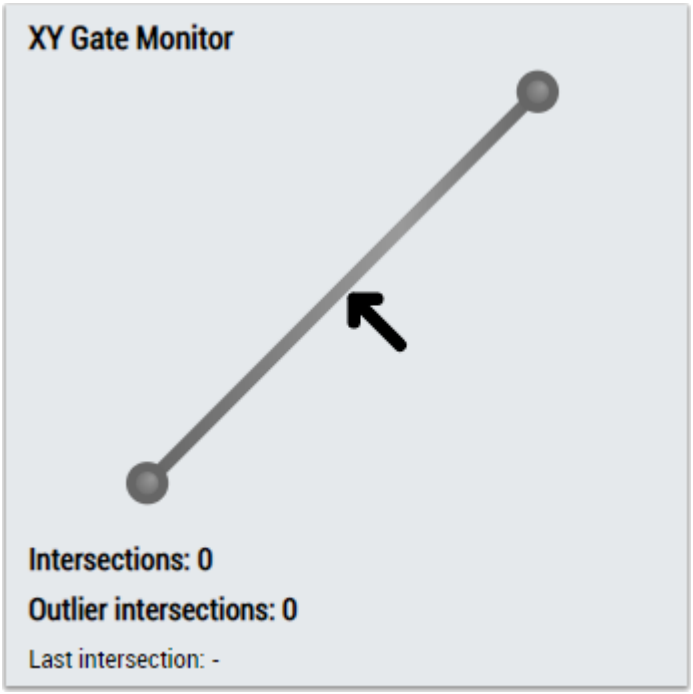
Output values

- **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.



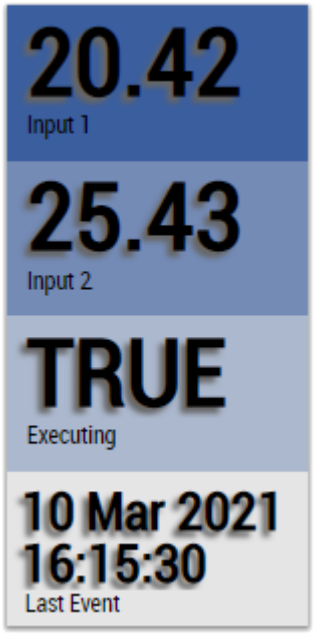
2. The Table Control or Multivalue Control visualizes all output values: Executing, Gate Intersection, Outlier Intersection, Position Intersection X, Position Intersection Y, Count Gate Intersections, Count Outlier Intersections, Last Intersection, Classification.

Data Table Vertical

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

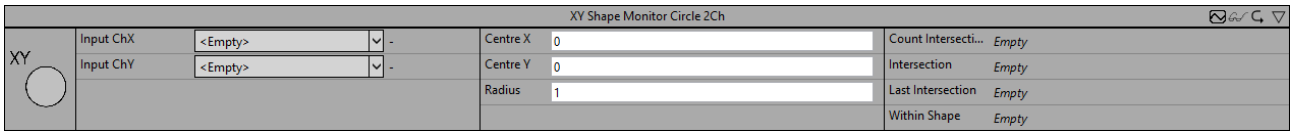
DataTable Horizontal

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



Alternatively, customer-specific HMI controls can be mapped in the XY Gate Monitor 2Ch algorithm using the Mapping Wizard.

6.5.9.2 XY Shape Monitor Circle 2Ch



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.

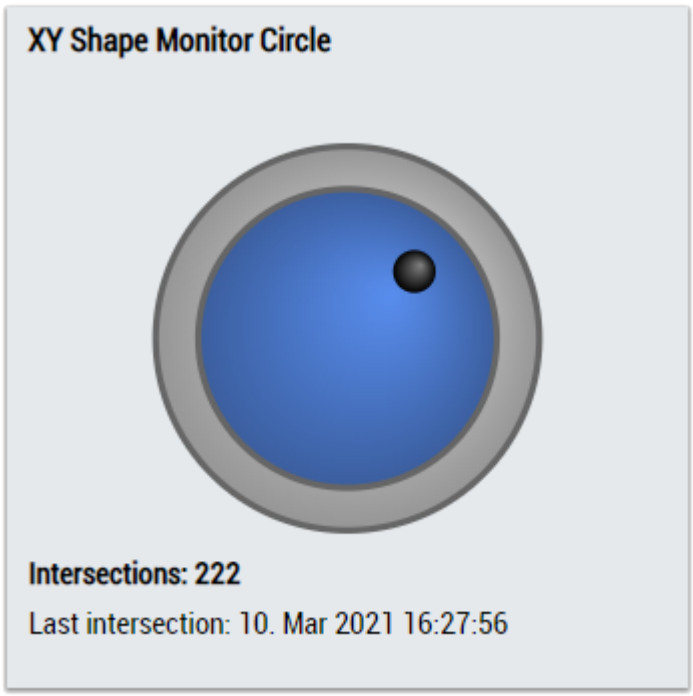
Output values

- **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.

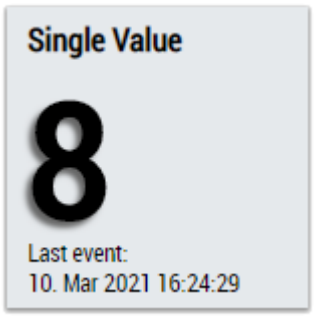
Standard HMI Controls

For the XY Shape Monitor Circle 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

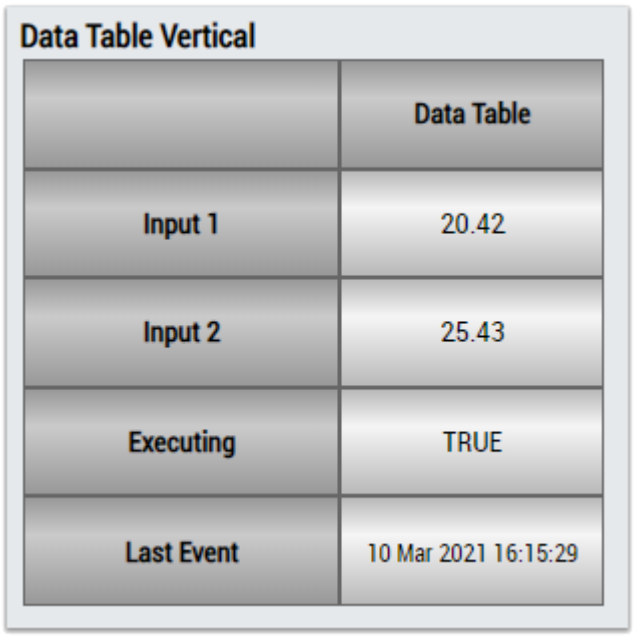
1. The XYShapeMonitor control visualizes the output values Within Shape, Intersection, Count Intersections and Last Intersection.



2. The SingleValue control visualizes the output values Intersection and Last Intersection.

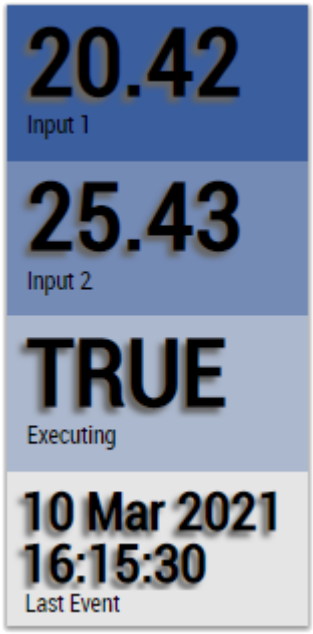


3. The Table Control or Multivalue Control visualizes all output values: Within Shape, Intersection, Count Intersections, Last Intersection.



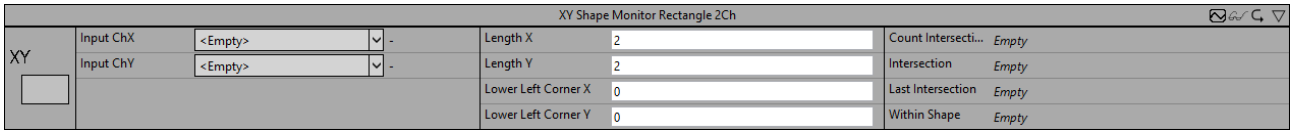
DataTable Horizontal

	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 Circle 2Ch algorithm using the Mapping Wizard.

6.5.9.3 XY Shape Monitor Rectangle 2Ch



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.

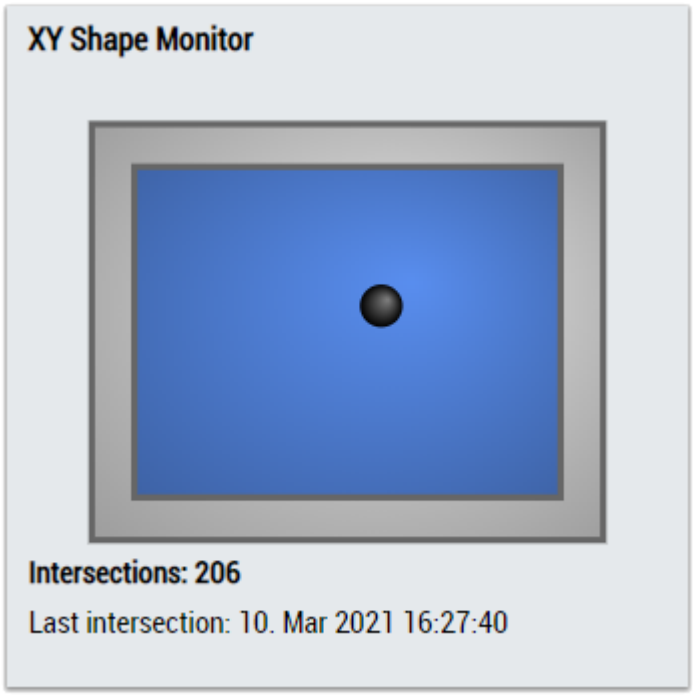
Output values

- **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.

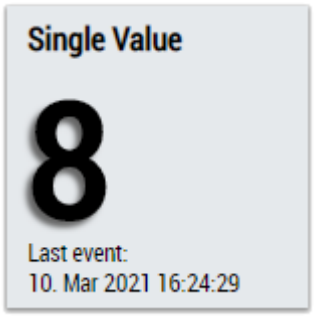
Standard HMI Controls

For the XY Shape Monitor Rectangle 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

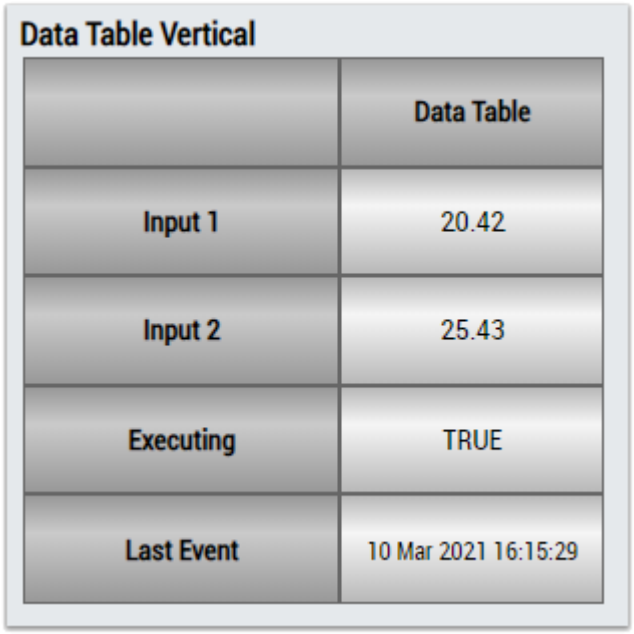
1. The XYShapeMonitor control visualizes the output values Within Shape, Intersection, Count Intersections and Last Intersection.

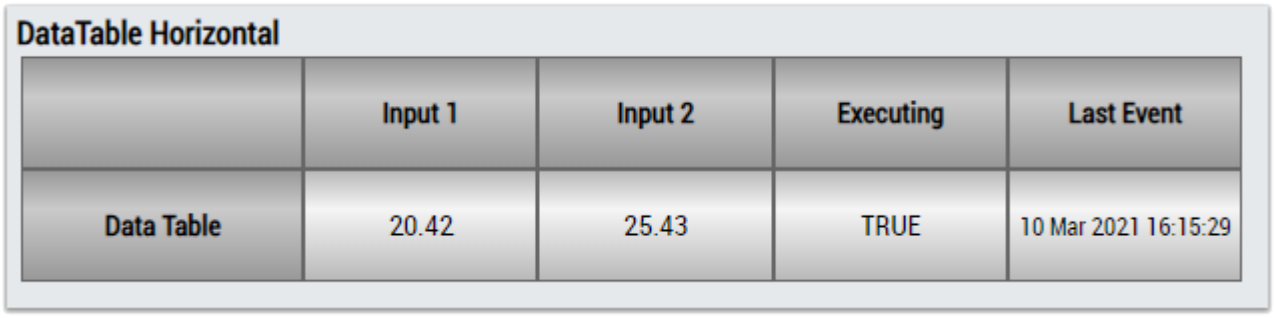


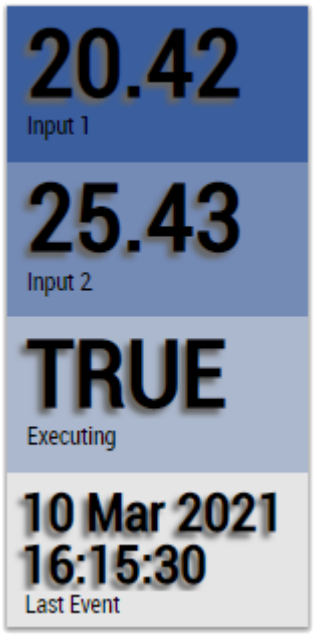
2. The SingleValue control visualizes the output values Intersection and Last Intersection.



3. The Table Control or Multivalue Control visualizes all output values: Within Shape, Intersection, Count Intersections, Last Intersection.

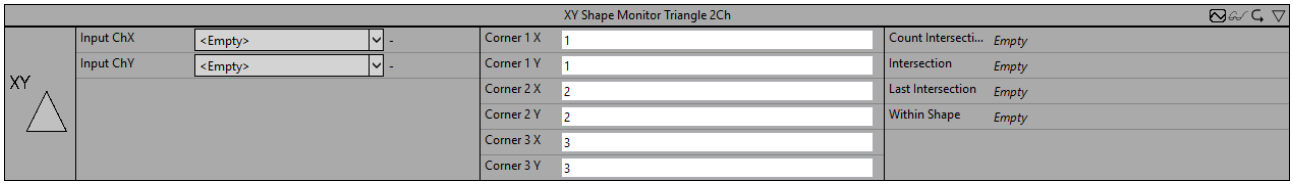






Alternatively, customer-specific HMI controls can be mapped in the XY Shape Monitor Rectangle 2Ch algorithm using the Mapping Wizard.

6.5.9.4 XY Shape Monitor Triangle 2Ch



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.

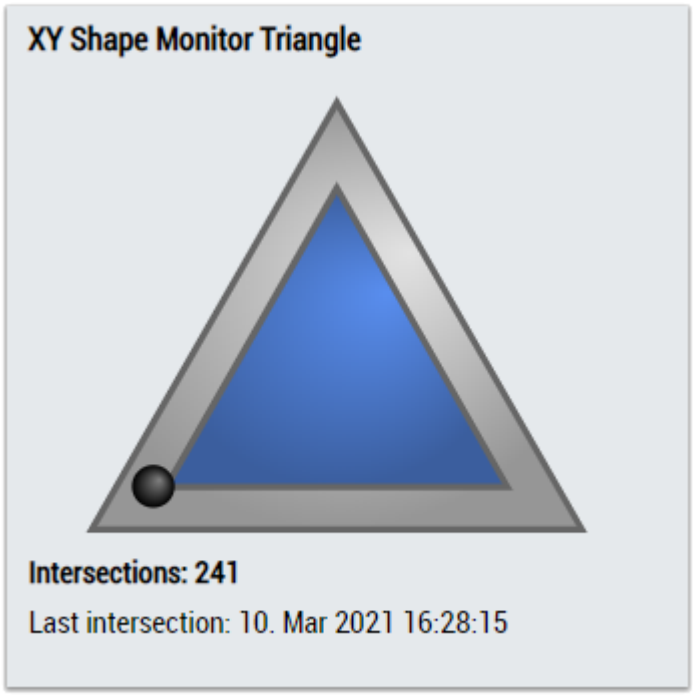
Output values

- **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.

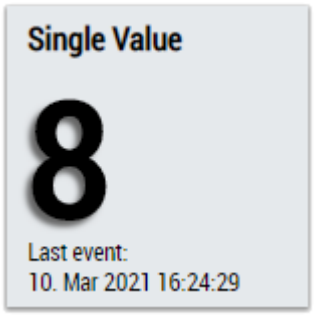
Standard HMI Controls

For the XY Shape Monitor Triangle 2Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

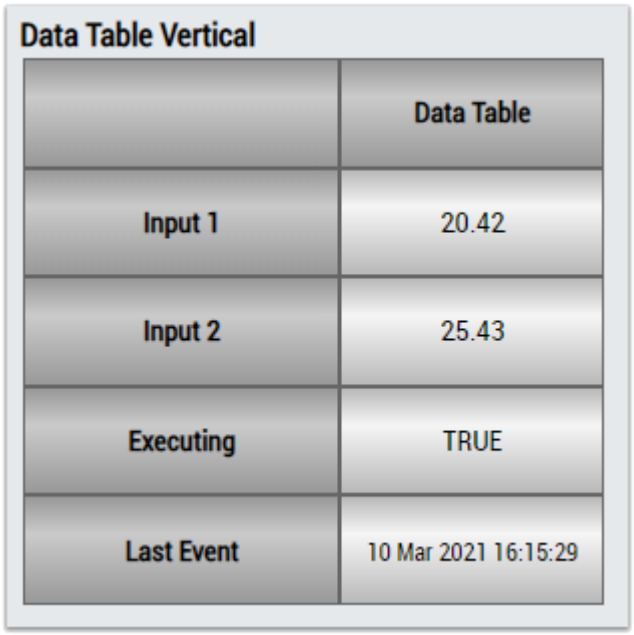
1. The XYShapeMonitor control visualizes the output values Within Shape, Intersection, Count Intersections and Last Intersection.

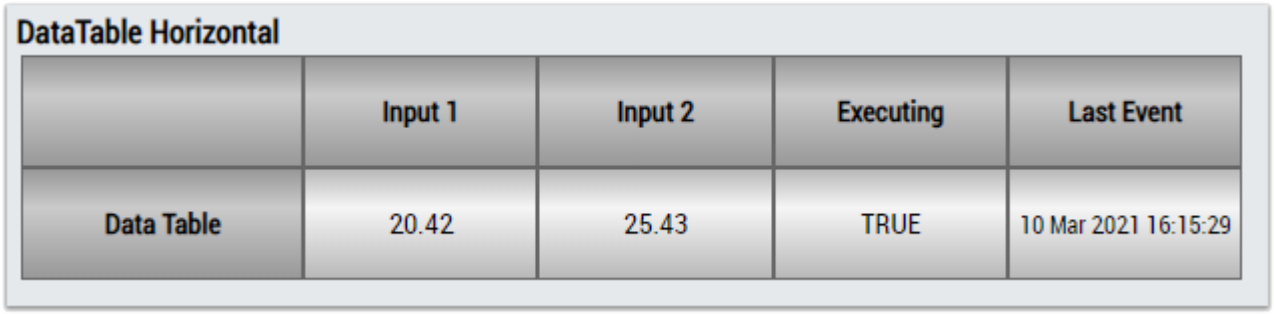


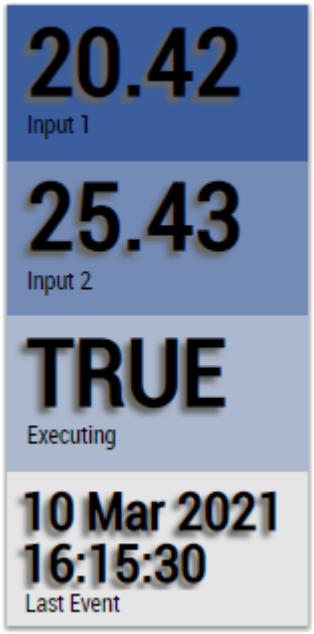
2. The SingleValue control visualizes the output values Intersection and Last Intersection.



3. The Table Control or Multivalue Control visualizes all output values: Within Shape, Intersection, Count Intersections, Last Intersection.





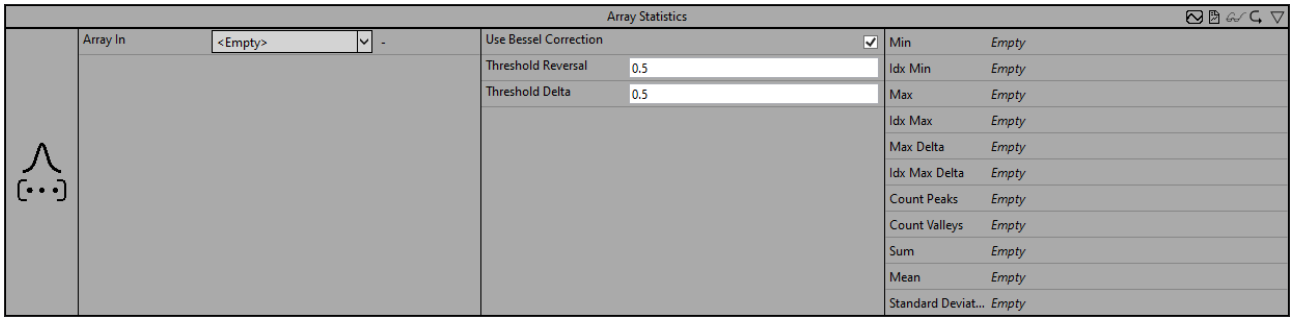


Alternatively, customer-specific HMI controls can be mapped in the XY Shape Monitor Triangle 2Ch algorithm using the Mapping Wizard.

6.5.10 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.

6.5.10.1 Array Statistics



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.

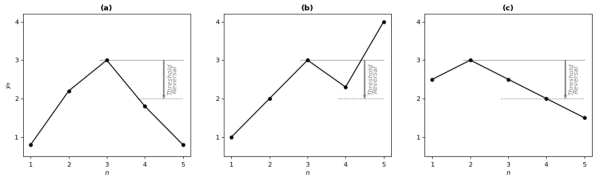
The empirical standard deviation, without Bessel's correction

$$s' = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x[n] - \overline{x})^2}$$

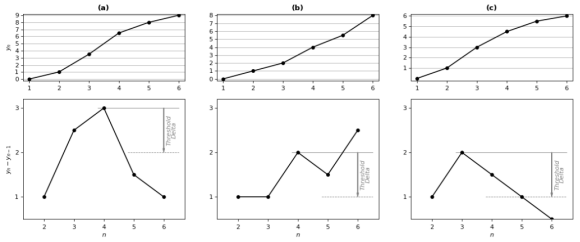
The empirical standard deviation, with Bessel's correction

$$s = \sqrt{\frac{1}{N-1}\sum_{n=0}^{N-1}(x[n]-\overline{x})^2}$$

- **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.

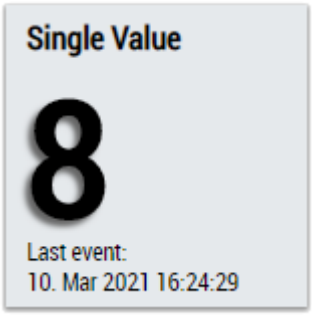
Output values

- **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.

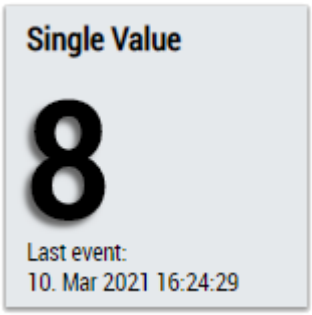
Standard HMI Controls

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

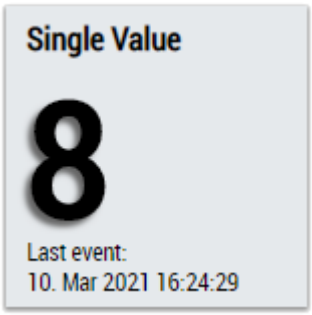
1. The SingleValue control visualizes the Sum output.



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

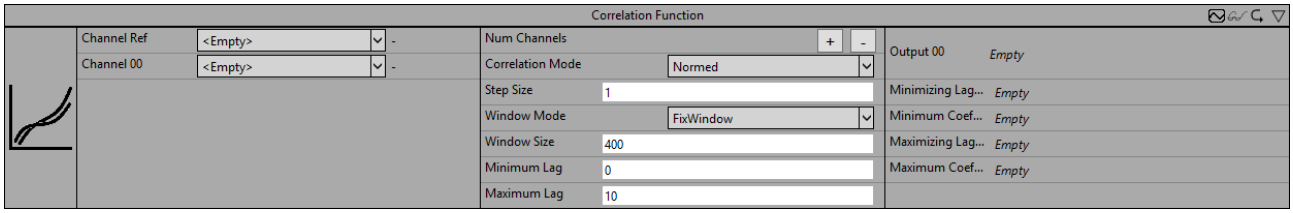
	Data Table
Input 1	20.42
Input 2	25.43
Executing	TRUE
Last Event	10 Mar 2021 16:15:29

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.

6.5.10.2 Correlation Function



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.

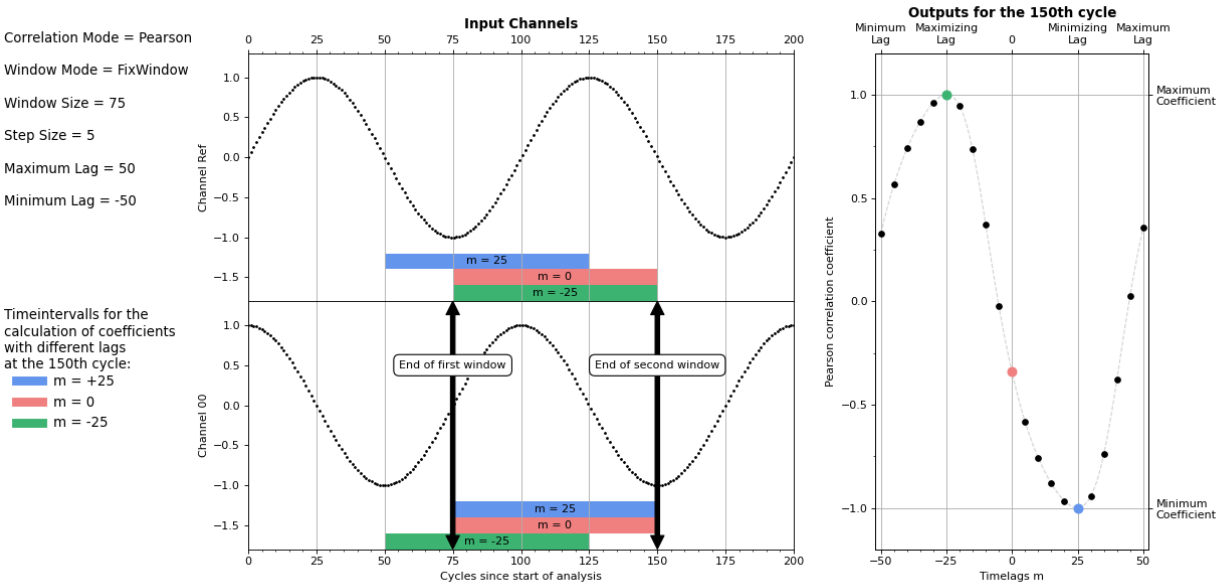
If *m* is not zero, the window for the corresponding signal shifts by *m* cycles. The number of values included in the calculation of the coefficients is the same for all values of *m*. The only exceptions to this are the results from the first cycles after the start of the analysis. If the number of 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 nth 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:*

$$\rho_{xy}[m,t_n] = \frac{cov(x,y)[m,t_n]}{\sigma_x[m,t_n]\sigma_y[m,t_n]}, where \;\; \sigma_x^2[m,t_n] = cov_{xx}[m,t_n]$$

- **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.

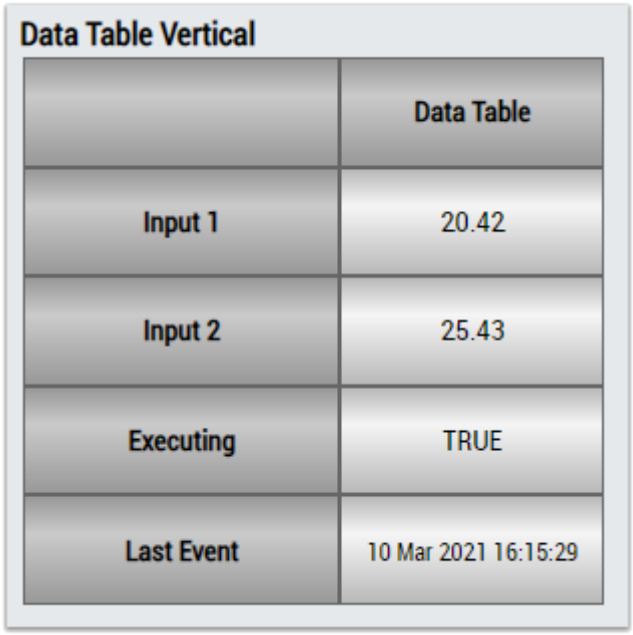
Output values

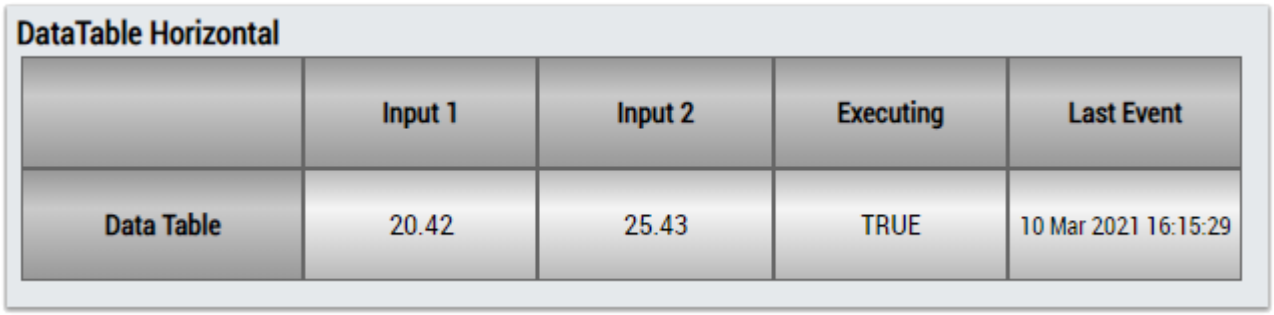
- **Output00 .. Output0n:** Shows for each input (Channel00, ..., Channel0n) the coefficients for the shifts *m* = *Minimum Lag*, *m* = *Minimum Lag* + *Step Size*, ..., *m* = - *Step Size*, *m* = 0, *m* = +*StepSize*, ..., m = *Maximum Lag* as an array.
- **Minimizing Lag00..Minimizing Lag0n:** For 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.

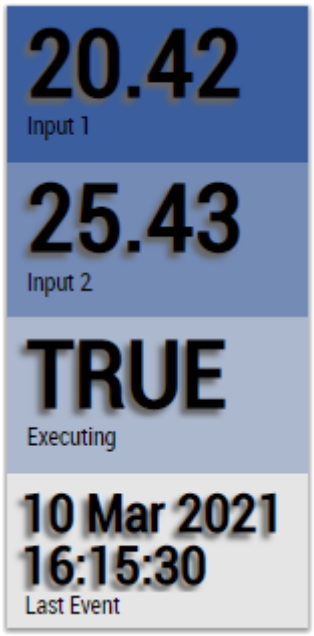
Standard HMI Controls

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.

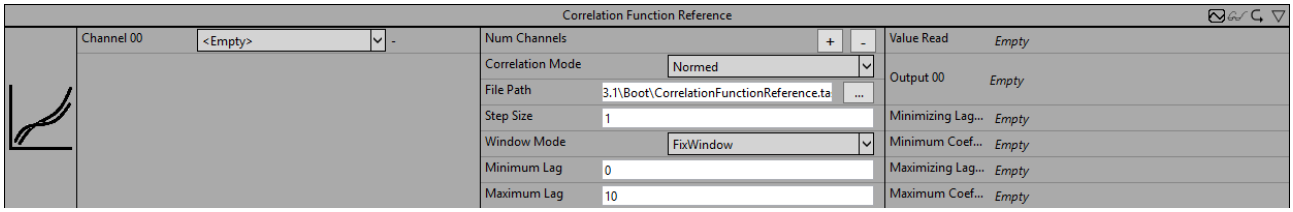






Alternatively, customer-specific HMI controls can be mapped in the Correlation Function algorithm using the Mapping Wizard.

6.5.10.3 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.

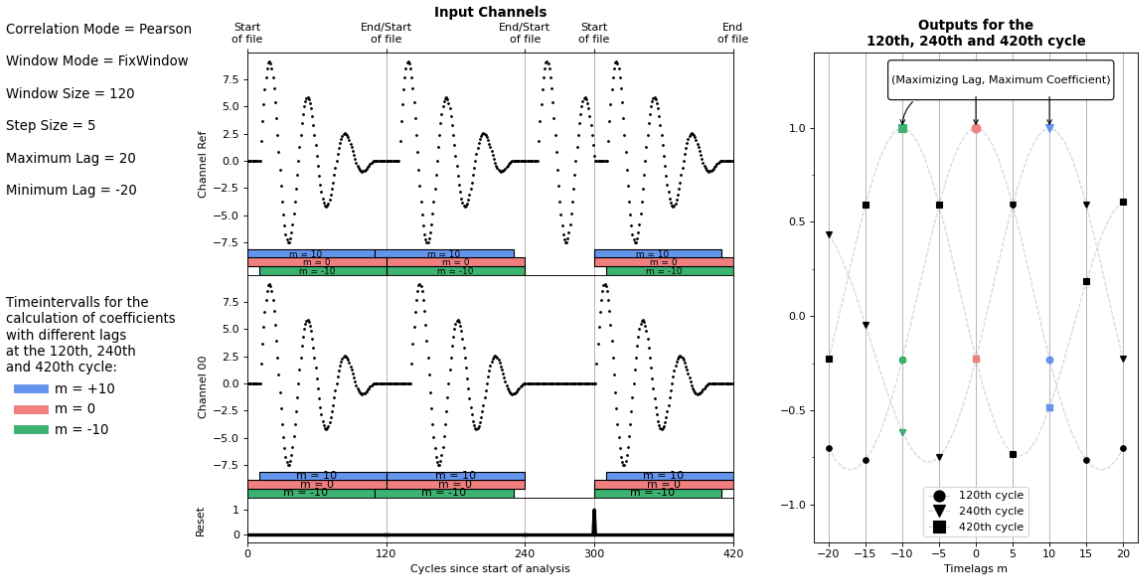
If *m* is not zero, the window for the corresponding signal shifts by *m* cycles. The number of values included in the calculation of the coefficients is the same for all values of *m*. The only exceptions to this are the results from the first cycles after the start of the analysis. If the number of 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, .. 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 input 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. 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:*

$$\rho_{xy}[m,t_n] = \frac{cov(x,y)[m,t_n]}{\sigma_x[m,t_n]\sigma_y[m,t_n]}, where \;\; \sigma_x^2[m,t_n] = cov_{xx}[m,t_n]$$

- **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

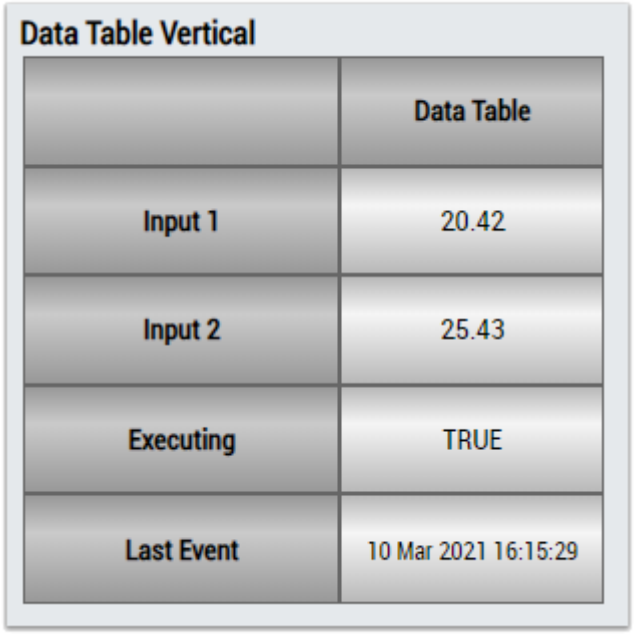
- **Output00 .. Output0n:** Shows for each input (Channel00, ..., Channel0n) the coefficients for the shifts *m* = *Minimum Lag*, *m* = *Minimum Lag* + *Step Size*, ..., *m* = - *Step Size*, *m* = 0, *m* = +*StepSize*,..., m = *Maximum Lag* as an array.
- **Minimizing Lag00..Minimizing Lag0n:** For each channel, specifies the shift for which the coefficient becomes minimum.
- **Minimum Coefficient00..Minimum Coefficient0n:** Specifies the minimum coefficient for each channel.

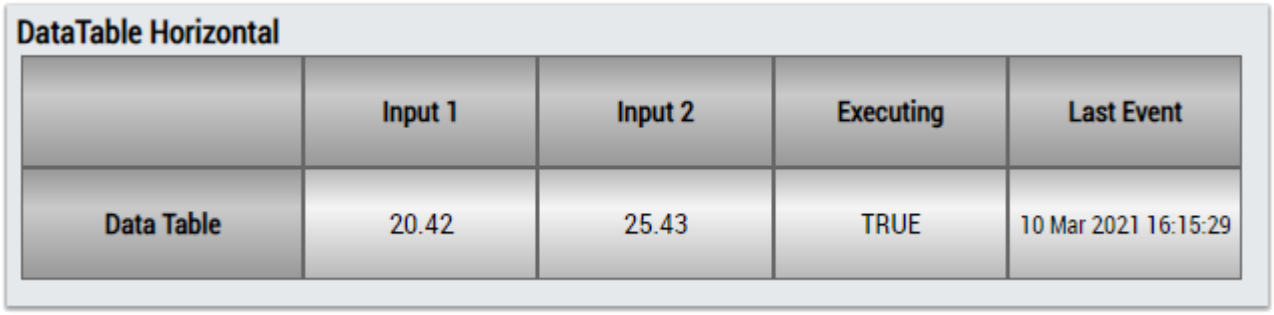
- **Maximizing Lag00..Maximizing Lag0n:** for each channel, specifies the shift for which the coefficient becomes maximum.
- **Maximum Coefficient 00..00.. Maximum Coefficient0n:** Specifies the maximum coefficient for each channel.

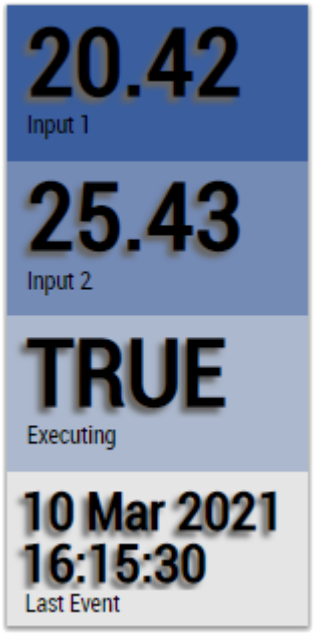
Standard HMI Controls

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.

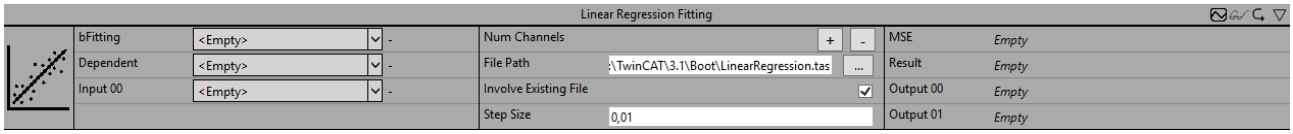






Alternatively, customer-specific HMI controls can be mapped in the Correlation Function Reference algorithm using the Mapping Wizard.

6.5.10.4 Linear Regression Fitting



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:

$$\beta_i = \begin{cases} \beta_i - \gamma \times \text{Input 0i} \times (y - \text{Dependent}), & i = 1, 2, \ldots, n \\ \beta_i - \gamma \times (y - \text{Dependent}), & i = 0 \end{cases}$$

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 path for the file where the coefficients are saved at the end of the approximation process.
- **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.

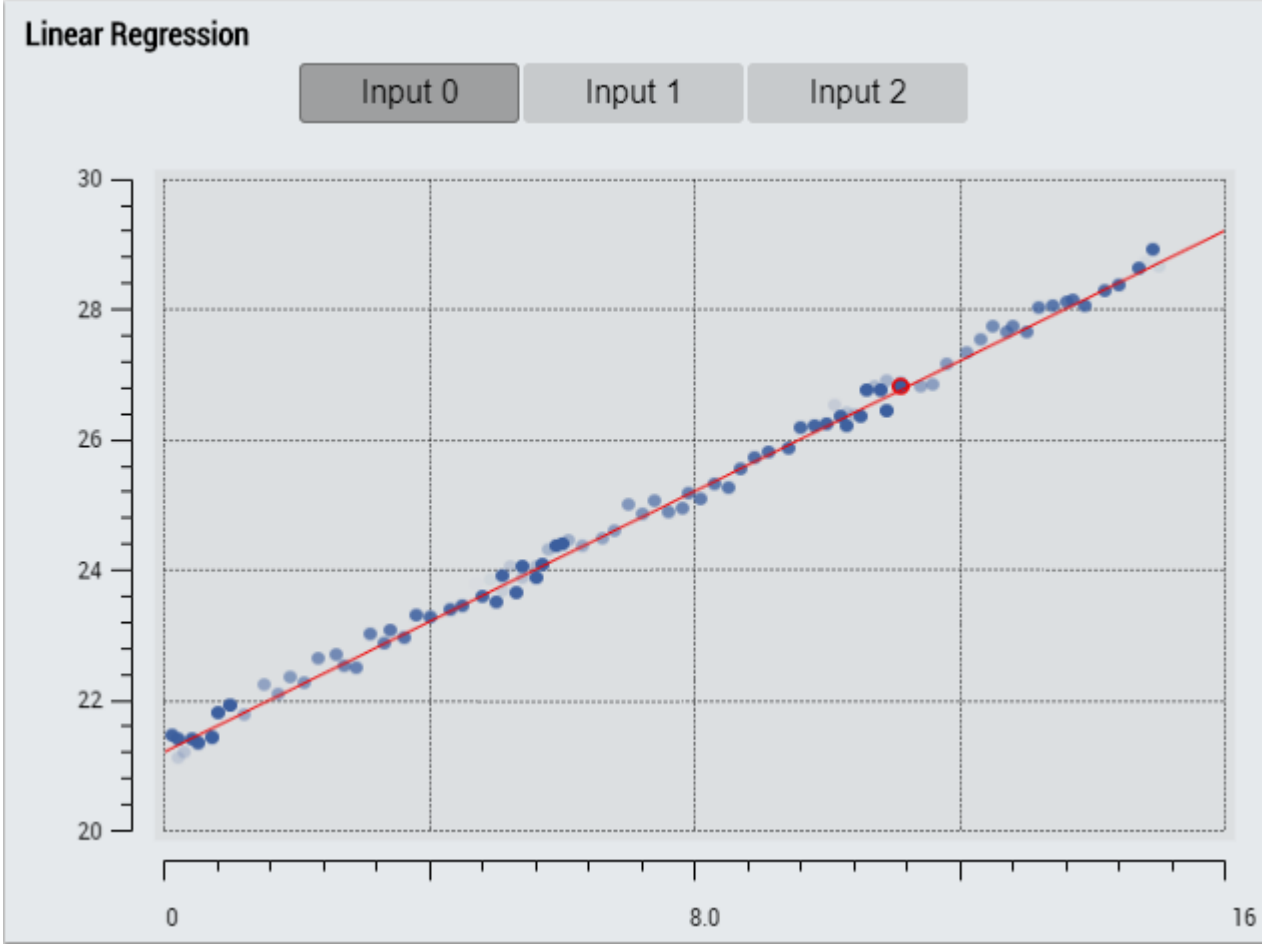
Output values

- **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.

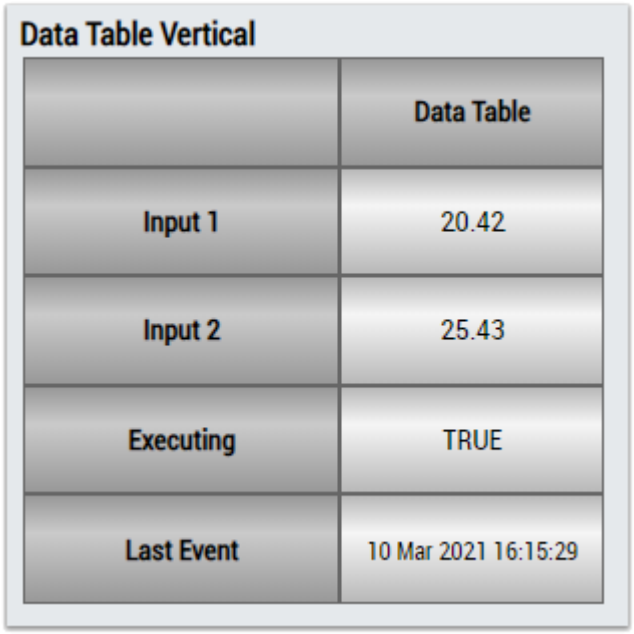
Standard HMI Controls

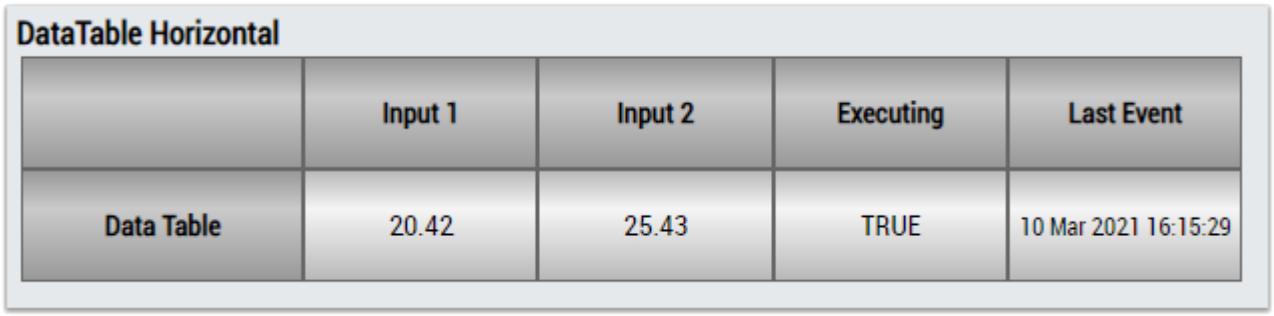
For the Linear Regression Fitting algorithm, the following HMI controls are available for generating an Analytics Dashboard:

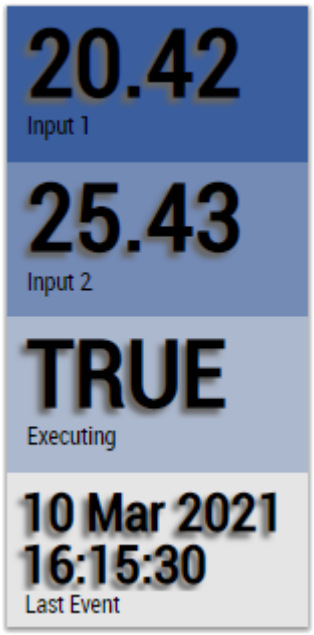
1. The Linear Regression Control visualizes the inputs and the calculated regression line. The buttons can be used to select the input channel to be displayed on the x axis. The y axis shows the target values of the regression. A new point is outlined in red, old points gradually fade.



2. The Table Control or Multivalue Control visualizes all output values: MSE, result, output coefficients.

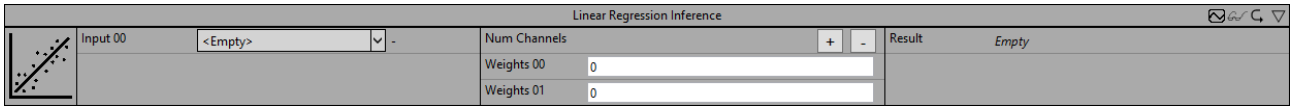






Alternatively, customer-specific HMI controls can be mapped in the Linear Regression Curve Fitting algorithm using the Mapping Wizard.

6.5.10.5 Linear Regression Inference



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.

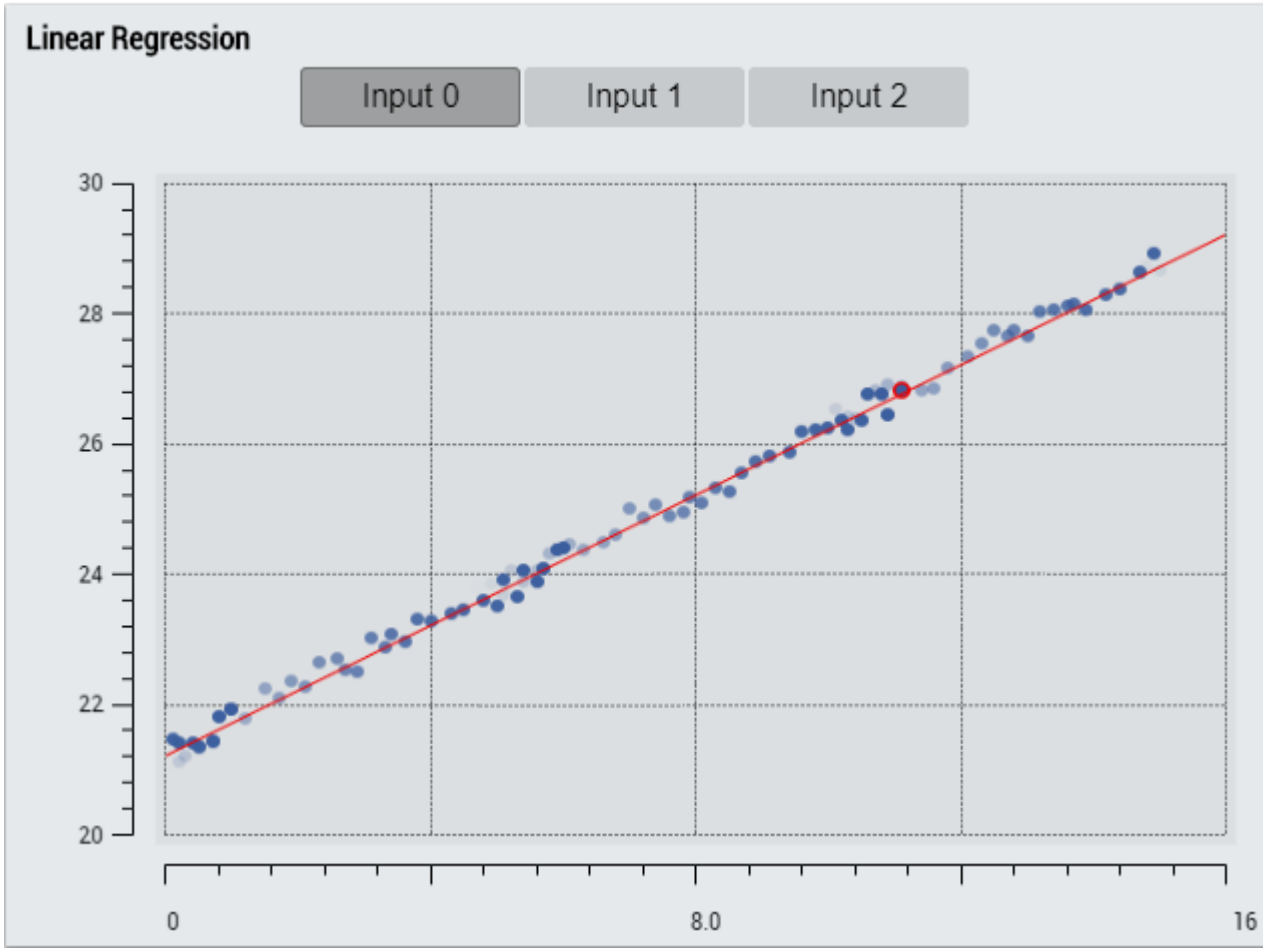
Output values

- **Result:** Specifies the value calculated from the linear combination.

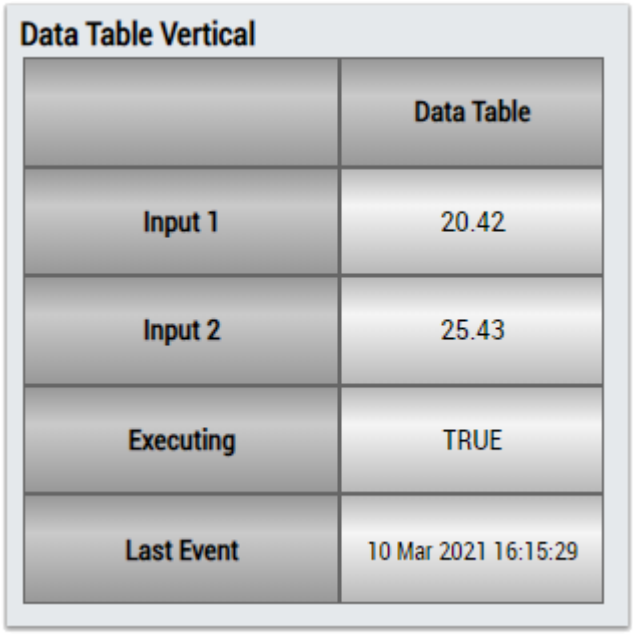
Standard HMI Controls

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.

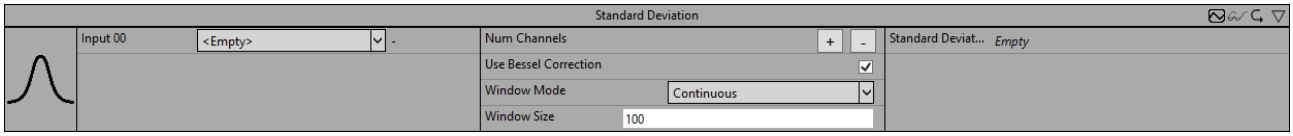


DataTable Horizontal

	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 Inference algorithm using the Mapping Wizard.

6.5.10.6 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

$$s' = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x[n] - \overline{x})^2}$$

The empirical standard deviation, with Bessel's correction

$$s = \sqrt{\frac{1}{N-1}\sum_{n=0}^{N-1}(x[n]-\overline{x})^2}$$

- **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:

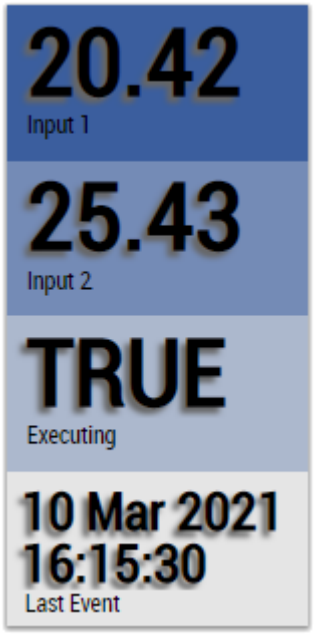
1. The Table Control or Multivalue Control visualizes all output values: Standard Deviation Results.

Data Table Vertical

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

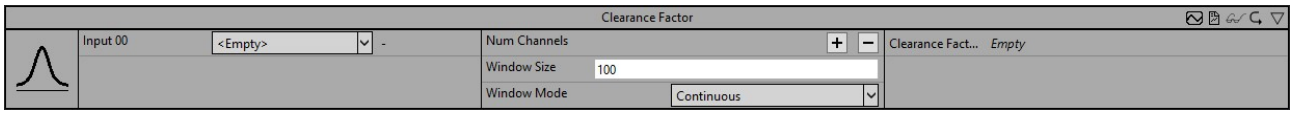
DataTable Horizontal

	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.

6.5.10.7 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.

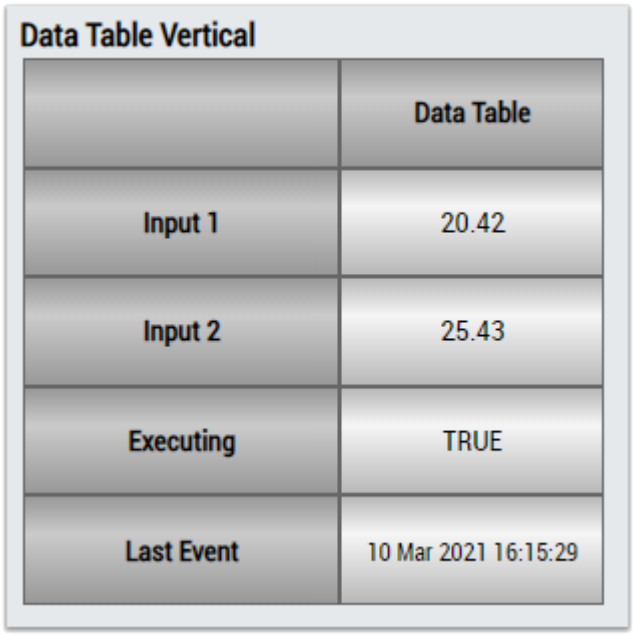
Output values

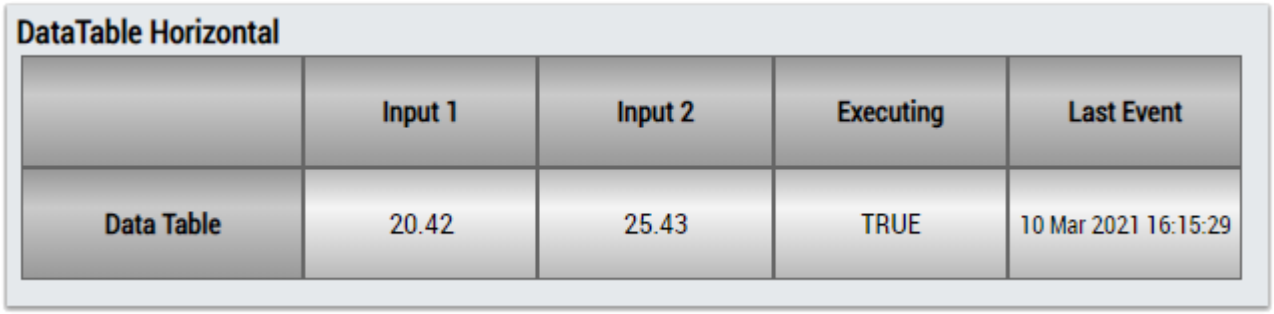
- **Clearance Factor 00..n:** Clearance factor of the input channels.

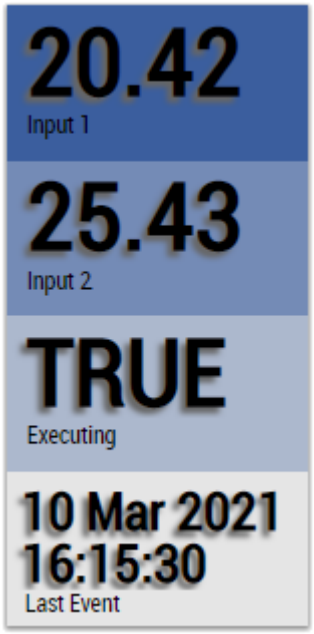
Standard HMI Controls

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.

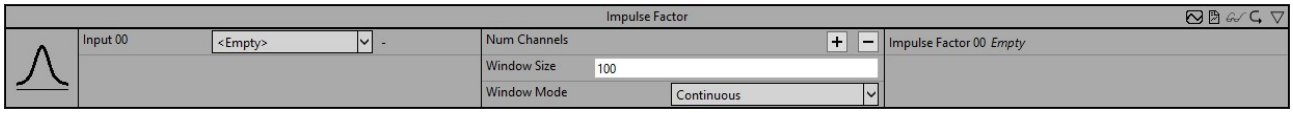






Alternatively, customer-specific HMI controls can be mapped using the Mapping Wizard.

6.5.10.8 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.

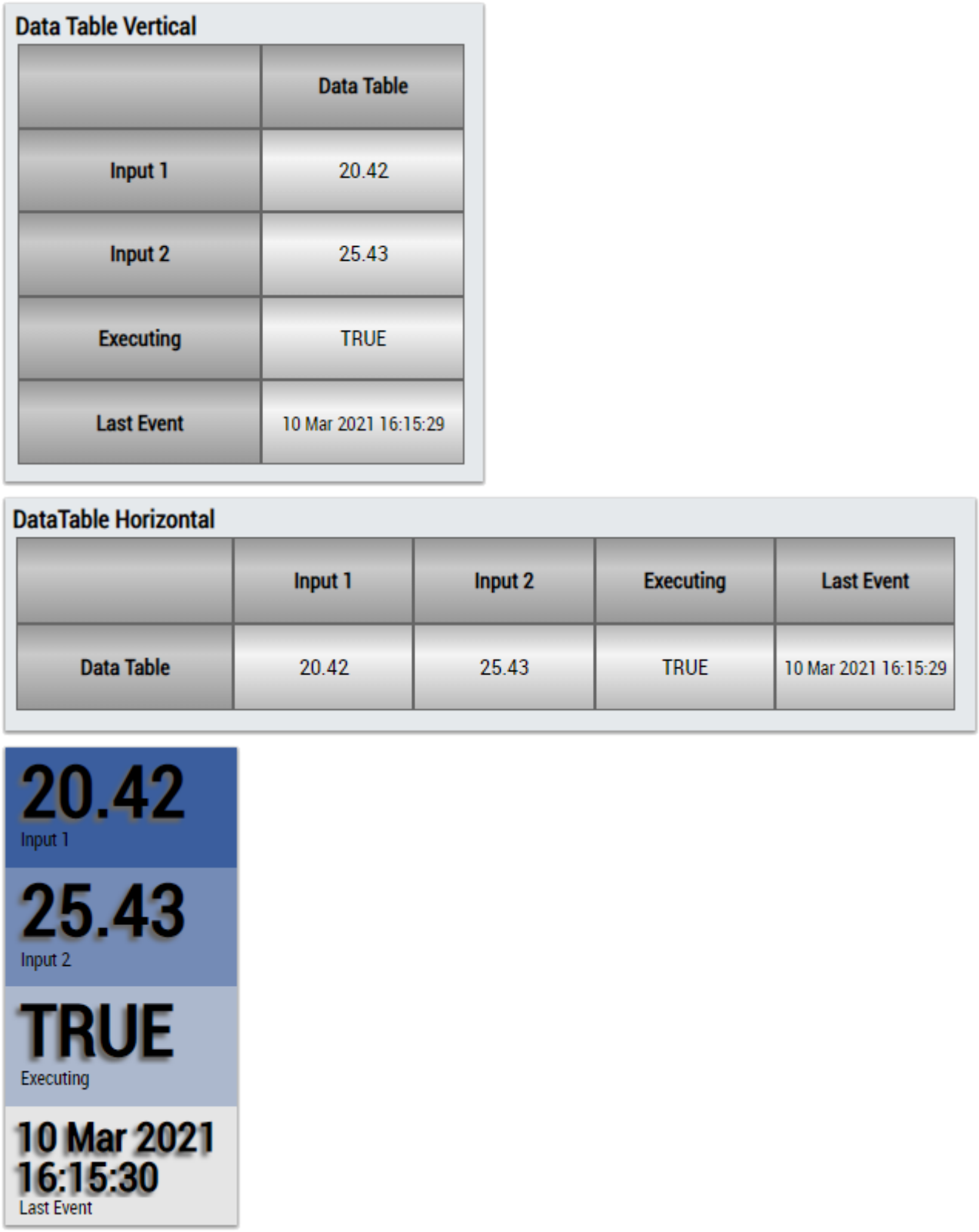
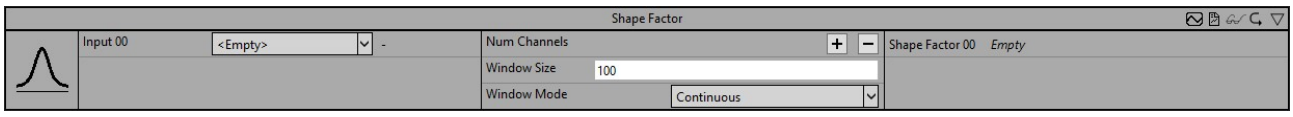


Fig. 3:

Alternatively, customer-specific HMI controls can be mapped using the Mapping Wizard.

6.5.10.9 Shape Factor



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.

$$\text{Shape Factor} = \frac{x_{rms}}{\frac{1}{N}\sum_{n=1}^{N}|x[n]|}$$

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.

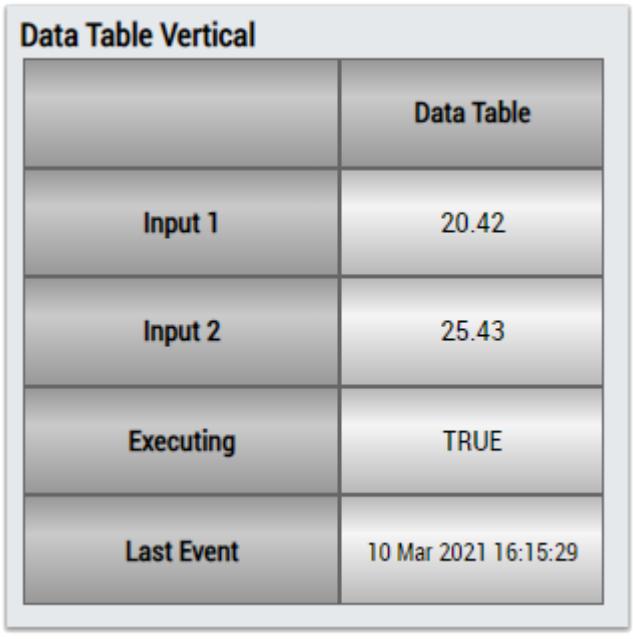
Output values

- **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.



DataTable Horizontal

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

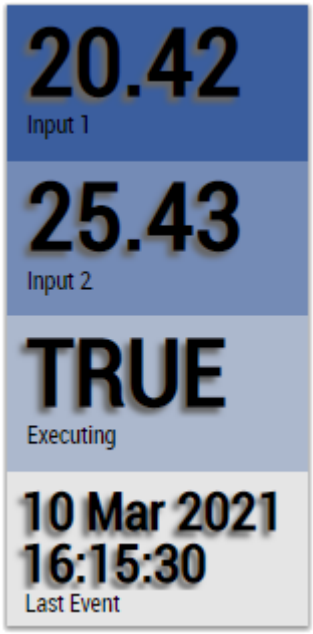


Fig. 4:

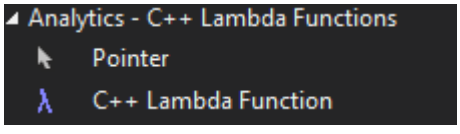
Alternatively, customer-specific HMI controls can be mapped using the Mapping Wizard.

6.5.11 Analytics - extension of the algorithms

In order to extend the algorithms of the TwinCAT Analytics Library, which are available in the TwinCAT Analytics Workbench configurator or in the TwinCAT Analytics Service Tool, two extension options are available. On the one hand, Analytics lambda functions can be used to develop user-specific algorithms and integrate them into the Analytics Workflow. On the other hand, algorithms from other libraries can extend the existing algorithms.

6.5.11.1 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.

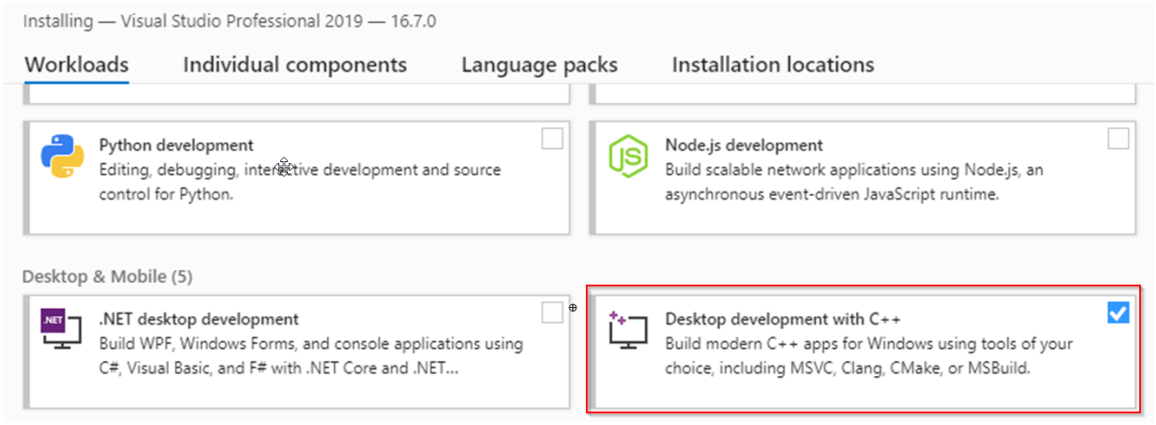
6.5.11.1.1 Requirements

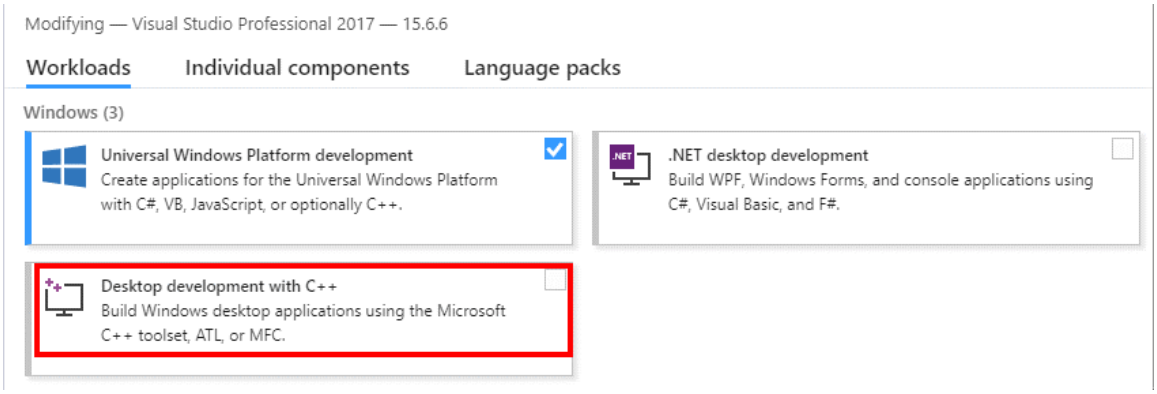
Overview of minimum requirements

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 the integration and use of 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 7 / 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)

6.5.11.1.2 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.
```

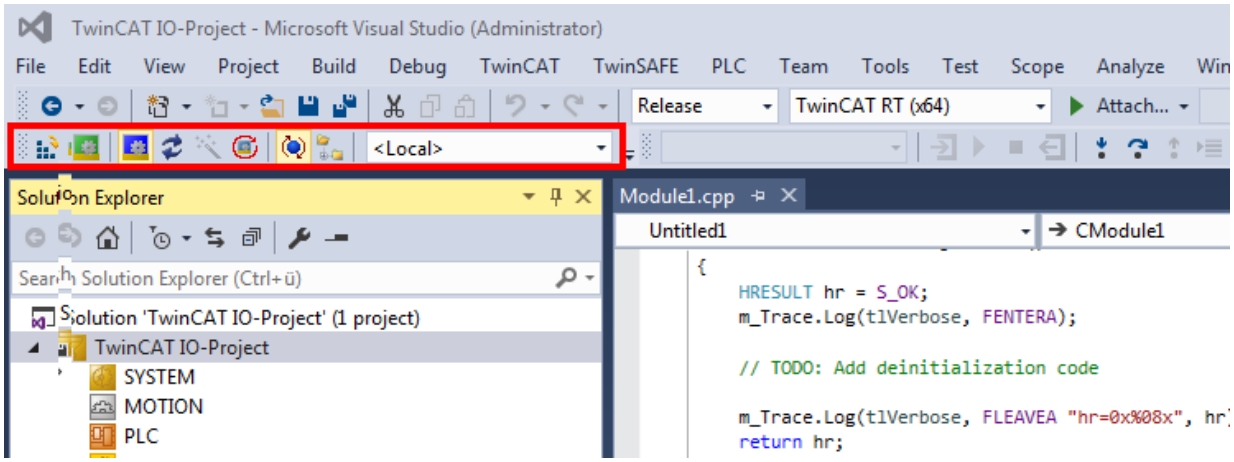
6.5.11.1.2.1 Visual Studio - TwinCAT XAE Base toolbar

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.



6.5.11.1.2.2 Driver signing

TwinCAT C++ modules must be signed with a certificate so that they can be executed.

The signature ensures that only C++ software whose origin can be traced is executed on productive systems.

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.

Engineering requires no signing

Only the execution requires certificates - the engineering does not.

There are two ways to load modules. For this purpose, different certificates are used for signing:

- TwinCAT: the C++ modules are loaded by the TwinCAT runtime system and must be signed with a TwinCAT user certificate.
  - With TwinCAT 3.1. 4024 and higher, this procedure is also available.
  - This procedure is required to perform new functions such as Versioned C++ Projects and thus also the Online Change.
  - Required for TwinCAT/BSD.
- (For <4024.0, no longer recommended for new projects. Existing projects should be migrated) Operating system: the C++ modules are loaded as normal kernel drivers and must therefore also have a signature.

- With TwinCAT 3.1. 4022 or earlier, only this procedure is available.
- Windows 7 (Embedded) x86 (32bit) does not require signing.
- Cannot be used with TwinCAT/BSD.

Since a published module should be executable on various PCs, signing is always necessary for publishing.

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.
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.

6.5.11.1.2.2.1 TwinCAT

Versioned C++ projects are stored as binary in a TMX file (TwinCAT Module Executeable).

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 [▶ 237], which is configured accordingly in the project for signing.

Test signing

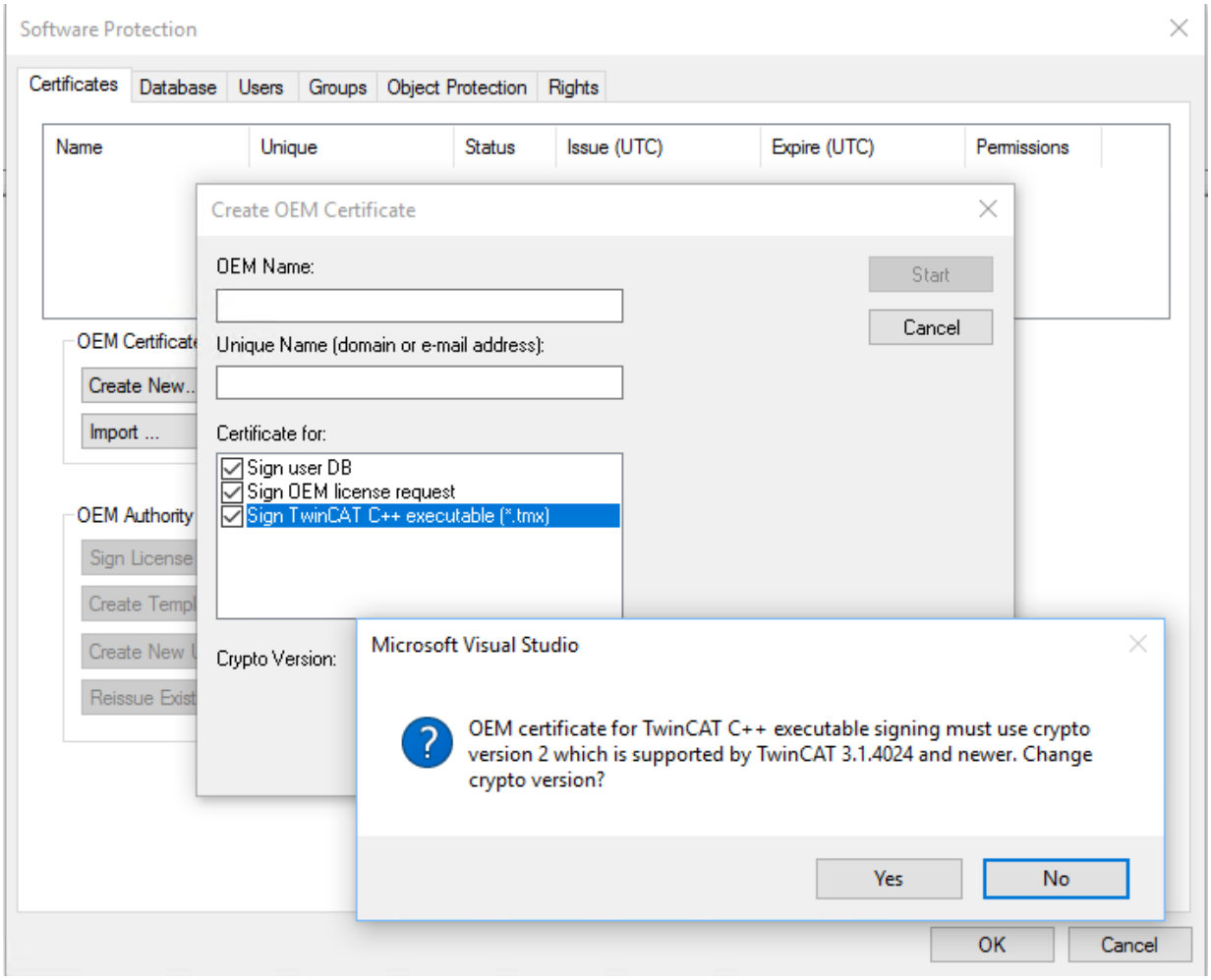
The user certificate can be created locally in TwinCAT. As long as it is not countersigned by Beckhoff, it is necessary to activate the test mode, as described here [▶ 237].

As soon as the TwinCAT user certificate has been countersigned by Beckhoff [▶ 239], the test mode can be dispensed with accordingly. It can be deactivated in the same way as it is activated.

6.5.11.1.2.2.1.1 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 [▶ 240]).

1. For test operation, e.g. during software development, the creation of a TwinCAT user certificate, as described Creation of the Certificate Request file for TC0008 [▶ 241], is sufficient. Make sure that you select the purpose "Sign TwinCAT C++ executable (*.tmx)". For this the Crypto version 2 is required, a message appears.

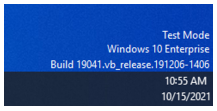


On XAR (and XAE, if it is a local test), activate the test mode so that the operating system can accept the self-signed certificates. This can be done on both engineering systems (XAE) and runtime systems (XAR).

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

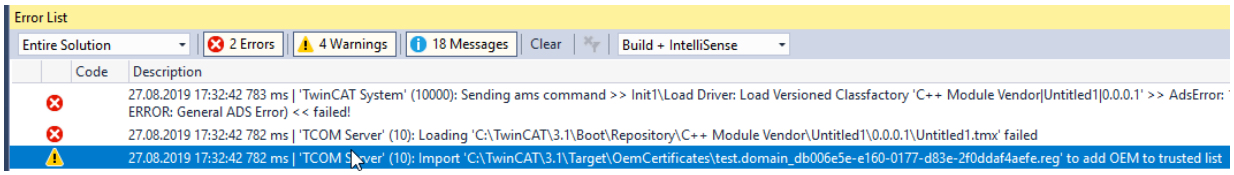
In the file `/usr/local/etc/TwinCAT/3.1/TcRegistry.xml` enter „`<Value Name="EnableTestSigning" Type="DW">1</Value>`" under Key "`System`".

```
<Key Name="System">
  <Value Name="RunAsDevice" Type="DW">1</Value>
  <Value Name="RTimeMode" Type="DW">0</Value>
  <Value Name="AmsNetId" Type="BIN">052445B00101</Value>
  <Value Name="LockedMemSize" Type="DW">33554432</Value>
  <Value Name="EnableTestSigning" Type="DW">1</Value>
</Key>
```

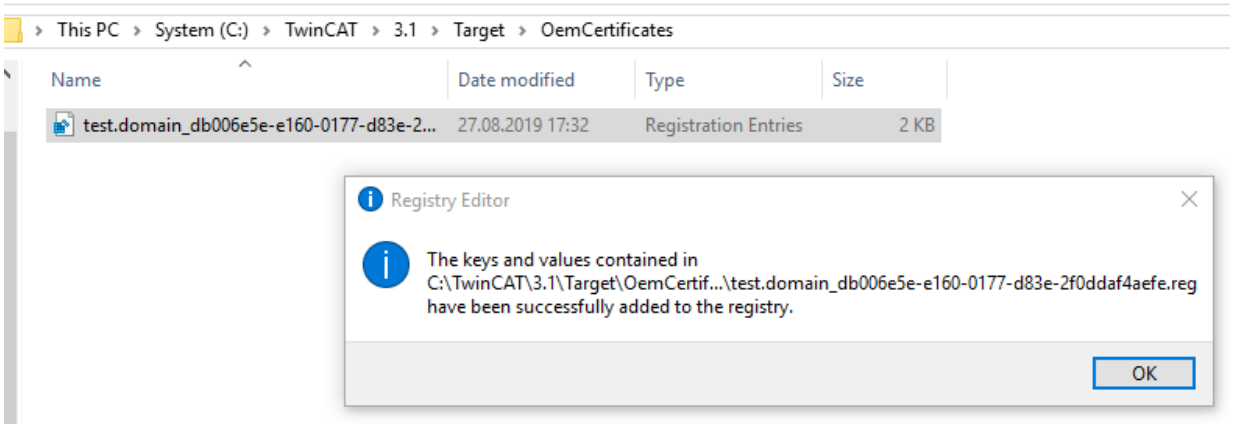
Then restart the TwinCAT System Service:
`doas service TcSystemService restart`

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 [▶ 250] 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 [▶ 239].

6.5.11.1.2.2.1.2 Signed TwinCAT 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**

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.

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.

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.

6.5.11.1.2.2.1.2.1 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

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 [▶ 241].
5. Determine the "File Fingerprint" of the OEM certificate file using TwinCAT Engineering (see Determining the file fingerprint of the OEM certificate file [▶ 244]). 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.

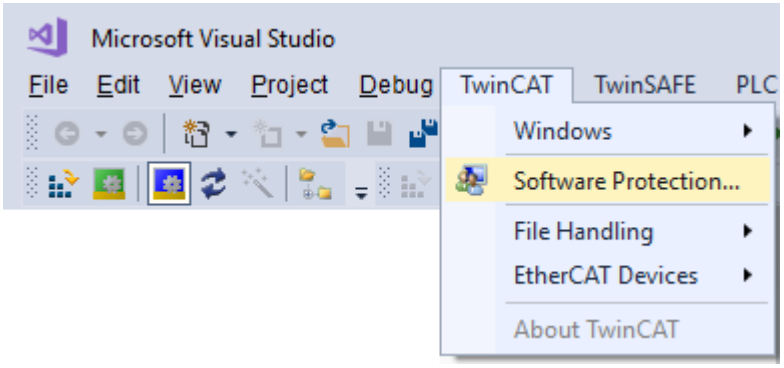
Please note that it may take a few days to validate your contact details and issue the certificate.

6.5.11.1.2.2.1.2.2 Creation of the Certificate Request file for TC0008

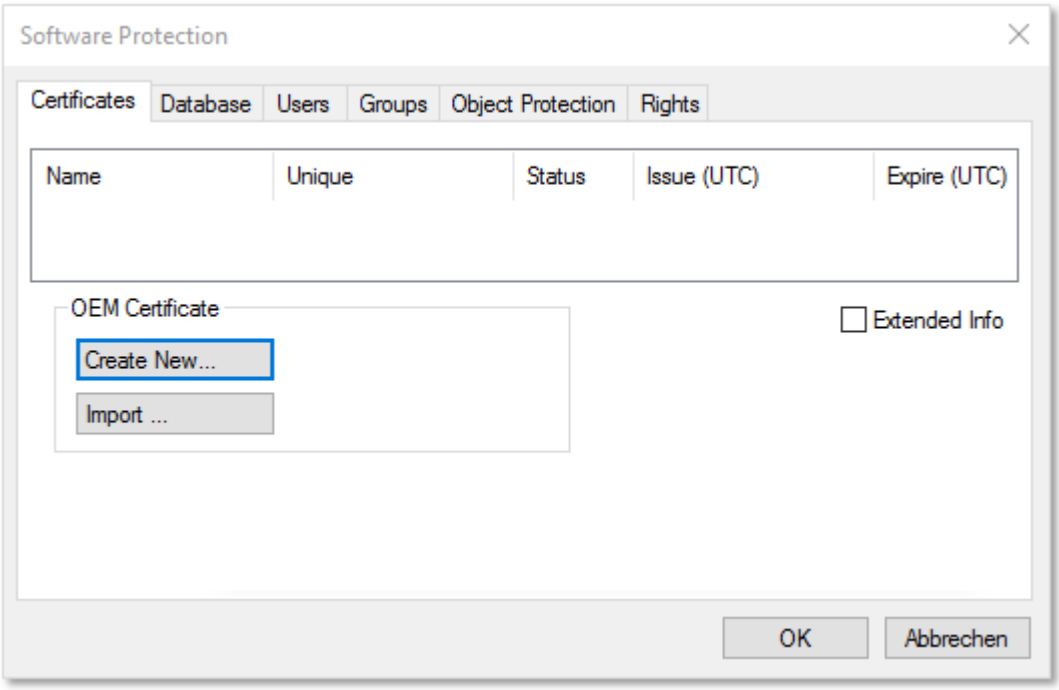
System requirements

- Min. TwinCAT 3.1 Build 4024
- Min. Windows 10 or TwinCAT/BSD (on the target system)

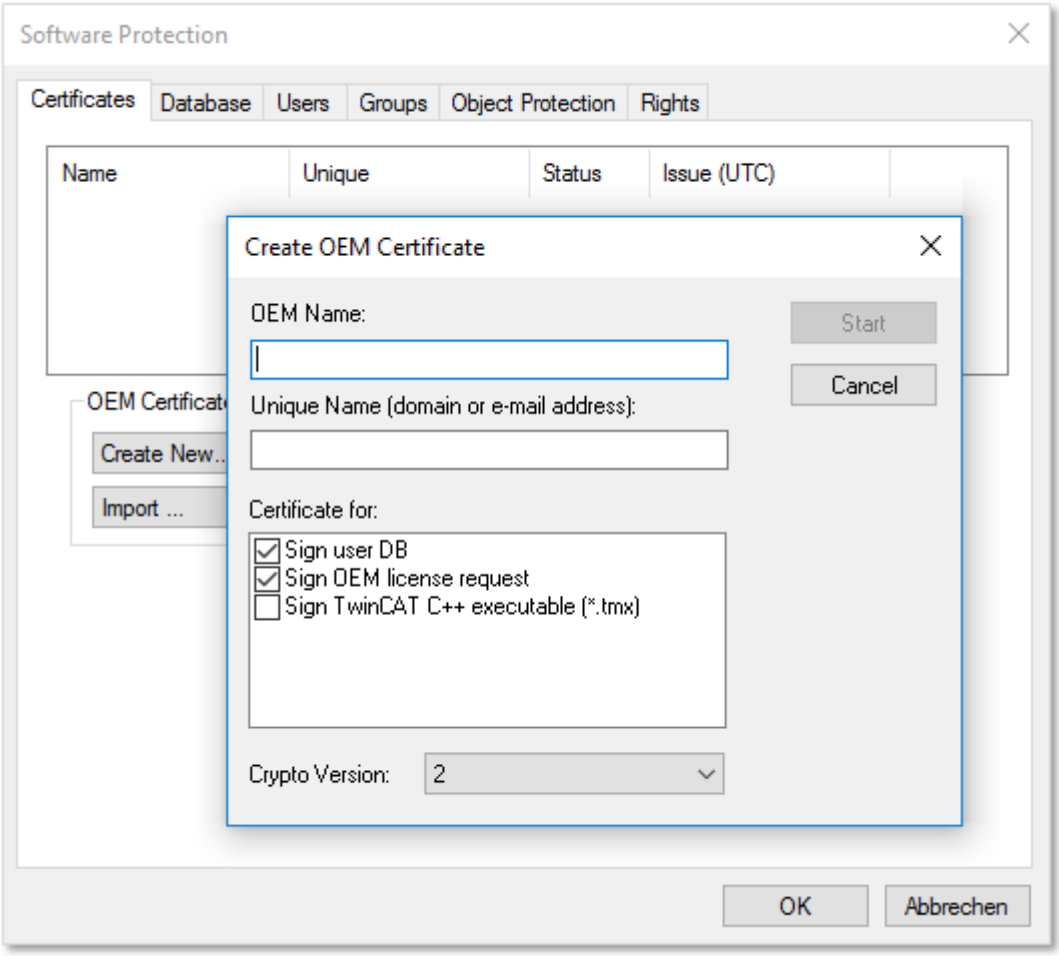
1. Call up the Software Protection configurator. To do this, select the menu item **Software Protection** in the main menu below the item **TwinCAT**:



2. In the window that opens, select the **Certificates** tab.
   Click **Create New....**:

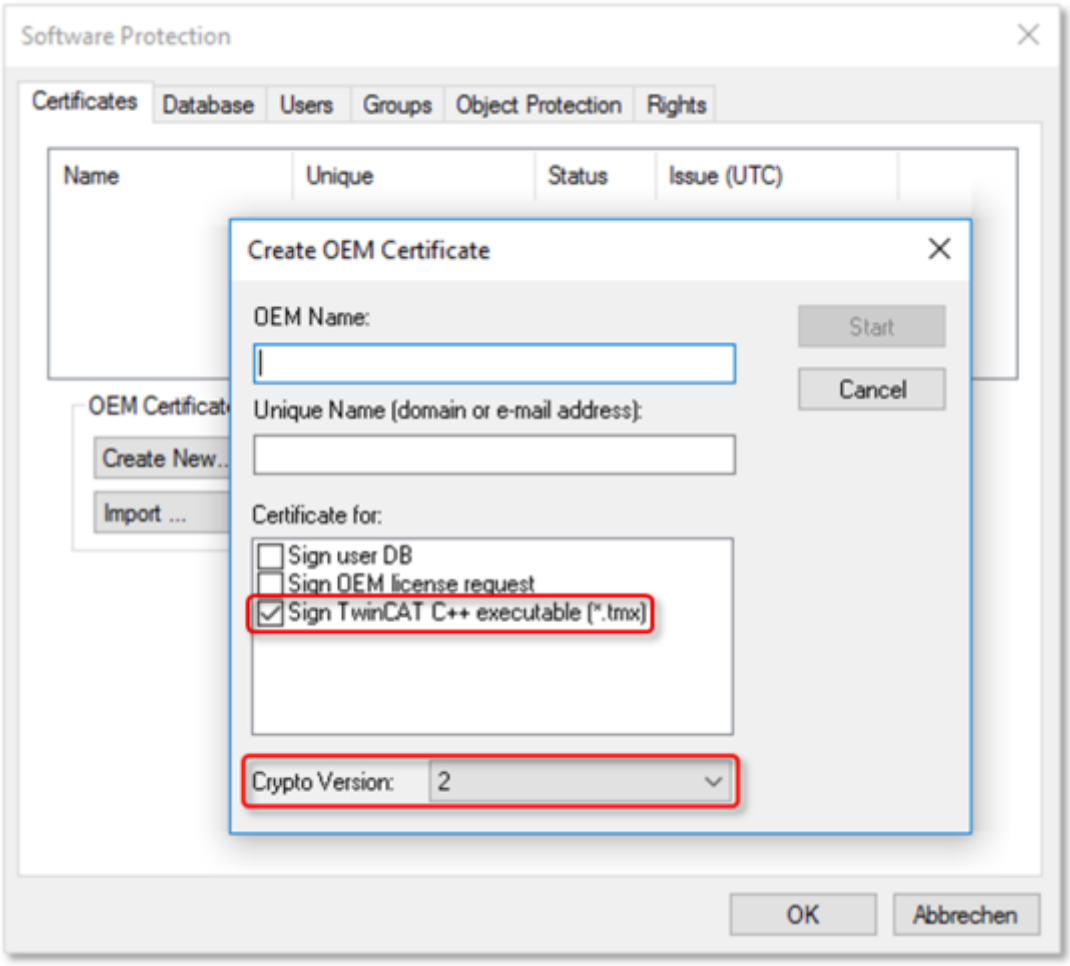


3. The **Create OEM Certificate** input window opens:



4. 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. (standard setting)

5. 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 [▶ 244] **in a subsequent step.**
   ⇨ A dialog for selecting a password for the OEM Private Key opens.
6. Issue a password for the OEM Private Key**.**

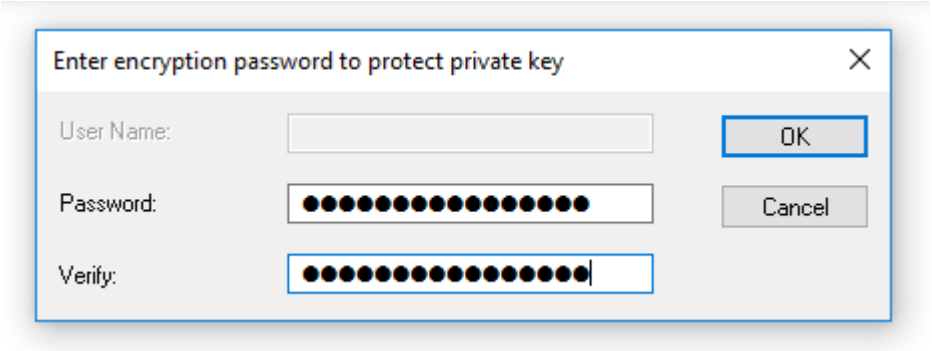
Important: Password security!

Be sure to use a strong password for your certificate!
Protect your password with suitable measures so that it cannot fall into unauthorized hands!

Password cannot be restored if lost

Beckhoff is unable to 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 [▶ 240].

6.5.11.1.2.2.1.2.3 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

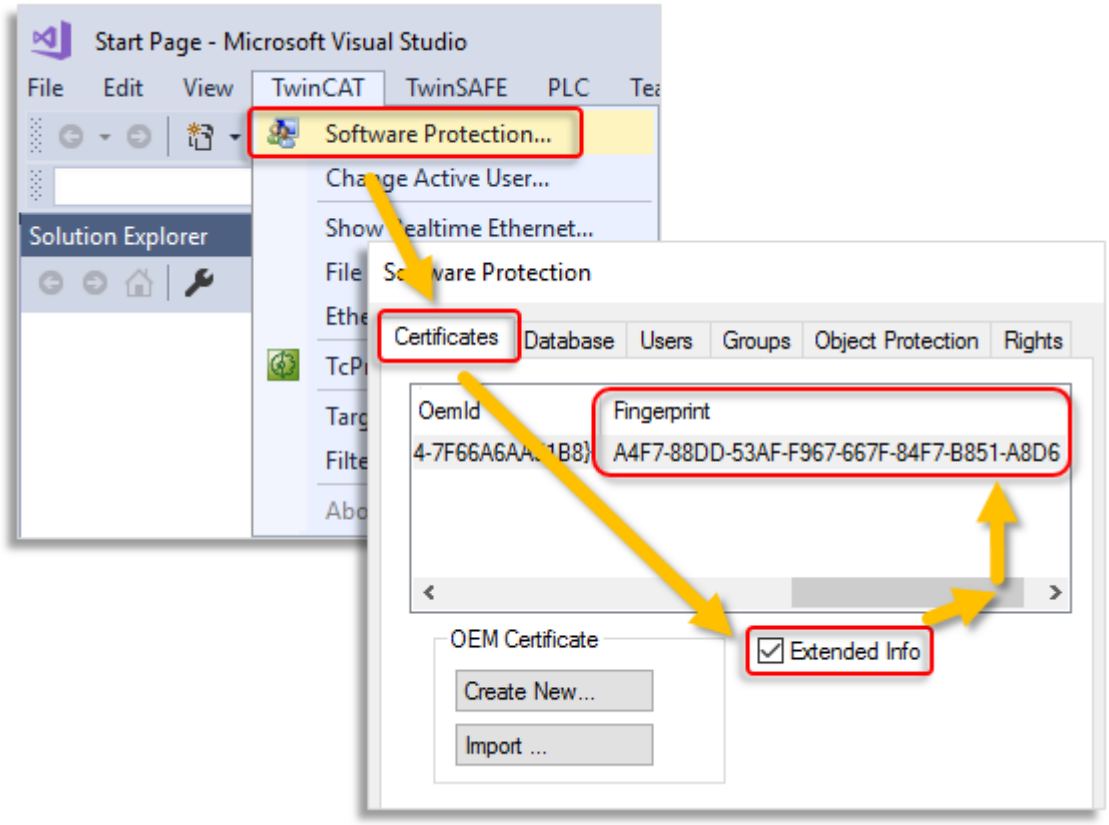
For this function it is necessary that the OEM certificate file is located in this directory: "c:\twincat\3.1\customconfig\certificates".

This directory contains your OEM certificate, if you already have a certificate and want to renew it.

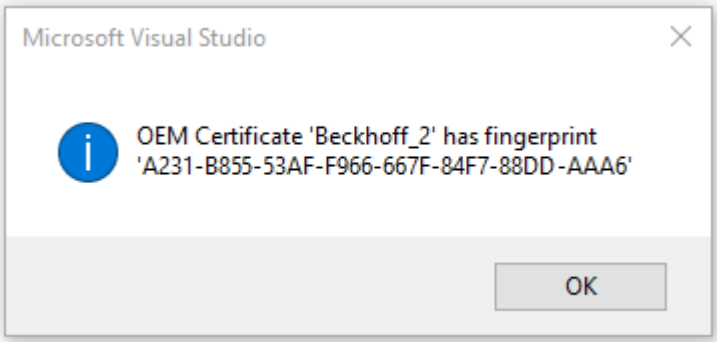
If you did not change the suggested directory when creating the "OEM Certificate Request File", the file is already in this directory.

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.

6.5.11.1.2.2.1.2.4 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)

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.

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.

6.5.11.1.2.2.2 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 [▶ 246]".

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 [▶ 246]
- Delete (test) certificates [▶ 248]

Further references:

MSDN, MakeCert test certificates (Windows drivers),
https://docs.microsoft.com/en-us/windows-hardware/drivers/install/makecert-test-certificate

6.5.11.1.2.2.2.1 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.

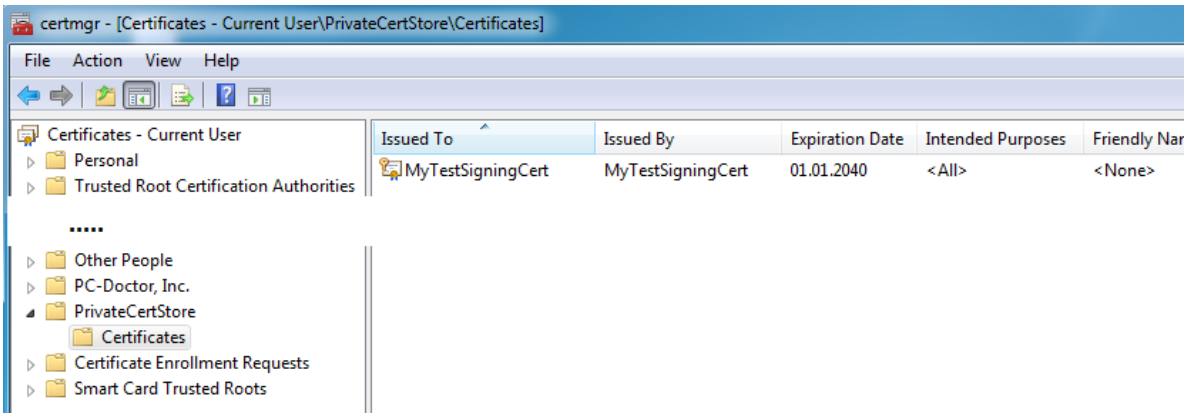
Note the procedure when creating test certificates

Developers may have a wide range of tools for creating certificates. Please follow this description exactly, in order to activate the test certificate mechanism.

The following commands must be executed from a command line that has been opened in either way:

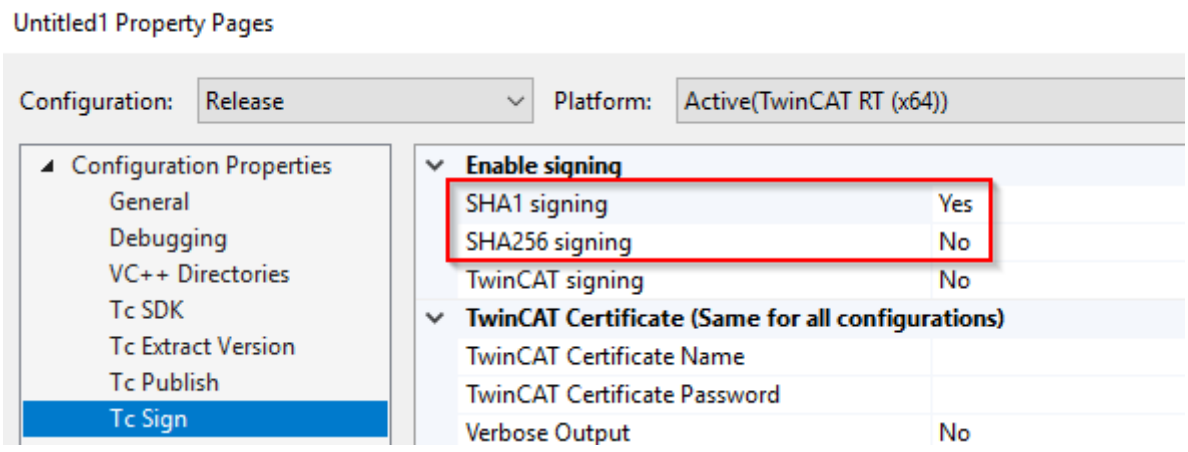
- **Visual Studio 2010 / 2012 prompt with administrator rights**. (Via: **All Programs** -> **Microsoft Visual Studio 2010/2012** -> **Visual Studio Tools** -> Visual Studio Command Prompt, then right-click **Run as administrator**)
- **Developer Command Prompt of Visual Studio 2017 / 2019 with administrator 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 Visual Studio 2010 / 2012 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 location. This must also be used in the PostBuild event, as described here [▶ 250]**.)**
   ⇨ 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).

Notice 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:

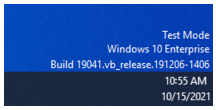


On XAR (and XAE, if it is a local test), activate the test mode so that the operating system can accept the self-signed certificates. This can be done on both engineering systems (XAE) and runtime systems (XAR).

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.

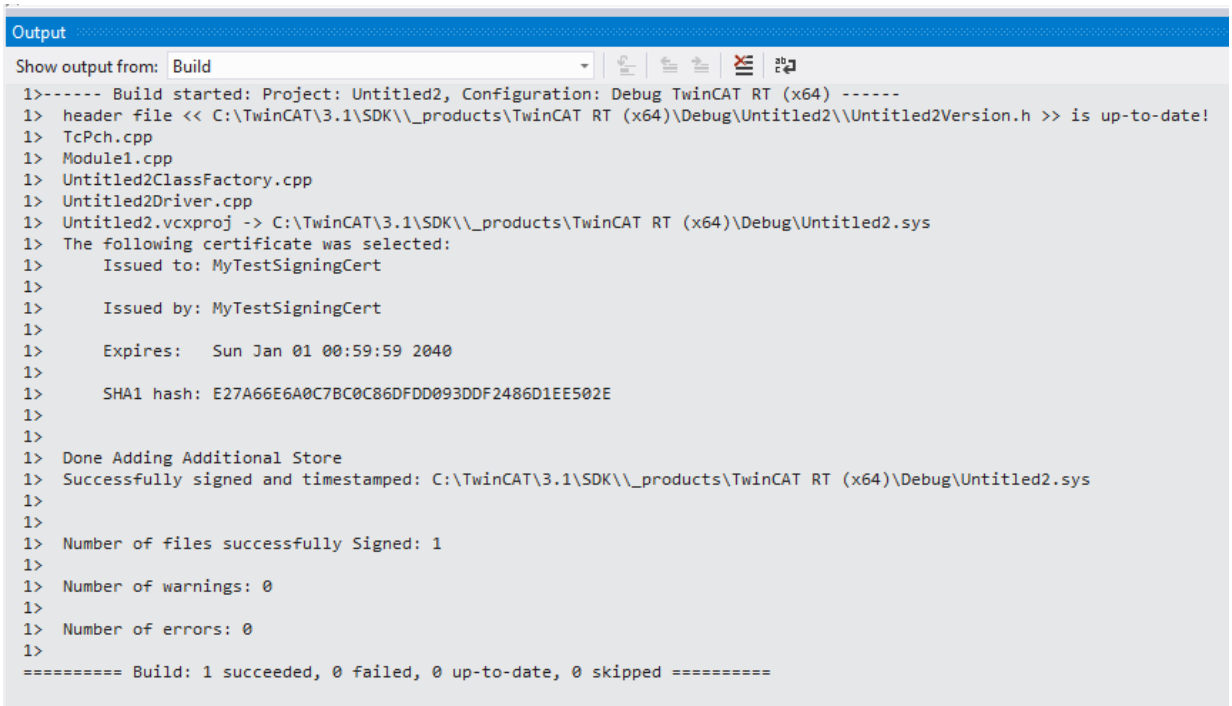
If test signing mode is enabled, this is displayed at the bottom right of the desktop. The system now accepts all signed drivers for execution.



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:



References:
MSDN, MakeCert test certificates (Windows drivers),
https://docs.microsoft.com/en-us/windows-hardware/drivers/install/makecert-test-certificate

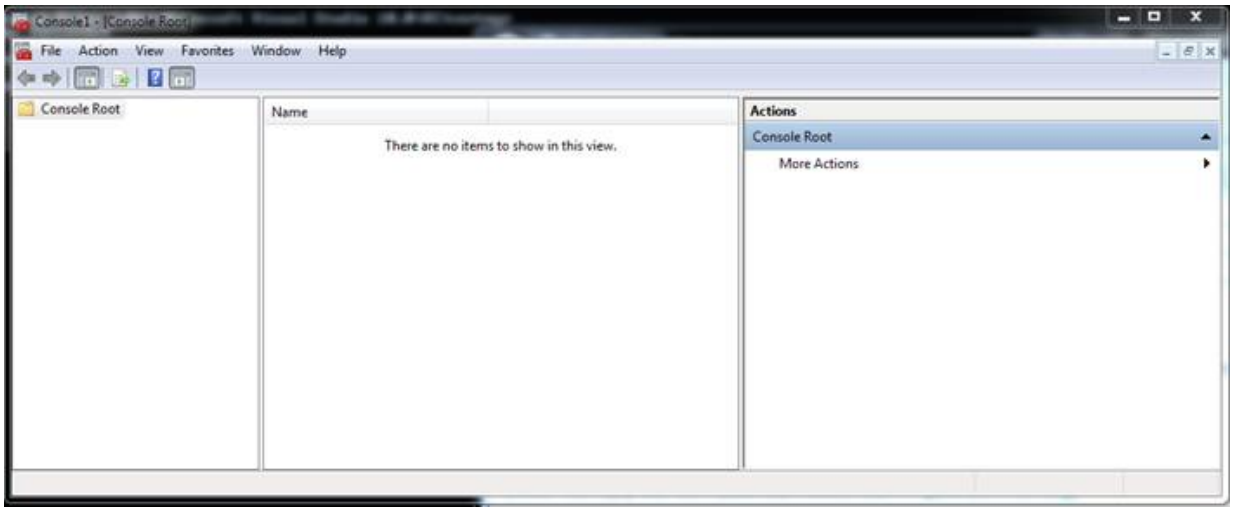
6.5.11.1.2.2.2.2 Delete test certificate

This article is about how to delete a test certificate.

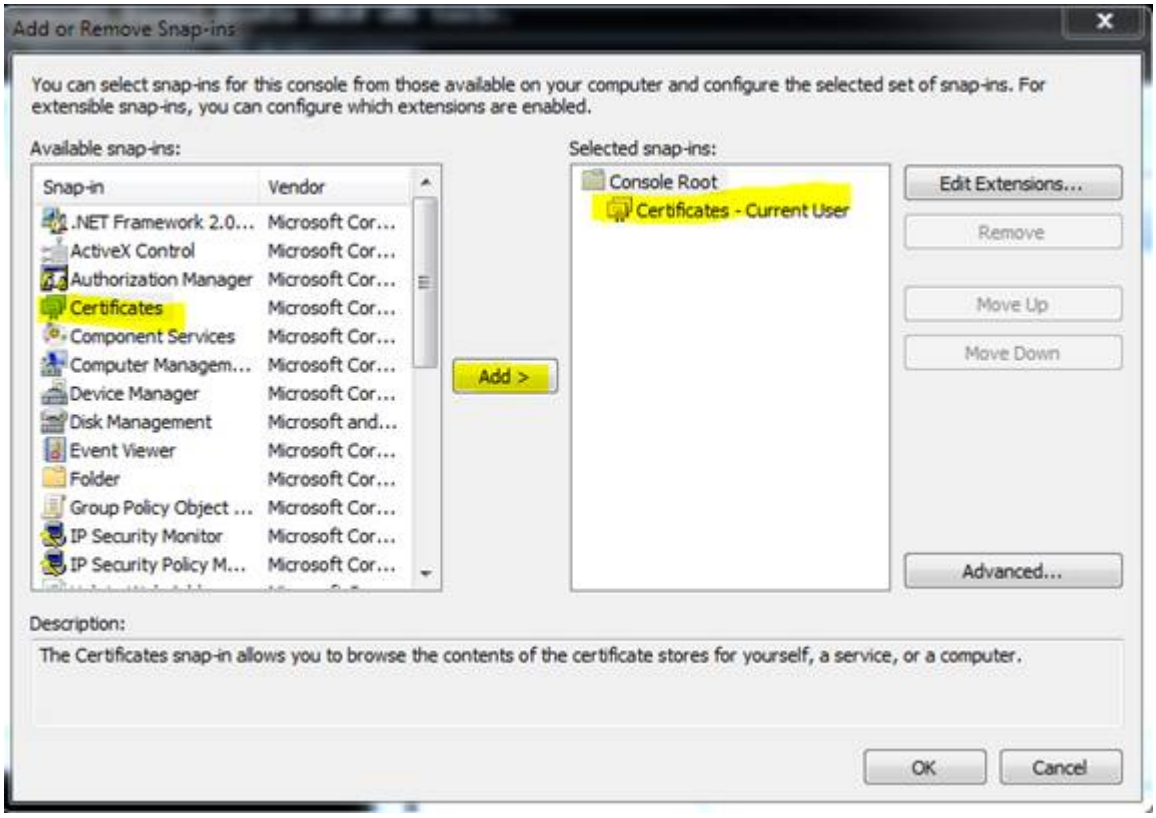
Overview

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.

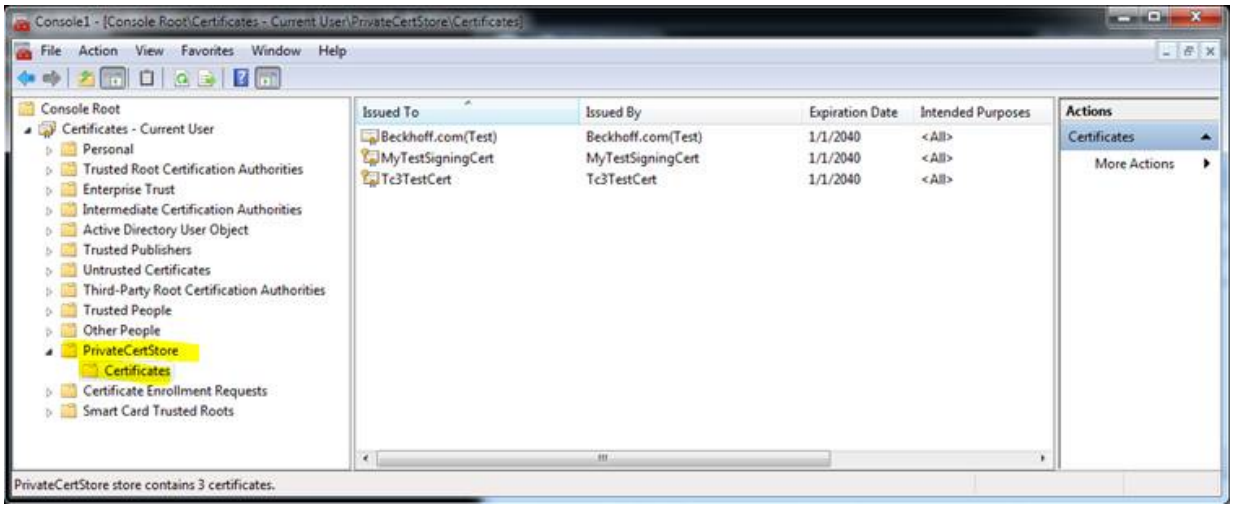


2. Click in the menu on **File** -> **Add/Remove Snap-in..** and select the certificate snap-in for the current user; conclude with **OK**.



⇨ The certificates are listed in the node under **PrivateCertStore/Certificates**.

3. Select the certificate to be deleted.



6.5.11.1.2.2.3 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. After this date it can no longer be used (depending on the expiry date of the individual CrossSigning certificate), as documented here.
Versioned C++ projects, which are loaded via the TwinCAT Loader, have been available for TwinCAT C++ for some time; they are not drivers for the purposes 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.

6.5.11.1.2.3 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.x\sdk\Bin*.

The storage of the password is carried out with the following parameters:

```
tcsigntool grant /f "C:\TwinCAT\3.1\CustomConfig\Certificates\MyCertificate.tccert" /p MyPassword
```

The password is deleted with the following parameters:

```
tcsigntool grant /f "C:\TwinCAT\3.1\CustomConfig\Certificates\MyCertificate.tccert" /r
```

The unencrypted password is stored under `HKEY_CURRENT_USER\SOFTWARE\Beckhoff\TcSignTool\`

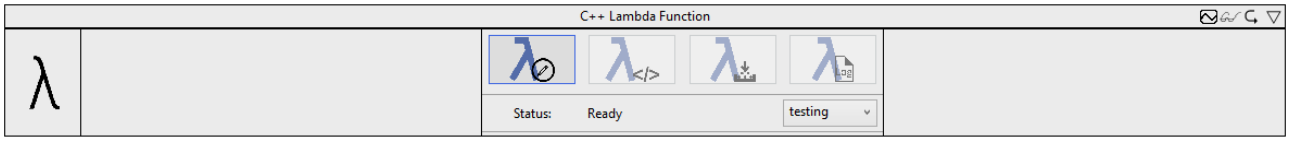
6.5.11.1.2.4 Environment variables

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.

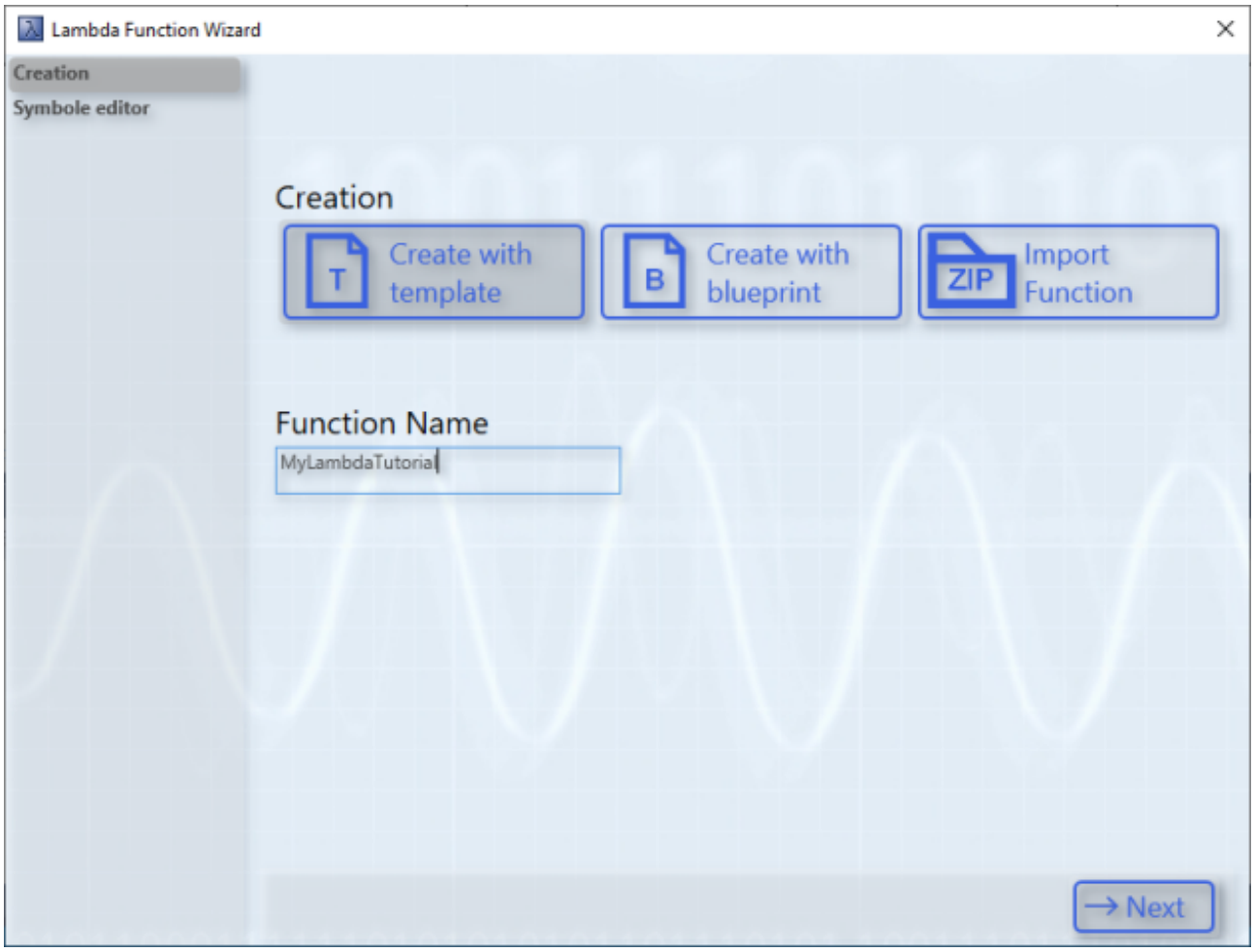
6.5.11.1.3 Lambda Template



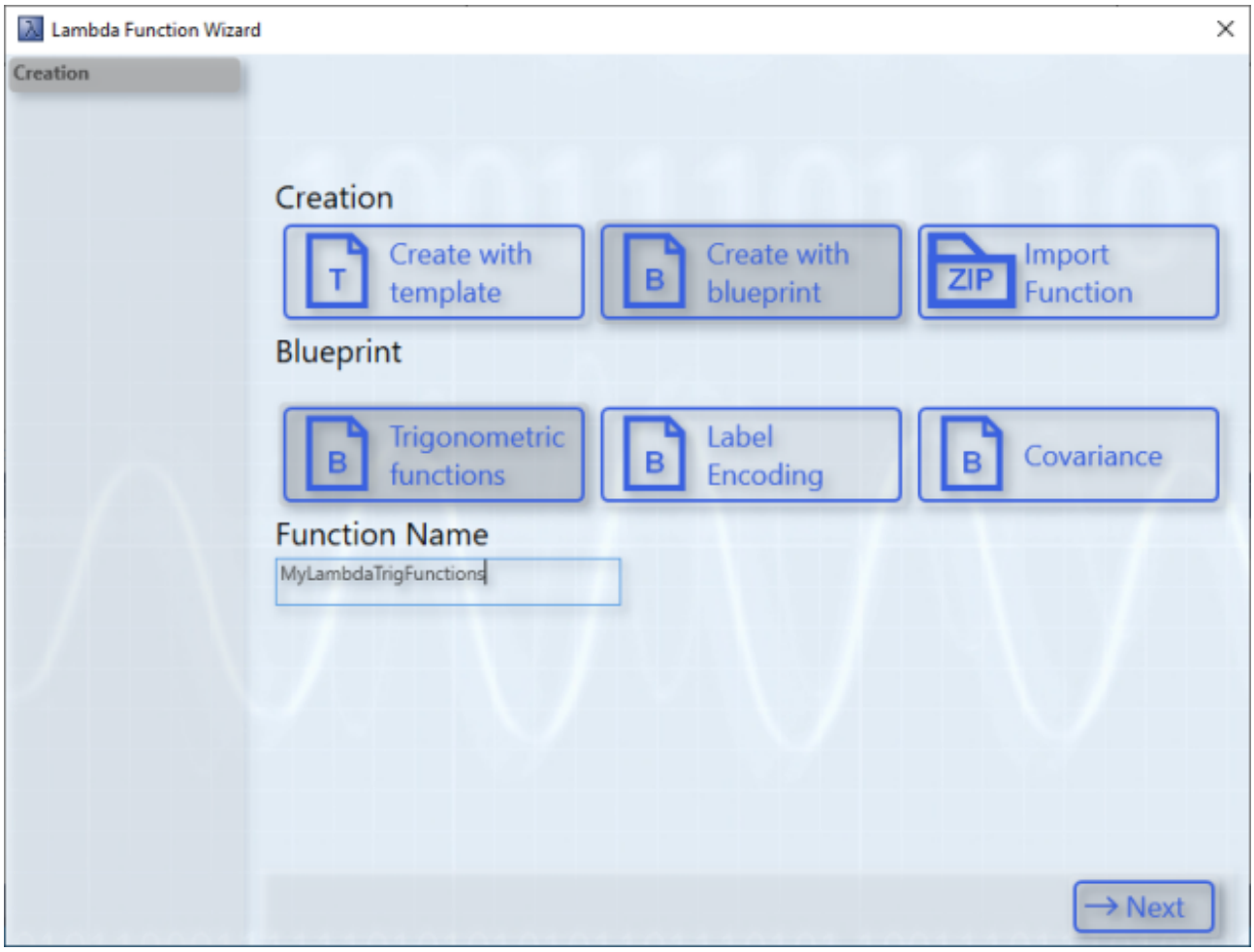
The lambda template guides through the development of a lambda function. Within the parameter area there is a control. This control contains four buttons for configuring, programming and publishing the lambda function. The fourth button is for viewing the log files of the development process. A status 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

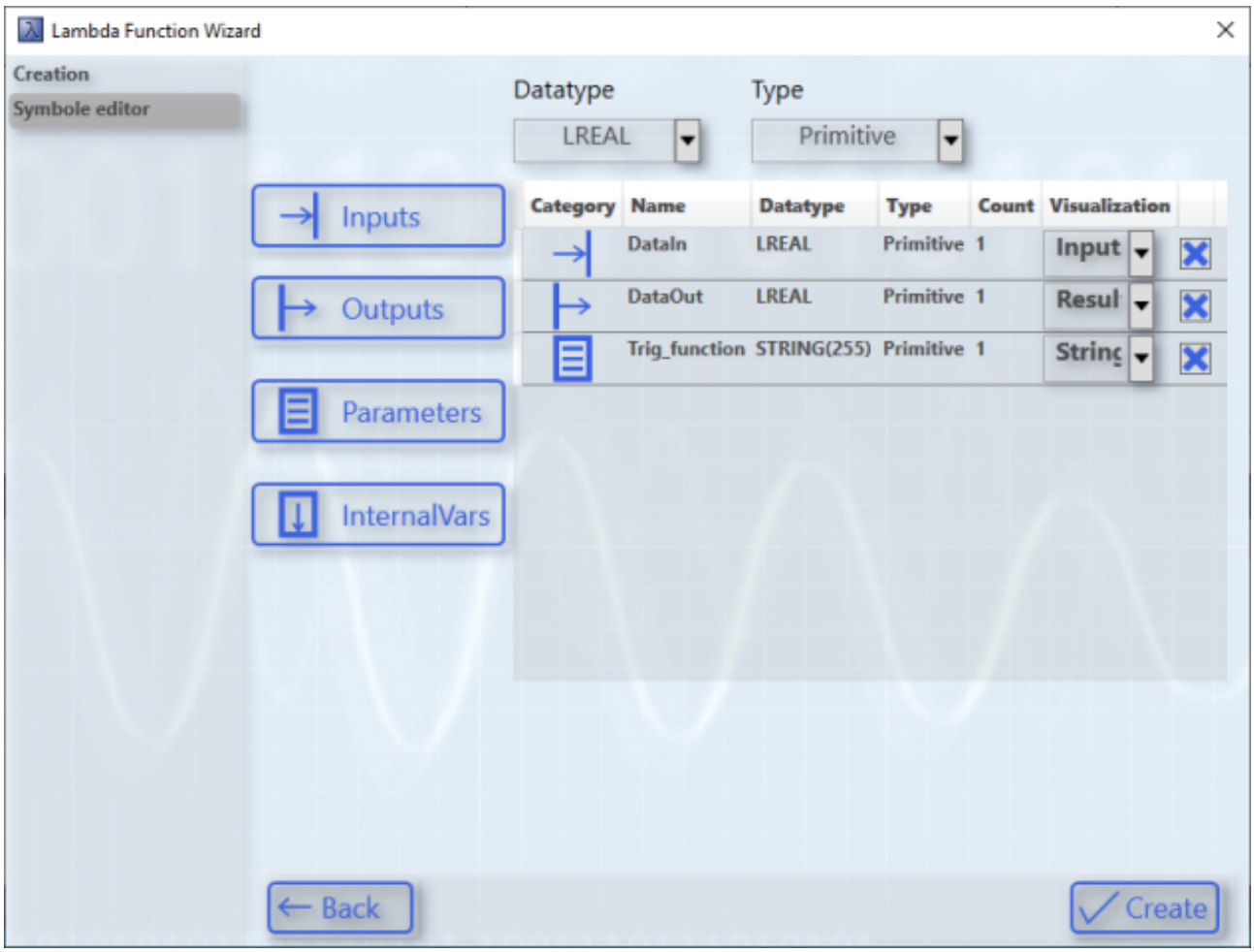
The first button from the left opens a wizard. With the help of this wizard you can configure the lambda function.



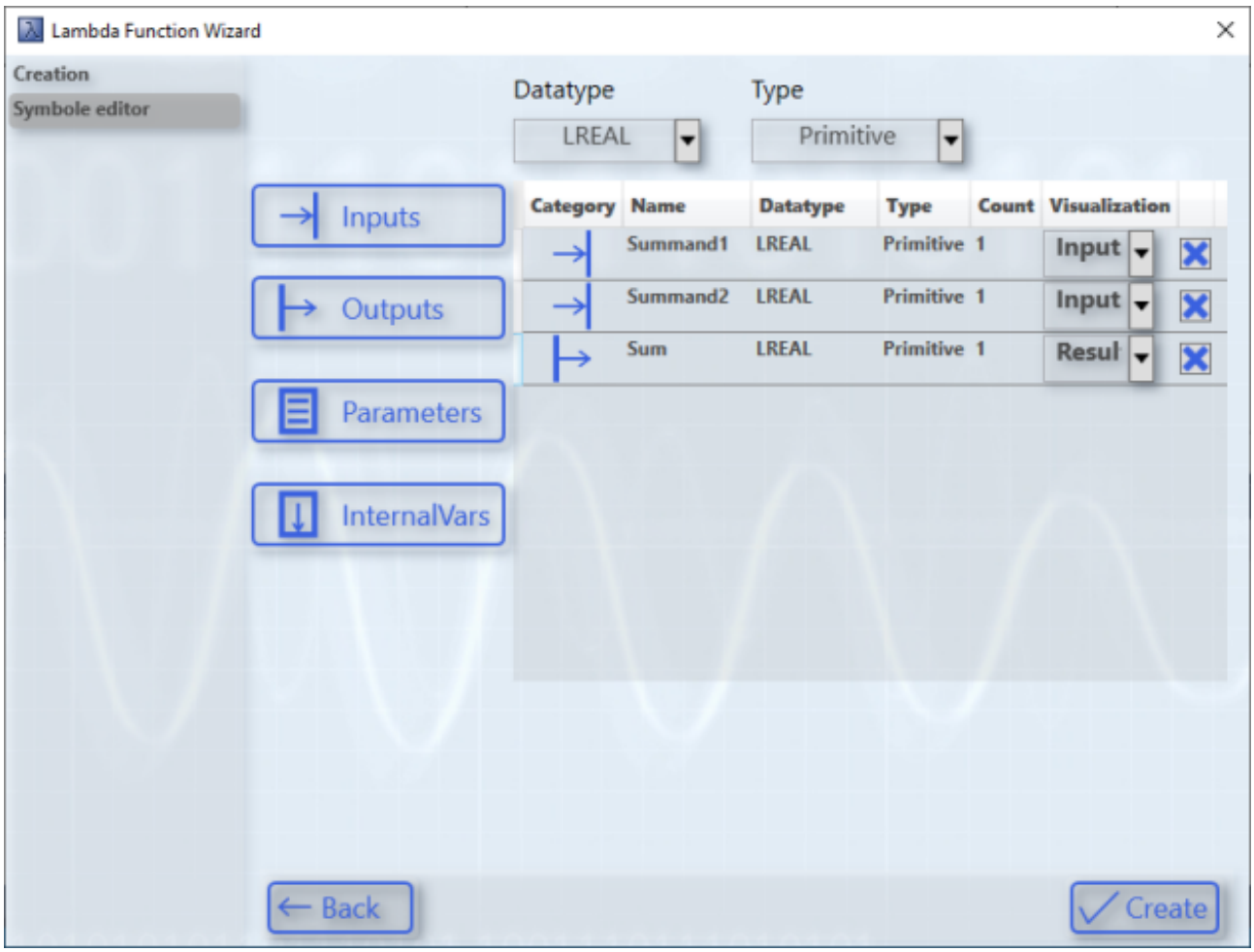
On the first page you can choose between three modes. The first mode is called Template. With this mode new lambda functions can be developed. All files and dependencies are prepared in a template and can be implemented with user-specific logic.



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 [▶ 259]).
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.



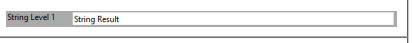
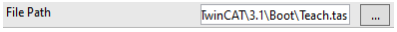
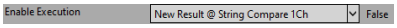
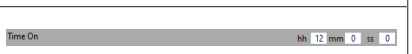
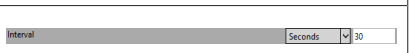
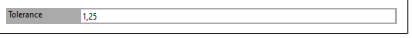
In the template mode the list of symbols is empty. As shown in the following figure, you can configure user-specific symbols.



Use the two drop-down boxes at the very top of the page to select the data type and the data structure type, respectively.

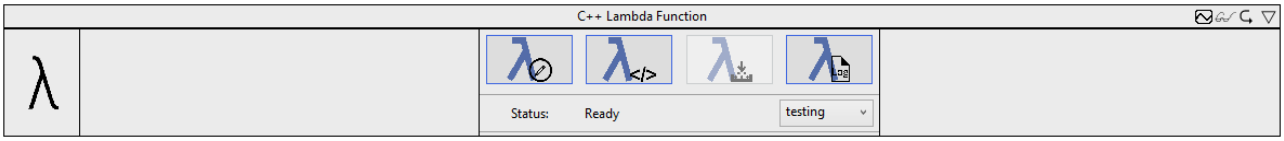
Available data types				
LREAL	REAL	BOOL	WORD	INT
LINT	ULINT	LTIME	STRING(255)	

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:

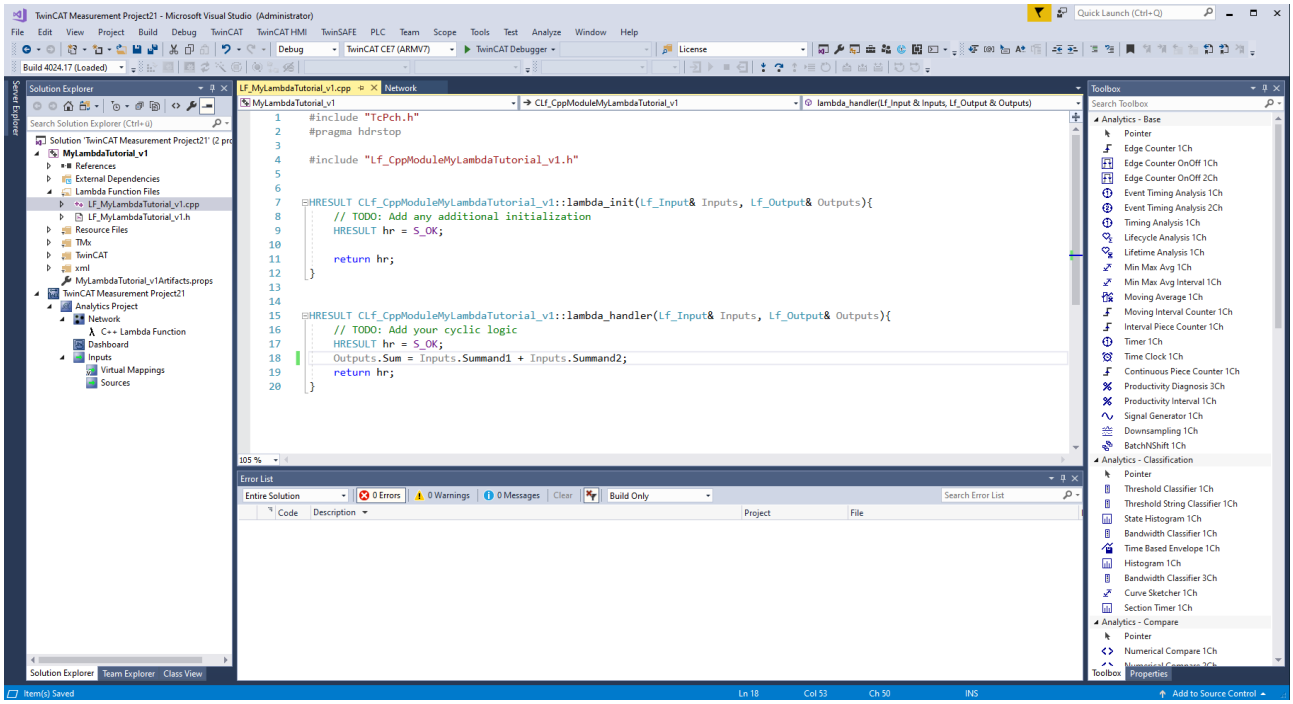
Data type	Visualization	Sample
STRING(255)	String (Input)	
	String (Output)	
	String (Parameter)	
	FileOpen/FileSave (Parameter)	
ULINT	Timestamp (Output)	
BOOL	Boolean (Input)	
	Boolean (Output)	
	Boolean (Parameter)	
REAL	Input (Input)	
	Result (Output)	
INT	Count (Output)	
LINT	Timespan (Output)	
	Clock (Parameter)	
	Interval (Parameter)	
WORD	DayOfWeekMask (Parameter)	
LREAL	Input (Input)	
	Result (Output)	
	Level (Parameter)	

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.



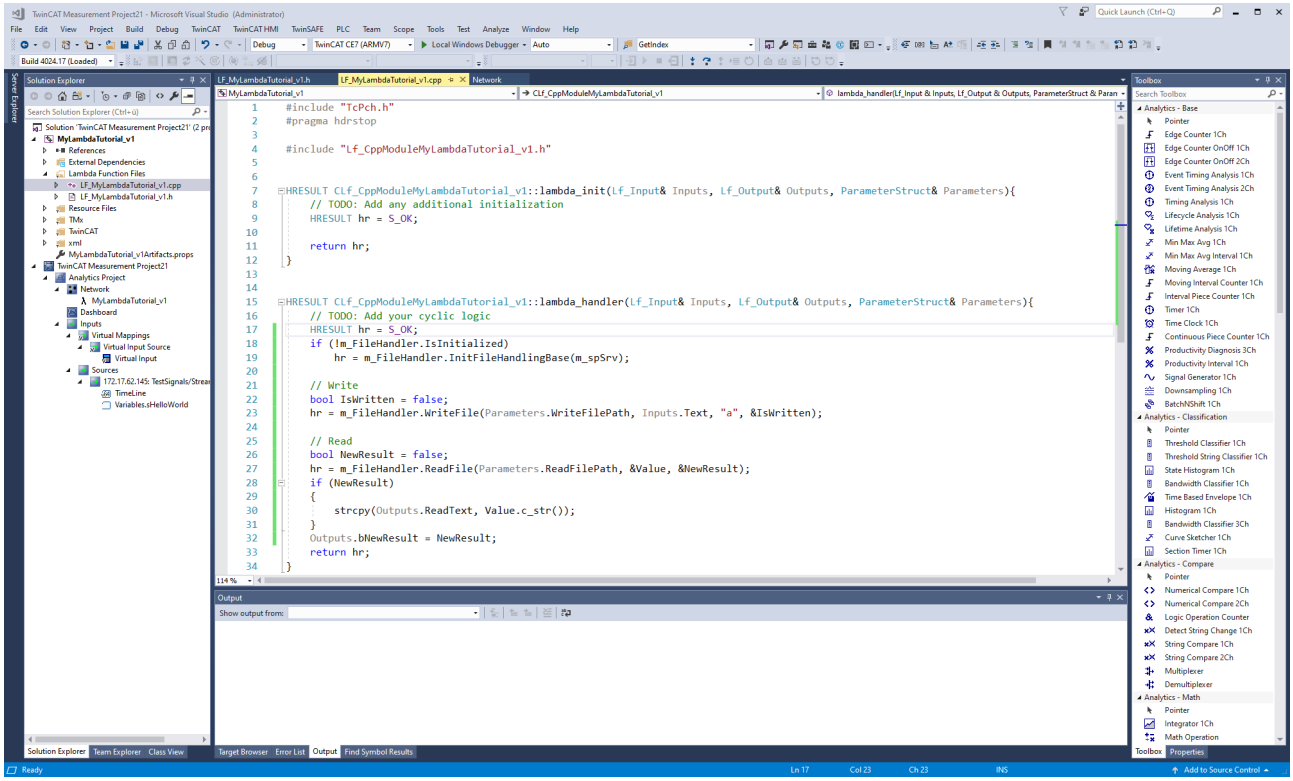
The automatically opened file contains several encapsulated functions. The so-called lambda_init method is for the initialization of variables. The lambda_handler method is executed cyclically. The lambda_uninit method is for deinitializing variables. In both methods the inputs, outputs, parameters and 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 lambda_handler 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:

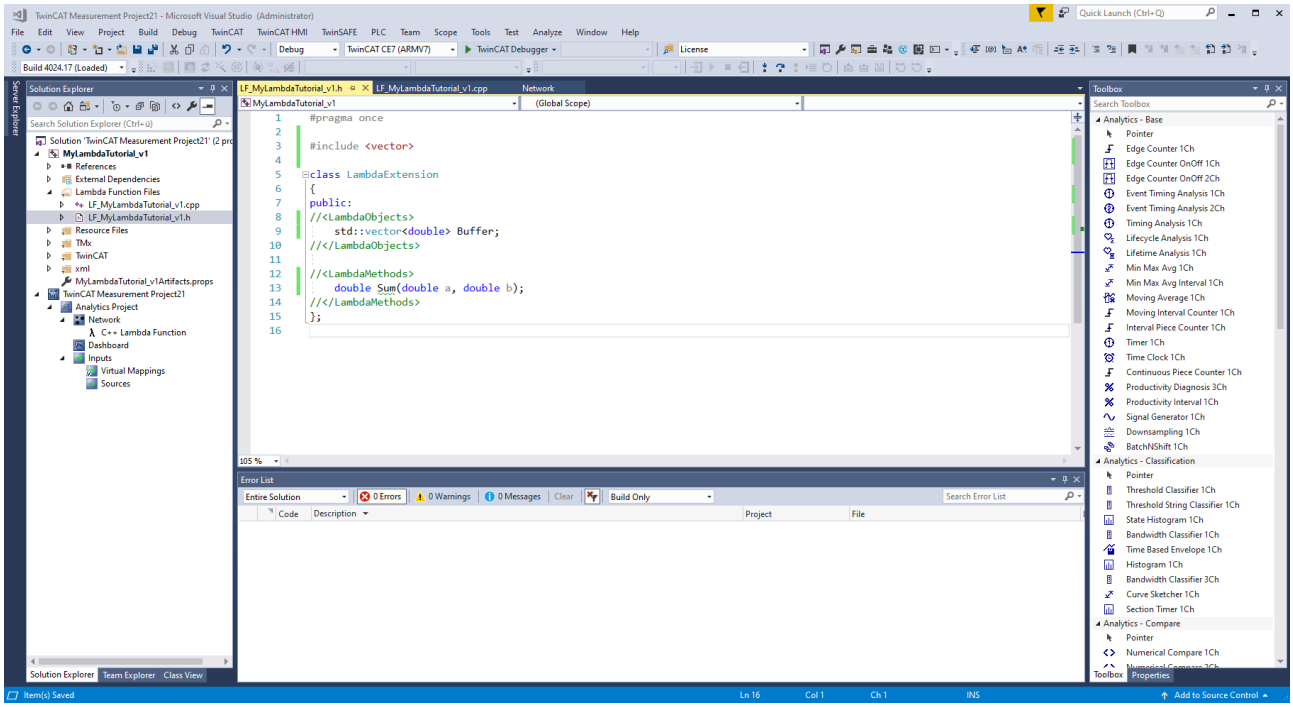
Function name	Parameter				Return value
InitFileHandlingBase	ITComObjectServer* ipSrv				Hresult
ReadFile	PCCH filename	Std::string* value	bool* bNewResult		Hresult
WriteFile	PCCH filename	Std::string value	Std::string mode	bool* bIsWritten	Hresult

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 bIsWritten parameter returns true. You can use the two parameters bNewResult and bIsWritten analogously for a synchronous file access. The following figure shows an example usage of the file access in the lambda functions:

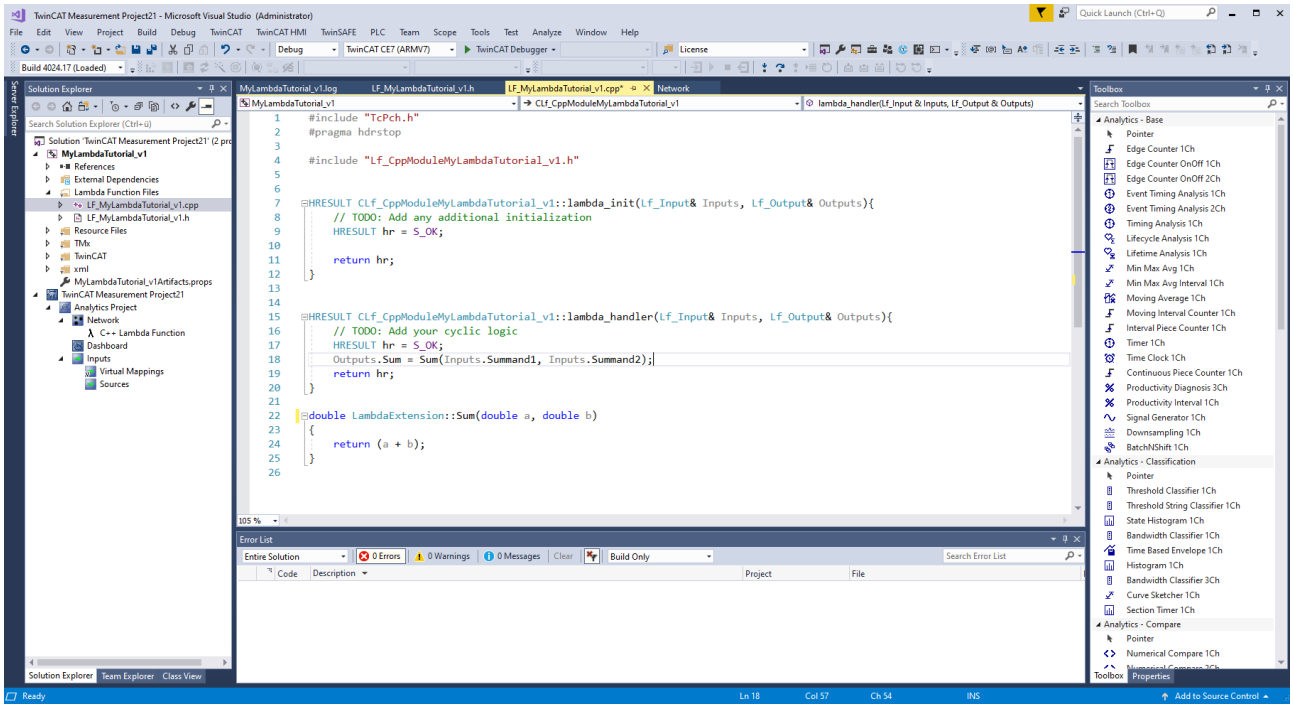


Create the project as usual in Visual Studio® to check the syntax.

In the header file in the "Lambda Function Files" folder you can declare user-specific objects and methods.



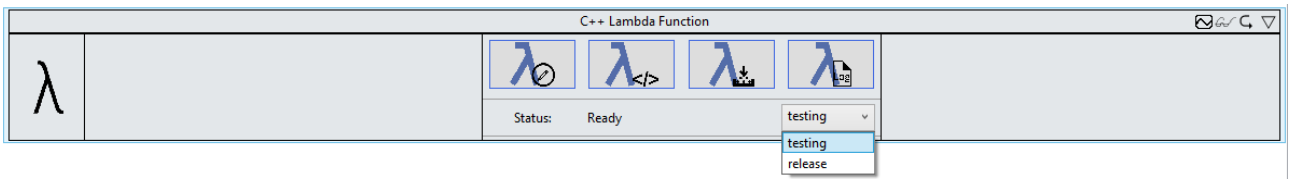
Above the class "LambdaExtension" libraries can be included. Within the class, objects and methods can be declared and used in the lambda functions. The newly declared methods can be implemented below the two lambda functions.



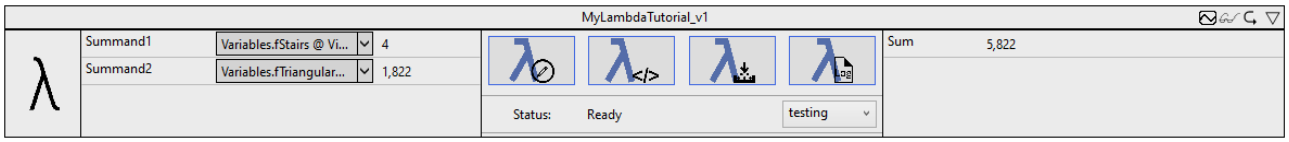
After the logic is integrated, you need to save the project. Then switch back to the lambda template in the Analytics project.
If sufficient know-how in dealing with TcCOM modules is available, the C++ project can be extended individually. If the TcCOM modules are used improperly, errors may occur during 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

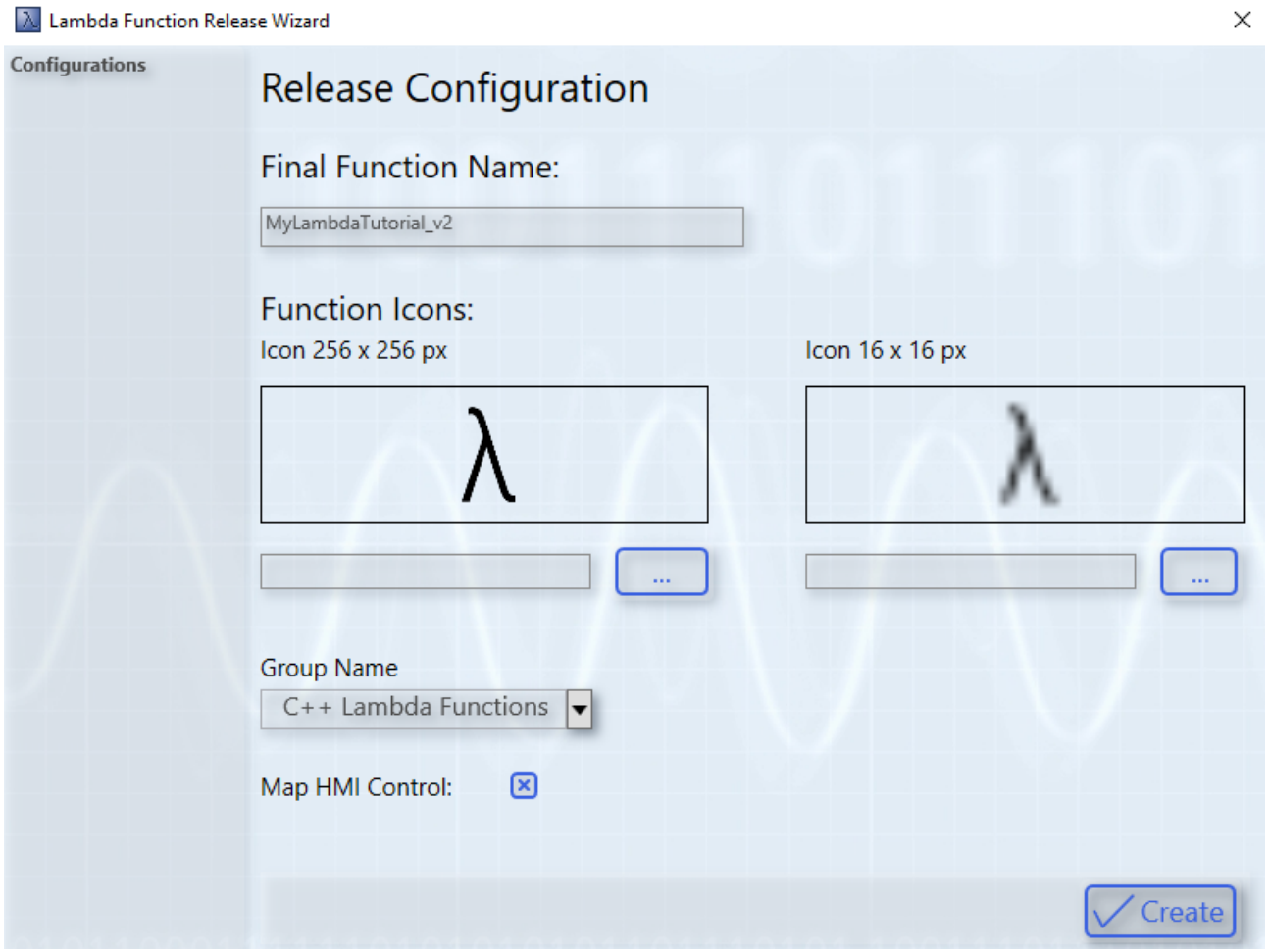
Before publishing, you can configure the type of lambda function using the drop-down menu at the bottom right.



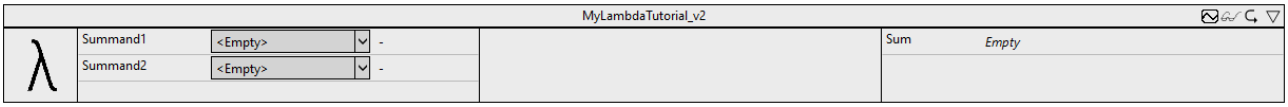
A lambda function of type "testing" contains the control with the four buttons for developing lambda functions. This type is useful during the development and testing phase.



Once the lambda function is working, select the "release" type. After clicking the third button, a Release Wizard opens.



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 Mapping Wizard). Use the Create button to generate the lambda function.



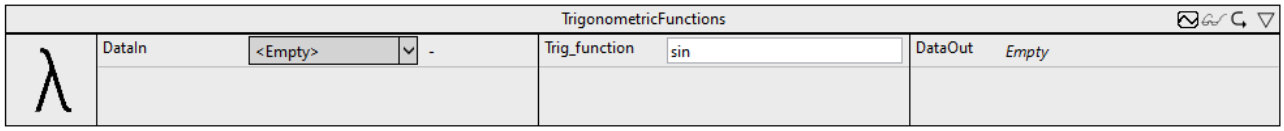
After successful publishing, the lambda template is automatically replaced by the newly generated lambda function.

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.

6.5.11.1.4 Lambda Blueprints

Trigonometric functions



The blueprint **Triginometric 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*.

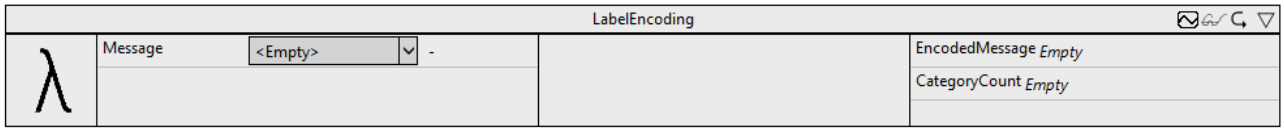
Configuration options

- **Trig_function**: the short form of the trigomic 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



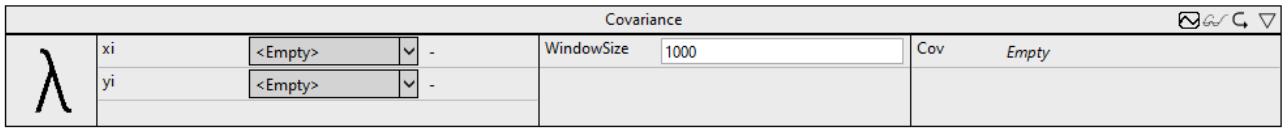
The blueprint **Label Encoding** converts text on an input channel to numeric values. For this purpose, the texts are classified into different categories at the input channel. Equal input texts return the same numeric value.

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



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

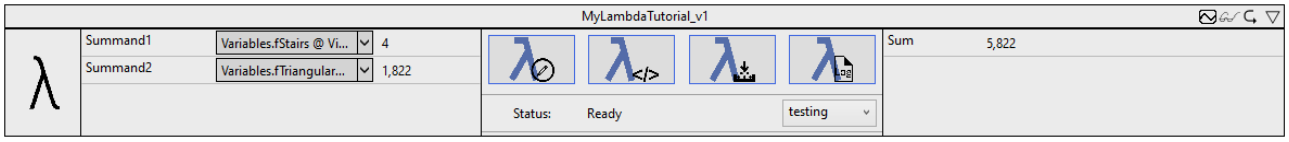
- **WindowSize**: this parameter specifies the number of cycles over which the covariance is calculated. A sliding window is implemented in the blueprint. The current values are included in the analysis and outputs are updated with each cycle. The size of the router memory should be taken into account when setting this parameter.

Output values

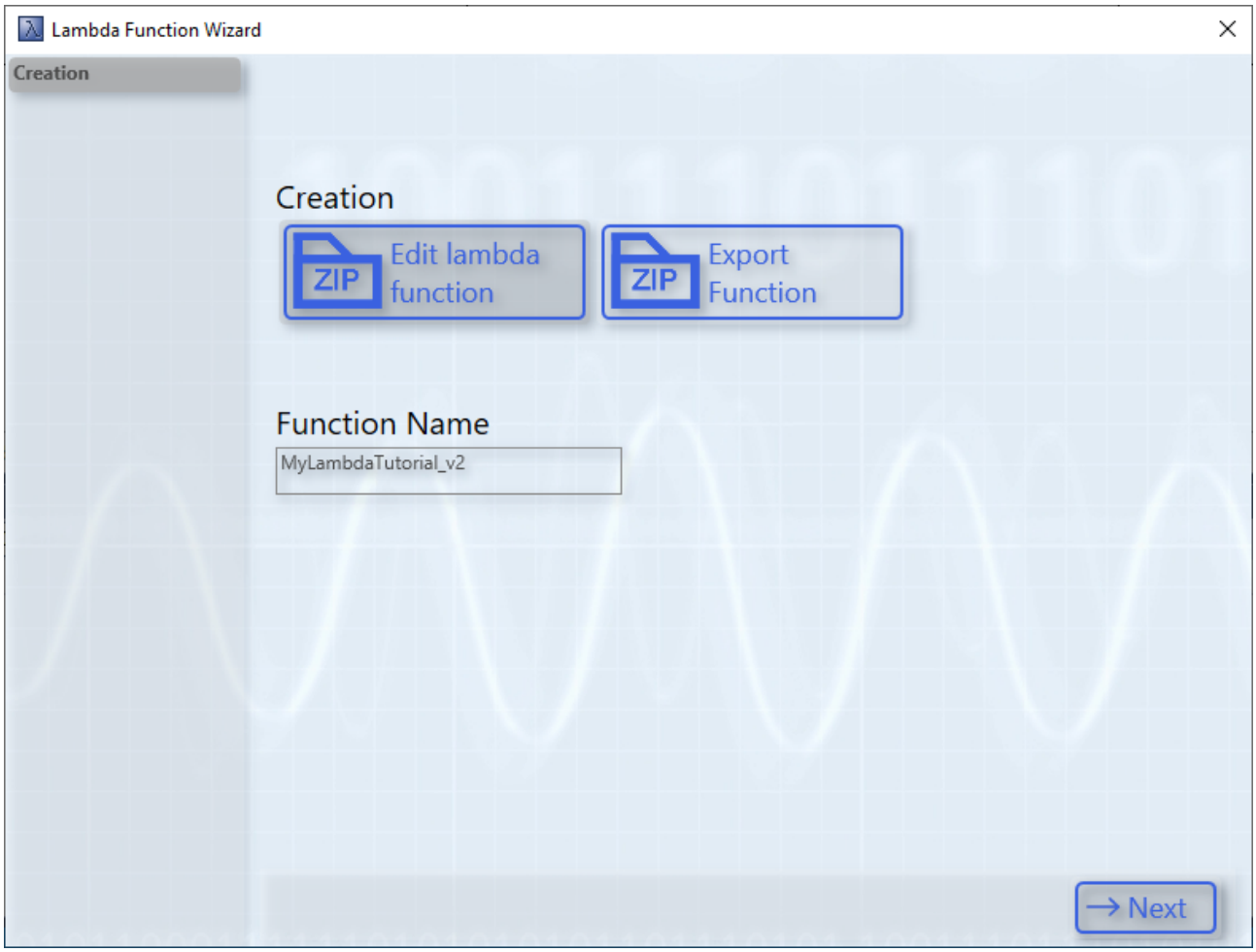
- **Cov**: this output channel indicates the covariance of the two input channels over the configured window size.

6.5.11.1.5 Lambda Function

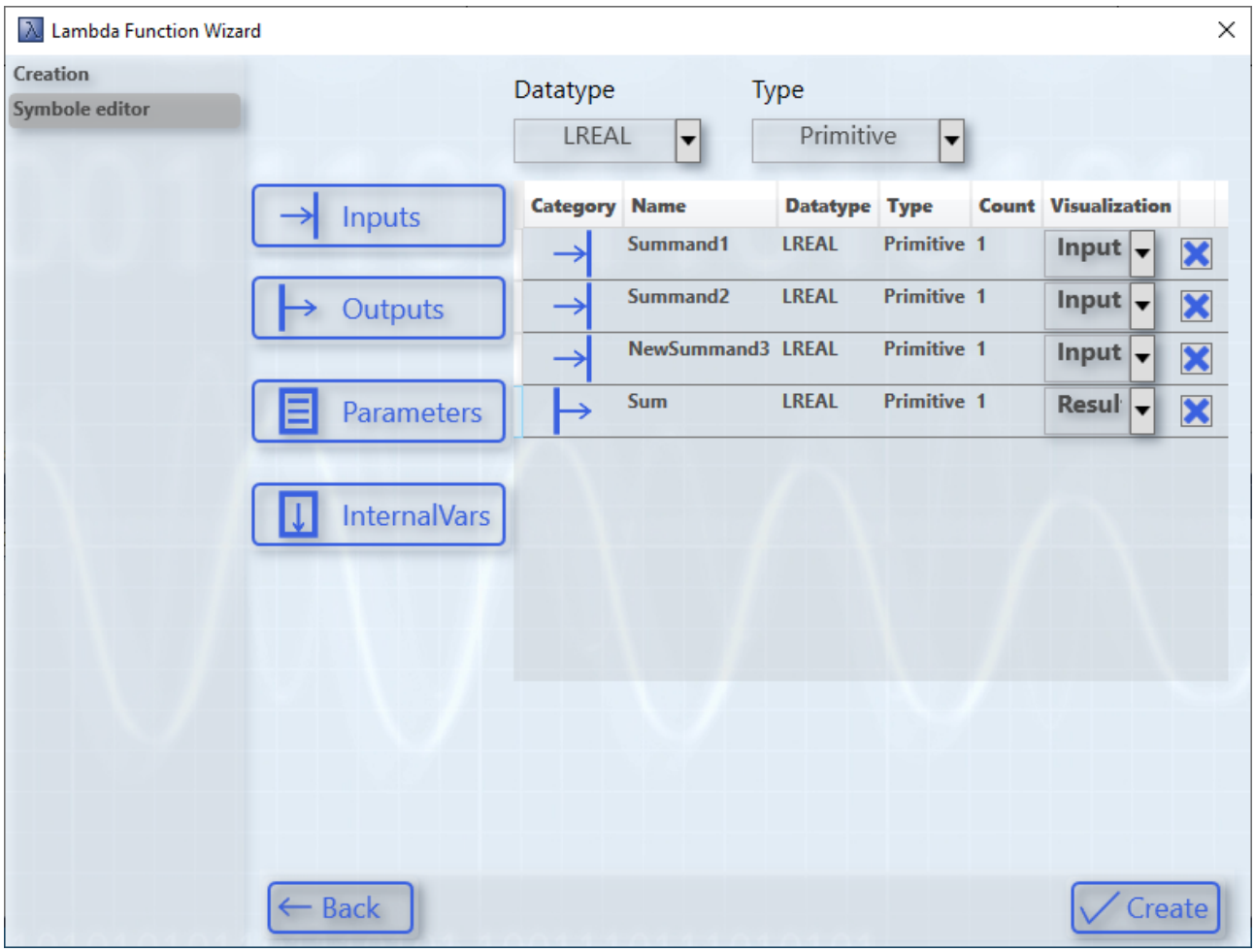
In the development and testing phase, lambda functions should be published as testing versions. These contain the configured symbols, as well as the Lambda Control with the four buttons.



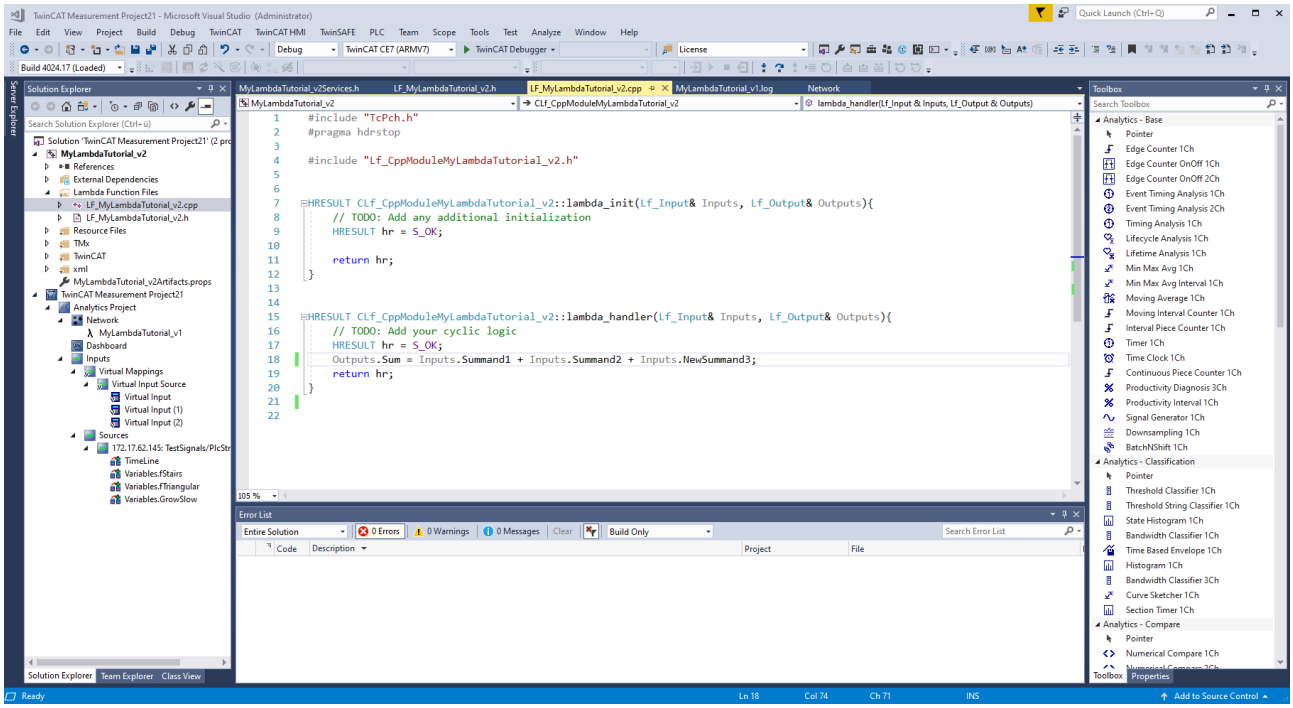
This lambda function can be tested like any other algorithm in Analytics Workbench. If changes are necessary, the Lambda Control guides you through the further development. After clicking the first button, a wizard for configuring the lambda function opens. The wizard for existing lambda functions offers 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*.

6.5.11.1.6 Further steps for the lambda functions

Runtime deployment

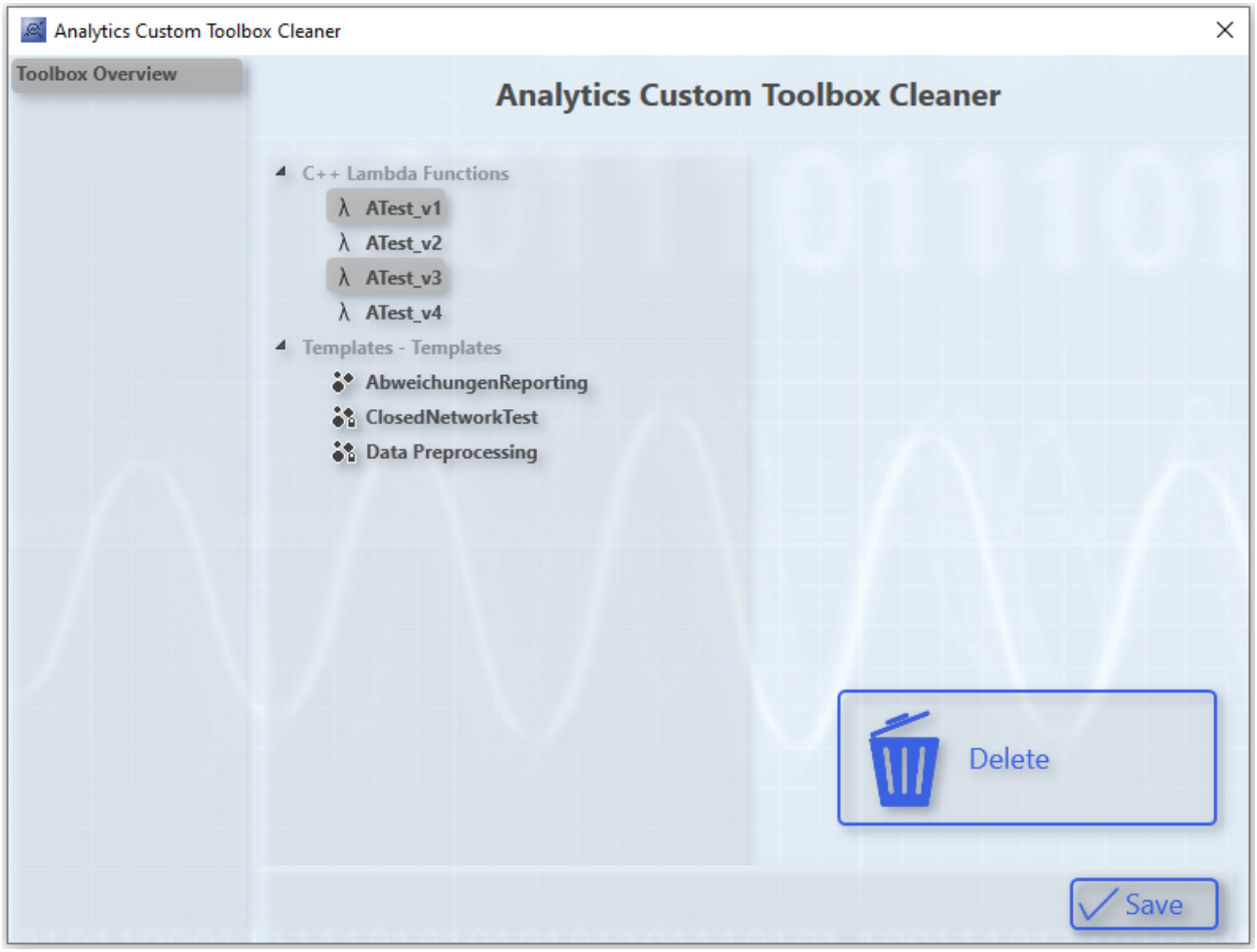
Lambda functions are also suitable for runtime deployment. The developed lambda function is created as a TcCOM module in the TwinCAT project for this purpose. A PLC function block handles the TcCOM module. The PLC function block calls the corresponding interface methods of the lambda function for this 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).

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 Dashboard node).

6.5.11.1.7 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.

6.5.11.2 Algorithms from other TwinCAT libraries

By implementing the TwinCAT Filter and TwinCAT Condition Monitoring libraries, a further variety of algorithms is available. The integration of these algorithms in TwinCAT Analytics turns programming tasks into pure configuration tasks. This reduces the effort required to create an analysis and also enables these algorithms to be used together with those of the TwinCAT Analytics Library directly in the TwinCAT Analytics Workbench configurator or Service Tool. They can be found in the toolbox in the same way as the other algorithm groups. The additional algorithms can be used for the engineering module without an additional license. However, the deployment of the algorithms in the TwinCAT Analytics runtime and the generation of an HMI One-Click Dashboard additionally require the licenses of the respective library. The following list shows an overview of the integrated libraries and algorithms, the license required for deployment or dashboard generation, and a link to the documentation of the respective PLC function block.

Library	Algorithms	License	Documentation
TwinCAT Filter	*Filter*	TF3680 TC3 Filter / TF3600 TC3 Condition Monitoring	
	PT1		PT1 [▶ 281]
	IIRCoeff		IIRCoeff [▶ 282]
	IIRSpec		IIRSpec [▶ 282]
	PT2		PT2 [▶ 283]
	PT3		PT3 [▶ 283]
	PTn		PTn [▶ 284]
	IIRSos		IIRSos [▶ 284]
	Notch		Notch [▶ 284]
	LeadLag		LeadLag [▶ 285]
	PT2oscillation		PT2oscillation [▶ 285]
	PTt		PTt [▶ 286]
	Median		Median [▶ 286]
	ActualValue		ActualValue [▶ 287]
	Gaussian		Gaussian [▶ 287]
TwinCAT Condition Monitoring	*Condition Monitoring*	TF3600 TC3 Condition Monitoring	
	ArgSort		ArgSort [▶ 267]
	Crest Factor		Crest Factor [▶ 268]
	Crest Factor Plus		Crest Factor Plus [▶ 268]
	DiscreteClassification		Discrete Classification [▶ 269]
	Envelope		Envelope [▶ 269]
	Envelope Spectrum		Envelope Spectrum [▶ 270]
	Fatigue Analysis		Fatigue Analysis [▶ 271]
	Instantaneous Frequency		Instantaneous Frequency [▶ 272]
	Instantaneous Phase		Instantaneous Phase [▶ 267]
	Integrated RMS		Integrated RMS [▶ 273]
	Magnitude Spectrum		Magnitude Spectrum [▶ 274]
	Multi-Band RMS		Multi Band RMS [▶ 275]
	Power Spectrum		Power Spectrum [▶ 276]
	RMS		RMS [▶ 277]
	Spike Energy Spectrum		Spike Energy Spectrum [▶ 278]
	Vibration Assessment		Vibration Assessment [▶ 279]
	Watch Upper Thresholds		Watch Upper Thresholds [▶ 280]

The **TwinCAT Filter** library enables the implementation of digital filters of different types using various function blocks.

The library **TwinCAT Condition Monitoring** offers mathematical algorithms for condition monitoring of machines and plants.

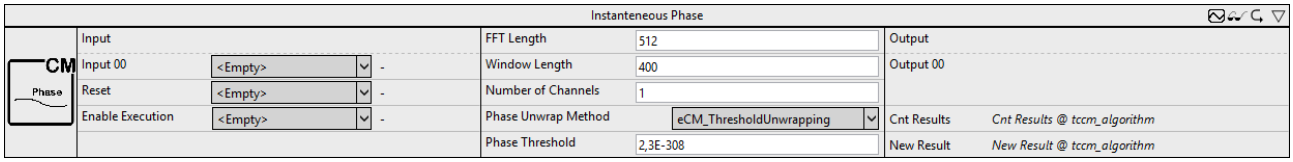
6.5.11.2.1 Condition Monitoring

The library **TwinCAT Condition Monitoring** offers mathematical algorithms for condition monitoring of machines and plants.

Algorithms Overview

Algorithm	Description	PLC function block
ArgSort [▶ 267]	Sorts the incoming arguments.	FB_CMA_ArgSort
Crest Factor [▶ 268]	Calculates the crest factor for each channel for single and multi-channel time series.	FB_CMA_CrestFactor
Crest Factor Plus [▶ 268]	Calculates the crest factor plus for each channel for single and multi-channel time series.	FB_CMA_CrestFactorPlus
Discrete Classification [▶ 269]	Classification of multi-channel data based on configurable threshold values.	FB_CMA_DiscreteClassification
Envelope [▶ 269]	Calculates the envelope of a time signal.	FB_CMA_Envelope
Envelope Spectrum [▶ 270]	Calculates the envelope spectrum of a time signal.	FB_CMA_EnvelopeSpectrum
Fatigue Analysis [▶ 271]	Service life analysis and damage calculation.	FB_CMA_RainflowCounting FB_CMA_MeanStressCorrection FB_CMA_MinersRule
Instantaneous Frequency [▶ 272]	Calculation of the instantaneous frequency of a time signal.	FB_CMA_InstantaneousFrequency
Instantaneous Phase [▶ 267]	Calculation of the instantaneous phase of a time signal.	FB_CMA_InstantaneousPhase
Integrated RMS [▶ 273]	Calculates (optionally integrated) RMS values for single- and multi-channel real-valued time series.	FB_CMA_IntegratedRMS
Magnitude Spectrum [▶ 274]	Calculates the magnitude spectrum (also referred to as amplitude spectrum) of a real-valued input signal.	FB_CMA_MagnitudeSpectrum
Multi Band RMS [▶ 275]	Calculated RMS value for single- and multi-channel real-valued time series for configurable frequency bands.	FB_CMA_MultiBandRMS
Power Spectrum [▶ 276]	Calculation of the power spectrum of a real-valued input signal, and optional decibel scaling.	FB_CMA_PowerSpectrum
RMS [▶ 277]	Calculates the temporal RMS value for single and multi-channel real-valued signals.	FB_CMA_RMS
Spike Energy Spectrum [▶ 278]	Analysis of peak energy of high-frequency signal components.	FB_CMA_SpikeEnergySpectrum
Vibration Assessment [▶ 279]	Vibration assessment of real-valued input signals.	FB_CMA_VibrationAssessment
Watch Upper Thresholds [▶ 280]	Configurable threshold value monitoring of multi-channel data.	FB_CMA_WatchUpperThresholds

6.5.11.2.1.1 Instantaneous Phase



Calculation of the instantaneous phase of a time signal.

The documentation of the corresponding PLC function block can be found here:
FB_CMA_InstantaneousPhase

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

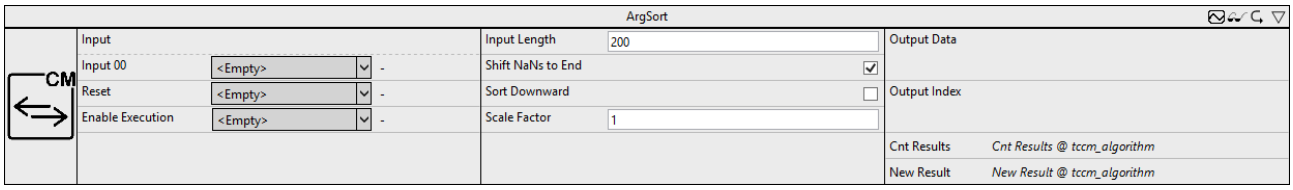
Configuration options

- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.

Phase Unwrap Method: defines the method used for phase-unwrapping with regard to the phase in multiples of 2 PI (see E_CM_UnwrapMethod).

- **Phase Threshold**: limit value for calculating the instantaneous phase. The value is related to the signal envelope. Interpretation: if the signal level is too low, the calculation of the phase is numerically too uncertain and cannot be evaluated reliably. 0 is then output as the phase.

6.5.11.2.1.2 ArgSort



Sorts the incoming arguments.

The documentation of the corresponding PLC function block can be found here: FB_CMA_ArgSort

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

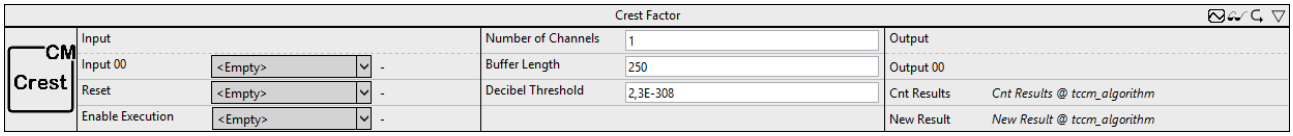
Configuration options

- **Input Length**: is the length of the input array.
- **Shift NaNs to End**: is a flag to select whether NaNs are shifted to the end of the output array.
- **Sort Downward**: is a flag with which you can select whether the data are to be sorted in ascending (`FALSE`) or descending (`TRUE`) order.
- **Scale Factor**: this parameter can be used to display directly the amplitudes with associated frequencies instead of the index position (Scale Factor = 1), for example.

Output values

- **Output Data**: output array with identical length of the input array, which contains the optionally ascending or descending sorted input values.
- **Output Index**: output array with identical length of the input array, which contains the (scalable) indices belonging to the sorted values in the input array.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.3 Crest Factor



Calculates the crest factor for each channel for single and multi-channel time series.

The documentation of the corresponding PLC function block can be found here: FB_CMA_CrestFactor

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

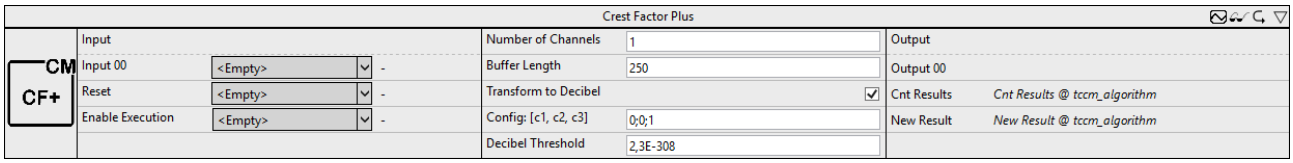
Configuration options

- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Buffer Length**: is the number of input values per channel held in the internal buffer.
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)

Output values

- **Output**: crest factor of the associated input data stream.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.4 Crest Factor Plus



Calculates the crest factor for each channel for single and multi-channel time series.

The documentation of the corresponding PLC function block can be found here: FB_CMA_CrestFactorPlus

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

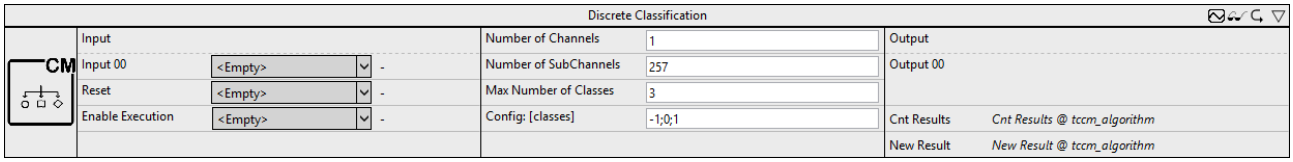
Configuration options

- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Buffer Length**: is the number of input values per channel held in the internal buffer.
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).
- **Config: [c1, c2, c3]:** configurable weights of the maximum amplitude (c1), the RMS value (c2) and the crest factor (c3).
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)

Output values

- **Output**: crest factor plus of the associated input data stream with respect to the configurable weights.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.5 Discrete Classification



Classification of multi-channel data based on configurable threshold values.

The documentation of the corresponding PLC function block can be found here:
FB_CMA_DiscreteClassification

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

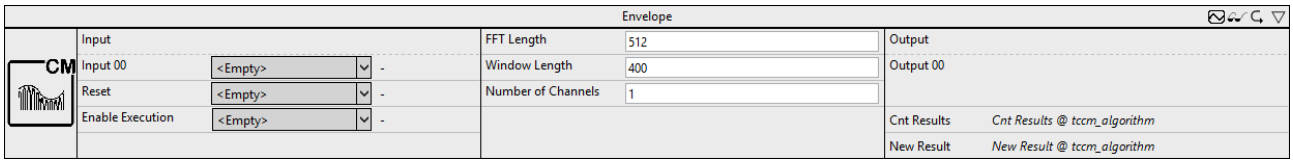
Configuration options

- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Number of SubChannels**: defines the number of independent subchannels. This value is automatically adjusted to the length of the linked input array. If multiple channels are used, all lengths must match.
- **Max Number of Classes**: defines the maximum number of classes that will be configured. The value must be at least one.
- **Config: [classes]**: definition of the configurable threshold values.

Output values

- **Output**: output array with identical length of the input array, which contains the classifications of the entries.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.6 Envelope



Calculates the envelope of a time signal.

The documentation of the corresponding PLC function block can be found here: FB_CMA_Envelope

For time-synchronous presentation and further processing, see the section Timeshift [▶ 269].

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

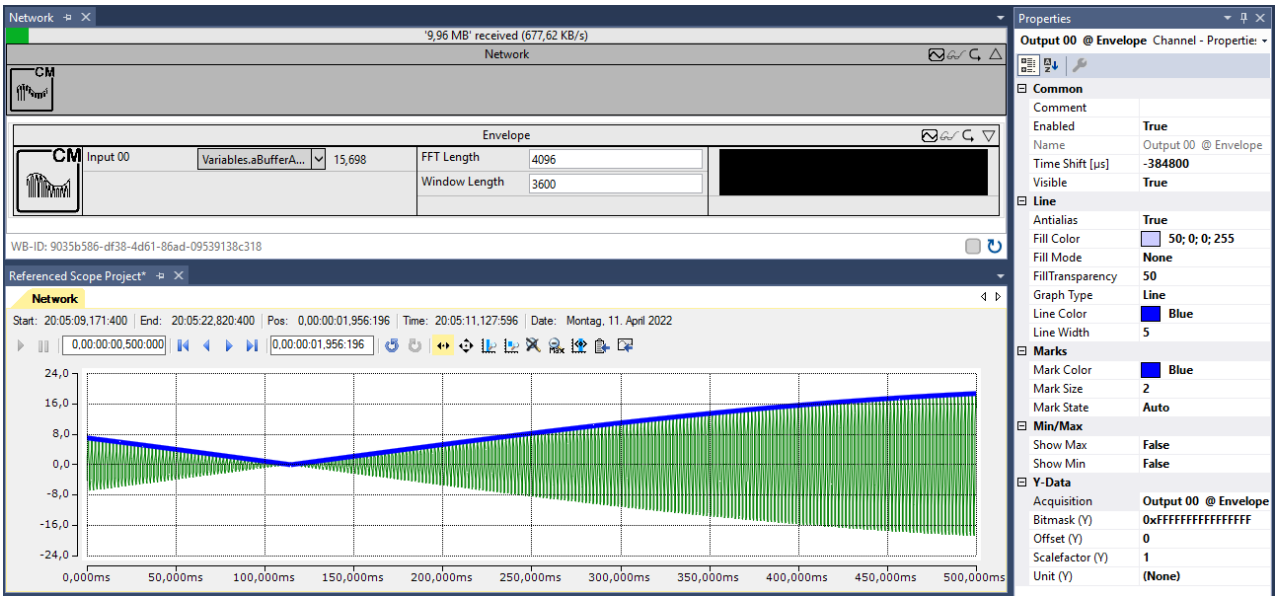
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.

Output values

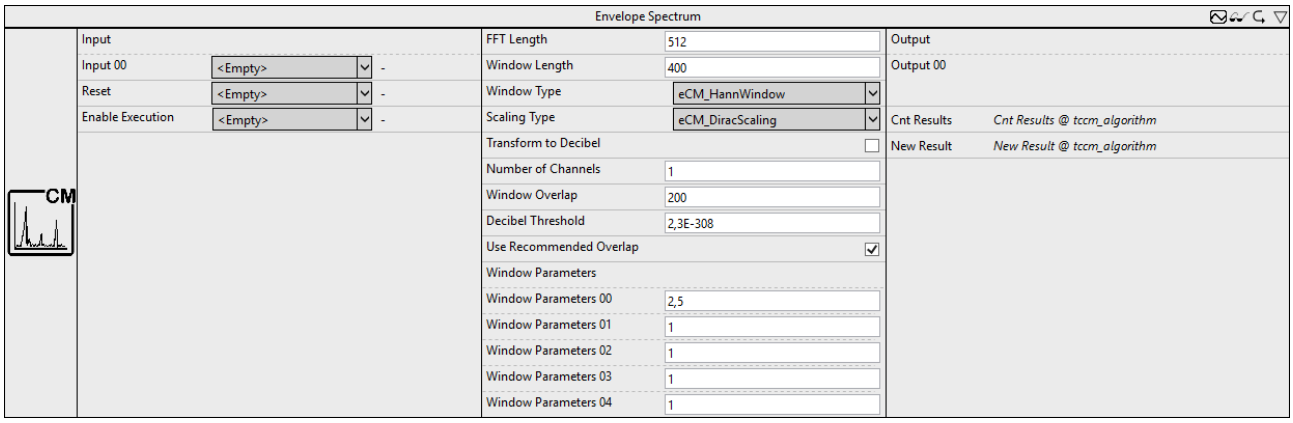
- **Output**: output array with half window length containing the calculated envelope.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

Time shift

Due to the use of the Welch method (50% overlap) for the analysis as well as a segmented convolution of the analytical signal in the synthesis, four complete data buffers of length `nWindowLength/2` are needed to calculate a valid output signal. Furthermore, the result is delayed by `(nFFTLentgth + nWindowLength)/2` samples. This can be corrected by a suitable selection of the Time Shift at the channel of the output.



6.5.11.2.1.7 Envelope Spectrum



Calculates the envelope spectrum of a time signal.

The documentation of the corresponding PLC function block can be found here: FB_CMA_EnvelopeSpectrum

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

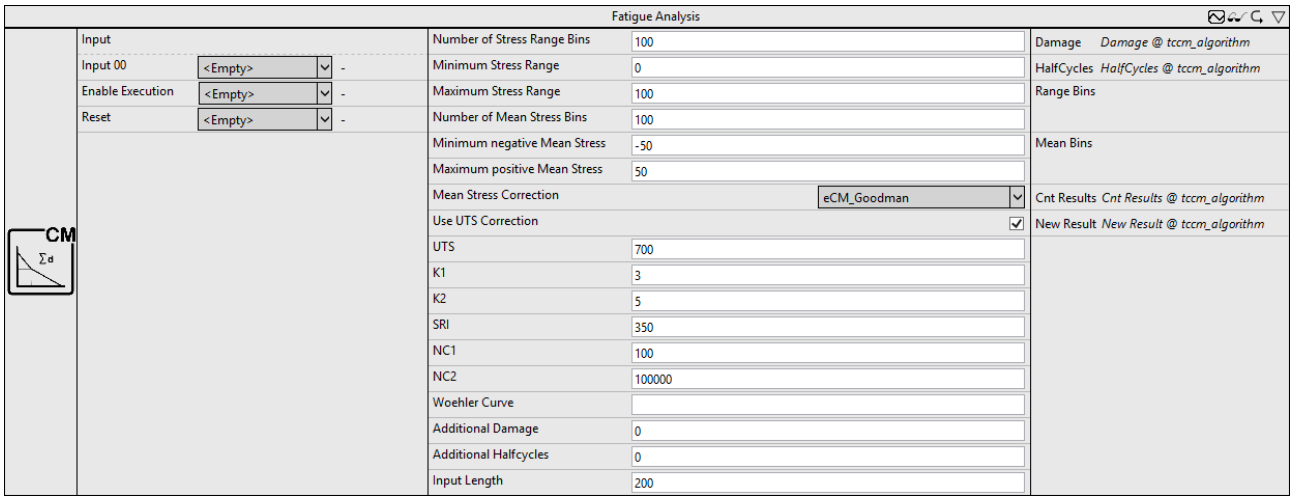
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **WindowType**: Defines the used window function (of the type E_CM_WindowType). A good default value is the window type `eCM_HannWindow`.
- **Scaling Type**: Enables the selection of the scaling (of the type E_CM_ScalingType) to be used, in case absolute scaling is required. The default value is `eCM_DiracScaling`. When selecting the scaling the type of signal should be considered: either deterministic signals or wide-band signals with stochastic portion. Both types require different scalings.
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).

- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Window Overlap**: defines the number of overlapping samples. This must be greater than or equal to zero.
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)
- **Use Recommended Overlap**: if selected, a recommended overlap is calculated internally (see F_CM_CalculateRecommendedOverlap).
- **Window Parameters**: contains the free parameters of selected window functions. When using `eCM_KaiserWindow`, the first entry defines the parameter beta; if `eCM_FlatTopWindow` is used, all parameters are used. See section Window functions.

Output values

- **Output**: output array of length `N / 2 + 1` (for FFT length `N`), which contains the calculated spectral lines of the envelope spectrum.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.8 Fatigue Analysis



Service life analysis and damage calculation to estimate the fatigue process of monitored components.

The documentation of the corresponding PLC function blocks can be found here: FB_CMA_RainflowCounting, FB_CMA_MeanStressCorrection and FB_CMA_MinersRule

For the calculation of the Wöhler curve, the function F_CM_CalculateWoehlerCurve is executed on the basis of fictitious material parameters.

For more detailed explanations, refer to the Condition Monitoring documentation in the chapter Application Concepts, section Service Life Analysis and Damage Calculation.

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

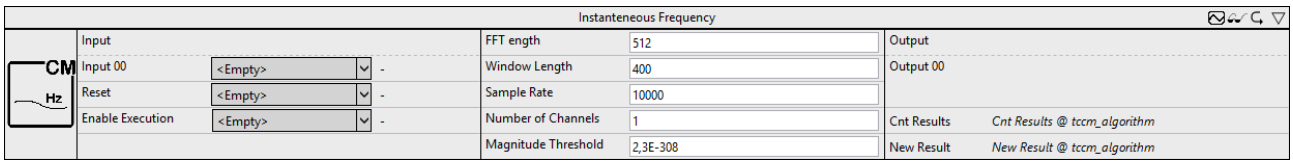
- **Number of Stress Range Bins**: defines the number of bins for the stress level. The total number of bins is "nBins + 2"; the value must be at least one. The first bin of each row contains the stress levels that are less than or equal to the defined minimum; analogously, all values greater than the defined maximum fall into the last bin of each row.
- **Minimum Stress Range**: defines the lower limit of the stress level. All values that are less than or equal to this value are accumulated in a bin.

- **Maximum Stress Range**: defines the upper limit of the stress level. Values that exceed the maximum are counted in the last bin.
- **Number of Mean Stress Bins**: defines the number of bins for the mean values. The value must be at least one. Two separate bins are added in the first/last column of the Halfcycle Count Matrix, where mean values are kept that are less than or equal to or, respectively, greater than the defined limits.
- **Minimum negative Mean Stress**: defines the lower limit of the mean values. All values that are less than or equal to this value are accumulated in a bin.
- **Maximum positive Mean Stress**: defines the upper limit of the mean values. Values that exceed the maximum are counted in the last bin.
- **Mean Stress Correction**: defines the correction mode to be used. The correction according to Goodman and Gerber is available as well as the option not to make a correction.
- **Use UTS Correction**: if active, a UTS correction of the stress matrix is performed.
- **UTS**: defines the tensile strength of the monitored material.
- **K1**: defines the gradient of the Wöhler curve in the range `N = 1..NC1.`
- **K2**: defines the gradient of the Wöhler curve starting at `N >= NC2.`
- **SRI**: defines the "Stress Range Intercept".
- **NC1**: defines the transition point for the UTS correction.
- **NC2**: defines the transition point between K1 and K2.
- **Woehler Curve**: definition of the Wöhler curve.
- **Additional Damage**: defines the constant damage from the beginning. The total damage is thus calculated from the sum of fDamage and the accumulated damage with respect to the configured Wöhler curve.
- **Additional Halfcycles**: defines the initial value of the counted half cycles at the beginning. The total number of cycles is calculated from the sum of nCycles and the current number of input data.
- **Input Length**: defines the length of the input data array for block processing.

Output values

- **Damage**: indicates the calculated fatigue damage with respect to the defined Wöhler curve.
- **HalfCycles**: indicates the number of half cycles counted.
- **Range Bins**: output array of length "Number of Stress Range Bins" with summed half cycles along the middle stress.
- **Mean Bins**: output array of length "Number of Mean Stress Bins" with summed half cycles along the absolute stress.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.9 Instantaneous Frequency



Calculation of the instantaneous frequency of a time signal.

The documentation of the corresponding PLC function block can be found here:
FB_CMA_InstantaneousFrequency

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

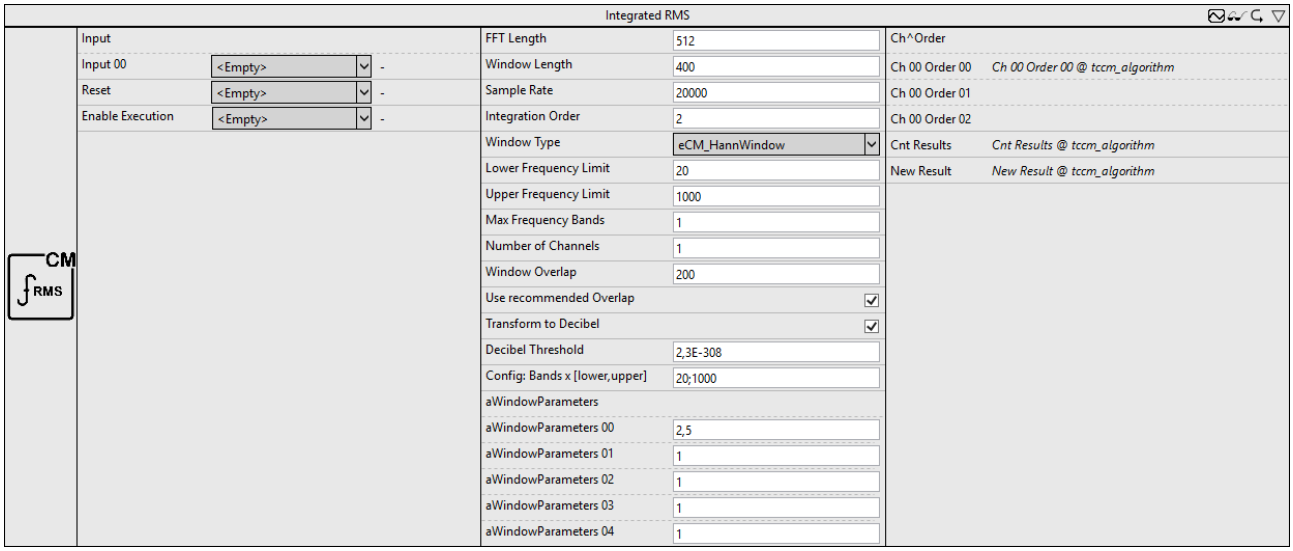
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.

- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **Sample Rate**: sampling rate of the incoming time signal. The value is used for the scaling of the result in Hz.
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Magnitude Threshold**: defines the limit value for the numerical calculability of the instantaneous frequency.

Output values

- **Output**: output array of half window length containing the calculated instantaneous frequency.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.10 Integrated RMS



Calculates (optionally integrated) RMS values for single- and multi-channel real-valued time series.

The documentation of the corresponding PLC function block can be found here: FB_CMA_IntegratedRMS

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

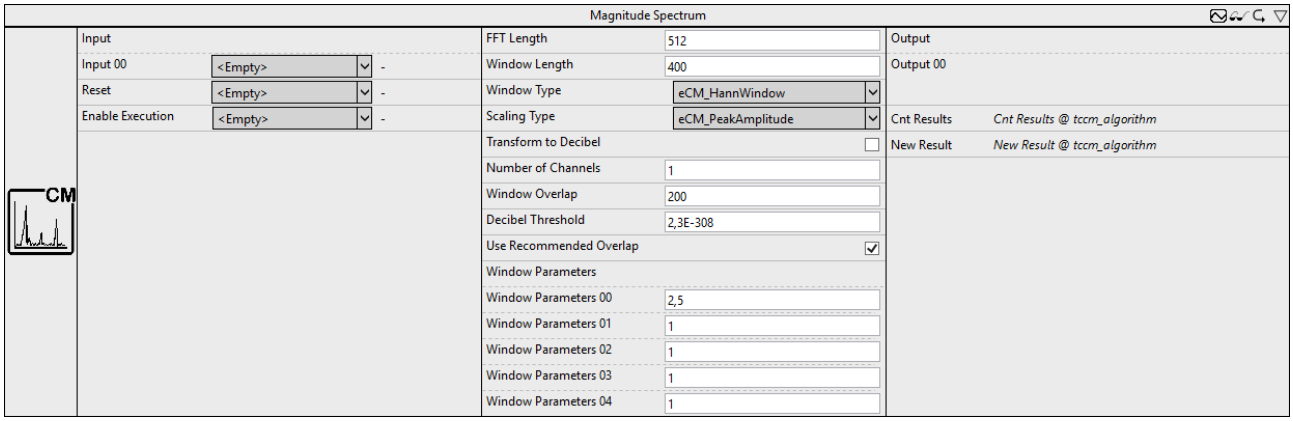
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **Sample Rate**: sampling rate of the incoming time signal. The value is used for the scaling of the result in Hz.
- **Max Integration Order**: is the maximum order of integration. This must be an integer between zero and two. The number of the values determined per channel is (`Order+1`).
- **WindowType**: Defines the used window function (of the type E_CM_WindowType). A good default value is the window type `eCM_HannWindow`.
- **Lower Frequency Limit**: lower limit of the considered frequency interval. The lower cut-off frequency must be at least the sampling rate divided by the FFT-length.
- **Upper Frequency Limit**: upper limit of the considered frequency interval. The upper cut-off frequency must be no greater than half the sampling rate and greater than the lower cut-off frequency.
- **Max Frequency Bands**: this value specifies the maximum number of frequency bands for which the RMS value is calculated.
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.

- **Window Overlap**: defines the number of overlapping samples. This must be greater than or equal to zero.
- **Use Recommended Overlap**: if selected, a recommended overlap is calculated internally (see F_CM_CalculateRecommendedOverlap).
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)
- **Config: MaxBands x [fmin, fmax]**: definition of the configurable lower and upper limit of the frequency bands.
- **Window Parameters**: contains the free parameters of selected window functions. When using `eCM_KaiserWindow`, the first entry defines the parameter beta; if `eCM_FlatTopWindow` is used, all parameters are used. See section Window functions.

Output values

- **Output**: output arrays corresponding to the number of frequency bands with channel and order specific RMS values.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.11 Magnitude Spectrum



Calculates the magnitude spectrum (also referred to as amplitude spectrum) of a real-valued input signal.

The documentation of the corresponding PLC function block can be found here:
FB_CMA_MagnitudeSpectrum

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

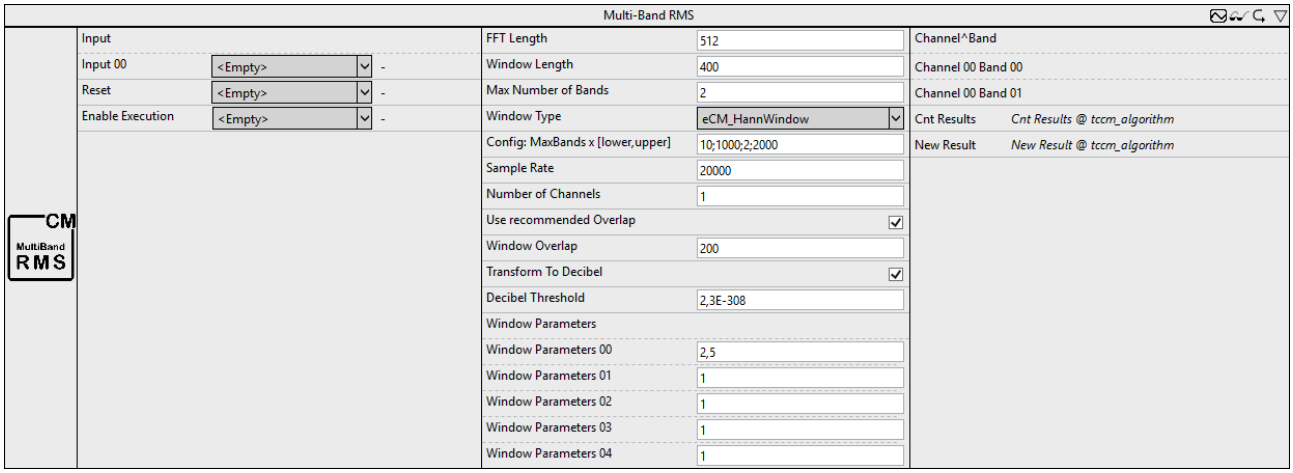
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **WindowType**: Defines the used window function (of the type E_CM_WindowType). A good default value is the window type `eCM_HannWindow`.
- **Scaling Type**: Enables the selection of the scaling (of the type E_CM_ScalingType) to be used, in case absolute scaling is required. The default value is `eCM_DiracScaling`. When selecting the scaling the type of signal should be considered: either deterministic signals or wide-band signals with stochastic portion. Both types require different scalings.

- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Window Overlap**: defines the number of overlapping samples. This must be greater than or equal to zero.
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)
- **Use Recommended Overlap**: if selected, a recommended overlap is calculated internally (see F_CM_CalculateRecommendedOverlap).
- **Window Parameters**: contains the free parameters of selected window functions. When using `eCM_KaiserWindow`, the first entry defines the parameter beta; if `eCM_FlatTopWindow` is used, all parameters are used. See section Window functions.

Output values

- **Output**: output array of length `N / 2 + 1` (for FFT length `N`), which contains the calculated spectral lines of the magnitude spectrum.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.12 Multi Band RMS



Calculated RMS value for single- and multi-channel real-valued time series for configurable frequency bands.

The documentation of the corresponding PLC function block can be found here: FB_CMA_MultiBandRMS

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

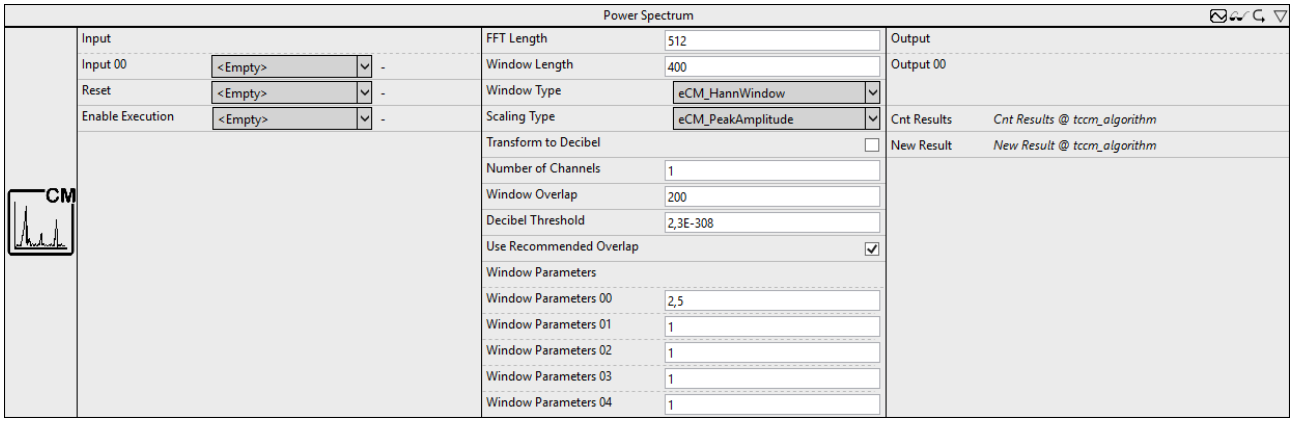
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **Max Number of Bands**: this value specifies the maximum number of frequency bands for which the RMS value is calculated.
- **WindowType**: Defines the used window function (of the type E_CM_WindowType). A good default value is the window type `eCM_HannWindow`.
- **Config: MaxBands x [fmin, fmax]**: definition of the configurable lower and upper limit of the frequency bands.

- **Sample Rate**: sampling rate of the incoming time signal. The value is used for the scaling of the result in Hz.
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Use Recommended Overlap**: if selected, a recommended overlap is calculated internally (see F_CM_CalculateRecommendedOverlap).
- **Window Overlap**: defines the number of overlapping samples. This must be greater than or equal to zero.
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)
- **Window Parameters**: contains the free parameters of selected window functions. When using `eCM_KaiserWindow`, the first entry defines the parameter beta; if `eCM_FlatTopWindow` is used, all parameters are used. See section Window functions.

Output values

- **Output**: channel and band specific RMS values.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.13 Power Spectrum



Calculation of the power spectrum of a real-valued input signal, and optional decibel scaling.

The documentation of the corresponding PLC function block can be found here: FB_CMA_PowerSpectrum

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

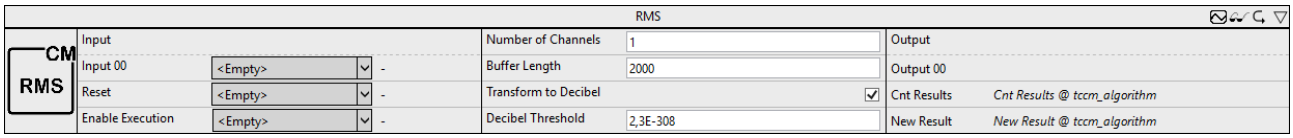
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **WindowType**: Defines the used window function (of the type E_CM_WindowType). A good default value is the window type `eCM_HannWindow`.
- **Scaling Type**: Enables the selection of the scaling (of the type E_CM_ScalingType) to be used, in case absolute scaling is required. The default value is `eCM_DiracScaling`. When selecting the scaling the type of signal should be considered: either deterministic signals or wide-band signals with stochastic portion. Both types require different scalings.
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).

- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Window Overlap**: defines the number of overlapping samples. This must be greater than or equal to zero.
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)
- **Use Recommended Overlap**: if selected, a recommended overlap is calculated internally (see F_CM_CalculateRecommendedOverlap).
- **Window Parameters**: contains the free parameters of selected window functions. When using `eCM_KaiserWindow`, the first entry defines the parameter beta; if `eCM_FlatTopWindow` is used, all parameters are used. See section Window functions.

Output values

- **Output**: output array of length `N / 2 + 1` (for FFT length `N`), which contains the calculated spectral lines of the magnitude spectrum.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.14 RMS



Calculates the temporal RMS value for single and multi-channel real-valued signals.

The documentation of the corresponding PLC function block can be found here: FB_CMA_RMS

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

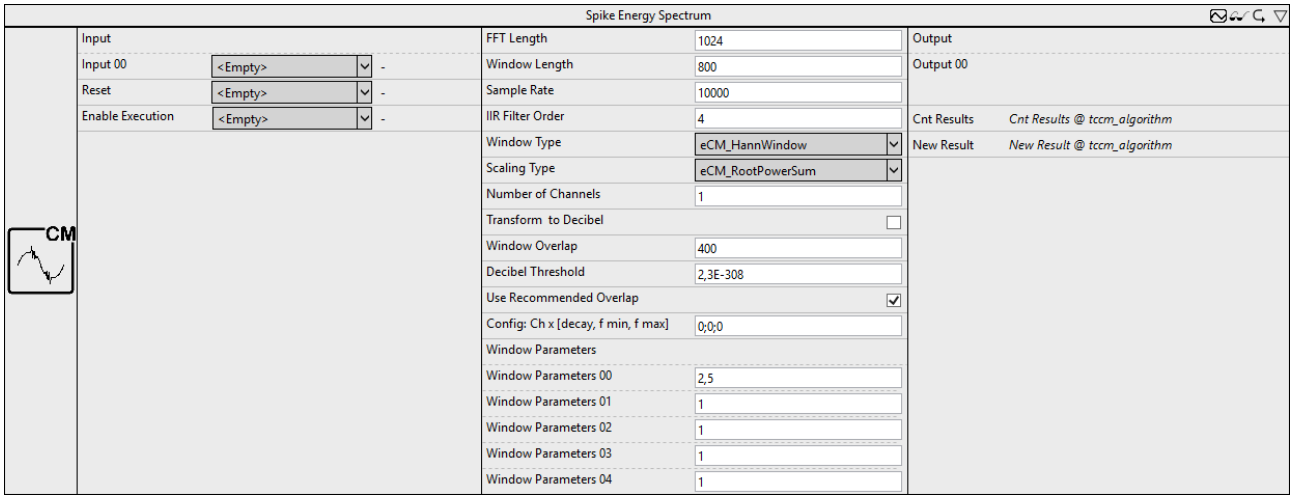
Configuration options

- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Buffer Length**: is the number of input values per channel held in the internal buffer.
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)

Output values

- **Output**: RMS value of the associated input data stream.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.15 Spike Energy Spectrum



Analysis of peak energy of high-frequency signal components.

The documentation of the corresponding PLC function block can be found here:
FB_CMA_SpikeEnergySpectrum

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

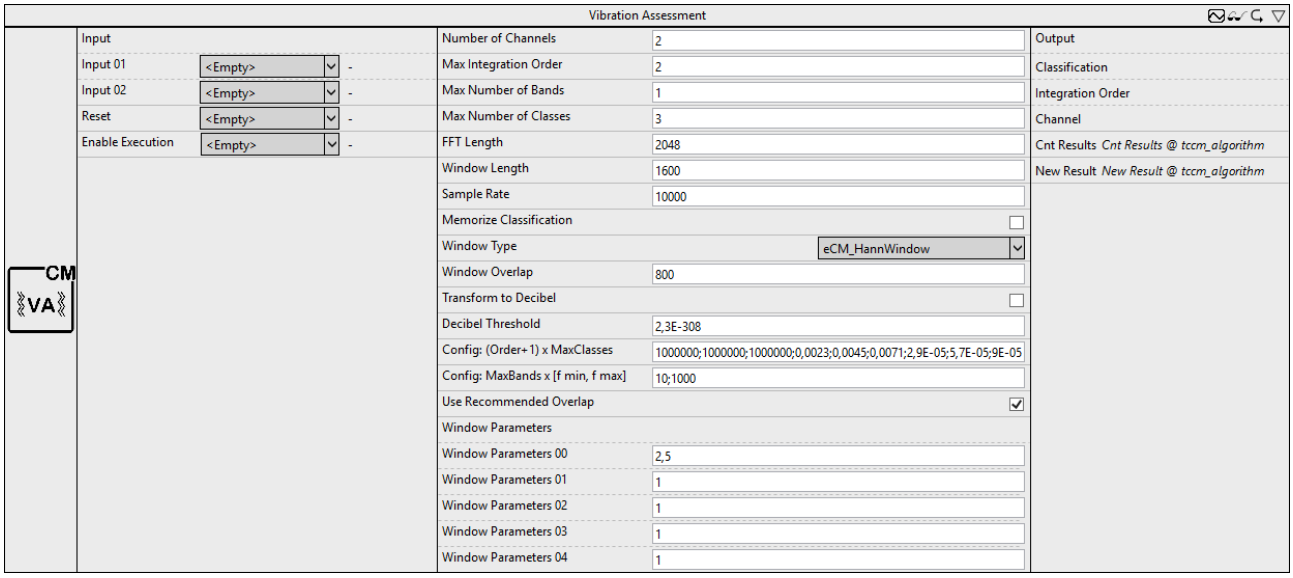
Configuration options

- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **Sample Rate**: sampling rate of the incoming time signal. The value is used for the scaling of the result in Hz.
- **IIR Filter Order**: defines the order of the IIR filter used.
- **WindowType**: Defines the used window function (of the type E_CM_WindowType). A good default value is the window type `eCM_HannWindow`.
- **Scaling Type**: Enables the selection of the scaling (of the type E_CM_ScalingType) to be used, in case absolute scaling is required. The default value is `eCM_DiracScaling`. When selecting the scaling the type of signal should be considered: either deterministic signals or wide-band signals with stochastic portion. Both types require different scalings.
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).
- **Window Overlap**: defines the number of overlapping samples. This must be greater than or equal to zero.
- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)
- **Use Recommended Overlap**: if selected, a recommended overlap is calculated internally (see F_CM_CalculateRecommendedOverlap).
- **Config: Ch x [decay, fmin, fmax]**: definition of the configurable parameters: the decay time and the considered frequency band (by lower and upper limits). The decay time should optimally be chosen so that the peak energy can completely decay, i.e. 'decay time' > 1 / 'error frequency'.
- **Window Parameters**: contains the free parameters of selected window functions. When using `eCM_KaiserWindow`, the first entry defines the parameter beta; if `eCM_FlatTopWindow` is used, all parameters are used. See section Window functions.

Output values

- **Output**: Output array of length `N / 2 + 1` (for FFT length `N`), which contains the calculated spectral lines of the spike energy spectrum.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.16 Vibration Assessment



Vibration assessment of real-valued input signals based on ISO 10816-3.

The documentation of the corresponding PLC function block can be found here:
FB_CMA_VibrationAssessment

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

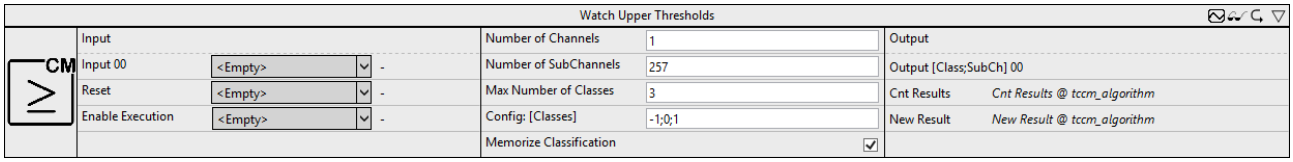
- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Max Integration Order**: is the maximum order of integration. This must be an integer between zero and two. The number of the values determined per channel is (`Order+1`).
- **Max Number of Bands**: this value specifies the maximum number of frequency bands for which the RMS value is calculated.
- **Max Number of Classes**: defines the maximum number of classes that will be configured. The value must be at least one.
- **FFT Length**: is the length of the FFT. It must be greater than one and an integral power of two.
- **Window Length**: is the length of the analysis window in samples. The length must be greater than one and an even number.
- **Sample Rate**: sampling rate of the incoming time signal. The value is used for the scaling of the result in Hz.
- **Memorize Classification**: if selected, the function block recalculates the number of the highest category and the corresponding channel at each step. Otherwise, the result values are stored when a limit value is exceeded until the reset is executed or a channel reaches a higher category.
- **WindowType**: Defines the used window function (of the type E_CM_WindowType). A good default value is the window type `eCM_HannWindow`.
- **Window Overlap**: defines the number of overlapping samples. This must be greater than or equal to zero.
- **Transform to Decibel**: is a Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x -> 20 * log10(x).

- **Decibel Threshold**: is a very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale. (The purpose is the avoidance of value range errors. The logarithm of zero is not defined and strives infinitely towards minus for the limit value of small arguments. The same applies to the argument of the number zero, arg(0). The smallest possible value is 2.3e-308)
- **Config: (Order+1) x MaxClasses**: definition of the configurable threshold values with respect to the integration order.
- **Config: MaxBands x [fmin, fmax]**: definition of the configurable lower and upper limit of the frequency bands.
- **Use Recommended Overlap**: if selected, a recommended overlap is calculated internally (see F_CM_CalculateRecommendedOverlap).
- **Window Parameters**: contains the free parameters of selected window functions. When using `eCM_KaiserWindow`, the first entry defines the parameter beta; if `eCM_FlatTopWindow` is used, all parameters are used. See section Window functions.

Output values

- **Output**: the result is a one-dimensional array that contains three values for each frequency band: the highest calculated classification (in the range -1..'Max Number of Classes'), the associated integration order (in the range 0..'Max Integration Order') and the channel (in the range 1..'Number of Channels').
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

6.5.11.2.1.17 Watch Upper Thresholds



Configurable threshold value monitoring of multi-channel data.

The documentation of the corresponding PLC function block can be found here:
FB_CMA_WatchUpperThresholds

The documentation of the TwinCAT 3 Condition Monitoring PLC library can be found here: Overview.

Configuration options

- **Number of Channels**: defines the number of independent channels. This must be greater than zero.
- **Number of SubChannels**: defines the number of independent subchannels. This value is automatically adjusted to the length of the linked input array. If multiple channels are used, all lengths must match.
- **Max Number of Classes**: defines the maximum number of classes that will be configured. The value must be at least one.
- **Config: [classes]**: definition of the configurable threshold values.
- **Memorize Classification**: if selected, the function block recalculates the number of the highest category and the corresponding channel at each step. Otherwise, the result values are stored when a limit value is exceeded until the reset is executed or a channel reaches a higher category.

Output values

- **Output**: output array of length two for each input data stream. The values of the tuple describe the classification as well as the subchannel in which this class was identified.
- **Cnt Results**: specifies the number of output arrays calculated.
- **New Result**: is a Boolean value that indicates whether a new result was obtained in the current cycle.

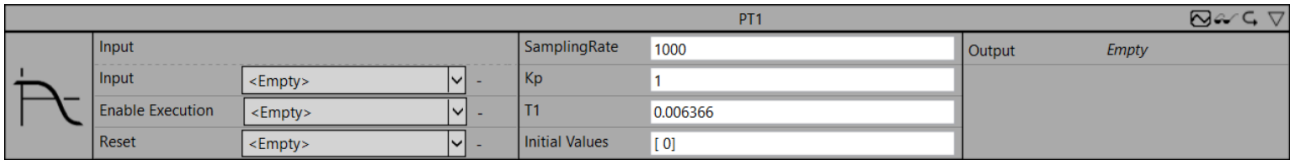
6.5.11.2.2 Filter

The library **TwinCAT Filter** offers algorithms for the implementation of digital filters.

Algorithms Overview

Algorithm	Description	PLC function block
PT1 [▶ 281]	Implementation of a first-order delay element (PT1 element).	FB_FTR_PT1
IIRCoeff [▶ 282]	Implementation of an IIR (Infinitive Impulse Response) filter via filter coefficients.	FB_FTR_IIRCoeff
IIRSpec [▶ 282]	Implementation of an IIR (Infinitive Impulse Response) filter via filter specifications.	FB_FTR_IIRSpec
PT2 [▶ 283]	Implementation of a second-order delay element (PT2 element).	FB_FTR_PT2
PT3 [▶ 283]	Implementation of a third-order delay element (PT3 element).	FB_FTR_PT3
PTn [▶ 284]	Implementation of a delay element of nth order with equal time constants.	FB_FTR_PTn
IIRSos [▶ 284]	Implementation of an IIR (Infinitive Impulse Response) filter via filter coefficients.	FB_FTR_IIRSos
Notch [▶ 284]	Implementation of a band-stop filter with narrow bandwidth.	FB_FTR_Notch
LeadLag [▶ 285]	Implementation of a first-order phase correction element.	FB_FTR_LeadLag
PT2oscillation [▶ 285]	Implementation of a second-order oscillatory delay element.	FB_FTR_PT2oscillation
PTt [▶ 286]	Implementation of a dead time element (PTt element).	FB_FTR_PTt
Median [▶ 286]	Implementation of a median filter.	FB_FTR_Median
ActualValue [▶ 287]	Implementation of a plausibility check and filtering of a measured input value.	FB_FTR_ActualValue
Gaussian [▶ 287]	Implementation of a Gaussian filter.	FB_FTR_Gaussian

6.5.11.2.2.1 PT1



Implementation of a first-order delay element (PT1 element).

The documentation of the corresponding PLC function block can be found here: FB_FTR_PT1

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

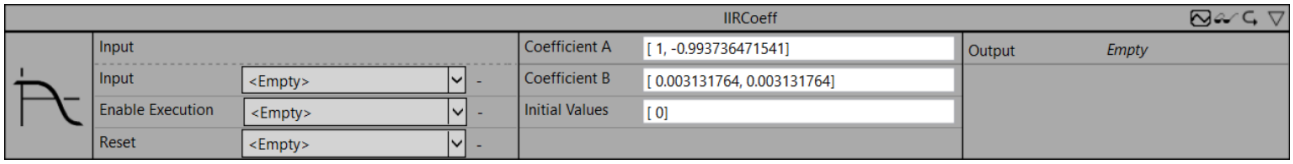
Configuration parameters

- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Kp**: gain factor (greater than zero).
- **T1**: time constant T_1 in seconds (greater than zero).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.2 IIRCoeff



Implementation of an Infinite Impulse Response filter (IIR).

The documentation of the corresponding PLC function block can be found here: FB_FTR_IIRCoeff

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

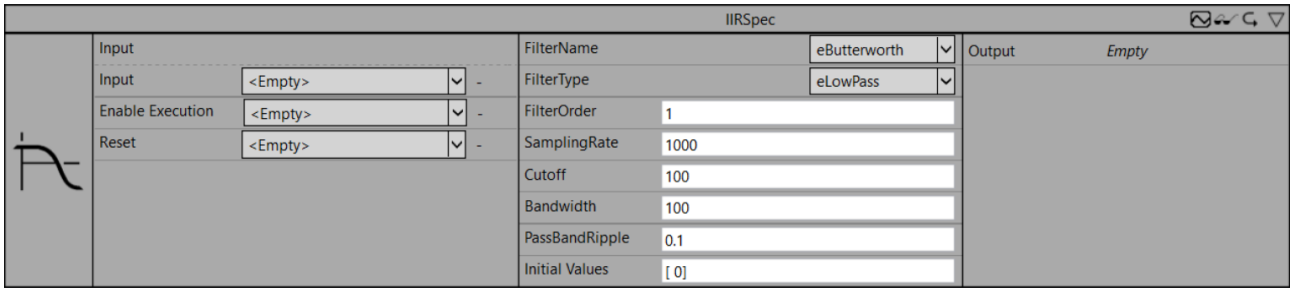
Configuration parameters

- **Coefficient A**: the coefficients a_k (denominator) [a_0, a_1, a_2,..., a_N] can be chosen freely.
- **Coefficient B**: the coefficients b_k (numerator) [b_0, b_1, b_2,..., b_M] can be chosen freely.
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.3 IIRSpec



Implementation of an Infinite Impulse Response filter (IIR).

The documentation of the corresponding PLC function block can be found here: FB_FTR_IIRSpec

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

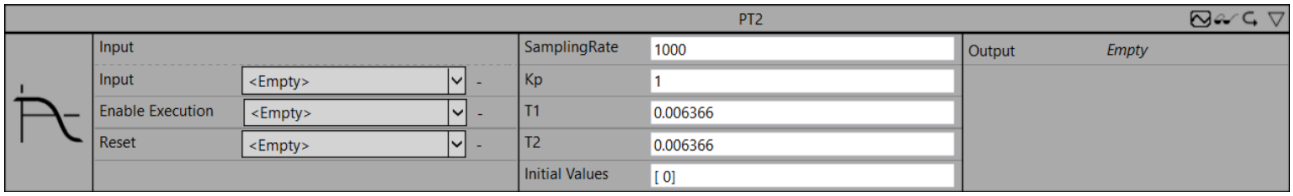
Configuration parameters

- **FilterName**: describes the filter implementation (Butterworth, Chebyshev).
- **FilterType**: describes the filter type (high-pass, low-pass, band-pass, band-stop).
- **FilterOrder**: filter order (max. 20 for high-pass and low-pass, max. 10 for band-pass and band-stop).
- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Cutoff**: cut-off frequency in Hz (greater than 0 and less than SamplingRate/2).
- **Bandwidth**: bandwidth in Hz with respect to band-pass and band-stop.
- **PassBandRipple**: passband ripple of the amplitude response in the passband of the filter in dB (greater than 0).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.4 PT2



Implementation of a second-order delay element (PT2 element).

The documentation of the corresponding PLC function block can be found here: FB_FTR_PT2

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

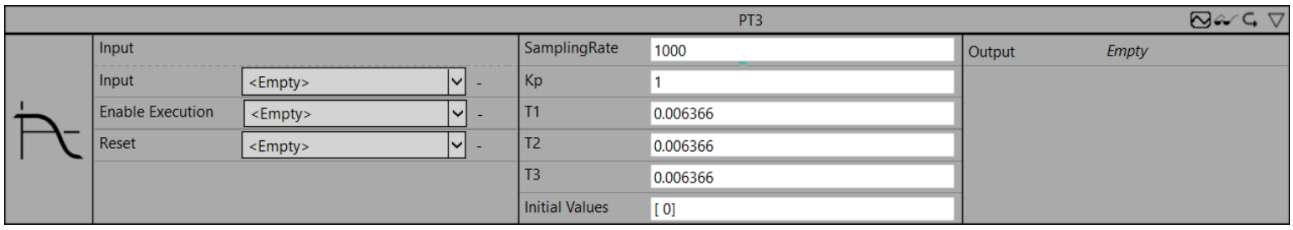
Configuration parameters

- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Kp**: gain factor (greater than zero).
- **T1**: time constant T_1 in seconds (greater than zero).
- **T2**: time constant T_2 in seconds (greater than zero).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.5 PT3



Implementation of a third-order delay element (PT3 element).

The documentation of the corresponding PLC function block can be found here: FB_FTR_PT3

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

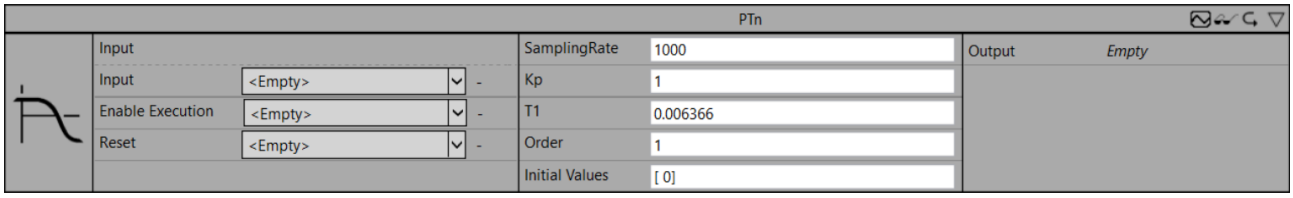
Configuration parameters

- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Kp**: gain factor (greater than zero).
- **T1**: time constant T_1 in seconds (greater than zero).
- **T2**: time constant T_2 in seconds (greater than zero).
- **T3**: time constant T_3 in seconds (greater than zero).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.6 PTn



Implementation of a delay element of nth order with equal time constants.

The documentation of the corresponding PLC function block can be found here: FB_FTR_PTn

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

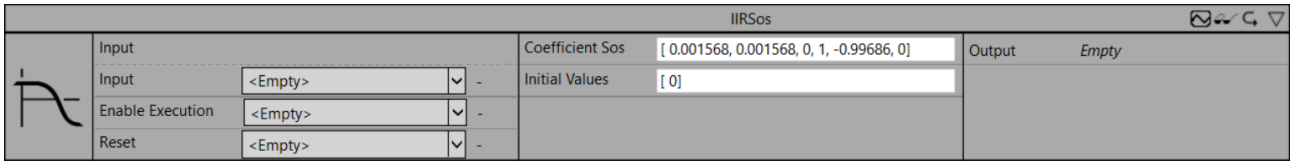
Configuration parameters

- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Kp**: gain factor (greater than zero).
- **T1**: time constant T_1 in seconds (greater than zero).
- **Order**: order of the filter (1..10).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.7 IIRSos



Implementation of an Infinite Impulse Response filter (IIR).

The documentation of the corresponding PLC function block can be found here: FB_FTR_IIRSos.

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

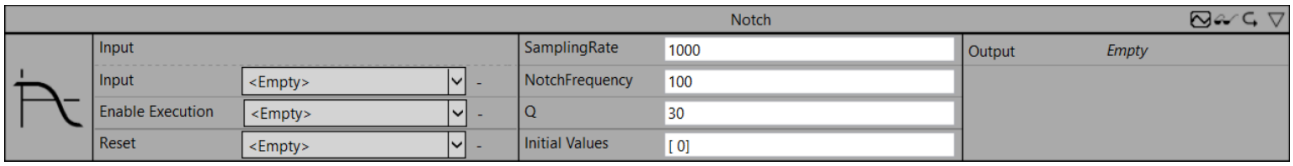
Configuration parameters

- **Coefficient Sos**: the coefficients [b_{01}, b_{11}, b_{21}, a_{01}, a_{11}, a_{21}, b_{02}, b_{12}, b_{22}, a_{02}, a_{12}, a_{22},…, b_{0M}, b_{1M}, b_{2M}, a_{0M}, a_{1M}, a_{2M}] are freely selectable.
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.8 Notch



Implementation of a band-stop filter with narrow bandwidth.

The documentation of the corresponding PLC function block can be found here: FB_FTR_Notch

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

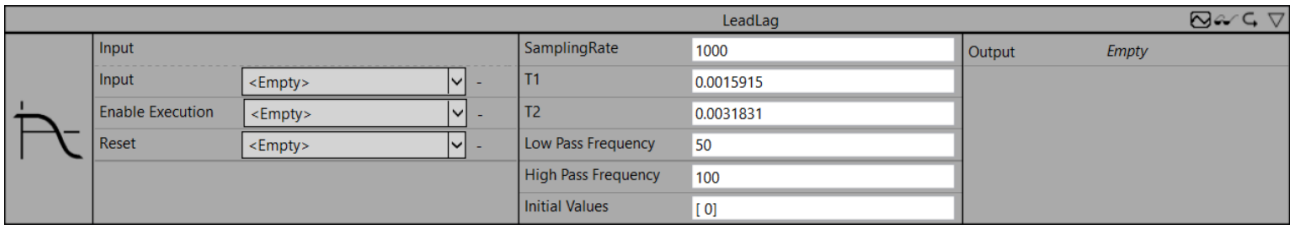
Configuration parameters

- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **NotchFrequency**: notch frequency in Hz (greater than 0 and less than SamplingRate/2).
- **Q**: Q-factor = notch frequency/bandwidth (greater than zero).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.9 LeadLag



Implementation of a first-order phase correction element.

The documentation of the corresponding PLC function block can be found here: FB_FTR_LeadLag.

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

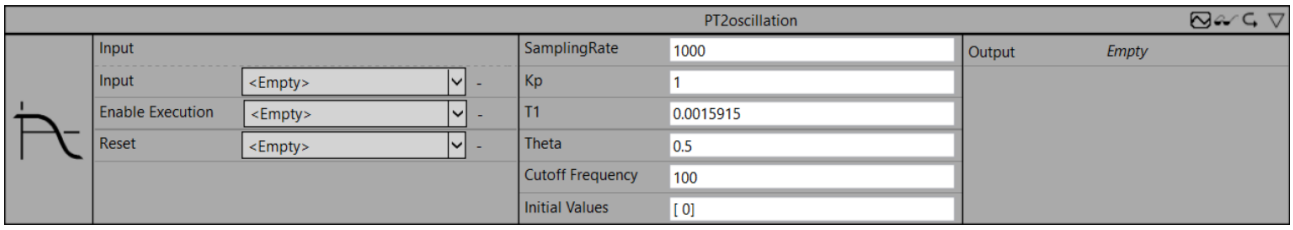
Configuration parameters

- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **T1**: time constant T_1 in seconds (greater than zero).
- **T2**: time constant T_2 in seconds (greater than zero).
- **Low Pass Frequency (optional)**: low-pass frequency in Hz (greater than 0 and less than SamplingRate/2).
- **High Pass Frequency (optional)**: high-pass frequency in Hz (greater than 0 and less than SamplingRate/2).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.10 PT2oscillation



Implementation of a second-order oscillatory delay element.

The documentation of the corresponding PLC function block can be found here: FB_FTR_PT2oscillation

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

Configuration parameters

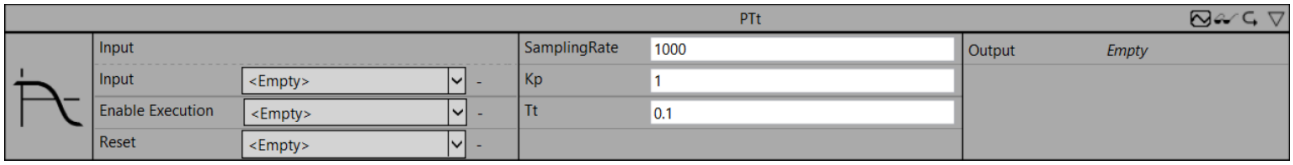
- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Kp**: gain factor (greater than zero).

- **T1**: time constant T_1 in seconds (greater than zero).
- **Theta**: damping factor (greater than zero).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.11 PTt



Implementation of a dead time element (PTt element).

The documentation of the corresponding PLC function block can be found here: FB_FTR_PTt

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

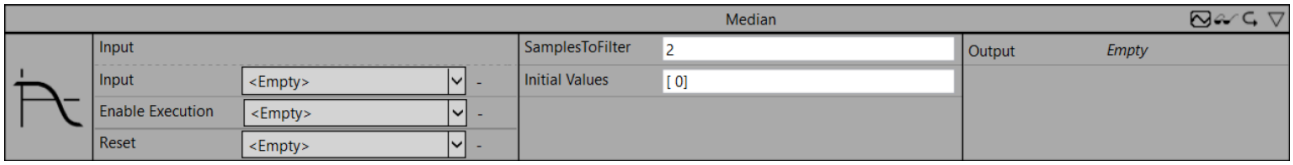
Configuration parameters

- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Kp**: gain factor (greater than zero).
- **Tt**: dead time T_t in seconds (greater than zero).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.12 Median



Implementation of a median filter.

The documentation of the corresponding PLC function block can be found here: FB_FTR_Median

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

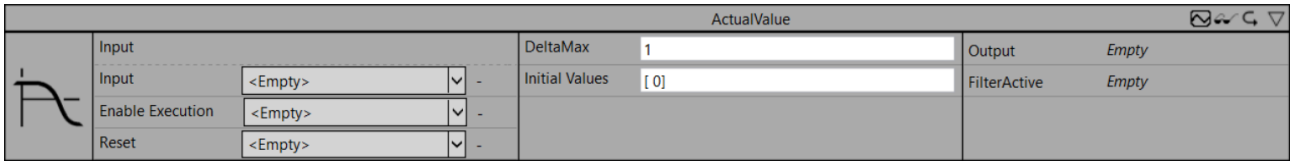
Configuration parameters

- **SamplesToFilter**: number of samples for calculating the sliding average value (often referred to as window size).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.

6.5.11.2.2.13 ActualValue



Implementation of a plausibility check and filtering of a measured input value.

The documentation of the corresponding PLC function block can be found here: FB_FTR_ActualValue

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

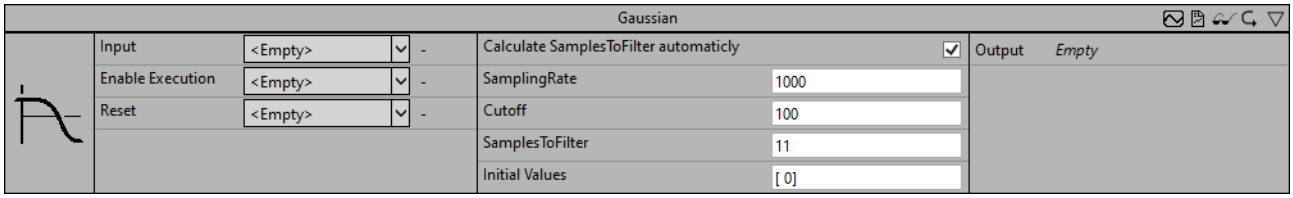
Configuration parameters

- **DeltaMax**: maximum difference between two input values in sequence (greater than or equal to zero).
- **Initial Values (optional)**: initial values define the internal state of the filter.

Output values

- **Output**: manipulated output signal.
- **FilterActive**: indicates which elements of the input signal have been changed.

6.5.11.2.2.14 Gaussian



Implementation of an Infinite Impulse Response filter (IIR).

The documentation of the corresponding PLC function block can be found here: FB_FTR_IIRSpec

The documentation of the TwinCAT 3 Filter PLC library can be found here: Overview

Configuration parameters

- **Calculate SamplesToFilter automatically**: Is a boolean value that indicates whether SamplesToFilter should be calculated automatically.
- **SamplingRate**: sampling rate f_s in Hz (greater than zero).
- **Cutoff**: cut-off frequency in Hz (greater than 0 and less than SamplingRate/2).
- **SamplesToFilter**: number of samples for calculating the sliding average value (often referred to as window size).
- **Initial Values (optional)**: initial values define the internal state of the filter.

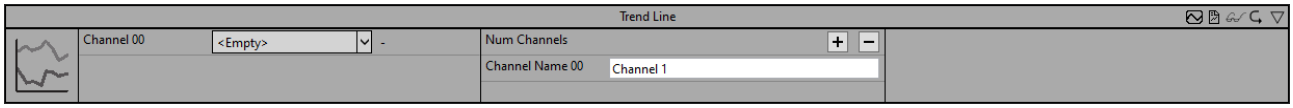
Output values

- **Output**: manipulated output signal.

6.5.12 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.

6.5.12.1 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*.

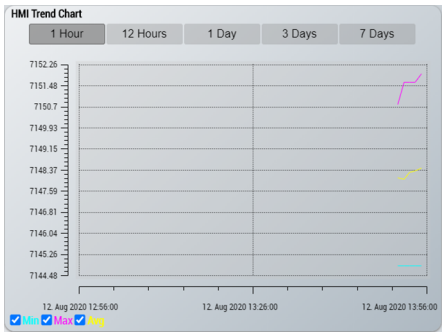
Configuration options

- **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.

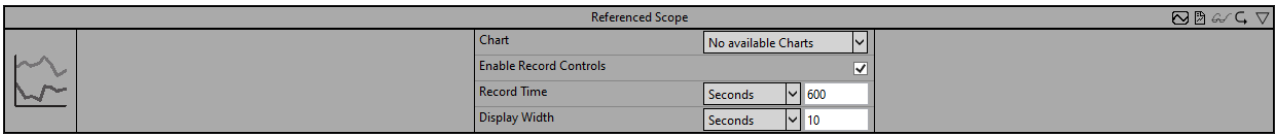
Trend Line to display the data over time.



No other HMI controls can be added to the function block, as it is intended exclusively for displaying the Trend Line Control.

6.5.12.2 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.

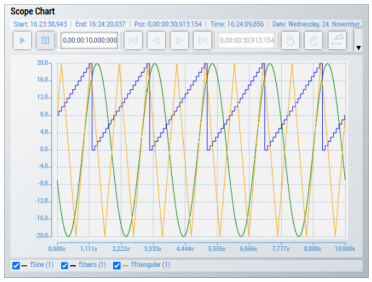
Configuration options

- **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.

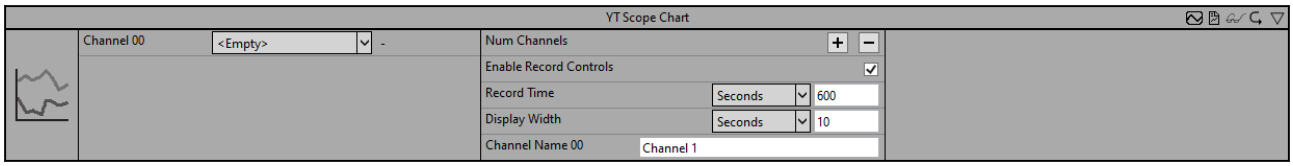
The HMI Scope Control for displaying a Scope Chart.



No other HMI controls can be added to the function block, as it is intended exclusively for displaying the HMI Scope Control.

6.5.12.3 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

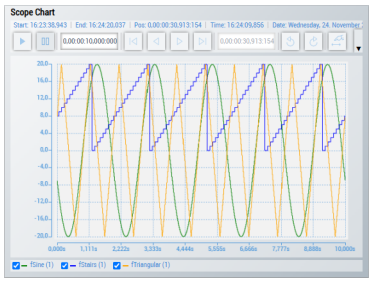
- **Num Channels:** the number of inputs to be displayed in the HMI Scope Control.
- **Channel Name 00/XX:** the display names of the inputs. These appear as legends in the HMI Scope Control.
- **Enable Record Controls:** if the checkbox is activated, a recording can be started and stopped from the HMI.
- **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.

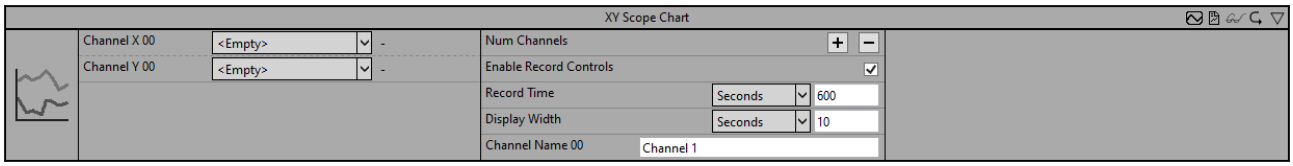
The HMI Scope Control for displaying a Scope Chart.



No other HMI controls can be added to the function block, as it is intended exclusively for displaying the HMI Scope Control.

6.5.12.4 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.



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.

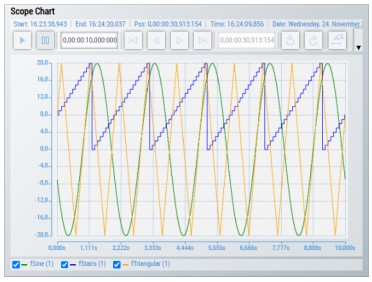
Configuration options

- **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

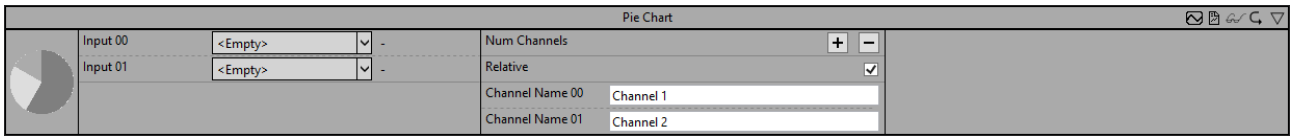
For the algorithm *XY Scope Chart* the following HMI Control is available for generating an Analytics Dashboard.

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.

6.5.12.5 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*.

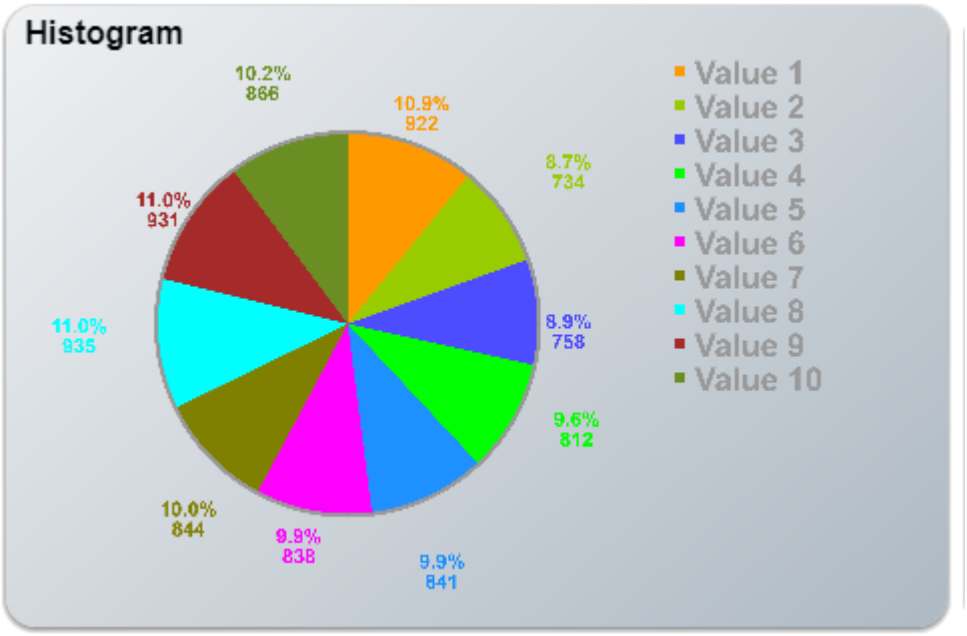
Configuration options

- **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.

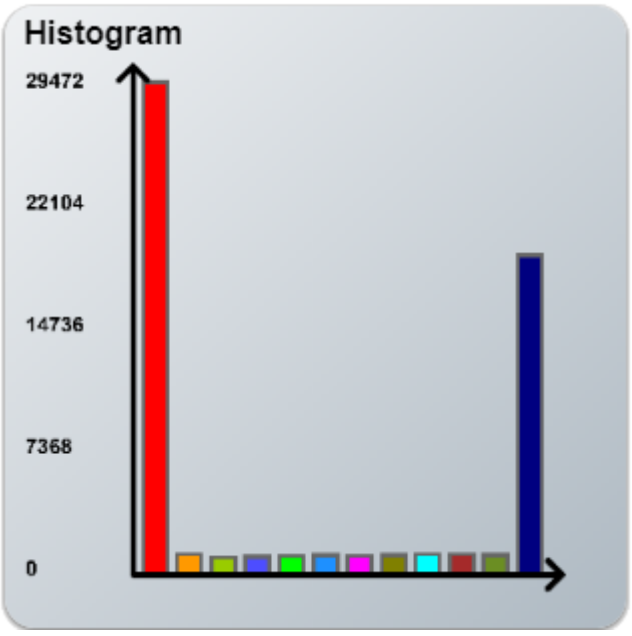
Standard HMI Controls

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.

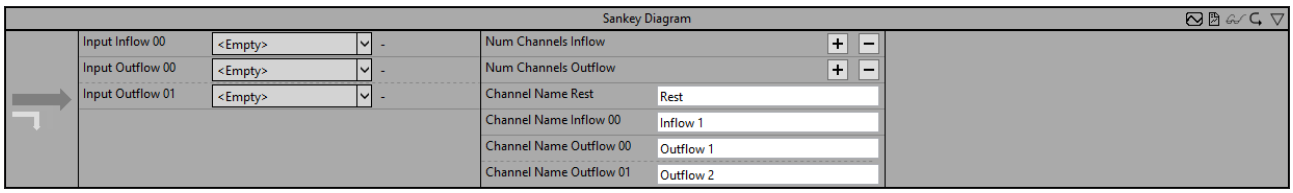


Histogram to display the data in a histogram.



Alternatively, customer-specific HMI controls can be mapped in the *Pie Chart* algorithm using the Mapping Wizard.

6.5.12.6 Sankey diagram



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*.

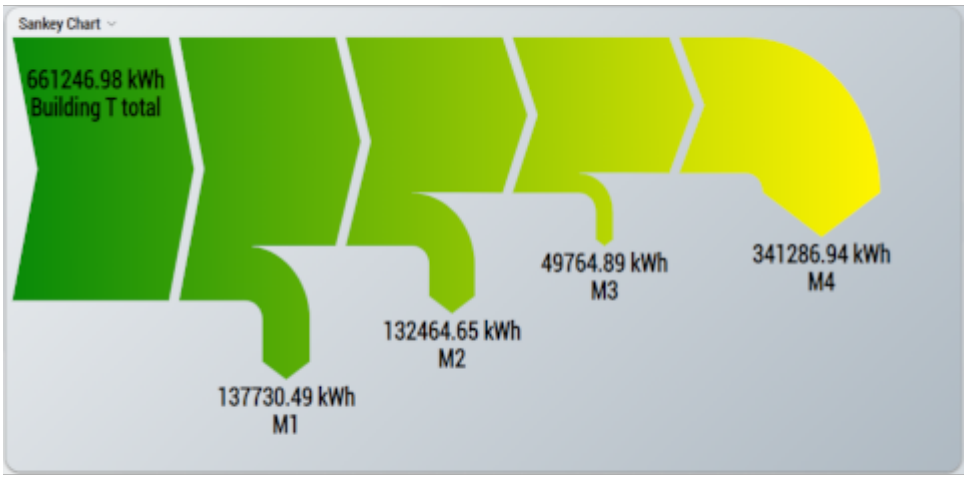
Configuration options

- **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.

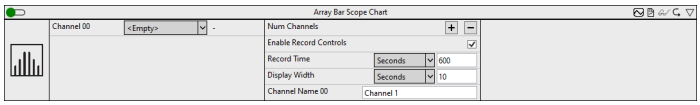
Sankey chart to display the data.



Alternatively, customer-specific HMI controls can be mapped in the *Sankey Diagram* algorithm using the Mapping Wizard.

6.5.12.7 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.

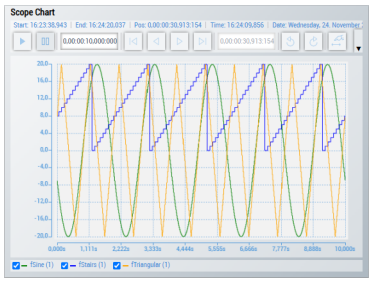
Configuration options

- **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.

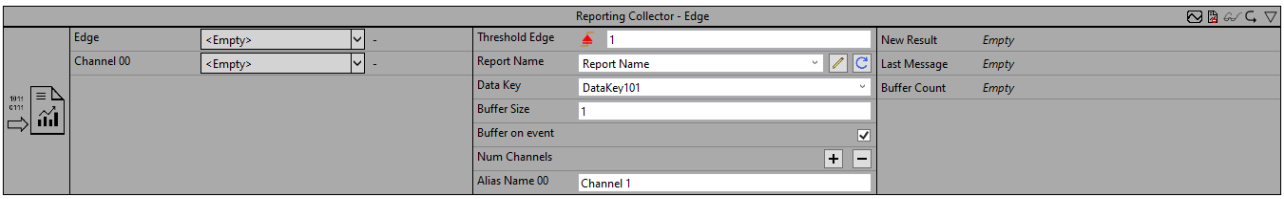
6.5.13 Analytics - Reporting

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.

6.5.13.1 Reporting Collector

The Reporting Collectors collect data and send it to the Reporting Server in a data message after an event.

6.5.13.1.1 Reporting Collector Edge



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*.

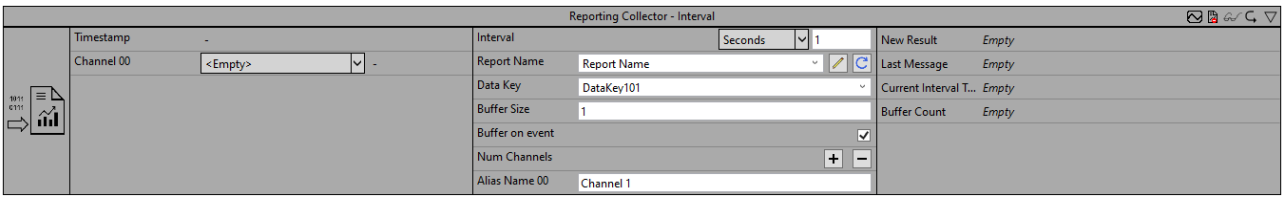
Configuration options

- **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.

6.5.13.1.2 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*.

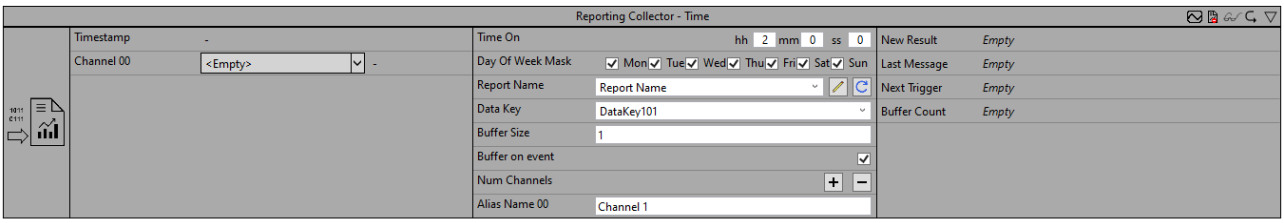
Configuration options

- **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.

6.5.13.1.3 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.

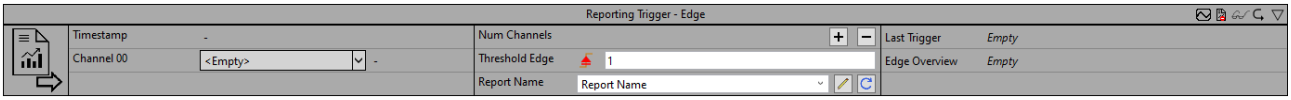
Output values

- **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.

6.5.13.2 Reporting Trigger

The Reporting Triggers send a trigger message to the Reporting Server after an event and thus trigger the creation of a report.

6.5.13.2.1 Reporting Trigger Edge



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*.

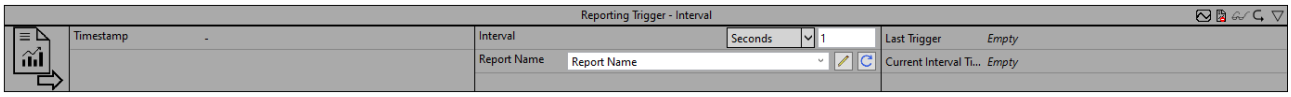
Configuration options

- **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.

6.5.13.2.2 Reporting Trigger Interval



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*.

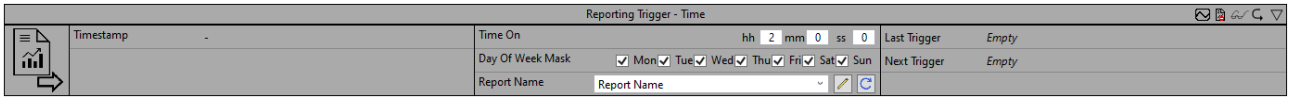
Configuration options

- **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.

6.5.13.2.3 Reporting Trigger Time



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

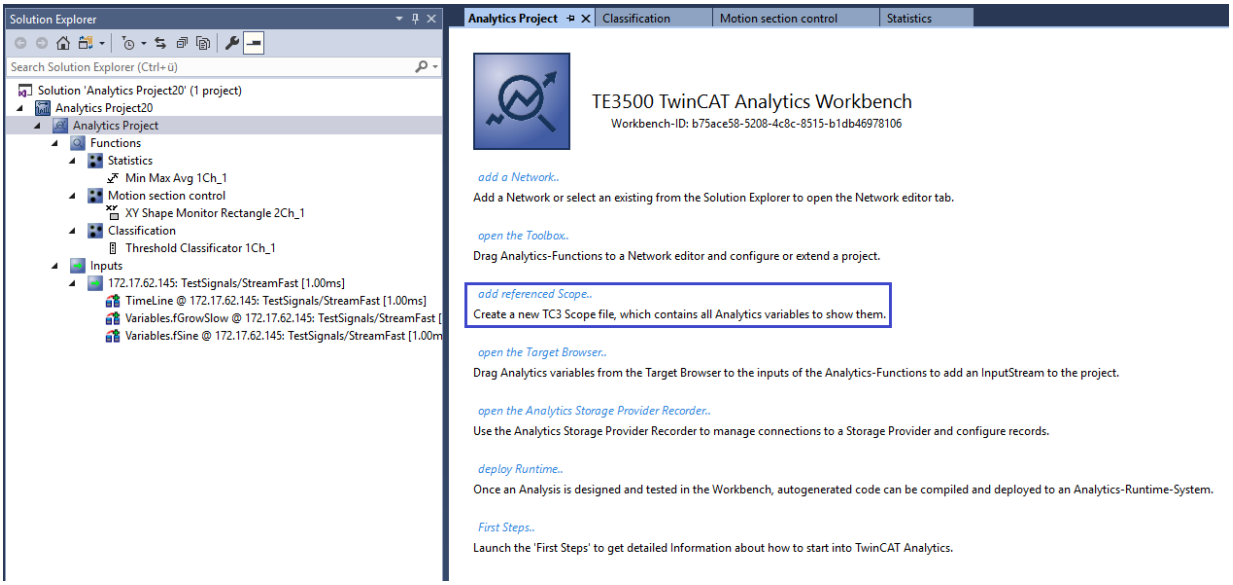
- **Last Message:** indicates when the last message was sent to the Reporting Server.
- **Next Trigger:** indicates the remaining time until the next message.

6.6 Interaction with Scope

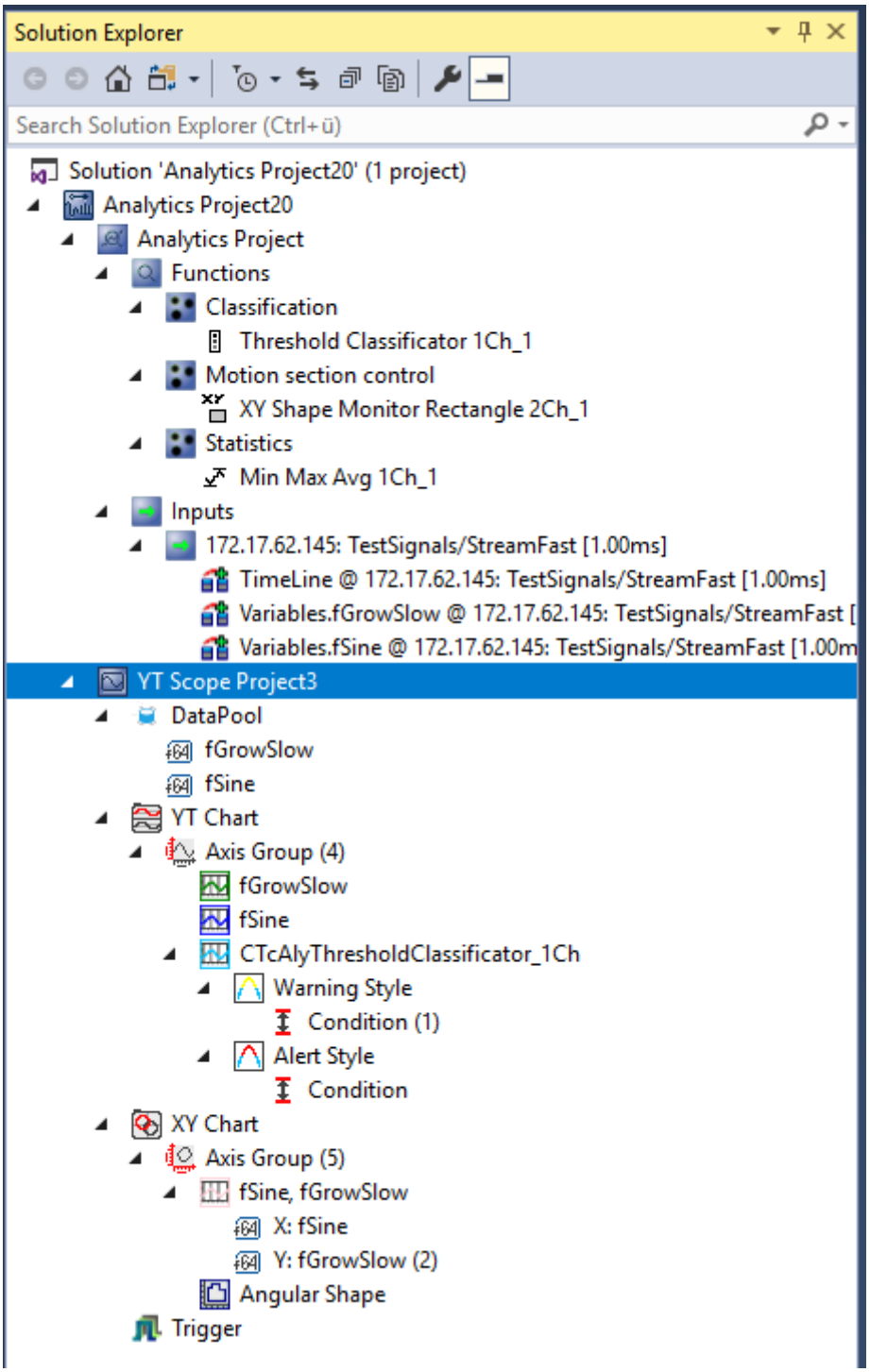
The TwinCAT Analytics engineering tools offer easy interaction between the Analytics Configurator and Scope View. They allow you to highlight significant values in the data stream and examine them with other process data within the exact cycle. In addition, it is possible to display the algorithm's result data, such as an average or maximum value, in the Scope View.

✓ After configuring the analysis, switch to the **Analytics Project** start page.

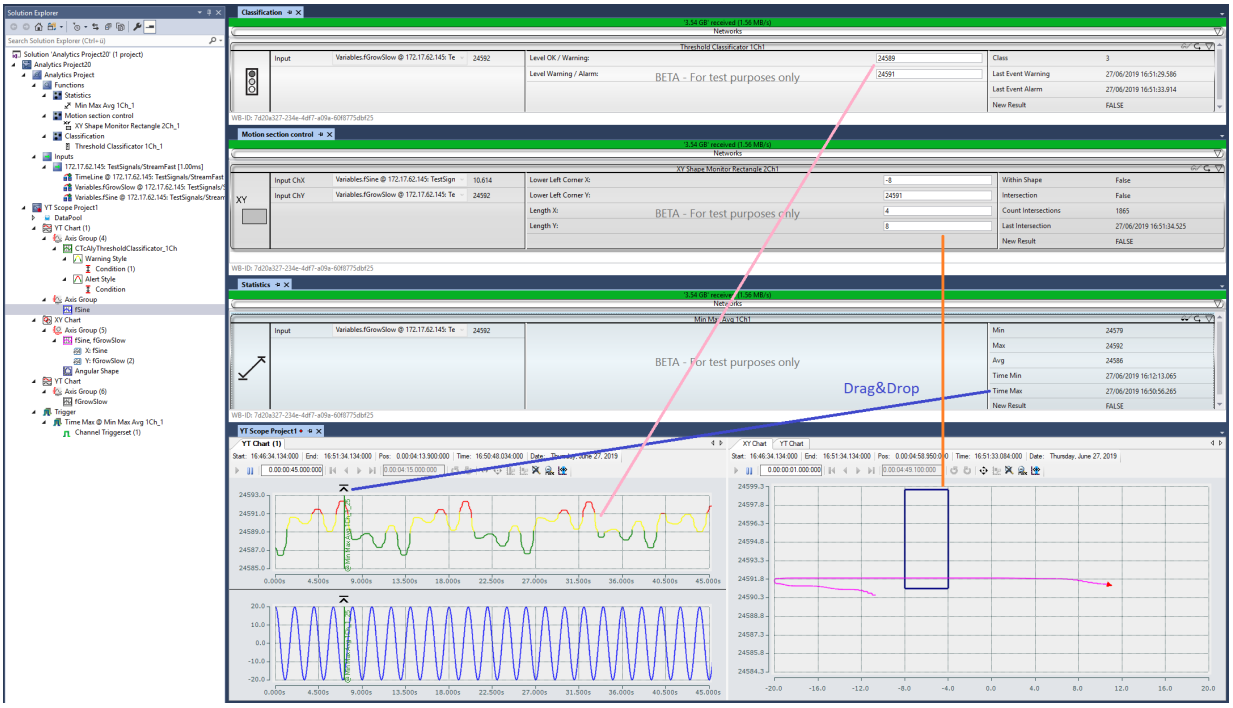
1. Click **add referenced Scope…**



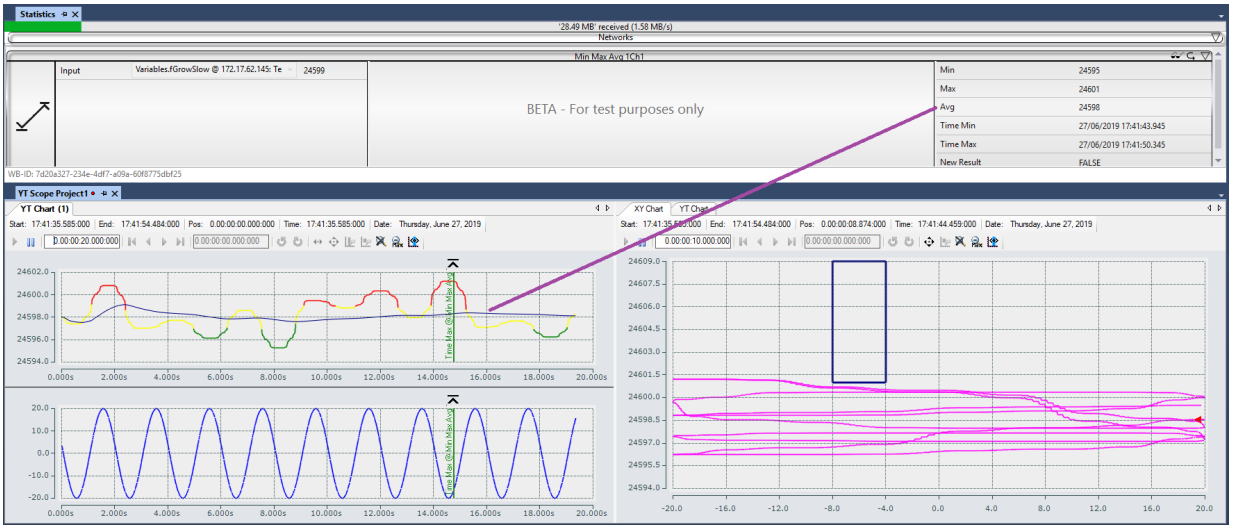
⇨ An appropriate Scope configuration is automatically added to the project.



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).

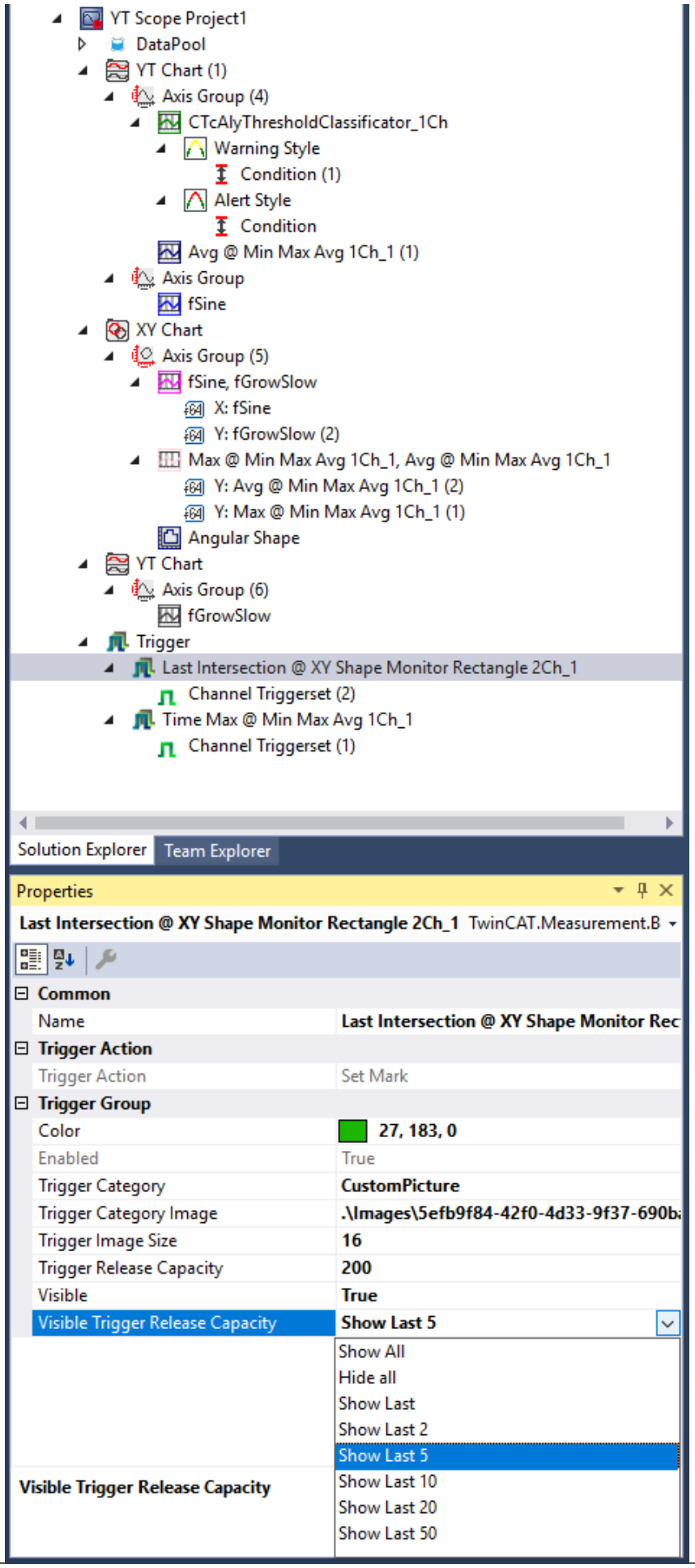


3. For each type of Shape algorithm, the SW creates an XY diagram in the Scope configuration including a shape with the given definition (see orange line in the image above). For the Threshold classification algorithm, a dynamic color change from channel color to yellow (warning) and to red (alarm) (pink 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.



⇨ 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 [▶ 303], 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!

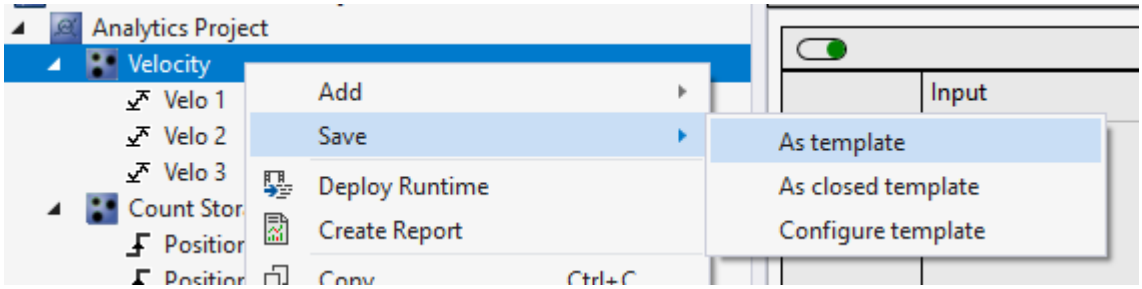
6.6.1 Scope configuration in the network template

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

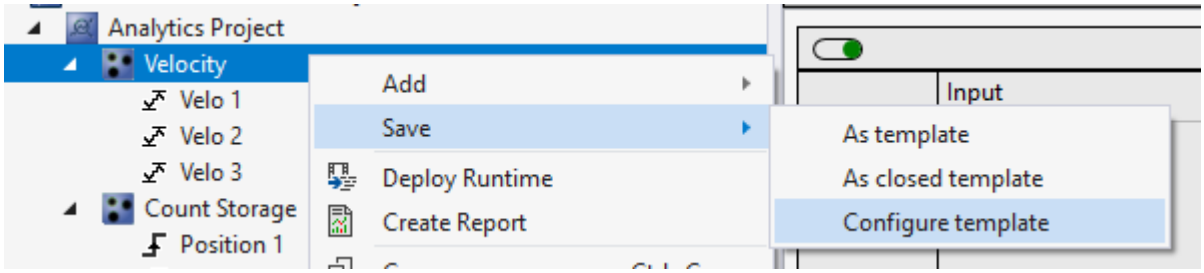
If you want to save the Scope configuration in the template without any manual adjustment, select the context menu items **Save-As template** and **Save-As closed template.**



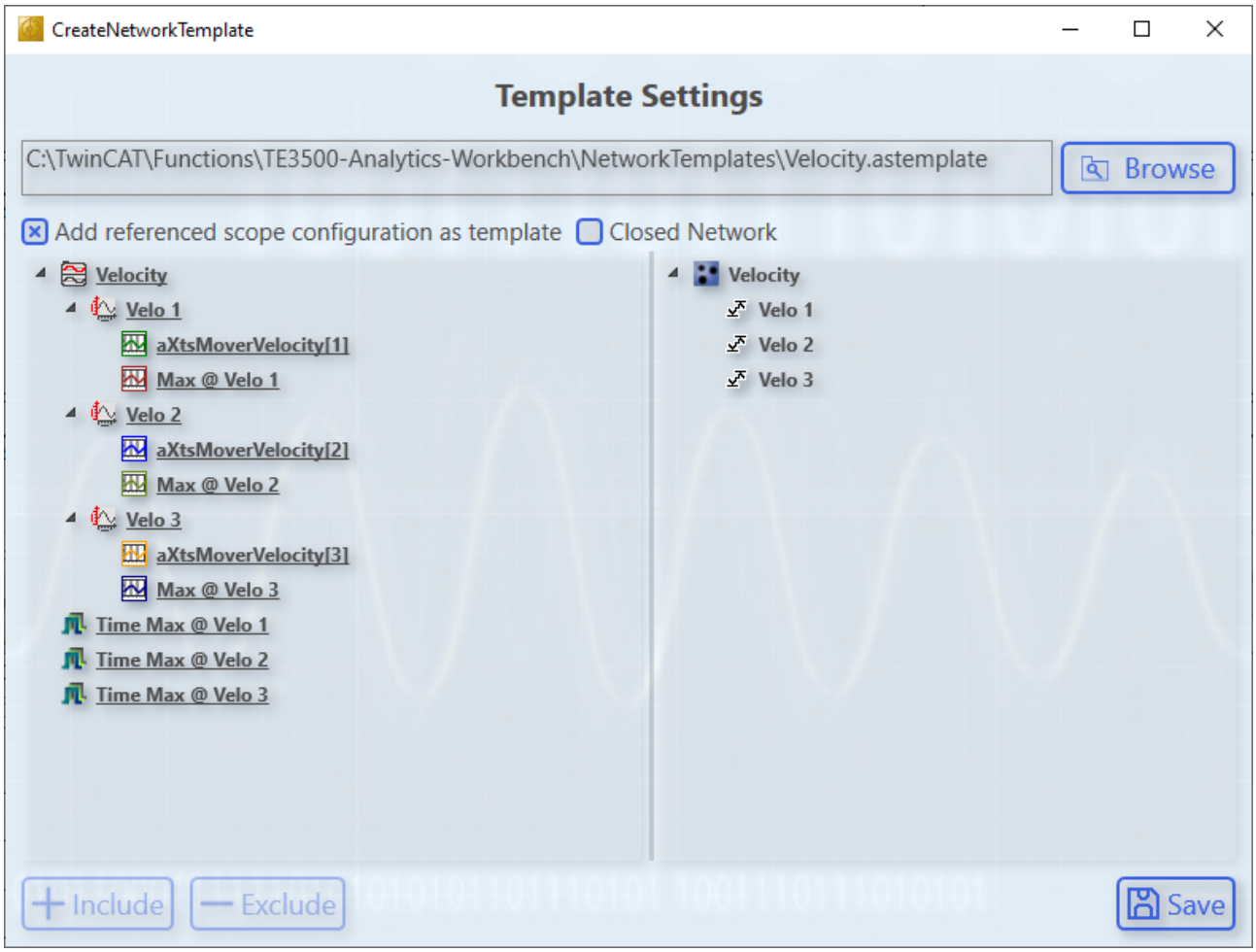
In the following dialog, specify the location and name of the template. At this point you can use the file type **Save as type** to specify whether the Scope configuration should also be saved (**Scope Network Template**) or whether only the network (**Network Template**) should be saved.

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:

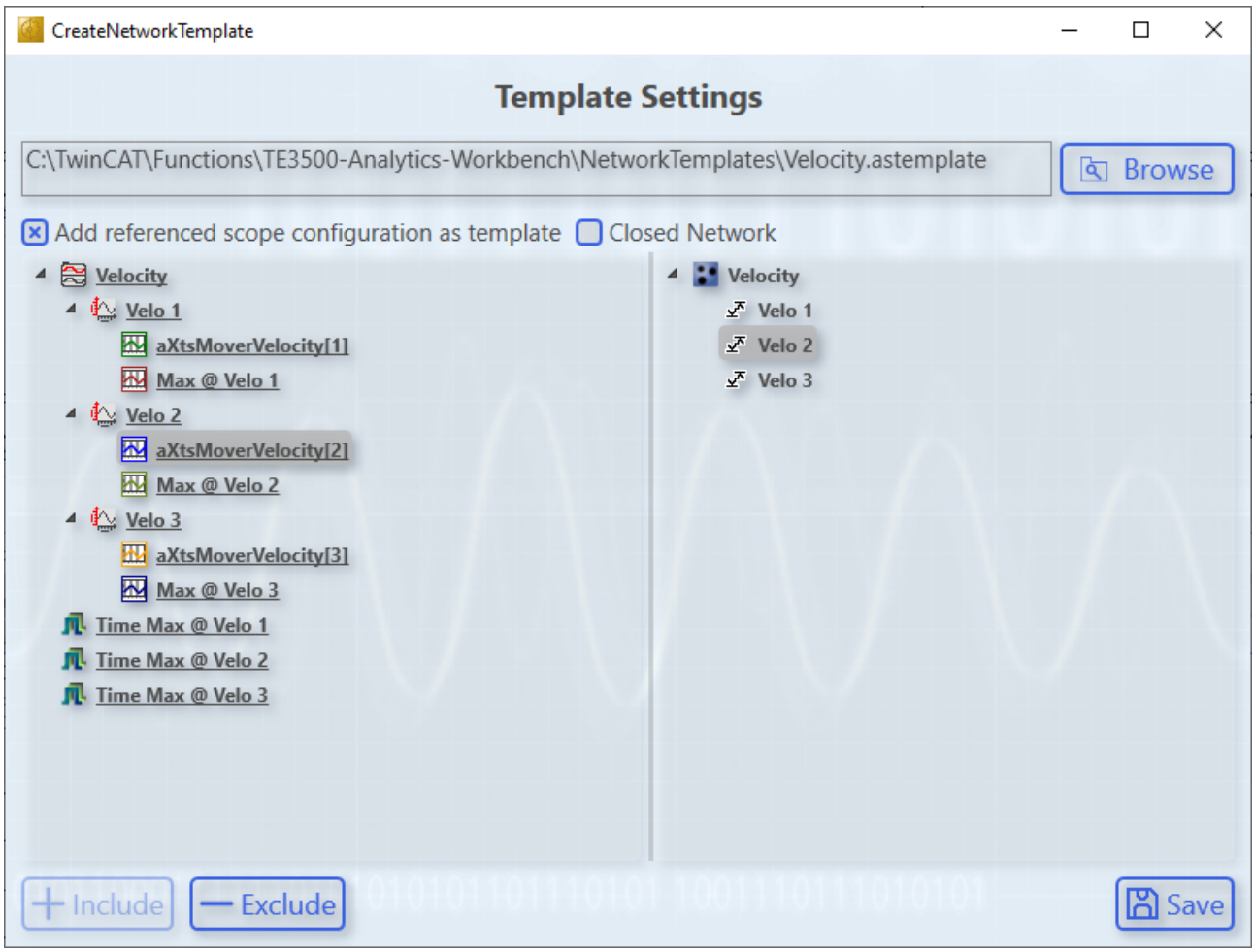


In the upper line you can set the location and name of the template. Directly below you can make general settings that affect the template.

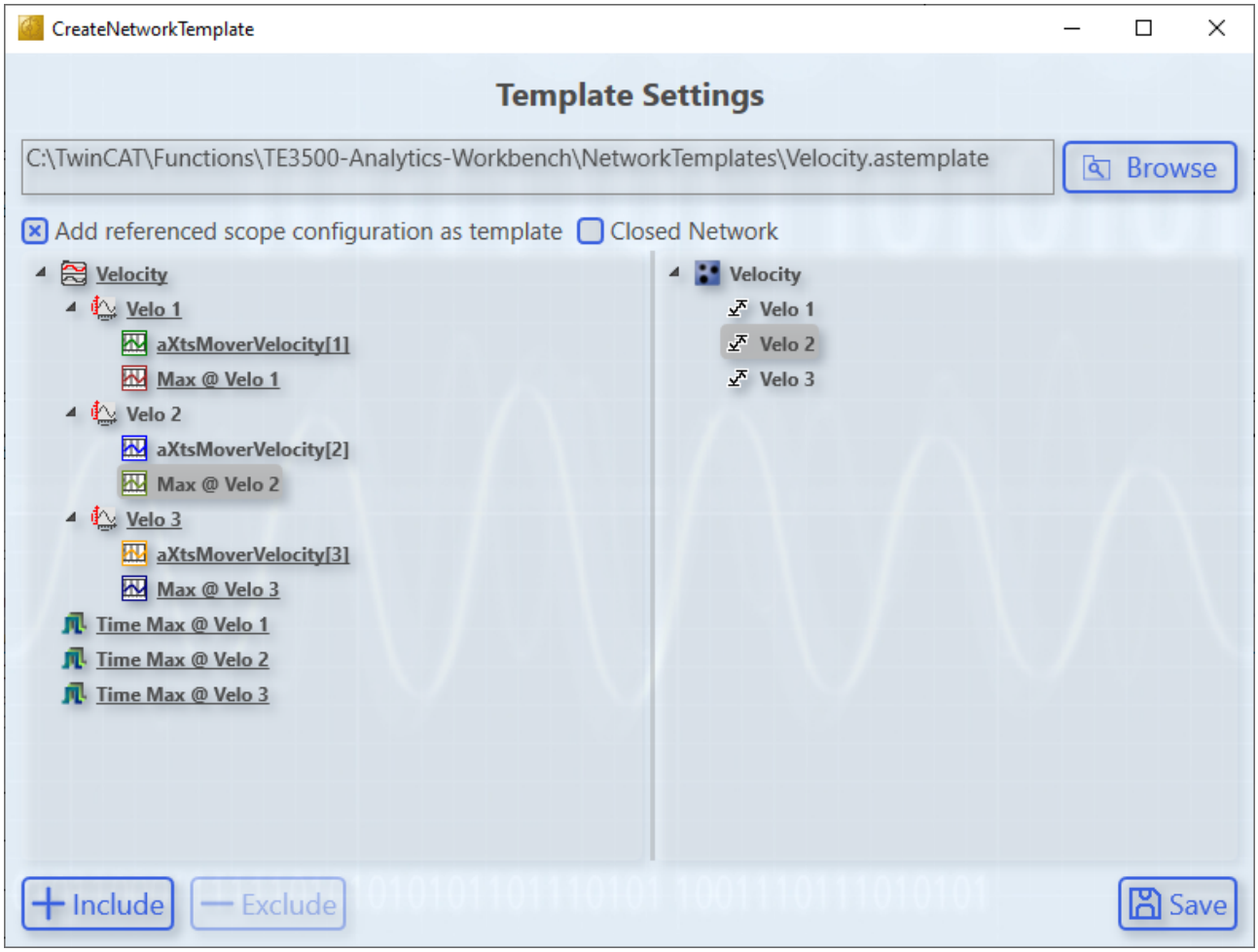
- **Add referenced scope configuration as template**
  Activate this setting to save the scope configuration in the template.
- **Closed Network**
  This setting determines in which format the template [▶ 66] should be saved. This setting is only visible if at least one network input and one network 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.

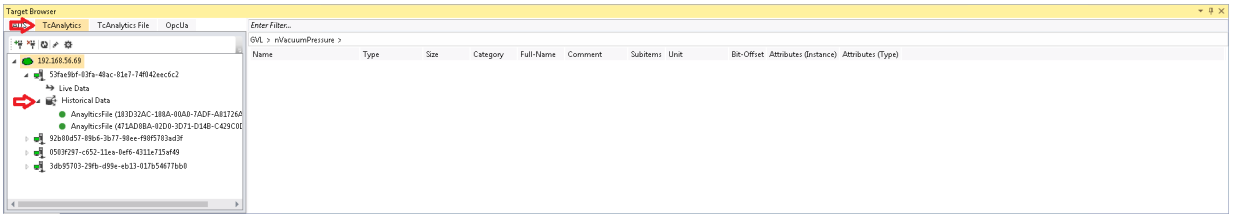
6.7 Working with Historical Data

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.

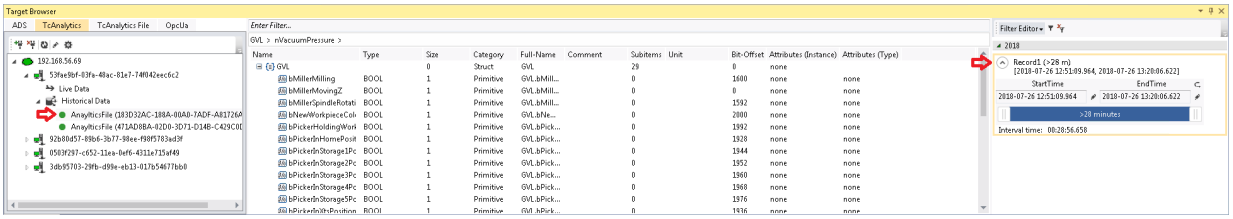
Selection of data from the TwinCAT Target Browser

The historical data can be pulled directly from the Target Browser to an input of an analysis algorithm.

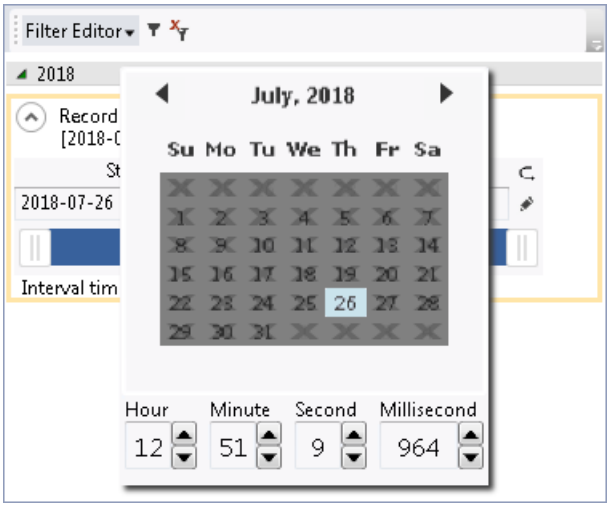
1. First, you need to click **TcAnalytics** in the left corner of Target Browser. There you can see your configured broker, which lists live and historical data from your various devices. This should look like the following figure.



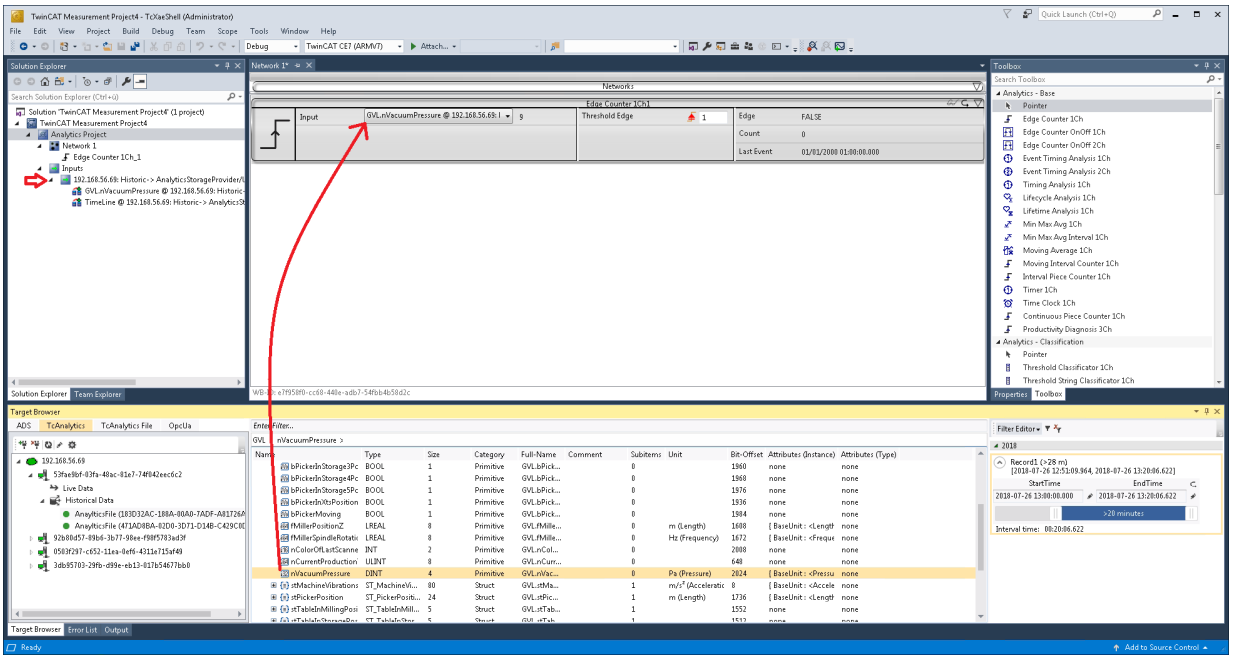
2. Go to the historical stream you created and select the recording to be analyzed. All your records are listed in the **Record** window on the right. The last recording is selected by default.



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.



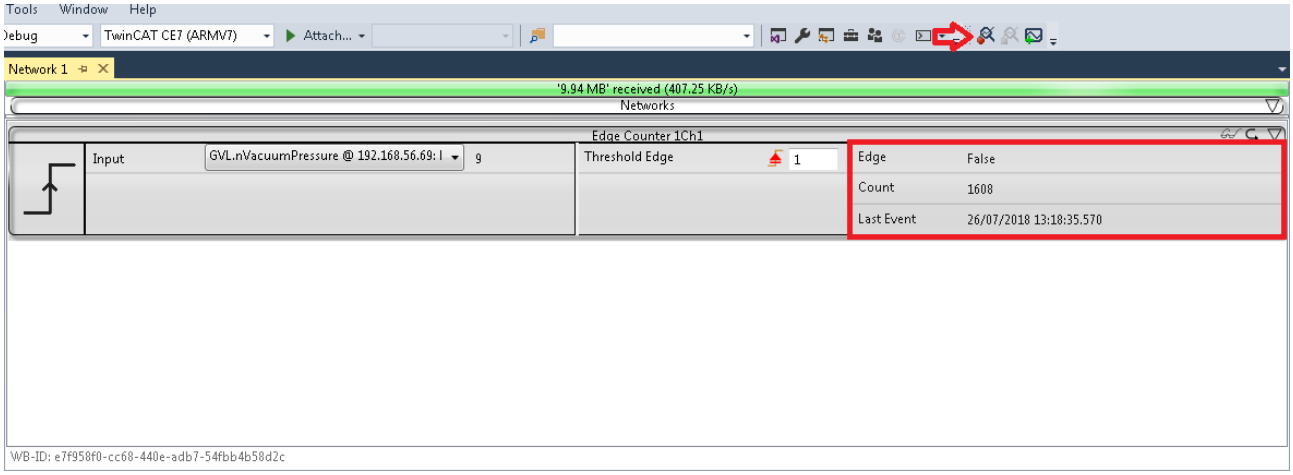
4. After these steps, you can drag and drop a symbol to an input of an algorithm just as you do with the symbols of the live data.



⇨ 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.



7 Appendix

7.1 FAQ - frequently asked questions and answers

In this section frequently asked questions are answered, in order to facilitate your work with the TwinCAT Analytics Service Tool. If you have any further questions, please contact our support team support@beckhoff.com.

1. Is it possible to define historical streams and analyze historical data with the Analytics Service Tool in addition to the Analytics Workbench? [▶ 309]
2. Are open source software components used in TwinCAT Measurement products? [▶ 309]

Is it possible to define historical streams and analyze historical data with the Analytics Service Tool in addition to the Analytics Workbench?

Yes, along with the Analytics Service Tool you will receive the Analytics Storage Provider Recorder. The recorder allows you to define new historical streams. You can also use the historical streams provided by the TwinCAT Target Browser for your analysis.

Are open source software components used in TwinCAT Measurement products?

Yes, various open source components are used. You can find a list including license conditions in the directory *…\TwinCAT\Functions\TwinCAT Measurement\Legal*.

More Information:
www.beckhoff.com/te3520

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com