# Table of contents

1 Foreword ........................................................................................................................................... 5
   1.1 Notes on the documentation ........................................................................................................ 5
   1.2 Safety instructions ......................................................................................................................... 6

2 Overview ........................................................................................................................................... 7

3 Installation ........................................................................................................................................ 9
   3.1 System requirements ...................................................................................................................... 9
   3.2 Installation and licensing ............................................................................................................... 9
   3.3 Licensing ....................................................................................................................................... 12

4 Analytics Workflow - First Steps .................................................................................................... 15
   4.1 Recording data from the machine ................................................................................................. 15
   4.2 Communication ............................................................................................................................. 18
   4.3 Historicize data ............................................................................................................................... 19
   4.4 Analyse data .................................................................................................................................. 24
   4.5 24h Analytics application .............................................................................................................. 28

5 Technical introduction ..................................................................................................................... 38
   5.1 Basic concept ................................................................................................................................ 38

6 Configuration .................................................................................................................................... 40
   6.1 Networks ...................................................................................................................................... 40
      6.1.1 Networks as template ............................................................................................................... 42
   6.2 Data Source .................................................................................................................................. 44
      6.2.1 Wizard .................................................................................................................................... 44
   6.3 Algorithms ................................................................................................................................... 51
      6.3.1 Analytics - Base ...................................................................................................................... 53
      6.3.2 Analytics - Classification ........................................................................................................ 95
      6.3.3 Analytics - Clustering ............................................................................................................ 116
      6.3.4 Analytics - Compare ............................................................................................................... 125
      6.3.5 Analytics - Math ...................................................................................................................... 142
      6.3.6 Analytics - Training Base ....................................................................................................... 153
      6.3.7 Analytics - XTS ....................................................................................................................... 156
      6.3.8 Analytics - WT ......................................................................................................................... 162
      6.3.9 Analytics - XY Path Analysis ................................................................................................. 164
      6.3.10 Analytics - Statistics ............................................................................................................ 174
      6.3.11 Analytics – Extensions with algorithms from other libraries .............................................. 190
      6.3.12 Analytics - Visualization Only ............................................................................................... 191
   6.4 Interaction with Scope .................................................................................................................. 192
   6.5 Runtime deployment .................................................................................................................... 196
      6.5.1 Algorithm properties .............................................................................................................. 202
      6.5.2 PLC Code ............................................................................................................................... 204
   6.6 HMI One Click Dashboard .......................................................................................................... 244
      6.6.1 Dashboard Configuration ....................................................................................................... 247
      6.6.2 Mapping Wizard .................................................................................................................... 254
      6.6.3 Configuration at Runtime ...................................................................................................... 273
      6.6.4 Switching multiple machines in the HMI Dashboard ............................................................. 275
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6.5</td>
<td>Change the language</td>
<td>281</td>
</tr>
<tr>
<td>6.6.6</td>
<td>Dashboard node</td>
<td>283</td>
</tr>
<tr>
<td>7</td>
<td>Samples</td>
<td>294</td>
</tr>
<tr>
<td>7.1</td>
<td>Pick&amp;Place Application</td>
<td>294</td>
</tr>
<tr>
<td>8</td>
<td>Appendix</td>
<td>303</td>
</tr>
<tr>
<td>8.1</td>
<td>FAQ - frequently asked questions and answers</td>
<td>303</td>
</tr>
</tbody>
</table>
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:
with corresponding applications or registrations in various other countries.

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DANGER</td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td>WARNING</td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td>CAUTION</td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td>NOTE</td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer

This symbol indicates information that contributes to better understanding.
## Overview

The TwinCAT 3 Analytics Workbench is a TC3 engineering product for creating continuous data analyses from various decentralized machine controllers. The configuration of the workbench is integrated into Microsoft Visual Studio® and is designed as a graphical user interface. Many algorithms, such as cycle time monitoring, life count, lifetime and minimum/maximum/mean, are available in a toolbox for configuring the analysis.

For simple visualization of the signal curves, the TC3 Analytics Workbench contains the TC3 Scope View Professional TE1300: The user can drag and drop analysis results from the Analytics configurator into the charting tool in order to mark significant points in the data stream. Such markings can be simple minima and maxima, counter readings or also the results of a logical operator that logically combines results from the machine controller so that they can be found in the data stream. This allows correlation with other signals in the Scope View to the exact cycle.

The MQTT input data is selected via the TwinCAT Target Browser, where live data and, via the TF3520 TC3 Analytics Storage Provider, historical data are also available. Once the created analysis is complete and tested in the graphical editor, this configuration can be converted into readable PLC code with an associated HTML5 dashboard in just a few clicks. The automatically generated PLC code and dashboard can be downloaded directly to a device with TF3550 TC3 Analytics Runtime and run there 24/7 in parallel with the actual production machines and provide analysis results. The display via the TwinCAT 3 HMI shows analysis results for machine operators, production managers and/or machine manufacturers, for example.

### Functionality

Compared to the Analytics Service Tool, the Analytics Workbench offers complementary features including code and dashboard generation. All functions of the Service Tool are also included in the Workbench.

### Components

- Analytics configurator
- Base Analytics algorithms
- Analytics PLC library
- Analytics Storage Provider Recorder incl. PLC library
- TwinCAT Scope (TE1300 and TF3300)
- IoT connectivity
- HMI engineering (TE2000)
<table>
<thead>
<tr>
<th>Features</th>
<th>TE3500 Analytics Workbench</th>
<th>TE3520 Analytics Service Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7-day trial</td>
<td>Full license</td>
</tr>
<tr>
<td><strong>General:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis configurator</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Analysis channels</td>
<td>max. 5</td>
<td>unlimited</td>
</tr>
<tr>
<td>Analysis modules/ algorithm</td>
<td>max. 3</td>
<td>unlimited</td>
</tr>
<tr>
<td>Long time records &gt; 1h</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Interaction with Scope View</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Storage Provider Recorder</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>MQTT</td>
<td>✔ (max. 5)</td>
<td>✔</td>
</tr>
<tr>
<td>Analytics File</td>
<td>✔ (max. 5)</td>
<td>✔</td>
</tr>
<tr>
<td>ADS</td>
<td>✔ (no auto deployment)</td>
<td>✔ (no auto deployment)</td>
</tr>
<tr>
<td>Data export tool</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Basic data export formats</td>
<td>✔ (max. 5)</td>
<td>✔</td>
</tr>
<tr>
<td>Extended data export formats</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Deploy Runtime:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy Wizard</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Add PLC Code to existing Sln</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Merge PLC Code</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Auto Create Boot project</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Auto Activate Runtime</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>Stream Results</td>
<td>❌</td>
<td>✔</td>
</tr>
<tr>
<td>HMI Dashboard</td>
<td>✔ (defaults)</td>
<td>✔</td>
</tr>
<tr>
<td>Reset Algorithm in PLC Code</td>
<td>❌</td>
<td>✔</td>
</tr>
</tbody>
</table>

- ✔: Full support
- ❌: No support
- /: No feature of this product
3 Installation

3.1 System requirements

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

**Supported operating systems**

Windows 7, Windows 8.1 and Windows 10

**TwinCAT**

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

**.NET Framework**

For the engineering a .NET Framework 4.6.2 is required.

**Visual Studio development environment**

- Microsoft Visual Studio © 2013 Update 5
- Visual Studio® 2015
- Visual Studio® 2017
- TwinCAT XAE Shell

In general, it is enough to use the Visual Studio® Shell. If you choose the “Full” setup it will install automatically the TwinCAT XAE Shell. The “Update” setup provides just an update of the Analytics sources and no Visual Studio® Shell.

3.2 Installation and licensing

The TwinCAT Analytics setup is part of the TwinCAT Measurement Suite setup. You can only select the Analytics option by clicking **Custom Setup**.
You can enable the option **Beckhoff TE35xx Analytics Engineering**. Depending on the license, the Analytics Workbench or Service Tool function will be provided later.
During the process, the Analytics setup checks whether your system has the necessary licenses. If not, a demo license can be activated. This demo license can be extended as often as you like, but no update of the software/setup is possible on this system. A license must be purchased for this purpose. An overview of functional limitations in the demo version can be found [here](#).
Setup requires license

Updates of TwinCAT Analytics engineering tools are only possible 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 platform level >= 90),
a TwinCAT 3 license dongle is highly recommended, if not a prerequisite for successful licensing of
TwinCAT Analytics!

TwinCAT Analytics Workflow

To use the complete Analytics Workflow you need the TE2000 HMI engineering setup. Today it is
not integrated into the Analytics Workbench setup. This means you must install it individually. It will
be integrated in one of the next setup versions. So, this step will be not necessary anymore.

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").
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

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 installation of the TwinCAT Analytics Storage Provider you are able to configure the service running in the background. You will find therefore the TcAnalyticsStorageProvider_Config application in the folder C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\WinService.
The main part of the topic can be set in the configuration as well as the comment which will be used for identification if more than one Storage Provider is registered at the Message Broker.

You are able to provide the Message Broker settings and to decide for storage type:

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

Finally, you can save the config and start the service. The next step is the configuration of the specific record. Therefore you should choose in your development environment the Storage Provider Recorder.
The recorder has also to connect to the Message Broker. So you have to provide the same settings as for the background service.

After this you can click to icon with the small cloud to search Storage Providers at the configured Message Broker. Here you will find also the comment you gave already by the service configuration to identify your Storage Provider.
The configuration of the record is very easy. You have just to choose your target in the Target Browser. Click on Live data and choose one or more variables by multiselect and put them by drag and drop to the recorder window.

The recorder will ask you if you like to add just the chosen variables or the complete source process image of the variables.
You can also configure record names and a duration (otherwise endless until manual stop). A ringbuffer can be set by memory or time.

Click the Start button to start the record. After this you can also disconnect the recorder, because the background service do the work. It is also possible that someone else connect to this Storage Provider service and control the running record.
After and also during the record you can choose the historical data as input for your analysis in the Target Browser. In the Target Browser you will find for historical data a new control on the right hand site. There you can choose the time span for your data.

4.4 Analyse data

Open your TwinCAT Engineering environment to start the data analysis.

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

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.

When you click on the network, an editor opens. Now you can drag and drop the desired algorithm into the editor interface. 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**
Now select the TcAnalytics or TcAnalyticsFile tab in the Target Browser. We continue with the tab TcAnalytics (MQTT). 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.

Select your MQTT Analytics client (TwinCAT Analytics Logger, TwinCAT IoT Data Agent or Beckhoff EK9160). Each controller has a unique ID. This ID is displayed in the Target Browser. Clicking the gear icon will take you 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.

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.
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.

### 4.5 24h Analytics application

The last big step in the TwinCAT Analytics workflow is the continuous 24h machine analysis. It runs parallel to the machine applications in the field. To do this in a very easy way the TwinCAT Analytics Workbench is able to generate automatically a PLC code and a HTML5-based Dashboard of your Analytics configuration. Both can be downloaded to a TwinCAT Analytics Runtime (TC3 PLC and HMI Server) and provide the same analysis results as the configurator tool in engineering environment.

At first save your configuration and open the Analytics Deploy Runtime wizard. You can do this via the context menu at the Analytics Project tree item or by the Start Page.
When the wizard is open, there are some tabs available for you to click through. First one is called Solution. Here it is possible to decide how you like to use your Analytics project in PLC code: As…

- completely new solution
- part of an existing solution
- update of an existing Analytics solution

In the TwinCAT PLC Target tab you can choose the ADS target system which runs the TwinCAT Analytics Runtime (TF3550). The generated project is immediately executable. Therefore, you can set the option Activate PLC Runtime. Also, that directly a boot project is created.
Specially for Virtual Machines it is important to run the project on isolated cores. Also, an option in this tab. The next tab Results is only necessary if you have chosen the Stream Results option in the algorithm properties. If you like to send results you can decide here in which way (local in a file/ by MQTT) and format (binary/JSON). Also, this is generated automatically and starts running after activation.
A down sampling of the results is possible by setting a cycle time. The next tab is reserved for the HMI Dashboard. The prerequisite for the automatic Dashboard generation is the selection of HMI controls for the corresponding algorithms whose results are to be displayed in the dashboard.
You can choose for your Analytics Dashboard different options like start page with a map, layouts, sorting algorithm, own colors and logos. If you choose more languages for the Analytics controls a language switch menu will be generated as well.
Choose one of the installed Visual Studio versions. And whether the generation should start the instance visibly or whether it should only be set up and activated in the background.
Finally you find a summery.
Now you can click the Deploy button to start the generation process. The PLC project and the HMI Dashboard will be generated now.
After the message Deploy Runtime succeeded you can find a new Visual Studio®/XAE Shell instance on your desktop. The new Solution and both Projects are generated.
5 Technical introduction

5.1 Basic concept

The following picture shows the basic concept of TwinCAT Analytics from the data source up to the Analytics dashboard based on TwinCAT 3 HMI. The communication in an Analytics scenario is realized by the IoT communication protocol MQTT.

Data sources:

Currently there are three different data sources for TwinCAT Analytics. All these sources are able to communicate with TwinCAT Analytics specific binary data format. This format is necessary to reach this high performance.

- TwinCAT 3 controller with TF3500 TwinCAT Analytics Logger
- TwinCAT 2, TwinCAT 3 and third party controller together with a gateway of TF6720 TwinCAT IoT Data Agent
- All EK9160 IoT Coupler devices

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 realize 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 make an automatic PLC code generation based on the realized analytics configuration in the configurator. The PLC code is ready to use. So you can start the data analysis immediately like in the configurator. But now for 24 hours 7 days per week if necessary. The auto generated code can be downloaded into the TF3550 TwinCAT Analytics Runtime. This runtime can be on a classic IPC or Embedded-PC, but also in a virtual machine. In the generated PLC project, the user is also able to realize his Analytics Dashboard by the TwinCAT 3 HMI.

Products:

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

<table>
<thead>
<tr>
<th>Product Number</th>
<th>Product Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TE3500</td>
<td>Analytics Workbench</td>
</tr>
<tr>
<td>TE3520</td>
<td>Analytics Service Tool</td>
</tr>
<tr>
<td>TF3500</td>
<td>Analytics Logger</td>
</tr>
<tr>
<td>TF3510</td>
<td>Analytics Library</td>
</tr>
<tr>
<td>TF3520</td>
<td>Analytics Storage Provider</td>
</tr>
<tr>
<td>TF3550</td>
<td>Analytics Runtime - including HMI Server and Client Pack</td>
</tr>
<tr>
<td>TF3551</td>
<td>Analytics Runtime Base – without HMI</td>
</tr>
<tr>
<td>TF3560</td>
<td>Analytics Controller Pack 4</td>
</tr>
<tr>
<td>TF3561</td>
<td>Analytics Controller Pack 8</td>
</tr>
<tr>
<td>TF3562</td>
<td>Analytics Controller Pack 16</td>
</tr>
<tr>
<td>TF3563</td>
<td>Analytics Controller Pack 32</td>
</tr>
<tr>
<td>TF3564</td>
<td>Analytics Controller Pack 64</td>
</tr>
<tr>
<td>TF3565</td>
<td>Analytics Controller Pack 128</td>
</tr>
<tr>
<td>TF6720</td>
<td>IoT Data Agent</td>
</tr>
<tr>
<td>EK9160</td>
<td>IoT Coupler</td>
</tr>
</tbody>
</table>

The minimal meaningful configuration is the usage of TF3500/TF6720/EK9160 together with TE3520 Analytics Service Tool. Also possible is just to use the TF3510 Analytics Library in a TwinCAT system.
6 Configuration

If you want to create an Analytics configuration, you will start at the Analytics Project start page, which you can see in the following screenshot.

From this start page you have different possibilities to go further as described below the respective option.

6.1 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.
6.1.1 **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 first have to select which of the available inputs should be visible on the network from outside. 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 SPS code generation by the Analytics Workbench.

Once the template has been saved, it can be selected in the toolbox and can be used for recurring analyses.
The Target Browser offers a special function. If the names for the network have been chosen in such a way that they correspond exactly to variables of a structure, you can drag a complete structure such as an axis structure directly onto one of the x inputs of a network. All matching names of all network inputs and structure variables are mapped automatically.

6.2 Data Source

6.2.1 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.

### 6.3 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 toolbox is empty, select the Analytics project to see the algorithms.
Currently there are ten 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 and Analytics-Statistics. In addition, it is possible to use algorithms from other libraries that are included in the Analytics toolbox (see Analytics – Extensions with algorithms from other libraries [190]). The different algorithm groups are described below.

Each algorithm has the same four icons in the upper right corner:
• **Include in Referenced Scope**: Clicking the *Included in Referenced Scope* icon adds the algorithm outputs to a referenced scope project.

• **Eyeglasses**: If you click on the eyeglasses icon, you can see the optional parameter *Enable Execution*. For this parameter you can select a boolean signal so that the algorithm is only active if the value of the selected signal is *TRUE*.

• **Reset arrow**: Clicking the arrow resets the output values of the specific algorithm.

• **Minimize arrow**: Click the minimize arrow on the right to collapse the algorithm.

### 6.3.1 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.

#### 6.3.1.1 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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 Piece Counter 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.

![Pie Chart](image)

**Number of intervals:** 4  
**Interval time:** 00:00:24

- Max: 10. Mar 2021 16:15:29

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.

![Table Control](image)

**DataTable Vertical**

<table>
<thead>
<tr>
<th></th>
<th>Data Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 1</td>
<td>20.42</td>
</tr>
<tr>
<td>Input 2</td>
<td>25.43</td>
</tr>
<tr>
<td>Executing</td>
<td>TRUE</td>
</tr>
<tr>
<td>Last Event</td>
<td>10 Mar 2021 16:15:29</td>
</tr>
</tbody>
</table>

**DataTable Horizontal**

<table>
<thead>
<tr>
<th></th>
<th>Input 1</th>
<th>Input 2</th>
<th>Executing</th>
<th>Last Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Table</td>
<td>20.42</td>
<td>25.43</td>
<td>TRUE</td>
<td>10 Mar 2021 16:15:29</td>
</tr>
</tbody>
</table>
Alternatively, customer-specific HMI controls can be mapped in the Continuous Piece Counter 1Ch algorithm using the Mapping Wizard [254].

6.3.1.2 Downsampling Buffer 1Ch

The Downsampling Buffer 1Ch buffers values of the input channel with a configurable down sampling factor. The size of the buffer, which corresponds to the number of output channels, is configurable as well.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

Configuration Options

- **Buffer Size**: Indicates the number of values that are stored and returned as output. The number of output channels equals the buffer size.
- **Downsampling Factor**: The number of cycles that pass between two values being saved to the buffer. If the Downsampling Factor is set to one, all values are buffered.
- **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 (each down sampling factor cycles).
  - **Wait**: At the start of the analysis or after reset all output channels are set to zero. Only after 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 down sampling buffer, according to the sample mode.
6.3.1.3  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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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**: Shows TRUE in the moment the event is raised, otherwise FALSE.
- **Count**: Counts the amount of raised events.
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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 [254].
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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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**: Shows TRUE within the time range between on-event and off-event, otherwise FALSE.
- **Edge On**: Shows TRUE if there is a raising edge.
- **Edge Off**: Shows TRUE if there is a falling edge.
- **Count On**: Counts the amount of raised on-events.
- **Count Off**: Counts the amount of raised off-events.
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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 Binary State 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 1Ch algorithm using the Mapping Wizard [254].

### 6.3.1.5 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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**: Shows *TRUE* within the time range between on-event and off-event, otherwise *FALSE*.
- **Edge On**: Shows *TRUE* if there is a raising edge.
- **Edge Off**: Shows *TRUE* if there is a falling edge.
- **Count On**: Counts the amount of raised on-events.
- **Count Off**: Counts the amount of raised off-events.
• **Last Event:** Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.

![Edge Counter OnOff](image1)

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

![Single Value](image2)

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

![Binary State](image3)

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 [p. 254].
**6.3.1.6 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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.
- **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 1Ch algorithm, the following HMI controls are available for generating an Analytics Dashboard:

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 1Ch algorithm using the Mapping Wizard. 

**6.3.1.7 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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:


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 [1, 254].

**6.3.1.8 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.

Optional a boolean signal for **Enable Execution** can be selected, so that the algorithm is just 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**: Shows **True**, if the calculation is active and the interval is running, otherwise **False**.
- **Num Intervals**: Shows the number of Intervals.
- **Count Last Interval**: Shows the amount of raised events in the last interval.
• **Count Current Interval**: Shows the amount of raised events in the current interval or if the calculation is currently inactive, the amount of raised events in the last interval.

• **Count Min**: Shows the minimum of raised events in an interval.

• **Count Max**: Shows the maximum of raised events in an interval.

• **Time Count Min**: Shows the time value of the minimum → this event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.

• **Time Count Max**: Shows the time value of the maximum → this event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.

• **Current Interval Time**: Shows the time of the current interval → this event can be pulled into the scope chart by Drag & Drop to show 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 [254].

6.3.1.9 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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.

![Process](image1)

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

![Data Table Vertical](image2)
Alternatively, customer-specific HMI controls can be mapped in the Lifecycle Analysis 1Ch algorithm using the Mapping Wizard [p. 254].

### 6.3.1.10 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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 [254].

### 6.3.1.11 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

#### Output Values

- **Min**: Shows the minimum of the input values.
- **Max**: Shows the maximum of the input values.
- **Avg**: Shows the average of the input values.
- **Time Min**: Shows the time value of the minimum → this event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.
- **Time Max**: Shows the time value of the maximum → this event can be pulled into the scope chart by Drag & Drop to show 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.
Alternatively, customer-specific HMI controls can be mapped in the Min Max Avg 1Ch algorithm using the Mapping Wizard [254].
6.3.1.12 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

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

Output Values
  - Min: Shows the minimum of the input values in the last time interval.
  - Max: Shows the maximum of the input values in the last time interval.
  - Avg: Shows the average of the input values in the last time interval.
  - Time Min: Shows the time value of the minimum in the last time interval → this event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.
  - Time Max: Shows the time value of the maximum in the last time interval → this event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.
  - Current Interval Time: Shows the timespan, that already proceeded 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.

![Energy Monitoring](image)

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

![Thermometer](image)

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

![Data Table Vertical](image)
Alternatively, customer-specific HMI controls can be mapped in the Min Max Avg Interval 1Ch algorithm using the Mapping Wizard [254].

6.3.1.13 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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.

![Min Max Avg](image1)

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

![Energy Monitoring](image2)

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.

![Thermometer Image]

Data Table Vertical

<table>
<thead>
<tr>
<th>Data Table</th>
<th>Input 1</th>
<th>Input 2</th>
<th>Executing</th>
<th>Last Event</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20.42</td>
<td>25.43</td>
<td>TRUE</td>
<td>10 Mar 2021 16:15:29</td>
</tr>
</tbody>
</table>

DataTable Horizontal

<table>
<thead>
<tr>
<th>Data Table</th>
<th>Input 1</th>
<th>Input 2</th>
<th>Executing</th>
<th>Last Event</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20.42</td>
<td>25.43</td>
<td>TRUE</td>
<td>10 Mar 2021 16:15:29</td>
</tr>
</tbody>
</table>
Alternatively, customer-specific HMI controls can be mapped in the Moving Average 1Ch algorithm using the Mapping Wizard [254].

### 6.3.1.14 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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**: Shows *TRUE* in the moment the event is raised, otherwise *FALSE*.
- **Limited**: Shows TRUE if the number of edges in the current interval exceeds the set Count Limit.
- **Counts in Interval**: Shows the amount of raised events in the current interval.
- **Time First Count**: Indicates the point of time of the first raised event → the event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.
- **Time Last Count**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.

![Moving Interval Counter](image1)

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

![Data Table Vertical](image2)

![Data Table Horizontal](image3)
Alternatively, customer-specific HMI controls can be mapped in the Moving Interval Counter 1Ch algorithm using the Mapping Wizard. 

6.3.1.15 **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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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.
Alternatively, customer-specific HMI controls can be mapped in the Productivity Diagnosis 3Ch algorithm using the Mapping Wizard [254].
6.3.1.16  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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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 [254].

6.3.1.17 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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.

6.3.1.18 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 timer 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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 [p 254].
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).

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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.

![Timer](image)

2. The Table Control or Multivalue Control visualizes all output values: Output, Elapsed Time.
Alternatively, customer-specific HMI controls can be mapped in the Timer 1Ch algorithm using the Mapping Wizard [254].
6.3.1.20 **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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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.

![Timing Analysis](image)

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

![Single Value](image)

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 [p. 254].

6.3.2 Analytics - Classification

The algorithms of the category *Analytics-Classification* provide functionalities for classification and state detection.
6.3.2.1 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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**: Shows the class to which the input values belong to (WithinBounds / Smaller / Bigger).
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.

![Bandwidth Classifier](image)

2. The MultiState control visualizes the output value Class.

![Multi State](image)

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 [254].

### 6.3.2.2 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

Output Values
- **Class**: Shows the class to which the input values belong to (WithinBounds / Smaller / Bigger).
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.

   ![Bandwidth Classifier](image)

   Last event: 10. Mar 2021 16:18:37

2. The MultiState control visualizes the output value Class.

   ![Multi State](image)

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 [254].
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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 Threshold Reversal value. Below you can see some examples of peak identification with the respective Threshold Reversal value. In diagram (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. In diagram (b) \( y_3 \) is not identified as a peak because the difference between \( y_3 \) and \( y_4 \) is less than Threshold Reversal and the curve rises again after \( y_4 \). In diagram (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.

- **Calculate Inflection**: boolean flag. Maxima of the rate of change are only identified if this flag is True. Otherwise, the values for Max Change Value, Max Change (Rate), Max Change (Timestamp) and Count Max Change are not calculated.

- **Threshold Delta**: threshold for identifying Delta maxima. Maxima of the absolute difference of two successive values (Delta) are only detected if the difference between successive Deltas exceeds Threshold Delta. Below you can see some examples of the maximum Delta identification with the respective Threshold Delta. The upper diagrams show the original input signals, the lower graphs show the corresponding Delta. In diagram (a) \( y_4 \) is identified as the maximum after processing the value \( y_5 \) because the difference of the two Deltas exceeds Threshold Delta. In diagram (b) no maximum is detected because the difference between the Deltas is less than Threshold Delta. In diagram (c) a maximum \( y_3 \) is detected only after the value \( y_5 \) has been processed.

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.
Alternatively, customer-specific HMI controls can be mapped in the Curve Sketcher 1Ch algorithm using the Mapping Wizard [254].

6.3.2.4 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

Configuration options

- **Min Binned**: minimum value to be analyzed.
- **Max Binned**: maximum value to be analyzed.
- **Bins**: total number of histogram classes to be calculated.

Output Values

- **Num Values**: Shows the total amount of analyzed values for the distribution.
- **Histogram**: Below the Num Values the Histogram is shown. If you move the cursor over the bars, you can see a tooltip with the value interval and the related amount of values that lies in the interval.

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 [254].
6.3.2.5 **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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just active, if the value of the selected signal is *TRUE*.

### Configuration Options

- **Num Sections**: This is the amount 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.

### 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 on*, 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 [P 254].

6.3.2.6 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

Configuration options
- **Hist Mode**: operating mode of the histogram. You can choose between eAbs for absolute values and eRel for relative values to display the percentage distribution.
- **Min**: minimum value to be analyzed.
- **Max**: maximum value to be analyzed.

Output Values
- **Num Values**: Shows the whole amount of executed states between the configured boarders.
- **Histogram**: Below the Num Values the Histogram is shown. On each bar the respective value is displayed.

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.

![Pie Chart](image1)

2. The StateHistogram control visualizes the output value histogram.

![State Chart](image2)

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

![Data Table Vertical](image3)
Alternatively, customer-specific HMI controls can be mapped in the State Histogram 1Ch algorithm using the Mapping Wizard [254].

### 6.3.2.7 Threshold Classifier 1Ch

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

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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**: Shows the class to which the input values belong to.

- **Last Event Warning**: Indicates the point of time of the last raised warning event → the event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.
• **Last Event Alarm**: Indicates the point of time of the last raised alarm event → the event can be pulled into the scope chart by Drag & Drop to show 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.

![Traffic Light](image1)

- **Warning Value**: 2.20
- **Last alarm**: 10. Mar 2021 16:19:25
- **Last warning**: 10. Mar 2021 16:19:25

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

![Multi State](image2)

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

![Table Control](image3)
Alternatively, customer-specific HMI controls can be mapped in the Threshold Classifier 1Ch algorithm using the Mapping Wizard.
6.3.2.8 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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**: Shows the class to which the input values belong to.
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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 Table Control or Multivalue Control visualizes all output values: Output String, Last Event.
Alternatively, customer-specific HMI controls can be mapped in the Threshold String Classifier 1Ch algorithm using the Mapping Wizard [254].

### 6.3.2.9 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* [154], for example. The comparison starts when the signal of the Start Period flag is *TRUE*. It is recommended that you do not use Time Based Envelope 1Ch at the same time as *Time Based Teach Path 1Ch* [154] due to competing file access. Instead, a reference signal should first be taught using *Time Based Teach Path 1Ch* [154]; only then should the evaluation be carried out using *Time Based Envelope 1Ch*.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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 Comparison**: Displays *TRUE* when 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.

![Time Based Envelope](image)

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.
Alternatively, customer-specific HMI controls can be mapped in the Time Based Envelope algorithm using the Mapping Wizard [254].

6.3.3 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.
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 (marked in red) contains six data points, the second (marked in green) also contains six, the third (marked in blue) contains seven and the fourth (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 \( \text{Beta} \times \text{Mu} \) 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 \( \text{Beta} \times \text{Mu} = 1 \).

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

The parameters \( \text{Epsilon (DBSCAN)} \) and \( \text{Min Weight (DBSCAN)} \) 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 \( \text{Epsilon (DBSCAN)} \). All micro-clusters forming this chain must satisfy a second condition. The sum of the weights of all micro-clusters within the distance \( \text{Epsilon (DBSCAN)} \) around each individual micro-cluster in this chain must exceed the threshold \( \text{Min Weight (DBSCAN)} \). 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 \( \text{Min Weight (DBSCAN)} \) 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 \( \text{Epsilon (DBSCAN)} \) to these micro-clusters. The micro-clusters marked with a small "x" or "+" are not core micro-clusters, but are located in the \( \text{Epsilon (DBSCAN)} \) 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 \( \text{Epsilon (DBSCAN)} \) 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 \( \text{Epsilon (DBSCAN)} \) neighborhood, there are only two of them. The \( \text{Min Weight (DBSCAN)} \) 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.


Optional a boolean signal for `Enable Execution` can be selected, so that the algorithm is just active, if the value of the selected signal is `TRUE`.

**Input values**

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

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

Output values

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

Standard HMI Controls

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

1. The DenStream control visualizes the inputs in the chart and their respective classification in a cluster (cluster index) in color. The buttons can be used to select the inputs to be displayed, up to two at a time. The slider on the right lists all output values: cluster count, number of potential micro-clusters, number of outlier micro-clusters, cluster index, last event, last switch.
2. The Table Control or Multivalue Control visualizes all output values: cluster count, number of potential micro-clusters, number of outlier micro-clusters, cluster index, last event, last switch.
Alternatively, customer-specific HMI controls can be mapped in the DenStream algorithm using the Mapping Wizard [254].

### 6.3.3.2 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 On:** 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](image)

![Data Table Horizontal](image)
Alternatively, customer-specific HMI controls can be mapped in the Sequential k-Means algorithm using the Mapping Wizard [254].

6.3.4 Analytics - Compare

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

6.3.4.1 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 [254].
6.3.4.2 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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**

- **Boolean Switch**: TRUE, if a string change was detected, otherwise FALSE.
- **Count**: Is Increased every time the Boolean switch is TRUE.
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.
Alternatively, customer-specific HMI controls can be mapped in the Detect String Change 1Ch algorithm using the Mapping Wizard [254].

### 6.3.4.3 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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 n\textsuperscript{th} channel.
- **Logic Operator**: Logical operator for the operation:

  - Logical OR
  - Logical XOR (EXCLUSIVE OR)
  - Logical AND
  - Logical NAND (NOT AND)
  - Logical NOR (NOT OR)
• **Count Mode**: Mode of the result counter. "OnChange": The counter increments each time the result changes. "Cyclic": The counter increments in each cycle if 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**: Is Increased every time the operation out is *TRUE*.

• **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.
Alternatively, customer-specific HMI controls can be mapped in the Logic Operation Counter algorithm using the Mapping Wizard [254].

6.3.4.4 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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](image1)

![Data Table Horizontal](image2)
Alternatively, customer-specific HMI controls can be mapped in the Multiplexer algorithm using the Mapping Wizard [☞ 254].

### 6.3.4.5 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.

Optional a boolean signal for **Enable Execution** can be selected, so that the algorithm is just 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 increments each time the result changes. "Cyclic": The counter increments in each cycle if 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**: Is increased every time the operation out is **TRUE**.
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.

![Comparison Control](image1)

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

![Single Value Control](image2)

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

![Binary State Control](image3)

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

![Table Control](image4)
Alternatively, customer-specific HMI controls can be mapped in the Numerical Compare 1Ch algorithm using the Mapping Wizard [254].

6.3.4.6 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 increments each time the result changes. "Cyclic": The counter increments in each cycle if 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**: Is increased every time the operation out is TRUE.
• **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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.

![Comparison Control](image1)

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

![Single Value](image2)

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

![Binary State](image3)

4. The Table Control visualizes 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 [254].

6.3.4.7 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 comparison types.
  - *Equals*: Input string corresponds to the reference string.
  - *BeginsWith*: Input string starts 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 increments each time the input string matches the reference string. "Cyclic": The counter increments in each cycle if the comparison operation is TRUE.

Output values

- **String Match**: TRUE if the input string matches the reference string, otherwise FALSE.
- **Count**: Increments every time String Match 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 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.
Alternatively, customer-specific HMI controls can be mapped in the String Compare 1Ch algorithm using the Mapping Wizard [p.254].

6.3.4.8 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

Configuration options

- **String Compare Mode**: Enumeration for different string comparison types.
  - **Equals**: Input string corresponds to the reference string.
  - **BeginsWith**: Input string starts 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 increments each time the input string matches the reference string. "Cyclic": The counter increments in each cycle if the comparison operation 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**: Is increased every time the string match is TRUE.
- **Last Event**: Indicates the point of time of the last raised event → the event can be pulled into the scope chart by Drag & Drop to show 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 [254].

### 6.3.5 Analytics - Math

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

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

**Configuration Options**

- **Integration Mode**: You can select an integration mode. "$\rightarrow$": the input value is will be integrated directly. "|$x$|": The absolute values of the input signal will be integrated.
- **Factor**: With this factor the integral is multiplied.

**Output Values**

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

**Standard HMI Controls**

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

1. The Integrator Control visualizes the last x output values Result.
2. The Table Control or Multivalue Control visualizes all output values: Result.

### Data Table Vertical

<table>
<thead>
<tr>
<th>Input 1</th>
<th>20.42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 2</td>
<td>25.43</td>
</tr>
<tr>
<td>Executing</td>
<td>TRUE</td>
</tr>
<tr>
<td>Last Event</td>
<td>10 Mar 2021 16:15:29</td>
</tr>
</tbody>
</table>

### DataTable Horizontal

<table>
<thead>
<tr>
<th>Data Table</th>
<th>Input 1</th>
<th>Input 2</th>
<th>Executing</th>
<th>Last Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.42</td>
<td>25.43</td>
<td>TRUE</td>
<td>10 Mar 2021 16:15:29</td>
<td></td>
</tr>
</tbody>
</table>
Alternatively, customer-specific HMI controls can be mapped in the Integrator 1Ch algorithm using the Mapping Wizard [254].

6.3.5.2 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 for 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 [254].

6.3.5.3 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

Configuration options
- **Mathematical Operator**: Mathematical operator for the operation ("+", ",", "x", ",", ",x").
- **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.

![Single Value](image1)

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

![Thermometer](image2)

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

![Data Table Vertical](image3)
Alternatively, customer-specific HMI controls can be mapped in the Math Operation algorithm using the Mapping Wizard [254].

6.3.5.4 **RMS 1Ch**

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

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

The number of samples N that are included in the calculation can be configured by specifying a time interval. A cascaded output can be configured to realize a long-term RMS in a resource-saving way and to pick up intermediate results. The time interval of the configured cascade must correspond to an integer multiple of the time interval of the previous cascade.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just active, if the value of the selected signal is **TRUE**.

**Configuration options**

- **Num Cascades**: Number of output cascades
Configuration

- **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](image1)

![Data Table Horizontal](image2)
Alternatively, customer-specific HMI controls can be mapped in the RMS algorithm using the Mapping Wizard [254].

6.3.5.5 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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**: Shows the current slope value.
- **Slope Min**: Shows the minimum of the slope values.
- **Slope Max**: Shows the maximum of the slope values.
- **Delta Value**: Shows the difference between the two values, which are used to calculate the newest slope.
- **Delta Time**: Shows the time period that was used to calculate the newest slope.
- **Time Slope**: Shows the time value of the newest slope value.
- **Time Slope Min**: Shows the time value of the slope minimum → this event can be pulled into the scope chart by Drag & Drop to show it as a trigger-event.
- **Time Slope Max**: Shows the time value of the slope maximum → this event can be pulled into the scope chart by Drag & Drop to show 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 SlopeAnalysis control visualizes the output values Slope, Slope Min, Slope Max, Time Slope Min and Time Slope Max.

![Slope Analysis](image)

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.
Alternatively, custom HMI controls can be mapped in the Edge Counter 1Ch algorithm using the Mapping Wizard [p. 254].

6.3.6 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.
6.3.6.1 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 [114] algorithm, for example.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 [114] due to competing file access. Instead, the reference signal should be learned first and only then should an evaluation with Time Based Envelope 1Ch [114] 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.
- **Involves 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.
Alternatively, customer-specific HMI controls can be mapped in the Time Based Teach Path 1Ch algorithm using the Mapping Wizard [254].

6.3.7 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.

6.3.7.1 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 XTS Velocity control visualizes the output value Acceleration.
2. The SingleValue control visualizes the output value Acceleration.

3. 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.

6.3.7.2 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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.[254]

6.3.7.3  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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 XTSVelocity control visualizes the output value Velocity.
2. The SingleValue control visualizes the output value Velocity.

3. 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 [254].

6.3.8 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.

6.3.8.1 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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.

![Wind Turbulence Control](image1)

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

![Data Table Vertical](image2)

| Input 1 | 20.42 |
| Input 2 | 25.43 |
| Executing | TRUE |
| Last Event | 10 Mar 2021 16:15:29 |
Alternatively, customer-specific HMI controls can be mapped in the WT Turbulence 1Ch algorithm using the Mapping Wizard [\textsuperscript{254}].

### 6.3.9 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.

#### 6.3.9.1 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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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:** Shows if there is currently a gate intersection.
- **Outlier Intersection:** Shows if there is currently an outlier intersection (intersection of the gates projection).
- **Position Intersection X:** X position of the last intersection.
- **Position Intersection Y:** Y position of the last intersection.
- **Count Gate Intersections:** Shows the amount of total gate intersections.
- **Count Outlier Intersections:** Shows the amount of total outlier intersections.
- **Last Intersection:** Indicates the point of time of the last intersection event → the event can be pulled into the scope chart by Drag & Drop to show 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:** Shows the classification result. OK or NOK. The classification depends on the gate mode as you can see 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.
Alternatively, customer-specific HMI controls can be mapped in the XY Gate Monitor 2Ch algorithm using the Mapping Wizard [254].

6.3.9.2 **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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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**: Shows if the input signal is currently within the specified shape.
- **Intersection**: Shows if the input signal currently intersects the specified shape.
- **Count Intersections**: Counts the total amount of intersections of input signal and shape.
- **Last Intersection**: Indicates the point of time of the last intersection event → the event can be pulled into the scope chart by Drag & Drop to show 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.
Alternatively, customer-specific HMI controls can be mapped in the XY Shape Monitor Circle 2Ch algorithm using the Mapping Wizard.[254]

6.3.9.3 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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:** Shows if the input signal is currently within the specified shape.
- **Intersection:** Shows if the input signal currently intersects the specified shape.
- **Count Intersections:** Counts the total amount of intersections of input signal and shape.
• **Last Intersection**: Indicates the point of time of the last intersection event → the event can be pulled into the scope chart by Drag & Drop to show 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.

![XY Shape Monitor](image)

**Intersections: 206**  
Last intersection: 10. Mar 2021 16:27:40

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

![Single Value](image)

**8**  
Last event: 10. Mar 2021 16:24:29

3. The **Table Control** or **Multivalue Control** visualizes all output values: Within Shape, Intersection, Count Intersections, Last Intersection.
Alternatively, customer-specific HMI controls can be mapped in the XY Shape Monitor Rectangle 2Ch algorithm using the Mapping Wizard [p. 254].
6.3.9.4  
**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.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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**: Shows if the input signal is currently within the specified shape.
- **Intersection**: Shows if the input signal currently intersects the specified shape.
- **Count Intersections**: Counts the total amount of intersections of input signal and shape.
- **Last Intersection**: Indicates the point of time of the last intersection event → the event can be pulled into the scope chart by Drag & Drop to show 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.

6.3.10 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.

6.3.10.1 Correlation Function

The Correlation 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 StepSize 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 StepSize. Accordingly, only multiples of the StepSize are also allowed for Minimum Lag and Maximum Lag. If the StepSize is set to one and, for
example, Minimum Lag is set to -6 and Maximum Lag to +4, correlation coefficients are calculated for shifts by -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 eachWindowSize 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. $N = n+1$ always applies in Continuous mode.

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.
Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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**:

Configuration

\[ \hat{C}_{xy}[m, t_n] = \begin{cases} 
\frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i}, & m \geq 0 \\
\frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m}, & m < 0 
\end{cases} \]

\[ \text{cov}_{xy}[m, t_n] = \begin{cases} 
\frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i}, & m \geq 0 \\
\frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i+m}, & m < 0 
\end{cases} \]

\[ \tilde{\text{cov}}_{xy}[m, t_n] = \frac{N}{N-1} \text{cov}_{xy}[m, t_n] \]

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

- **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 cycle and calculated via a window with the length Window Size.

**Output values**
- **Output00 .. Output0n**: Shows for each input (Channel00, ..., Channel0n) the coefficients for the shifts \( m = \text{Minimum Lag}, \ m = \text{Minimum Lag} + \text{Step Size}, ..., \ m = - \text{Step Size}, \ m = 0, \ m = + \text{Step Size}, ..., \ m = \text{Maximum Lag} \) as an array.
- **Minimizing Lag00..Minimizing Lag0n**: For each channel, specifies the shift for which the coefficient becomes minimum.
- **Minimum Coefficient00..Minimum Coefficient0n**: Specifies the minimum coefficient for each channel.
- **Maximizing Lag00..Maximizing Lag0n**: for each channel, specifies the shift for which the coefficient becomes maximum.
- **Maximum Coefficient 00..00.. Maximum Coefficient0n**: Specifies the maximum coefficient for each channel.

**Standard HMI Controls**

For the Correlation Function 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 [p. 254].
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, \( x \) denotes the value of the reference signal and \( y \) denotes the value of the input signal (in each case Channel00, ..., Channel0n) at the timestamp \( t \), (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 \( x_{n+m} \) (or \( y_{n+m} \)) 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 On. 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:

\[ \text{cov}_{xy}[m, t_n] = \begin{cases} \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot y_{n-i} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i-m} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i}, & m \geq 0 \\ \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot y_{n-i+m} - \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i} \cdot \frac{1}{N} \sum_{i=0}^{N-1} y_{n-i+m}, & m < 0 \end{cases} \]

- CovarianceBessel:

\[ \text{cov}_{xy}[m, t_n] = \frac{N}{N-1} \cdot \text{cov}_{xy}[m, t_n] \]

- Pearson:

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

- Window Mode: Specifies the type of window used to calculate the coefficients:
  
  - Continuous: All values since the start of the analysis are included in the analysis with equal weighting.
  
  - SlidingWindow: The calculation is done via a window of the size Window Size. The current values are always included in the analysis and outputs are updated with each cycle.
  
  - FixWindow: The coefficients are calculated over the entire length of the read-in signal section and the outputs are updated after processing the last value.

Output values:

- Output00 .. Output0n: Shows for each input (Channel00, ..., Channel0n) the coefficients for the shifts \( m = \text{Minimum Lag}, m = \text{Minimum Lag} + \text{Step Size}, ..., m = - \text{Step Size}, m = 0, m = + \text{Step Size}, ..., m = \text{Maximum Lag} \) as an array.

- Minimizing Lag00..Minimizing Lag0n: For each channel, specifies the shift for which the coefficient becomes minimum.

- Minimum Coefficient00..Minimum Coefficient0n: Specifies the minimum coefficient for each channel.
• **Maximizing Lag0..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[254].

6.3.10.3 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 \( \beta_0 \) to \( \beta_n \) are recalculated using the following rule:

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

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 \( Y \) 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just 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 LinearRegression control visualizes the inputs and the calculated regression line. The buttons can be used to select the inputs to be displayed, up to two at a time. 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 [254].

6.3.10.4 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 Curve Fitting function block by dragging / dropping onto the parameter field.

Optional a boolean signal for *Enable Execution* can be selected, so that the algorithm is just 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 LinearRegression control visualizes the inputs and the calculated regression line. The buttons can be used to select the inputs to be displayed, up to two at a time. 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 Inference algorithm using the Mapping Wizard.¹ ²

6.3.10.5 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

Configuration options

- **Num Channels**: Configuration of the number of independent input and output channels.

- **Use Bessel Correction**: If the checkbox is checked, Bessel's correction is applied. This parameter must be enabled in order to obtain an expected result for random samples.

The empirical standard deviation, without Bessel's correction

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

The empirical standard deviation, with Bessel's correction

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

- **Window Mode**: Window mode used. Influences the amount of input data included in the calculation and the timing of the calculations.
  - *Continuous*: All input values since the start of the algorithm are included in the calculation. The calculation is performed cyclically.
  - *Fix Window*: Only the last $N$ input values are included in the calculation. The calculation is performed every $N$ calls.
  - *Sliding Window*: Only the last $N$ input values are included in the calculation. The calculation is performed cyclically.
- **Window Size**: The number of values $N$ that are included in the calculation can be configured depending on the window mode used. With the window mode *Continuous* this parameter is ignored.

**Output values**

- **Standard Deviation 00..n**: Empirical standard deviation of the input channels.

**Standard HMI Controls**

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

1. The Table Control or Multivalue Control visualizes all output values: Standard Deviation Results.
Alternatively, customer-specific HMI controls can be mapped in the Standard Deviation algorithm using the Mapping Wizard [254].

6.3.11 Analytics – Extensions with algorithms from other libraries

In addition to the algorithms of the TwinCAT Analytics Library, which are available in the TwinCAT Analytics Workbench configurator or in the TwinCAT Analytics Service Tool, further algorithms from other PLC libraries are also 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.
The TwinCAT Filter library enables the implementation of digital filters of different types using various function blocks.

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

### 6.3.12 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.

#### 6.3.12.1 Trend Line

The Trend Line function block deals with forwarding of input signal trends for subsequent 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.

Optional a boolean signal for Enable Execution can be selected, so that the algorithm is just active, if the value of the selected signal is TRUE.

**Configuration options**

- **Num Channels**: Specifies the number of configurable channels for forwarding the input signals.
- **Channel 00**: Specifies 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**: Specifies 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**: Specifies 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 xx**: Analogous for all other channels.

### 6.4 Interaction with Scope

The TwinCAT Analytics Engineering tools offer an easy interaction between Analytics Configurator and Scope View. They allow you to mark significant values in the data stream and examine them with other process data within the exact cycle. Also, it is possible to see result data of the algorithm in Scope View like an average or a max value.

After configuration of the analysis you can move to the Analytics Project start page. There you can click on **add referenced Scope**.

Automatically a suitable Scope configuration is added to the project.
There are various possibilities to visualize the data from different algorithms. It is possible to take any timestamp from an algorithm output by drag and drop to the charting. The Scope will mark the position of the event with a colored marker (see blue line in picture below).
For any kind of Shape algorithm, the SW generates a XY Chart in the Scope configuration including a shape with the given definition (see orange line in the picture above). For the Threshold classification algorithm, a dynamic style with color change from channel color to yellow (warning) and to red (alert) is also automatically done in Scope View (rose line).

You can take additionally the result values of the algorithm by drag and drop to a chart in front of the record or during the record to add a new channel which shows e.g. the average, min or max value.

To control the number of shown events/marker you can click on the corresponding Trigger Group in the Scope View tree. In the Property window you are able to set a number of Visible Trigger Release Capacity.
You can choose between

- All
Time relation between Analytics configuration and Scope View

Keep in mind that the record time in Scope View can be different than the one in Analytics. Especially if you choose the ring buffer in Scope View it could be possible that some significant values of your analysis are in past of the Scope record!

### 6.5 Runtime deployment

PLC code can be generated with all modules and parameters configured in the TwinCAT Analytics Workbench Configurator. This code can be downloaded into a TwinCAT Analytics runtime to realize 24/7 data analysis.

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
</table>

**Compatibility of automatically generated PLC code**

The automatically generated PLC code is based on the TwinCAT Analytics Library. Interfaces of the base models of the library are compatible with earlier versions after release of the library. The automatically generated code itself is only a sample code! This code generation may change from version to version. As far as possible, this is solved by the code generation version.

After configuration, you can click on the **Deploy Analytics Runtime** command in the context menu, as shown in the image below.
The Deploy Wizard starts and it is possible to set up the entire required configuration step by step for use.
On the first tab, **Solution**, you can select whether a new Solution is created, whether the analysis PLC code is integrated or added to an existing Solution as a new project, a new PLC or to an existing PLC, or whether the new project is inserted into an existing Solution using the TwinCAT Project Compare Tool.

On the second tab, **TwinCAT PLC Target**, all PLC-specific parameters such as target system, task cycle time or the task assignment to corresponding CPU cores can be defined.
If you have set the "PLC Result" property of some functions in the configurator, the Result tab is enabled in the Deploy wizard. There you can set up where the results will be streamed or stored.

By clicking Select Result Items it is possible to select only the desired values.
If you have assigned HMI controls to one or more functions, the **HMI Dashboard** tab is enabled in the Deploy wizard. Here, various settings can be made to generate a customized dashboard that fits your needs.

On the next tab, **Visual Studio**, you can select which Visual Studio version or TwinCAT XAE shell should be used for generation, if several are installed.
The last tab shows you all the settings you have made for the generation. Now you can start the generation process by clicking Deploy.

In the overview window, each step during the generation process is displayed, clearly arranged and divided into categories.
6.5.1 Algorithm properties

Each algorithm of the Analytics Configurator is providing some properties. The sections of HMI and PLC are necessary for the automatic code generation.
**HMI**

- **Generate GVL**: Enable the generation of a Global Variable List with a collection of variables and corresponding data type mapping for TwinCAT HMI
- **GlobalVariableType**: Choose the type with InOutVariables just for inputs and outputs of the algorithm or KeyValueCollection for general mapping to STRING for tables

**PLC**

- **Persistent Results**: Enable this flag to store results of algorithm persistent to target system of the Analytics Runtime
- **Stream Results**: Enable this flag to add the In- and Outputs of the algorithm to a result stream which will be generated by the code generation
6.5.2 PLC Code

NOTE

Compatibility of automatically generated PLC code

The automatically generated PLC code is based on the TwinCAT Analytics Library. Interfaces of the base models of the library are compatible with earlier versions after release of the library. The automatically generated code itself is only a sample code! This code generation may change from version to version.

Code generation version compatibility

<table>
<thead>
<tr>
<th></th>
<th>Version 2.1</th>
<th>Version 3.0</th>
<th>Version 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics algorithms</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Filter algorithms</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HMI support</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HMI with advanced input stream handling</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Support of array inputs</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Support of oversampling inputs</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Network templates</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>&quot;Closed&quot; network templates</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Network with inputs, outputs and parameters</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>&quot;Condition Monitoring&quot; network templates</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>
6.5.2.1 Code version 2.1

Task:
A separate task is created for the analytics analysis.

StreamHelper:
If one or more data sources are of type MQTT binary stream, the code generation creates an instance of a StreamHelper object to process the incoming binary stream patterns.
DataTypes:
The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

HMI GVL:
To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables.

DataSource/M2M Mapping:
The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs. The FB ValueMapping_M2M manages the value mapping between the modules (M2M - Module to Module) from the module INPUTs to the module OUTPUTs.

Network:
All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:
The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Results:
If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Analysis:
The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN:
The FB Analysis is called in the program MAIN_Analytics. The program is assigned to the separate task.
It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.
All reset calls are defined in the action A_Reset.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics Version &gt;= 3.1.0.0</td>
</tr>
</tbody>
</table>

6.5.2.1.1 FB_DataSource
The DataSource FB manage the receiving of the input values of the different sources. In the OUTPUT declaration you can find all configured inputs.

Syntax
Definition:
**FUNCTION_BLOCK FB_T[n]_DataSource** IMPLEMENTS I_T[n]_DataSource
VAR
END_VAR

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [207]</td>
<td>Local</td>
<td>Method for background communication with the TwinCAT driver. The method must be called cyclically.</td>
</tr>
<tr>
<td>GetData [207]</td>
<td>Local</td>
<td>Method to get the data of the specified element</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

### GetData

**Syntax**

METHOD GetData : BOOL
VAR_INPUT
  nElement : UDINT;
END_VAR

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Element ID to get the specific sample</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetData</td>
<td>BOOL</td>
<td>Is TRUE if a new element is selected</td>
</tr>
</tbody>
</table>

### Call

**Syntax**

METHOD Call : BOOL

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

#### 6.5.2.1.2 FB_Network

All modules are sorted in a specific network to get a better overview and structure of the configured analysis.
### Syntax

**Definition:**

```
FUNCTION_BLOCK FB_N[n]_1[Network1]
VAR_INPUT
    [module FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

---

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module FBs</td>
<td></td>
<td>FBs of the configured modules</td>
</tr>
</tbody>
</table>

---

#### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE as soon as an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.</td>
</tr>
</tbody>
</table>

---

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [212]</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset [213]</td>
<td>Local</td>
<td>Reset the Network with all sub modules</td>
</tr>
<tr>
<td>ValueMapping [208]</td>
<td>Local</td>
<td>Map the input values to the different module inputs</td>
</tr>
<tr>
<td>SetHMIValues [209]</td>
<td>Local</td>
<td>Optional: Map in-outputs of the modules to the global HMI variable</td>
</tr>
</tbody>
</table>

---

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

---

#### ValueMapping

**Syntax**

```
METHOD ValueMapping : BOOL
VAR_INPUT
    ipDataSource : I_T[n]_DataSource;
END_VAR
```

---

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDataSource</td>
<td>I_T[n]_DataSource</td>
<td>Data for the analysis from the specific data source</td>
</tr>
</tbody>
</table>
Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueMapping</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

SetHMIValues

Syntax

METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pHMI_N[n]_[Network1]</td>
<td>POINTER TO ST_HMI_N[n]_[Network1]</td>
<td>Pointer to global HMI struct</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetHMIValues</td>
<td>BOOL</td>
<td>Is TRUE if done</td>
</tr>
</tbody>
</table>

Reset

Syntax

METHOD Reset : BOOL
VAR
END_VAR

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE if done</td>
</tr>
</tbody>
</table>

Call

Syntax

METHOD Call : BOOL
VAR_INPUT
    ipDataSource: I_T[n]_DataSource;
    [ipValueMapping_M2M: I_ValueMapping_M2M;]
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDataSource</td>
<td>I_T[n]_DataSource</td>
<td>Data for the analysis.</td>
</tr>
<tr>
<td>ipValueMapping_M2M</td>
<td>I_ValueMapping_M2M</td>
<td>Optional: Needed for mapping values between modules</td>
</tr>
</tbody>
</table>
Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

6.5.2.1.3 FB_Module

The module FBs contains all inputs and outputs of the configured modules from the Workbench Configurator. It is also possible to reconfigure the modules at runtime. You only have to change the parameter and then start the reconfigure process with a rising edge at the bReconfigure INPUT.

Syntax

Definition:

FUNCTION_BLOCK FB_N[n]_M[n]_Module
VAR_INPUT [module inputs]
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module inputs</td>
<td></td>
<td>Inputs of the selected module</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE as soon as an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.</td>
</tr>
<tr>
<td>Module outputs</td>
<td></td>
<td>Outputs of the selected module</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [211]</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset [211]</td>
<td>Local</td>
<td>Reset the module</td>
</tr>
<tr>
<td>SetHMI [211]</td>
<td>Local</td>
<td>Optional: Sets the in-outputs to the global HMI structs</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>
Call

Syntax
METHOD Call : BOOL
VAR
END_VAR

�� Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

Reset

Syntax
METHOD Reset : BOOL
VAR
END_VAR

�� Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE if done</td>
</tr>
</tbody>
</table>

SetHMI

Syntax
METHOD SetHMIValues : BOOL
VAR_INPUT
  pHMI_N[n][Network1] : POINTER TO ST_HMI_N[n][Network1];
END_VAR

�� Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pHMI_N[n][Network1]</td>
<td>POINTER TO ST_HMI_N[n][Network1]</td>
<td>Pointer to global HMI struct</td>
</tr>
</tbody>
</table>

�� Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetHMI</td>
<td>BOOL</td>
<td>Is TRUE if done</td>
</tr>
</tbody>
</table>

6.5.2.1.4 FB_Analysis

In the analysis FB the whole analytics routine is defined. All configured networks with their modules and error handling is created.

Syntax

Definition:
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
[network FBs]
END_VAR
VAR_OUTPUT
bError: BOOL;
ipTcResult: I_TcMessage;
END_VAR

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network FBs</td>
<td></td>
<td>FBs of the configured networks</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE as soon as an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>Local</td>
<td>Method for background communication with the TwinCAT driver. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset</td>
<td>Local</td>
<td>Reset the whole analysis</td>
</tr>
<tr>
<td>ResultStream</td>
<td>Local</td>
<td>Optional: If a result stream has to be created</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

**Call**

**Syntax**

```
METHOD Call : BOOL
VAR_INPUT
  ipDataSource: I_T[n]_DataSource;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipDataSource</td>
<td>I_T[n]_DataSource</td>
<td>Data for the analysis.</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
Reset

Syntax

METHOD Reset : BOOL
VAR_IN_OUT
   stReset: ST_AnalysisReset;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stReset</td>
<td>ST_AnalysisReset</td>
<td>Struct to define which module or network should be reset.</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE if done</td>
</tr>
</tbody>
</table>

ResultStream

Syntax

METHOD ResultStream : BOOL
VAR_INPUT
   ipResults: I_Results;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResults</td>
<td>I_Results</td>
<td>Interface pointer to the Result FB</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

6.5.2.1.5 FB_Results

If results of the analysis has to be stored or streamed, the result FB managed this and streamed the selected variables to the message broker or store the data into the analytics binary file.

Syntax

Definition:

FUNCTION_BLOCK FB_Results
VAR_OUTPUT
   stResults: ST_Results;
END_VAR
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stResults</td>
<td>ST_Results</td>
<td>Result struct which contains all items of the result stream</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [214]</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>AddResult [214]</td>
<td>Local</td>
<td>Add a sample to the result stream</td>
</tr>
<tr>
<td>SendResults [215]</td>
<td>Local</td>
<td>Sends all buffered samples of the result stream</td>
</tr>
<tr>
<td>Release [215]</td>
<td>Local</td>
<td>Close stream or file of the result stream</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

### Call

#### Syntax

METHOD Call : BOOL  
VAR  
END_VAR  

#### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

### AddResult

#### Syntax

METHOD AddResult : BOOL  
VAR_INPUT  
  tTimestamp: ULLINT;  
  stSample: ST_Results;  
END_VAR  

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tTimestamp</td>
<td>ULLINT</td>
<td>Timestamp of the sample</td>
</tr>
<tr>
<td>stSample</td>
<td>ST_Results</td>
<td>Sample struct</td>
</tr>
</tbody>
</table>

#### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddResult</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
SendResults

Syntax

METHOD SendResults : BOOL
VAR
END_VAR

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendResults</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

Release

Syntax

METHOD Release : BOOL
VAR
END_VAR

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

6.5.2.1.6 MAIN_Analytics

In the MAIN_Analytics program the analysis FB is called. The program is assign to the separated Task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First you have to choose the component you would like to reset. Then a rising edge at the bReset INPUT starts the reset process.

Inside of the A_Reset Action all reset calls are defined.

Syntax

Definition:

```plaintext
PROGRAM MAIN_Analytics
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>
6.5.2.2 Code version 3.0

Tasks:
A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

StreamHelper:
For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.
DataTypes:
The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

GVLS:
To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

VirtualInputSource / DataSource / M2M Mapping:
The Virtual Input Source interfaces abstract the Data Source symbols from the analysis. The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs. The FB ValueMapping_M2M manages the value mapping between the modules (M2M - Module to Module) from the module INPUTs to the module OUTPUTs.

Network:
All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:
The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Results:
If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

Analysis:
The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN PRGs:
In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

The FB Analysis is called in the programs MAIN_Analytics_Vx_Cx. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics Version &gt;= 3.1.0.0</td>
</tr>
</tbody>
</table>

6.5.2.2.1 FB_DataSource
The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.
Syntax

Definition:

FUNCTION_BLOCK FB_T[n]_DataSource IMPLEMENTS I_DataSource, I_V[n]_Virtual_Input_Source
VAR_OUTPUT
eDataState: E_DataSourceState;
END_VAR

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>Local</td>
<td>Method for background communication with the TwinCAT driver. The method must be called cyclically.</td>
</tr>
<tr>
<td>GetData</td>
<td>Local</td>
<td>Method to retrieve the data of the specified element.</td>
</tr>
<tr>
<td>GetDataOversampling</td>
<td>Local</td>
<td>Method to retrieve the oversampling data of the specified element.</td>
</tr>
<tr>
<td>NewDataAvailable</td>
<td>Local</td>
<td>Method to check if new data is available.</td>
</tr>
<tr>
<td>HistoricalCtrl</td>
<td>Local</td>
<td>Method for retrieving historical data.</td>
</tr>
<tr>
<td>UpdateRecordList</td>
<td>Local</td>
<td>Method for updating the historical record list.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

GetData

Syntax

METHOD GetData : BOOL
VAR_INPUT
nElement : UDINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Element ID to obtain the specific example.</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetData</td>
<td>BOOL</td>
<td>Is TRUE if a new element is selected.</td>
</tr>
</tbody>
</table>

Call

Syntax

METHOD Call : BOOL
**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

**GetDataOversampling**

**Syntax**

```plaintext
METHOD GetDataOversampling : BOOL
VAR_INPUT
nElement : UDINT;
nSample : UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Element ID to obtain the specific sample element.</td>
</tr>
<tr>
<td>nSample</td>
<td>UDINT</td>
<td>Sample ID to obtain the specific sample.</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDataOversampling</td>
<td>BOOL</td>
<td>Is TRUE if a new element is selected.</td>
</tr>
</tbody>
</table>

**NewDataAvailable**

**Syntax**

```plaintext
METHOD NewDataAvailable : BOOL
VAR_INPUT
nLastDataHandle : ULLINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLastDataHandle</td>
<td>ULLINT</td>
<td>Handle from the last fetched data packet.</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewDataAvailable</td>
<td>BOOL</td>
<td>Is TRUE if new data is available.</td>
</tr>
</tbody>
</table>

**HistoricalCtrl**

**Syntax**

```plaintext
METHOD HistoricalCtrl : BOOL
VAR_INPUT
  stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
  stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
  stRecordInfo : REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stCtrl</td>
<td>ST_HMI_DataSourceCtrl</td>
<td></td>
</tr>
<tr>
<td>stHistStreamInfo</td>
<td>ST_HMI_DataSourceHist</td>
<td></td>
</tr>
<tr>
<td>stRecordInfo</td>
<td>ST_HMI_DataSourceHistRecordInfo</td>
<td></td>
</tr>
</tbody>
</table>

## Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HistoricalCtrl</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

## UpdateRecordList

**Syntax**

```plaintext
METHOD UpdateRecordList : BOOL
VAR_INPUT
  stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
  stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
  sStreamSystemID : GUID;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stCtrl</td>
<td>ST_HMI_DataSourceCtrl</td>
<td></td>
</tr>
<tr>
<td>stHistStreamInfo</td>
<td>ST_HMI_DataSourceHist</td>
<td></td>
</tr>
<tr>
<td>sStreamSystemID</td>
<td>GUID</td>
<td></td>
</tr>
</tbody>
</table>

## Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateRecordList</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

## 6.5.2.2.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

**Syntax**

```plaintext
FUNCTION_BLOCK FB_N[n][Network1]
VAR_INPUT
  [module FBs]
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module FBs</td>
<td></td>
<td>Function blocks of the configured modules.</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE when an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [225]</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset [226]</td>
<td>Local</td>
<td>Resetting the network with all submodules.</td>
</tr>
<tr>
<td>ValueMapping [221]</td>
<td>Local</td>
<td>Assignment of the input values to the various module inputs.</td>
</tr>
<tr>
<td>SetHMIValues [222]</td>
<td>Local</td>
<td>Optional: Mapping of the inputs/outputs of the modules to the global HMI variable.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

### ValueMapping

#### Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipVirtual_Input_Source</td>
<td>I_V[n]_Virtual_Input_Source</td>
<td>Data for analysis from the specific data source.</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueMapping</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
SetHMIValues

Syntax
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n][Network1] : POINTER TO ST_HMI_N[n][Network1];
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pHMI_N[n][Network1]</td>
<td>POINTER TO ST_HMI_N[n][Network1]</td>
<td>Pointer to the global HMI structure.</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetHMIValues</td>
<td>BOOL</td>
<td>Is TRUE when completed.</td>
</tr>
</tbody>
</table>

Reset

Syntax
METHOD Reset : BOOL
VAR
END_VAR

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE when completed.</td>
</tr>
</tbody>
</table>

Call

Syntax
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M: I_ValueMapping_M2M;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipVirtual_Input_Source</td>
<td>I_T[n]_Virtual_Input_Source</td>
<td>Data for the analysis.</td>
</tr>
<tr>
<td>ipValueMapping_M2M</td>
<td>I_ValueMapping_M2M</td>
<td>Optional: Necessary for mapping values between modules.</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
6.5.2.2.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

**Definition:**

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
[module inputs]
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
[module outputs]
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module inputs</td>
<td></td>
<td>Inputs of the selected module.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE when an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.</td>
</tr>
</tbody>
</table>

**Module outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Outputs of the selected module.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset</td>
<td>Local</td>
<td>Resetting the module.</td>
</tr>
<tr>
<td>SetHMI</td>
<td>Local</td>
<td>Optional: Sets the inputs/outputs to the global HMI structures.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

**Call**

**Syntax**

```
METHOD Call : BOOL
VAR_INPUT
  ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
  [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipVirtual_Input_Source</td>
<td>I_V[n]_Virtual_Input_Source</td>
<td>Data for the analysis.</td>
</tr>
<tr>
<td>ipValueMapping_M2M</td>
<td>I_ValueMapping_M2M</td>
<td>Optional: Necessary for mapping values between modules.</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

### Reset

**Syntax**

```
METHOD Reset : BOOL
VAR
END_VAR
```

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE when completed.</td>
</tr>
</tbody>
</table>

### SetHMI

**Syntax**

```
METHOD SetHMIValues : BOOL
VAR_INPUT
  pHMI_N[n][Network1] : POINTER TO ST_HMI_N[n][Network1];
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pHMI_N[n][Network1]</td>
<td>POINTER TO ST_HMI_N[n][Network1]</td>
<td>Pointer to the global HMI structure.</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetHMI</td>
<td>BOOL</td>
<td>Is TRUE when completed.</td>
</tr>
</tbody>
</table>

### 6.5.2.2.4 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

**Syntax**

Definition:
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
  [network FBs]
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
END_VAR

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network FBs</td>
<td></td>
<td>Function blocks of the configured networks.</td>
</tr>
</tbody>
</table>

## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE when an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [225]</td>
<td>Local</td>
<td>Method for background communication with the TwinCAT driver. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset [226]</td>
<td>Local</td>
<td>Resets the entire analysis.</td>
</tr>
<tr>
<td>ResultStream [226]</td>
<td>Local</td>
<td>Optional: If a result stream needs to be created.</td>
</tr>
</tbody>
</table>

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

## Call

### Syntax

```plaintext
METHOD Call : BOOL
VAR_INPUT
  ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipVirtual_Input_Source</td>
<td>I_V[n]_Virtual_Input_Source</td>
<td>Data for the analysis.</td>
</tr>
</tbody>
</table>

## Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
Reset

Syntax

METHOD Reset : BOOL
VAR_IN_OUT
   stReset: ST_AnalysisReset;
END_VAR

.inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stReset</td>
<td>ST_AnalysisReset</td>
<td>Structure to define which module or network is to be reset.</td>
</tr>
</tbody>
</table>

.return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE when completed.</td>
</tr>
</tbody>
</table>

ResultStream

Syntax

METHOD ResultStream : BOOL
VAR_INPUT
   ipResults: I_Results;
END_VAR

.inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResults</td>
<td>I_Results</td>
<td>Interface pointer to the FB Results.</td>
</tr>
</tbody>
</table>

.return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

6.5.2.2.5 FB_Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Syntax

Definition:

FUNCTION_BLOCK FB_Results
VAR_OUTPUT
   stResults: ST_Results;
END_VAR
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stResults</td>
<td>ST_Results</td>
<td>Result structure that contains all elements of the result stream.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>AddResult</td>
<td>Local</td>
<td>Add a sample to the result stream.</td>
</tr>
<tr>
<td>SendResults</td>
<td>Local</td>
<td>Sends all buffered samples of the result stream.</td>
</tr>
<tr>
<td>Release</td>
<td>Local</td>
<td>Close stream or file of the result stream.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

### Call

**Syntax**

METHOD Call : BOOL  
VAR  
END_VAR

#### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

### AddResult

**Syntax**

METHOD AddResult : BOOL  
VAR_INPUT  
    tTimestamp: ULINT;  
    stSample: ST_Results;  
END_VAR

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tTimestamp</td>
<td>ULINT</td>
<td>Timestamp of the sample</td>
</tr>
<tr>
<td>stSample</td>
<td>ST_Results</td>
<td>Sample structure</td>
</tr>
</tbody>
</table>

#### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddResult</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
SendResults

Syntax

METHOD SendResults : BOOL
VAR
END_VAR

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendResults</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

Release

Syntax

METHOD Release : BOOL
VAR
END_VAR

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

6.5.2.2.6 MAIN_Analytics

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI Dashboard. The program is assigned to a separate task.

Definition:

PROGRAM MAIN_Analytics
VAR_INPUT
  sCurrentStreamSystemID: GUID;
  stHistStreamInfo: REFERENCE TO ST_HMI_DataSourceHist;
  stHistRecordInfo: REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
END_VAR

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

6.5.2.2.7 MAIN_Analytics_V[n]_C[n]

The FB Analysis is called in the program MAIN_Analytics_V[n]_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.
All reset calls are defined in the action A_Reset.

Mapping of the HMI values takes place in the action A_MapToHMI.

**Syntax**

**Definition:**

```plaintext
PROGRAM MAIN_Analytics_V[n]_C[n]
VAR_INPUT
  stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
  nLastDataHandle: ULINT;
END_VAR
```

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>
**Tasks:**

A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

**StreamHelper:**

For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.
DataTypes:
The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

GVLs:
To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

ClosedNetwork:
The ClosedNetwork FBs are generated once with all subnetworks and modules. They can be instantiated multiple times in the analysis. In this way, the generated code can be reduced and simplified.

VirtualInputSource / DataSource:
The VirtualInputSource interfaces abstract the DataSource symbols from the analysis. The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Network:
All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:
The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

ClosedNetwork Instance:
In this FB the corresponding ClosedNetwork is instantiated for the analysis. Internally used modules are no longer generated as module FBs in this case.

Results:
If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

Analysis:
The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN PRGs:
In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

The FB Analysis is called in the programs MAIN_Analytics_Vx_Cx. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics Version &gt;= 3.1.0.0</td>
</tr>
</tbody>
</table>

### 6.5.2.3.1 FB_DataSource

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

### Syntax

**Definition:**

```plaintext
FUNCTION_BLOCK FB_T[n]_DataSource IMPLEMENTS I_DataSource, I_V[n]_Virtual_Input_Source
  VAR_OUTPUT
eDataState: E_DataSourceState;
END_VAR
```

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [233]</td>
<td>Local</td>
<td>Method for background communication with the TwinCAT driver. The method must be called cyclically.</td>
</tr>
<tr>
<td>GetData [232]</td>
<td>Local</td>
<td>Method to retrieve the data of the specified element.</td>
</tr>
<tr>
<td>GetDataOversampling [233]</td>
<td>Local</td>
<td>Method to retrieve the oversampling data of the specified element.</td>
</tr>
<tr>
<td>NewDataAvailable [233]</td>
<td>Local</td>
<td>Method to check if new data is available.</td>
</tr>
<tr>
<td>HistoricalCtrl [234]</td>
<td>Local</td>
<td>Method for retrieving historical data</td>
</tr>
<tr>
<td>UpdateRecordList [234]</td>
<td>Local</td>
<td>Method for updating the historical record list.</td>
</tr>
</tbody>
</table>

###.GetData

**Syntax**

```plaintext
METHOD GetData : BOOL
  VAR_INPUT
    nElement : UDINT;
  END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Element ID to obtain the specific sample</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetData</td>
<td>BOOL</td>
<td>Is TRUE if a new element is selected</td>
</tr>
</tbody>
</table>
Call

Syntax
METHOD Call : BOOL

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

GetDataOversampling

Syntax
METHOD GetDataOversampling : BOOL
VAR_INPUT
VAR_INPUT
nElement : UDINT;
nSample : UDINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nElement</td>
<td>UDINT</td>
<td>Element ID to obtain the specific sample element</td>
</tr>
<tr>
<td>nSample</td>
<td>UDINT</td>
<td>Sample ID to obtain the specific sample</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDataOversampling</td>
<td>BOOL</td>
<td>Is TRUE if a new element is selected</td>
</tr>
</tbody>
</table>

NewDataAvailable

Syntax
METHOD NewDataAvailable : BOOL
VAR_INPUT
nLastDataHandle : ULINT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLastDataHandle</td>
<td>ULINT</td>
<td>Handle of the last fetched data packet</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewDataAvailable</td>
<td>BOOL</td>
<td>Is TRUE if new data is available</td>
</tr>
</tbody>
</table>
**HistoricalCtrl**

**Syntax**

```plaintext
METHOD HistoricalCtrl : BOOL
VAR_INPUT
  stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
  stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
  stRecordInfo : REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stCtrl</td>
<td>ST_HMI_DataSourceCtrl</td>
<td></td>
</tr>
<tr>
<td>stHistStreamInfo</td>
<td>ST_HMI_DataSourceHist</td>
<td></td>
</tr>
<tr>
<td>stRecordInfo</td>
<td>ST_HMI_DataSourceHistRecordInfo</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HistoricalCtrl</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

**UpdateRecordList**

**Syntax**

```plaintext
METHOD UpdateRecordList : BOOL
VAR_INPUT
  stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
  stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
  sStreamSystemID : GUID;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stCtrl</td>
<td>ST_HMI_DataSourceCtrl</td>
<td></td>
</tr>
<tr>
<td>stHistStreamInfo</td>
<td>ST_HMI_DataSourceHist</td>
<td></td>
</tr>
<tr>
<td>sStreamSystemID</td>
<td>GUID</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateRecordList</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

**NextData**

**Syntax**

```plaintext
METHOD NextData : BOOL
VAR_INPUT
END_VAR
```
NextDataOversample

Syntax
METHOD GetDataOversampling : BOOL
VAR_INPUT
nMaxOversampling : UDINT;
END_VAR

Inputs
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nMaxOversampling</td>
<td>UDINT</td>
<td>Specifies the maximum oversampling factor.</td>
</tr>
</tbody>
</table>

Return value
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NextDataOversample</td>
<td>BOOL</td>
<td>Is TRUE if a new element is selected</td>
</tr>
</tbody>
</table>

6.5.2.3.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Syntax
Definition:
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_OUTPUT
   bError: BOOL;
   ipTcResult: I_TcMessage;
END_VAR
VAR
   [module FBs]
END_VAR

Outputs
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE when an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.</td>
</tr>
</tbody>
</table>

Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [\ref{225}]</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset [\ref{226}]</td>
<td>Local</td>
<td>Resetting the network with all submodules.</td>
</tr>
<tr>
<td>ValueMapping [\ref{236}]</td>
<td>Local</td>
<td>Assignment of the input values to the various module inputs.</td>
</tr>
<tr>
<td>SetHMIValues [\ref{236}]</td>
<td>Local</td>
<td>Optional: Mapping of the inputs/outputs of the modules to the global HMI variable.</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

ValueMapping

Syntax

```plaintext
METHOD ValueMapping : BOOL
  VAR_INPUT
    pAnalysis : POINTER TO FB_Analysis;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pAnalysis</td>
<td>FB_Analysis</td>
<td>Instance of the analysis FB</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueMapping</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

SetHMIValues

Syntax

```plaintext
METHOD SetHMIValues : BOOL
  VAR_INPUT
    pHMI_N[n][Network1] : POINTER TO ST_HMI_N[n][Network1];
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pHMI_N[n][Network1]</td>
<td>POINTER TO ST_HMI_N[n][Network1]</td>
<td>Pointer to the global HMI structure</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetHMIValues</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

Call

Syntax

```plaintext
METHOD Call : BOOL
  VAR_INPUT
    pAnalysis: POINTER TO FB_Analysis;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pAnalysis</td>
<td>FB_Analysis</td>
<td>Instance of the analysis FB.</td>
</tr>
</tbody>
</table>

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

Reset

Syntax

METHOD Reset : BOOL
VAR
END_VAR

Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE when completed.</td>
</tr>
</tbody>
</table>

6.5.2.3.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

Definition:

FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
  [module inputs]
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
  [module outputs]
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module inputs</td>
<td></td>
<td>Inputs of the selected module.</td>
</tr>
</tbody>
</table>
## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE when an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.</td>
</tr>
</tbody>
</table>

---

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [238]</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset [238]</td>
<td>Local</td>
<td>Resetting the module.</td>
</tr>
<tr>
<td>SetHMI [239]</td>
<td>Local</td>
<td>Optional: Sets the inputs/outputs to the global HMI structures.</td>
</tr>
</tbody>
</table>

---

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

---

## Call

**Syntax**

METHOD Call : BOOL

VAR_INPUT
  _ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
  [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR

---

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipVirtual_Input_Source</td>
<td>I_V[n]_Virtual_Input_Source</td>
<td>Data for the analysis</td>
</tr>
<tr>
<td>ipValueMapping_M2M</td>
<td>I_ValueMapping_M2M</td>
<td>Optional: Necessary for mapping values between modules</td>
</tr>
</tbody>
</table>

---

## Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

---

## Reset

**Syntax**

METHOD Reset : BOOL

VAR
END_VAR
6.5.2.3.4 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
  ipV[n]_VirtualInputs: I_V[n]_Virtual_Input_Source;
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
END_VAR
VAR
  [network FBs]
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipV[n]_VirtualInputs</td>
<td>I_V[n]_Virtual_Input_Source</td>
<td>Data for analysis from the specific data source</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE when an error situation occurs.</td>
</tr>
<tr>
<td>ipTcResult</td>
<td>I_TcMessage</td>
<td>Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [240]</td>
<td>Local</td>
<td>Method for background communication with the TwinCAT driver. The method must be called cyclically.</td>
</tr>
<tr>
<td>Reset [240]</td>
<td>Local</td>
<td>Reset the whole analysis</td>
</tr>
<tr>
<td>ResultStream</td>
<td>Local</td>
<td>Optional: If a result stream has to be created</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

### Call

**Syntax**

```plaintext
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipVirtual_Input_Source</td>
<td>I_V[n]_Virtual_Input_Source</td>
<td>Data for the analysis.</td>
</tr>
</tbody>
</table>

### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

### Reset

**Syntax**

```plaintext
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```
**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stReset</td>
<td>ST_AnalysisReset</td>
<td>Structure to define which module or network is to be reset.</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>BOOL</td>
<td>Is TRUE when completed.</td>
</tr>
</tbody>
</table>

**ResultStream**

**Syntax**

```plaintext
METHOD ResultStream : BOOL
VAR_INPUT
  ipResults: I_Results;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResults</td>
<td>I_Results</td>
<td>Interface pointer to the FB Results</td>
</tr>
</tbody>
</table>

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

**6.5.2.3.5 FB_Results**

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

**Syntax**

**Definition:**

```plaintext
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
  stResults: ST_Results;
END_VAR
```

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stResults</td>
<td>ST_Results</td>
<td>Result structure that contains all elements of the result stream.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call [242]</td>
<td>Local</td>
<td>Method for background communication. The method must be called cyclically.</td>
</tr>
<tr>
<td>AddResult [242]</td>
<td>Local</td>
<td>Add a sample to the result stream</td>
</tr>
<tr>
<td>SendResults [243]</td>
<td>Local</td>
<td>Sends all buffered samples of the result stream</td>
</tr>
<tr>
<td>Release [243]</td>
<td>Local</td>
<td>Close stream or file of the result stream</td>
</tr>
</tbody>
</table>

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

### Call

**Syntax**

METHOD Call : BOOL  
VAR  
END_VAR

#### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

### AddResult

**Syntax**

METHOD AddResult : BOOL  
VAR_INPUT  
  tTimestamp: ULI;  
  stSample: ST_Results;  
END_VAR

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tTimestamp</td>
<td>ULI</td>
<td>Timestamp of the sample</td>
</tr>
<tr>
<td>stSample</td>
<td>ST_Results</td>
<td>Sample structure</td>
</tr>
</tbody>
</table>

#### Return value

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddResult</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>
SendResults

**Syntax**
METHOD SendResults : BOOL
VAR
END_VAR

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendResults</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

Release

**Syntax**
METHOD Release : BOOL
VAR
END_VAR

**Return value**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

6.5.2.3.6 MAIN_Analytics

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI Dashboard. The program is assigned to a separate task.

**Syntax**

**Definition:**
PROGRAM MAIN_Analytics
VAR_INPUT
  sCurrentStreamSystemID: GUID;
  stHistStreamInfo: REFERENCE TO ST_HMI_DataSourceHist;
  stHistRecordInfo: REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
END_VAR

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

6.5.2.3.7 MAIN_Analytics_C[n]

The FB Analysis is called in the program MAIN_Analytics_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.
All reset calls are defined in the action A_Reset.

Mapping of the HMI values takes place in the action A_MapToHMI.

Syntax

Definition:

PROGRAM MAIN_Analytics_V[n]_C[n]
VAR_INPUT
  stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
  bError: BOOL;
  ipTcResult: I_TcMessage;
  nLastDataHandle: ULINT;
END_VAR

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>Plc libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.0</td>
<td>PC or CX (x64, x86)</td>
<td>Tc3_Analytics</td>
</tr>
</tbody>
</table>

### 6.6 HMI One Click Dashboard

It is possible to automatically generate an HMI dashboard with HMI Controls for all modules and parameters configured in the TwinCAT Analytics Workbench Configurator. The HMI Dashboard is based on the TwinCAT HMI and visualizes the PLC data from the runtime deployment.

The automatically generated One Click Dashboard is only available with the new HMI version 1.12. An Analytics Runtime license is required in order to use the Analytics HMI Controls.

After configuring your Analytics Workbench project, an HMI Control can be selected for each algorithm. Open the Properties window of the module and select an HMI Control from the drop-down list. You can change the display text for the title in the HMI dashboard (display text). You can also choose whether the control should be docked to the start page (Dock on Desktop). In the Solution Explorer all controls to be generated are stored under the Dashboard node.

After configuring your Analytics Workbench project, an HMI Control can be selected for each algorithm. Open the Properties window of the module and select an HMI Control from the drop-down list. You can change the display text for the title in the HMI dashboard (display text). You can also choose whether the control should be docked to the start page (Dock on Desktop). In the Solution Explorer all controls to be generated are stored under the Dashboard node.
After completion of the configuration, click the **Deploy Analytics Runtime** command in the context menu. The Deploy Wizard starts and it is possible to set up the entire required configuration step by step for use.

You can configure your HMI Dashboard on the **HMI Dashboard** tab.

Check the checkbox **Generate HMI Dashboard**. It is also possible to create only one HMI project without a PLC. Furthermore, you can also assign an **HMI Project Name** to the dashboard and set a Dashboard Title as well as the Desktop Height and Desktop Width in order to generate a tailor-made dashboard that suits your needs. The remaining configurations are explained in Dashboard Configuration [247].

As usual, the last tab shows you all the settings you have made for the generation. Now you can start the generation process by clicking **Deploy**.
The HMI generation begins immediately after runtime deployment (if selected). Each step for generating the HMI dashboard is also displayed in the overview window during the generation process.

The dashboard opens automatically in your default browser.
6.6.1  Dashboard Configuration

HMI Dashboard tab

The **HMI Dashboard** tab contains all the configurations for the dashboard.
Configuration

HMI generation Settings

- Generate HMI Dashboard
- Create only HMI Project (no PLC)
- HMI Project Name: AnalyticsHMIProjectOneClickDashboard

Dashboard Options
- Dashboard Title: HMI DashboardOneClickDashboard
- Desktop Height: 854, Desktop Width: 1520
- Create Startpage
- Show current time

Dashboard Styles
- Dashboard Layout: Without Dock, Space Saving
- Dashboard Theme: Shiny
- Select Color: Header Color, Color Gradient
- Control Style: Round

Languages
- German, Chinese, Belgian, Swedish, English, Italian, Finnish, Brazilian

HMI Server
- Publish to HMI Server
- Address: localhost
- Port: 1020
- User: SystemAdministrator
- Password: ******

Validate Connection
<table>
<thead>
<tr>
<th>Configuration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate HMI Dashboard</td>
<td>Enables the generation of the HMI Dashboard, if enabled. This automatically activates/deactivates the checkboxes Create Bootproject and Activate PLC Runtime in the TwinCAT PLC Target tab.</td>
</tr>
<tr>
<td>Create only HMI Project (No PLC)</td>
<td>Creates only one HMI project and not a PLC project, if enabled.</td>
</tr>
<tr>
<td>HMI Project Name</td>
<td>Name of the TwinCAT HMI project.</td>
</tr>
</tbody>
</table>

**Dashboard Options**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboard Title</td>
<td>Title of the HMI Dashboard, which is displayed in the dashboard header.</td>
</tr>
<tr>
<td>Desktop Height</td>
<td>Height of the target screen in pixels.</td>
</tr>
<tr>
<td>Desktop Width</td>
<td>Width of the target screen in pixels.</td>
</tr>
<tr>
<td>Create Startpage</td>
<td>Creates a start page for the dashboard that displays a map with all machine locations. The location data is adopted from the machine management data.</td>
</tr>
<tr>
<td>Show current time</td>
<td>Creates a clock in the dashboard header that shows the current local time.</td>
</tr>
</tbody>
</table>

**Dashboard Styles**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboard Layout</td>
<td>Defines the layout of the dashboard. Doc requires the “Doc” property of a module to be TRUE. Doc Left: Fixed left column, Doc Right: Fixed right column, Without Doc: No fixed column</td>
</tr>
<tr>
<td>Dashboard Sorting</td>
<td>Defines the sorting of the dashboard. Space Saving: Arranges the controls without free space, Control Type: Arranges the controls by Control type, Control Size: Arranges the controls by size from large to small, Filled: Arranges the controls so that the entire screen is filled, Network Groups: Groups the controls by network and summarizes them on a screen. Grouping begins at the defined level (0 = All, 1 = First Level...)</td>
</tr>
<tr>
<td>Dashboard Theme</td>
<td>Defines the theme of the HMI dashboard, affects controls and backgrounds: Light: Bright skin, especially for day mode, Shiny: Similar to the bright skin, color gradient in the controls, Dark: Dark skin, especially for night mode, Select Color</td>
</tr>
<tr>
<td>Control Style</td>
<td>Defines the style of the controls: Round: The controls have rounded corners, Flat: The controls have angular corners.</td>
</tr>
<tr>
<td>Change default background image</td>
<td>If enabled, a customer-specific background image can be set for the HMI Dashboard. If nothing is defined, the default image is used.</td>
</tr>
<tr>
<td>Use Logo</td>
<td>If enabled, a logo is added to the dashboard header. A customer-specific image can be defined for the logo.</td>
</tr>
<tr>
<td>Use custom Map Icon</td>
<td>If enabled, a custom map icon can be defined for the map on the start page. If nothing is defined, the default icon is used.</td>
</tr>
</tbody>
</table>

**Languages**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>Enables the enabled language for the language change in the HMI Dashboard. (Only for standard text)</td>
</tr>
</tbody>
</table>
### HMI Server

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish to TwinCAT HMI Server</td>
<td>Publishes the dashboard to a TwinCAT HMI server, if enabled.</td>
</tr>
<tr>
<td>Address</td>
<td>Enter the IP address or host name of the TwinCAT HMI server.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the port of the TwinCAT HMI server.</td>
</tr>
<tr>
<td>User</td>
<td>Enter the user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password. (The administrator password must be set on the server via the configuration page).</td>
</tr>
<tr>
<td>Validate Connection</td>
<td>Press the button to validate your server connection.</td>
</tr>
</tbody>
</table>

### Visual Studio tab

On the **Visual Studio** tab you can select which version of Visual Studio or TwinCAT XAE shell is to be used to generate PLC and HMI. In addition, it is possible to generate HMI and PLC in two different solutions.

- **Keep Visual Studio open** Leaves Visual Studio open after generation, if enabled.
- **Target VS Version** The target Visual Studio version for PLC and HMI.
- **Create HMI in another Visual Studio** If enabled, the HMI is generated into a second solution. A different version of Visual Studio can be selected.

Request: TwinCAT or TwinCAT HMI must be installed in the selected VS version.

Click **Next** to display the **Summary** tab, then click **Deploy** to start generating the dashboard.

### Impressions

### Topics
Styles

Min Max Avg HMI

最小: 24. 一月 2020 07:40:26
最大: 24. 一月 2020 07:40:27

最小 -5
最大 5
平均值 0

值 -1.15
Min Max Avg HMI

Views for mobile devices
Map on customer-specific dashboard

Historical data and machine switching
6.6.2 Mapping Wizard

The HMI Control Mapping wizard enables the following:

1. Adding your own controls.
2. Mapping controls to module classes or changing existing mappings.
3. Mapping controls to module instances or virtual inputs (Analytics project must be open)

Open the wizard via the tab TwinCAT > Analytics > HMI Control Mapping.

Additional help is available via the question mark ? in the wizard.

1. Adding your own controls

The TwinCAT 3 HMI allows you to create your own HMI Framework controls and export them as a NuGet package.
1. To assign your own framework controls to the Analytics modules, click **Import HMI Controls**.
2. Select the NuGet package via **Browse**.
3. Next, choose a name, size, and image for your control. Then click **Next** and for the last control click **Create**.

You will automatically be redirected so that you can create a mapping between your controls and Analytics modules.

2. **a) Mapping of controls to modules (further to 1.)**

   Under 2. b) this step is explained in more detail using "Binary State" control.
1. Now select your control.
2. Now select the module to which you want to assign the control.
3. Select **Continue with Mapping**.
4. In the next steps, connect the control inputs to the module data.
5. Last, click **Create** to add the mapping to the Analytics workbench.
6. You can now close the wizard and the mapping will automatically be available for the module. Select it and generate your dashboard.

![Dashboard Image]

⇒ The dashboard is ready.

2.b) Mapping controls to modules (without own controls)

Analogous to 2.a), this section describes how to perform a mapping between a control and a module. An existing Analytics Control is used as an example. You can try this example directly.
1. To do this, select the **Mapping Template** option.
2. Select a control. The properties of the control are displayed on the right. You can also see the size of the control on the dashboard.
3. Select the module from which you want to display the data. The inputs, outputs and parameters are displayed as a preview, which can then be linked to the properties of the selected control. Modules that have already been mapped are underlined. These mappings can be edited.
4. In the future, a control will be able to display data from different modules. Since this is not currently possible, select **Continue with Mapping**.
5. Next, select a control property to be mapped. All impossible variables that cannot be assigned due to their data type are grayed out. All others can be mapped. Now select the module variable that you want to map with the control property.

![Configuration Wizard](image)

6. You can also check the checkbox **Set default value** to assign a default value to the control. This can be used, for example, to change default colors or to set boolean values such as “ShowTitle” to “False” if no title is to be displayed in the Analytics control.
In this example, the "OnColor" is changed. Based on the data type, a specific selection option is provided.
7. If you do not want to add any more entries via **Add**, you can complete the mapping via **Create**. It is advisable to assign a meaningful mapping name.
8. You can now close the wizard and the mapping will automatically be available for the module. Select it and generate your dashboard.

⇒ The dashboard is ready.

3. **Mapping of controls to module instances**

In addition to mapping controls to module classes, the module instances of a project can also be directly linked to controls. This is possible via the Dashboard node [283] and via the wizard.
1. At the start of the wizard, click **HMI Dashboard Configurator**.
2. There you have several choices.

1. Overview and editing options for all controls
   - This function is also available by double-clicking a control in the Solution Explorer.

2. Adding a control that displays your input data
   - This function is also available via the Dashboard node [p. 283].

3. Adding a control that displays module data (selection via a template)
   - This function can also be set for an individual module via the Properties window. This is the only way to display data from several modules.
   - You can see directly which data from the template are linked and can adjust them directly.

4. Adding a control that displays data from modules (fully manual)
   - This function is also available via the Dashboard node [p. 283].

For the individual points you have to follow the steps of the wizard. The steps are the same or very similar to section "2. Mapping controls to modules".

### 6.6.3 Configuration at Runtime

**TwinCAT 3 HMI project**

The result of dashboard generation is a complete TwinCAT 3 HMI project. Therefore, all options offered by TwinCAT 3 HMI Engineering can be used and included.
During runtime, the dashboard can be configured on the Options page and in the dashboard header:

**Options page**

**Layout**
- Select dashboard theme: Change the dashboard theme between shiny, dark and light.
- Select control style: Change the control style between flat and round.
- Show background image: Displays the standard or customer-specific background image

**Networks**
- Reset buttons for networks: Enables reset buttons for entire networks. Reset all controls within the selected network.

**Controls**
- Reset buttons for controls: Enables reset buttons for controls. Reset a single control.
- Show control titles: Enables control titles for all controls.
6.6.4 Switching multiple machines in the HMI Dashboard

In TwinCAT Analytics you can use different data streams from several machines, which you can switch in an analysis. This is possible both in the Analytics Workbench via the Virtual Input Source and in the fully generated PLC and HMI. Both live or historical data can be used. For each data stream, you can add a brief description and the location either in the Analytics Logger or in the Machine Administration.

Machine Administration page

If you have not already done so in the Overview, set the metadata of your machine on the Analyse data page. Open the TwinCAT Target Browser (TwinCAT > Target Browser >> Target Browser) and click the gear icon. Now you can enter the location of your machine, a short description and the name of your machine. Note that existing data in an Analytics project is not assigned this meta information, since the information is only transferred from the Target Browser during drag and drop. In this case, you can delete the data stream and recreate it.

For each data stream you use in your Analytics configuration, the Runtime deployment creates a map entry. These map entries are used as input variables for the general map on the start page of your HMI dashboard.
In the Analyse data [24] the data streams are listed under Sources. These are created by using data from various sources from the TwinCAT Target Browser. These sources are listed in the Virtual Input Source. It is possible to switch between the sources. Check that all individual inputs are linked and that none is set to Empty by clicking each source once (which corresponds to a switch in the workbench). You can then generate an HMI with PLC.

After a successful HMI and PLC generation, the names, positions, and descriptions are entered into the global variable list of the PLC. If something is not right or you want to change a value afterwards, you can do this directly in the PLC.
Map with machines

If you selected to create a start page, the Map Control will be created on the home page of your HMI Dashboard. The map shows all machine locations and lists the names of the machines (system alias) in the legend. The legend can be opened and closed using the arrow on the right. The icon color indicates the current machine status: green = OK, yellow = Warning, red = Alarm.

You can zoom in on a single icon by double-clicking it and zoom out to an overview of all machines using the icons on the left. It is also possible to click on an icon on the map. The corresponding legend entry is automatically highlighted. This also works the other way around, i.e. when you click on a legend entry, the corresponding icon is automatically highlighted.
Switching machine data

In addition to the map with the individual machines, the analyses for the individual machines can also be switched. In the PLC all analyses run in separate tasks, which means that all analysis data is available at the same time. In the generated HMI, the various machines can be switched over in a specially developed control. The control can be opened and closed via the icon highlighted in the image below.

The machines that were previously shown on the map with the locations can be selected in the control. A machine can consist of live and historical data; for Production Verl only live data is available. When you click on another machine, it is initially only displayed in the control.

1. The currently active machine.
2. Button to activate the selected machine.
3. Selection window for minimizing (and automatic activation of the selected machine).
For the machine **Production Berlin** only historical data is configured. Historical data can be analyzed in the HMI in the same way as in Working with Historical Data. There are several interaction options for configuration and use.

1. Switching between live and historical data of a machine (if available).
2. Reloading the record list. This allows data recorded subsequently to be dynamically loaded and analyzed in the dashboard.
3. Selection of the start and end times via a slider.
4. Selection of the start and end date via a selection window with calendar.
5. Resetting the start and end times.
6. Starting the selected analysis.
7. Canceling the analysis.

The slider is ideal for quick tests and approximate setting of the start and end times. For a more precise setting down to the millisecond, click the **text box**. This opens a selection window with a calendar for selecting the day.
The analysis is triggered in the PLC by clicking the **Start** button. The button turns yellow until the analysis starts. It turns green when the first data appears. A marker also shows the current time of the historical analysis. The process speed depends on the bandwidth, the number of data and the recorded cycle time. It is possible to analyze several historical recordings at the same time by simply switching machines. Internally, all analyses run in parallel. This makes it easy to switch between analyses in the dashboard.

The **Minimize** button in the top right corner hides the selection of the machine and, if available, the record. This allows you to work with a record and view the data at the same time. You can hide the window completely by clicking the blue icon at the top. Only the machine name is always shown in the bottom right corner of the dashboard.
6.6.5 Change the language

In the Deploy wizard configuration window, up to 8 languages can be selected for language switching in the HMI area.

The texts in the supplied Analytics Controls and all other texts can be switched automatically. Only your network and module names have to be translated, if you want them to be included in language switching. In the following screenshot, the main affected names are marked in the Solution Explorer.
In the generated TwinCAT HMI Engineering project this is easily possible, since the translation entries are already prepared. The texts for the respective languages are stored in the Localization files. The names of your networks and modules are automatically entered there and only have to be translated. To do this, open all the files of the languages for which you need a translation. The following screenshot lists the entries that need to be translated for German. The number of entries to be processed varies depending on the complexity of the Analytics project.

Once completed, you can open the dashboard by clicking the Google Chrome button (or the name of your default browser). Simply reloading in the browser is not sufficient, because the project has to be rebuilt internally. In the following image the texts which are now also switched with the language switch are highlighted.
6.6.6 Dashboard node

An Analytics project has a configuration of HMI contents (pages) and HMI controls (display elements) that are created during dashboard generation. This configuration can be viewed and changed via the Dashboard node in the Solution Explorer. You can rename the contents and controls at any time, move them to other contents via drag & drop, copy them (Ctrl-C, Ctrl-V) or delete them. A control can also be edited by double-clicking on it.

Each Analytics module has existing control mappings, which can be selected via the Properties window (you can also create/edit these yourself in the Mapping Wizard [254] (point 2). Once a control is selected for a module, it is listed under the Dashboard node.
You can create new content for the **Dashboard** and **Content** nodes by right-clicking.

Likewise, you can add **new controls** by right-clicking on a **Content** node.
This opens the **Analytics Dashboard Wizard**. This wizard guides you step-by-step through the configuration for adding a control.

### NOTE

#### Analytics Dashboard Wizard

Familiarize yourself with the wizard by adding a control (select a control from the Properties window of the module). You can then look up the entire configuration in the wizard by double-clicking on the control. There you can follow the individual configuration steps.

**Analytics Dashboard Wizard**

Select a control. All available controls are listed on the left. By default, only controls that are not algorithm-specific are listed. All controls can be made available by unchecking the **Show only default Controls** checkbox.
1. Select **Single Value** to display a single value in the dashboard. Click the **Next** button to continue.
2. You can now link data from one module (1-Module), several modules (N-Modules) or from virtual inputs (Virtual Inputs). For this example, select 1-Module.
3. All modules from the Analytics project are listed. Select the module from which you want to display the data. A preview of the selected module is available on the right.
4. Here the first control property is linked to a module variable. To do this, select the **Value** to be displayed with the variable **Max**.
5. This overview page lists all existing links. Add another link with **Add**.
6. Select **Unit** and check **Set default value** to assign a static value. A text field opens on the right; enter °C.
7. Change the title of the control to **Max Temperature** and add the control mapping via **Create**.

8. The **Max Temperature** control that was created appears in the **Dashboard** node. **Right-click > Rename** to rename the content.
After a successful HMI and PLC generation, you can open the previously created My Custom Page via the navigation. There you can see the manually created control.
7

Samples

7.1 Pick&Place Application

The following example shows an analysis of an XTS application together with a pick&place robot. It offers you two options:

1. You can use the sample data and build your own TwinCAT Analytics project.
2. You can use the ready-to-use TwinCAT Analytics project in order to understand how TwinCAT Analytics works.

In both cases, it makes sense to understand the machine application that provides the data for the analysis. The application sequence is shown in the video:

Video: https://infosys.beckhoff.com/content/1033/TE3500_TC3_Analytics_Workbench/Resources/mp4/6815846411.mp4.

Application description

This application example describes a simulation for the processing of production goods. The application transports workpieces from a store to a drill via a Beckhoff XTS system. Depending on the color of the workpiece, a different pattern is milled into each part. The XTS system shown has a length of 4000 mm. It consists of ten movers, where two movers carry a tray together. A pick&place robot (1) is positioned at the workpiece store (2). Two red and three yellow parts are located in five store positions, which the picker can pick up or set down via a vacuum pump. On the opposite side of the XTS there is a vision system (3) for identifying the workpiece color and a miller (4) for the specific milling contour.

The trays drive to the workpiece store. The picker picks up the parts and loads or unloads the trays. The tray then drives to the scanner, where the color of the workpiece is scanned and communicated to the downstream miller. Depending on the color, the workpieces are processed differently: a circle is milled into yellow parts, and an oval shape is milled into red parts. The workpiece then drives back to the store and is stored there by the picker. The tray is loaded with another part and drives onwards.

Variable description

The following variables are recorded by the TwinCAT Analytics Logger in the attached Analytics File Format. You can use the data for your own Analytics configurator project or for the default configuration in the tnzip file provided. The meaning and function of each variable are shown in the following table.
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bMillerMilling</td>
<td>BOOL</td>
<td>Information regarding the activity of the drill: 1, when the drill is active and 0, when the drill is inactive.</td>
</tr>
<tr>
<td>bMillerMovingZ</td>
<td>BOOL</td>
<td>Information regarding the movement of the drill in z-direction: Is 1, when the drill is moving in z-direction and 0, when he is not moving in z-direction.</td>
</tr>
<tr>
<td>bMillerSpindleRotation</td>
<td>BOOL</td>
<td>Information regarding the activity of the drill needle: Is 1, when the needle is moving and the drill is active and 0, when it is not moving.</td>
</tr>
<tr>
<td>bNewWorkpieceColorScanned</td>
<td>BOOL</td>
<td>Registers if a new colour has been scanned (=1), and then will inform the drill. If the colour is changing the drill pattern has to be changed. When the colour is still the same (=0) the drill pattern will not change.</td>
</tr>
<tr>
<td>bPickerHoldingWorkpiece</td>
<td>BOOL</td>
<td>Is 1, when the picker is holding a component and 0, when not.</td>
</tr>
<tr>
<td>bPickerInStorage1Position</td>
<td>BOOL</td>
<td>Is 1, when the picker is in storage position 1.</td>
</tr>
<tr>
<td>bPickerInStorage2Position</td>
<td>BOOL</td>
<td>Is 1, when the picker is in storage position 2.</td>
</tr>
<tr>
<td>bPickerInStorage3Position</td>
<td>BOOL</td>
<td>Is 1, when the picker is in storage position 3.</td>
</tr>
<tr>
<td>bPickerInStorage4Position</td>
<td>BOOL</td>
<td>Is 1, when the picker is in storage position 4.</td>
</tr>
<tr>
<td>bPickerInStorage5Position</td>
<td>BOOL</td>
<td>Is 1, when the picker is in storage position 5.</td>
</tr>
<tr>
<td>bPickerInXtsPosition</td>
<td>BOOL</td>
<td>Is 1, when the picker is Xts position, which means over the Xts.</td>
</tr>
<tr>
<td>bPickerMoving</td>
<td>BOOL</td>
<td>Information, when the picker is moving (=1) or is inactive (=0).</td>
</tr>
<tr>
<td>fMillerPositionZ</td>
<td>LREAL</td>
<td>Indicates the z-position of the drill. At a value of 224 the drill is in drill position, at 245 he is inactive.</td>
</tr>
<tr>
<td>fMillerSpindleRotation</td>
<td>LREAL</td>
<td>Indicates the amount of revolutions per minute of the drill needle.</td>
</tr>
<tr>
<td>nColorOfLastScannedWorkpiece</td>
<td>INT</td>
<td>Registers the colour of the last scanned component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 → yellow component</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 → red component</td>
</tr>
<tr>
<td>nCurrentProductionTime</td>
<td>UINT (64 Bit)</td>
<td>Registers the current production time.</td>
</tr>
<tr>
<td>nVacuumPressure</td>
<td>DINT</td>
<td>The picker grasps the components via compressed air. This variable indicates the current pressure in mbar.</td>
</tr>
<tr>
<td>stMachineVibrations</td>
<td>STRUCT with ARRAY of LREAL</td>
<td>The structure stMachineVibrations indicates the vibrations of the machine base in the unit m/s² with oversampling.</td>
</tr>
<tr>
<td>aMachineVibrations[1...10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stPickerPosition</td>
<td>STRUCT with ARRAY of LREAL</td>
<td>The structure stPickerPosition contains an array with three values. The values indicate the x-/y- and z-position of the picker.</td>
</tr>
<tr>
<td>aPickerPosition[1...3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stTableInMillingPosition</td>
<td>STRUCT with ARRAY of BOOL</td>
<td>The structure stTableInMillingPosition contains an array with five values, each value stands for a tray: value 1 for tray 1, value 2 for tray 2... if the value is 1, the tray is in drill position.</td>
</tr>
<tr>
<td>aTableInMillingPosition[1...5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stTableInStoragePosition</td>
<td>STRUCT with ARRAY of BOOL</td>
<td>The structure stTableInStoragePosition contains an array with five values, each value stands for a tray: value 1 for tray 1, value 2 for tray 2... if the value is 1, the tray is in the component storage.</td>
</tr>
<tr>
<td>aTableInStoragePosition[1...5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>stWorkpieceColors</td>
<td>STRUCT with ARRAY of INT</td>
<td>The structure stWorkpieceColors contains an array with five values, each value stands for one component and indicates its colour (as Int-value): Component 1: yellow Component 2: yellow Component 3: red Component 4: red Component 5: yellow</td>
</tr>
<tr>
<td>aWorkpieceColors[1...5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stWorkpiecePositions</td>
<td>STRUCT with ARRAY of INT</td>
<td>The structure stWorkpiecePositions contains an array with five values for each component. The Int-value indicates the current position of the component: 1: Storage 1 2: Storage 2 3: Storage 3 4: Storage 4 5: Storage 5 6: Tray 1 7: Tray 2 8: Tray 3 9: Tray 4 10: Tray 5 11: Picker</td>
</tr>
<tr>
<td>aWorkpiecePosition[1...5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stXtsMoverAcceleration</td>
<td>STRUCT with ARRAY of LREAL</td>
<td>The structure stXtsMoverAcceleration contains an array with ten values for the movers 1-10 and indicates the acceleration of the particular mover.</td>
</tr>
<tr>
<td>aXtsMoverAcceleration[1...10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stXtsMoverMovingNegative</td>
<td>STRUCT with ARRAY of BOOL</td>
<td>The structure stXtsMoverMovingNegative contains an array with ten values for the movers 1-10 and is 1, when the particular mover is moving backward.</td>
</tr>
<tr>
<td>aXtsMoverMovingNegative[1..10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stXtsMoverMovingPositive</td>
<td>STRUCT with ARRAY of BOOL</td>
<td>The structure stXtsMoverMovingPositive contains an array with ten values for the movers 1-10 and is 1, when the particular mover is moving forward.</td>
</tr>
<tr>
<td>aXtsMoverMovingPositive [1...10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stXtsMoverPositionsX</td>
<td>STRUCT with ARRAY of LREAL</td>
<td>The structure stXtsMoverPositionsX contains an array with ten values for the movers 1-10. The value indicates the x-position of the particular mover on the XTS.</td>
</tr>
<tr>
<td>aXtsMoverPositionsX[1...10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stXtsMoverVelocity</td>
<td>STRUCT with ARRAY of LREAL</td>
<td>The structure stXtsMoverAcceleration contains an array with ten values for the movers 1-10 and indicates the speed of the particular mover.</td>
</tr>
<tr>
<td>aXtsMoverVelocity[1...10]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**How to get the data**

In the download area you will find a 30-minute recording of the XTS system that can be used for further analysis. To access these data, you must make the data available in a historical stream using the Analytics Storage Provider. A message broker should be made available for this. For the samples prepared, all components must be running on the local system.
Install the Analytics Storage Provider. Follow the instructions given.

1. Copy the Analytics File Data in the Storage Provider (see Import of Analytics Files).
2. Use the Analytics Storage Provider Configurator to start the Windows service.
3. Open the TwinCAT Target Browser in the tcXaeShell or in Visual Studio. Select the TcAnalytics tab and your message broker. In the tree on the left-hand side you will now find an entry for the historical data, which will be displayed in the symbol list on the right-hand side when selected.
4. Connect individual symbols to the inputs of their Analytics algorithms.
5. Start the analysis.

**Analysis with the Analytics Workbench**

The analysis that is the subject of the TwinCAT Analytics project from the download area is explained below. Download and open the finished project.

Select the Virtual Mappings node and also browse for the topic specified by your storage provider in the Target Browser. Select the data and link them to the appropriate virtual inputs in the editor.
Description of the Analytics Project
<table>
<thead>
<tr>
<th>Network name</th>
<th>Used variables</th>
<th>Used Algorithms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1_Storage_Picker</td>
<td>bPickerinStoragePosition, bPickerHoldingWorkpiece, bPickerInXtsPosition, bPickerInHomePosition, bPickerMoving, stPickerPosition</td>
<td>Edge Counter, Math Operation, Edge Counter OnOff, Logic Operation Counter, XY Shape Monitor, Min Max Avg</td>
<td>Picker: This network analyses the picker actions. It counts the quantities if the picker holds a component, is in home position or in XTS position. It is also analyzed how often the picker is transporting a component, how often the picker is loading a mover as well as its xyz-position. How often does the picker load a component? Storage: This network analyses the storage use of the five storage positions next to the picker. The quantities and proportions of storage uses are counted. Is each storage is used equally?</td>
</tr>
<tr>
<td>2_Components</td>
<td>bNewWorkpieceColorScanned, bMillerMilling, nColorOfLastScannedWorkpiece, stWorkpiecePosition</td>
<td>Edge Counter, Logic Operation Counter, Threshold String Classifier, Histogram</td>
<td>This network analyses the components itself. The quantity how many components of each color are processed at the scanner and the miller and the current position of each component can be detected. The histogram algorithms show the distribution of the component positions. How many red workpieces were processed?</td>
</tr>
<tr>
<td>3_Mover</td>
<td>stTableInMillingPosition, sXtsMoverPosition</td>
<td>XTS Velocity Analysis, XTS Distance Integrator, Min Max Avg Interval, Math Operation, Threshold Classifier, Lifecycle Analysis, Logic Operation Counter</td>
<td>Distance: This network analyses the distance travelled of each tray and calculates the overall distance of all movers. As the length of the XTS system is known (4000 mm), the average amount of rounds for each tray can be derived. The Threshold Analysis gives a warning after 3 rounds and an alarm after 4 rounds. Which distance did all trays travel in total? Tray position: This network analyses how often the trays are stopping at the storage or at the miller and stops how long it takes a tray to move one round. How often did all trays stop at the miller? Velocity: This network analyses the current velocity of the trays as well as the minimum and maximum velocity in the last 20 seconds. The trays can move backwards. Therefore, the minimum velocity can be negative. What is the maximum velocity you can read?</td>
</tr>
<tr>
<td>Network name</td>
<td>Used variables</td>
<td>Used Algorithms</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>4_Miller</td>
<td>bMillerMilling, bMillerSpindleRotating, bPickerMoving</td>
<td>Event Timing Analysis, Lifetime Analysis</td>
<td>This network does analysis for the miller. Drilling time: Time the miller is active. Moving time of the drill bit: Time the drill bit is active. Moving time of the driller: Time the driller is active. Which of the three actions did the miller as longest?</td>
</tr>
<tr>
<td>5_System</td>
<td>bNewWorkpieceColorScanned, bMillerMilling, nVacuumPressure, stMachineVibrations</td>
<td>Min Max Avg, Min Max Avg Interval, Timing Analysis, Math Operation Lifecycle Analysis, Threshold Classifier, Continuous Piece Counter</td>
<td>Productivity: The productivity is analyzed by counting the processed components per 20 seconds and comparing it to previous intervals. Based on the maximum count the productivity of the following intervals is derived. What is the maximum count of components in an interval? Technical components: This network analyses the vacuum pressure of the picker and gives a warning if the pressure comes under -836. How can you count the amount of warnings?</td>
</tr>
</tbody>
</table>

**PLC code generation and HMI**

On completion of the analysis in the Analytics Workbench, you can generate PLC code and an HMI dashboard (see also: 24h Analytics application [p. 28]):

1. Double-clicking the project node opens the overview page. Select deploy Runtime there.

2. The **Deploy Wizard** opens, which guides you step by step. If no individual settings are desired, you can simply proceed.

3. Optional: On the PLC tab, select a CPU core on which the analysis should be executed. An isolated core must be selected for execution in a virtual machine.

4. On the HMI Dashboard tab, enable generation. Adapt the design to suit your taste.

5. After subsequent confirmation via the **Deploy** button, a new development environment opens. A TwinCAT XAE project and a TwinCAT HMI project are automatically generated.

6. At the end of the generation, a dashboard opens in the browser. You can observe your original networks via the navigation.

7. You have to start the analysis in order to analyze your historical data. To do this, open the corresponding option element via the third icon from the right at the top.

8. Select the time range of your recorded data that is to be analyzed and start the recording in order to track the analysis results.
Download the finished PLC code generation

If you only want to view a fully generated Analytics PLC project with the corresponding HMI, a download (PickPlaceAnalyticsSolution.zip) is also available. In order for this project to be executable, the main topic and the identification GUID of the storage provider may have to be adapted in the code. Appropriate warnings are output, allowing you to navigate conveniently to the appropriate location within TwinCAT, where you will also find indications where the information is available.
Downloads

- https://infosys.beckhoff.com/content/1033/TE3500_TC3_Analytics_Workbench/Resources/zip/9007206180112139.zip
- https://infosys.beckhoff.com/content/1033/TE3500_TC3_Analytics_Workbench/Resources/zip/8817652491.zip
- https://infosys.beckhoff.com/content/1033/TE3500_TC3_Analytics_Workbench/Resources/zip/8817654667.zip
Appendix

8.1 FAQ - frequently asked questions and answers

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

1. Are open source software components used in TwinCAT Measurement products? [303]

Are open source software components used in TwinCAT Measurement products?

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