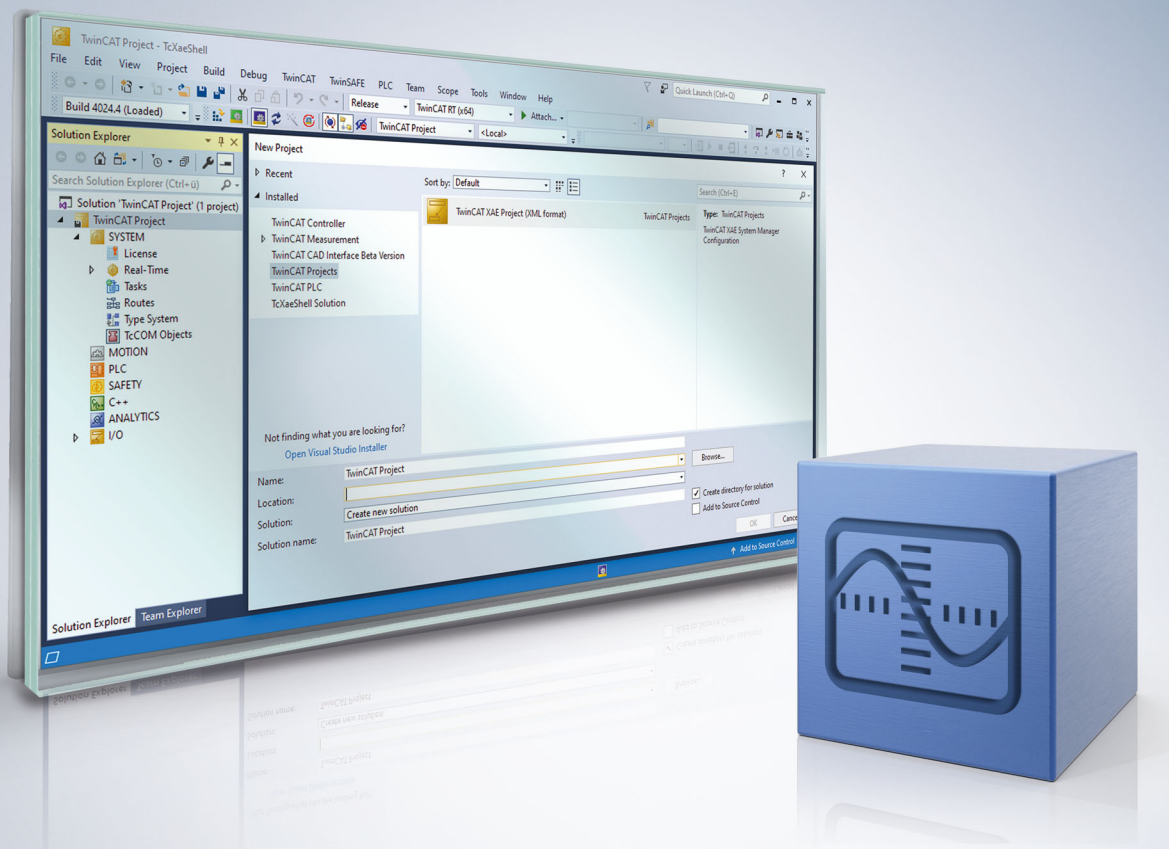
BECKHOFF New Automation Technology

Manual | EN

TF3510

TwinCAT 3 | Analytics Library



2023-07-18 | Version: 1.3.0

Table of contents

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

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

Patent Pending

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

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

Copyright

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

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

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Overview

The TwinCAT Analytics Library is a collection of base analysis algorithm. Functions such as time and life cycle analysis, peak detection, comperators, basic math operations, logical operators and much more are available. This library can be used as standard PLC library for your machine application, but also in a TwinCAT Analytics Runtime (TF3550) for a central PLC-based 24h analysis application. In the second case the library will be used in an automatic PLC code generation powered by TE3500 TwinCAT Analytics Workbench.

Components

- Driver TcAnalyticsKernel.sys
- PLC Library Tc3_Analytics.compiled-library

List of algorithm

FB_ALY_MathOperation [▶ 198]
FB_ALY_MathOperation_1Ch [▶ 201]
FB_ALY_RMS_1Ch [▶ 204]
FB_ALY_SlopeAnalysis_1Ch [▶ 208]
Statistics
FB_ALY_CorrelationFunction [▶ 215]
FB_ALY_CorrelationFunctionReference [▶ 224]
FB_ALY_StandardDeviation [▶ 248]
FB_ALY_LinearRegressionFitting [▶ 236]
FB_ALY_LinearRegressionInference [▶ 243]
Training Base
FB_ALY_TimeBasedTeachPath_1Ch [▶ 263]
Wind Turbine
FB_ALY_WtTurbulence_1Ch [▶ 290]
XTS
FB_ALY_XtsAccelerationAnalysis_1Ch [▶ 282]
FB_ALY_XtsDistanceIntegrator_1Ch [▶ 285]
FB_ALY_XtsVelocityAnalysis_1Ch [▶ 287]
XY Path Analysis
FB_ALY_XyGateMonitor_2Ch [▶ 269]
FB_ALY_XyShapeMonitor_Circle_2Ch [▶ 273]
FB_ALY_XyShapeMonitor_Rectangle_2Ch [▶ 276]
FB_ALY_XyShapeMonitor_Triangle_2Ch [▶ 279]

3 Installation

The setup of TE3500 Analytics Workbench is providing the TwinCAT Analytics library. The Workbench self is not necessary to use the TF3510 product. It is enough to install TE3500 on the engineering system in “demo” mode. Important is the licence for the target system, all necessary driver components will be downloaded to the target system by activation of the configuration and download of the project.

3.1 System requirements

Technical data	TF3500 TwinCAT 3 Analytics Library
Target System	Windows 10, WinCE, TwinCAT/BSD PC (x86, x64 and ARM)
Min TwinCAT version	3.1.4022.29
Min. TwinCAT level	TC1200 TC3 | PLC

3.2 Installation

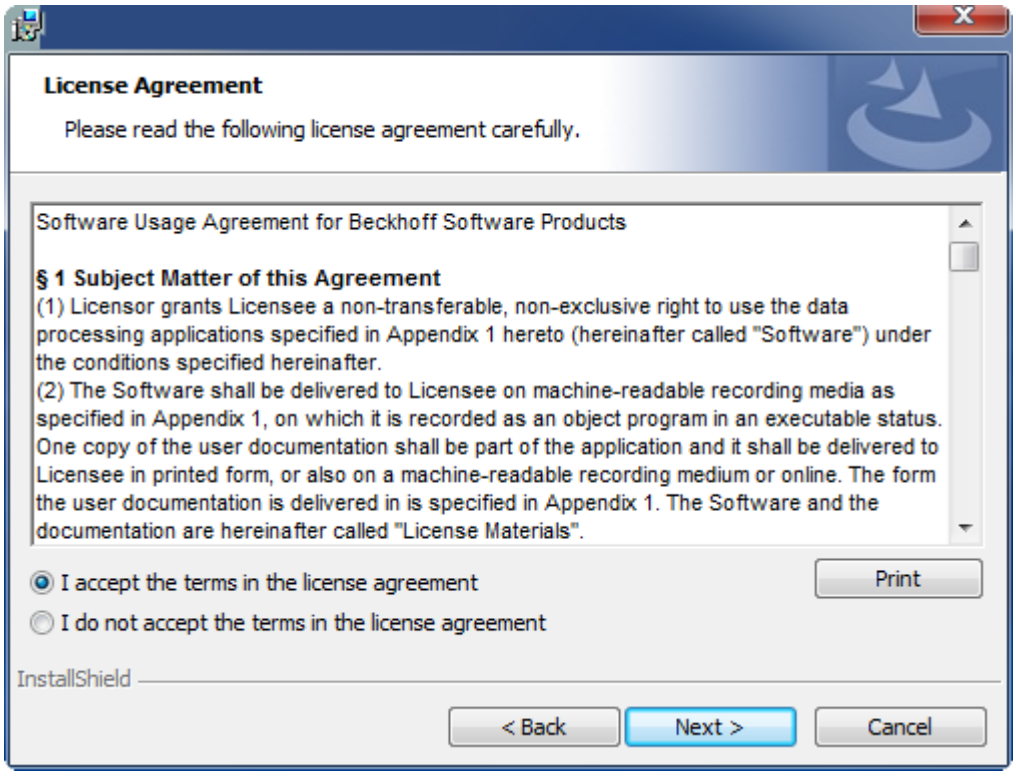
The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.

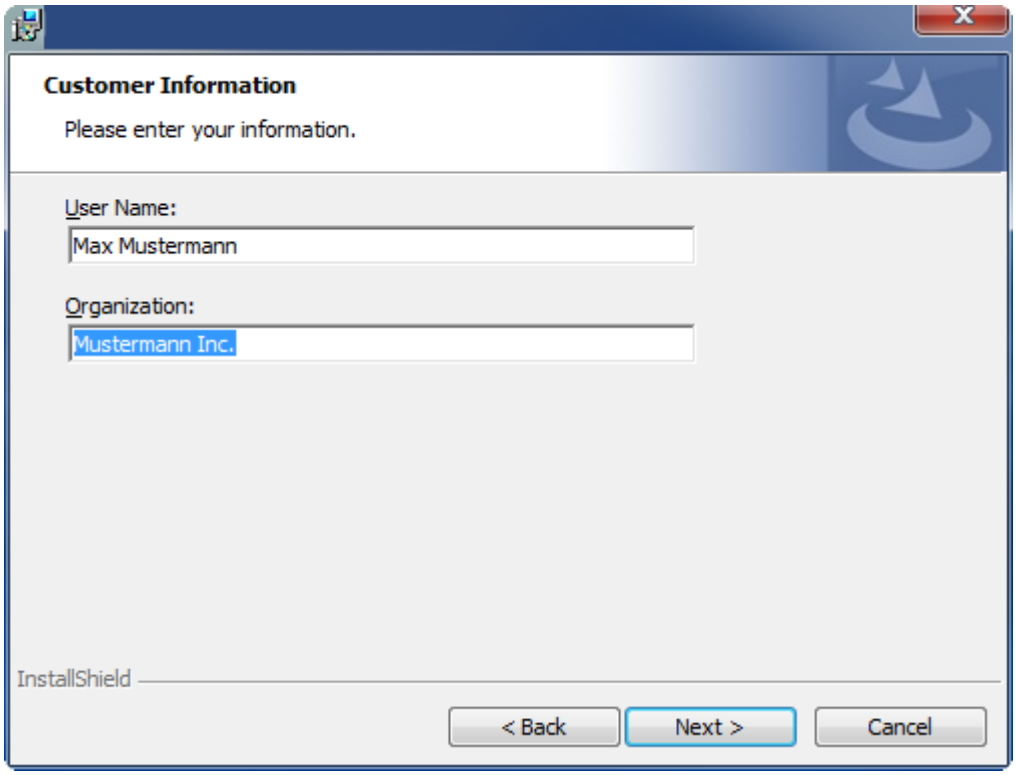
1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.

   ⇨ The installation dialog opens.

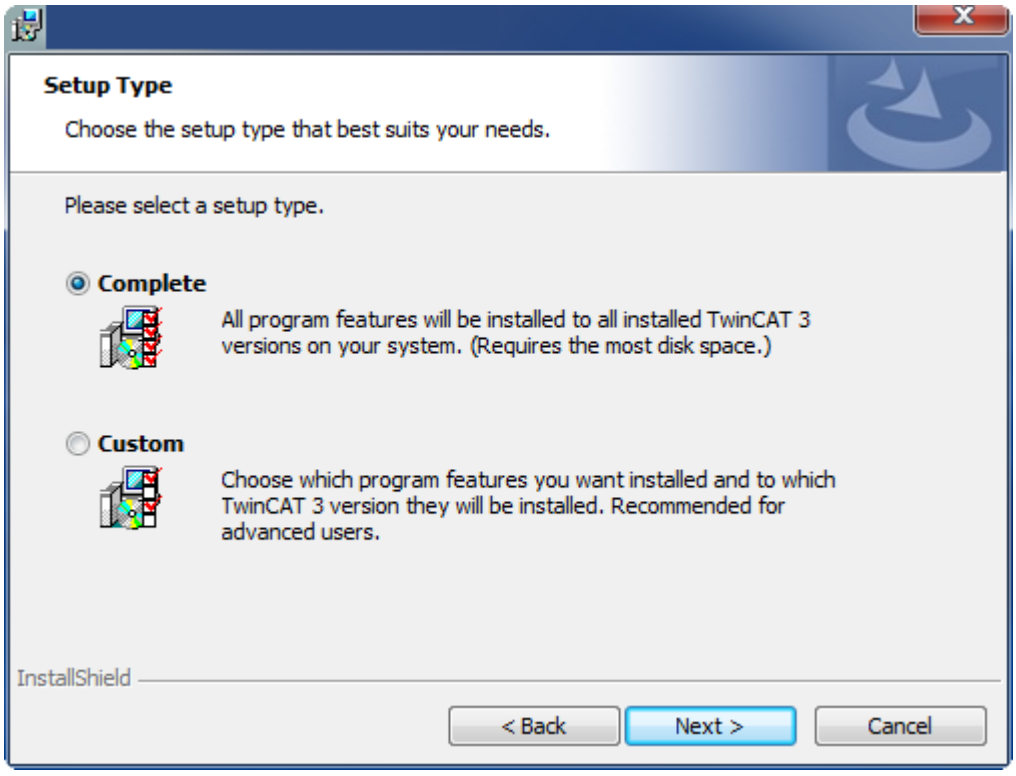
2. Accept the end user licensing agreement and click **Next**.



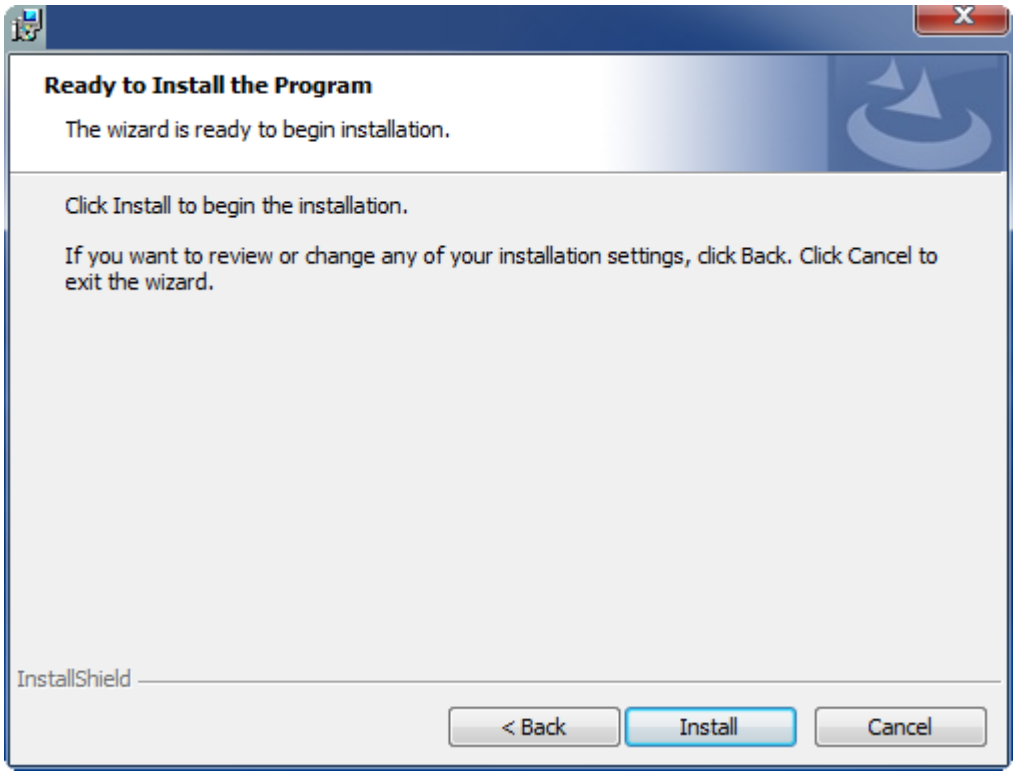
3. Enter your user data.



4. If you want to install the full version of the TwinCAT 3 Function, select **Complete** as installation type. If you want to install the TwinCAT 3 Function components separately, select **Custom**.

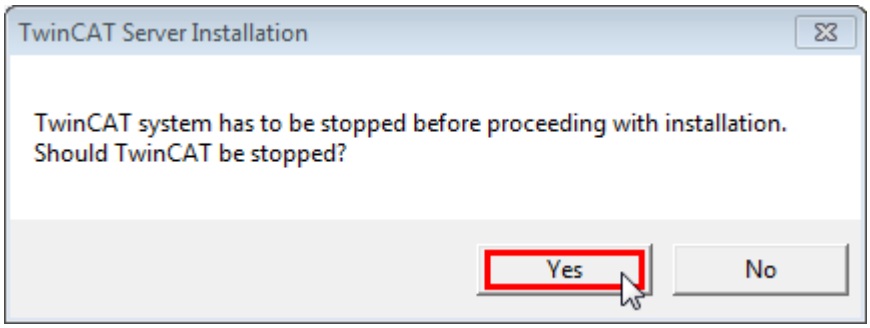


5. Select **Next**, then **Install** to start the installation.

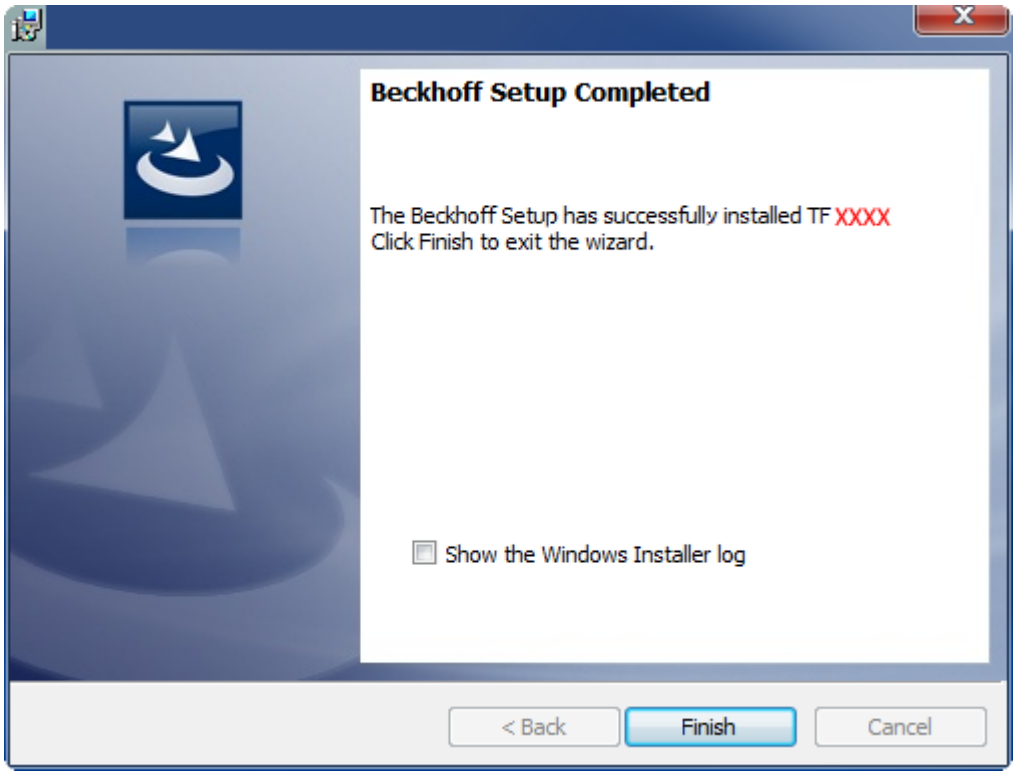


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

6. Confirm the dialog with **Yes**.



7. Select **Finish** to exit the setup.



⇨ The TwinCAT 3 Function has been successfully installed and can be licensed (see Licensing [▶ 15]).

3.3 Licensing

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

Licensing the full version of a TwinCAT 3 Function

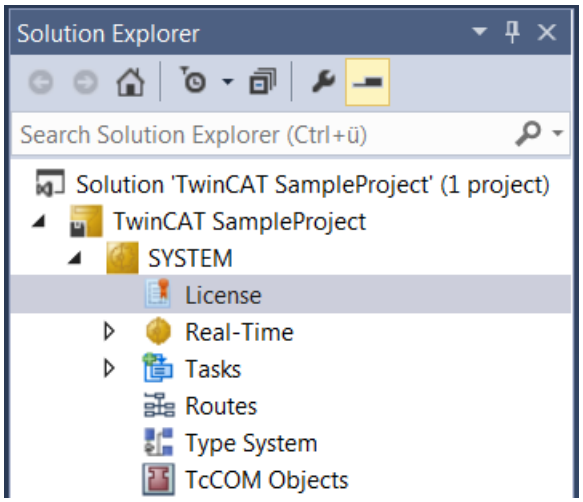
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

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

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

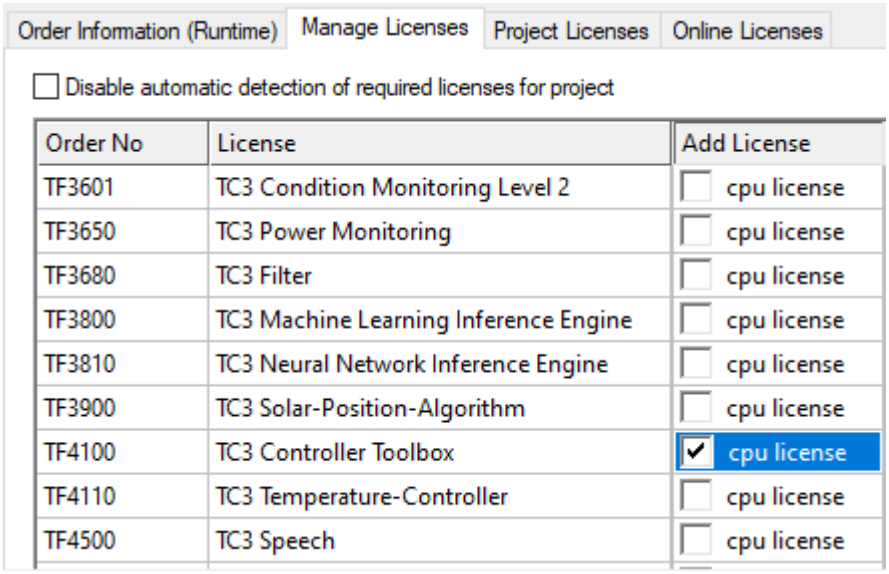
1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the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

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



⇨ The TwinCAT 3 license manager opens.

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

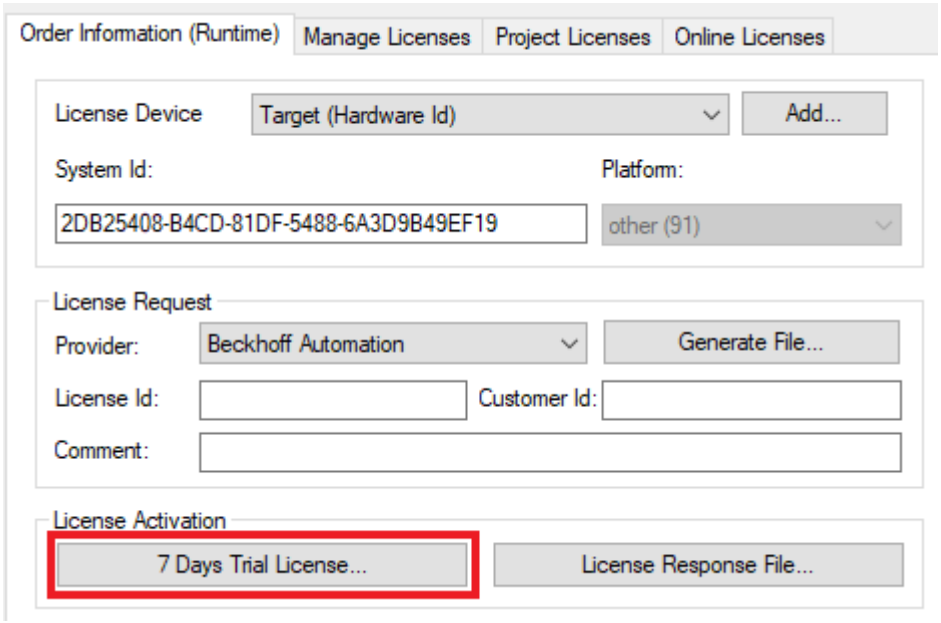


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu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

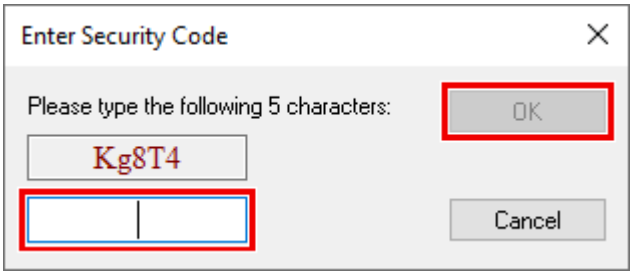
6. Open the **Order Information (Runtime)** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

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



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



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

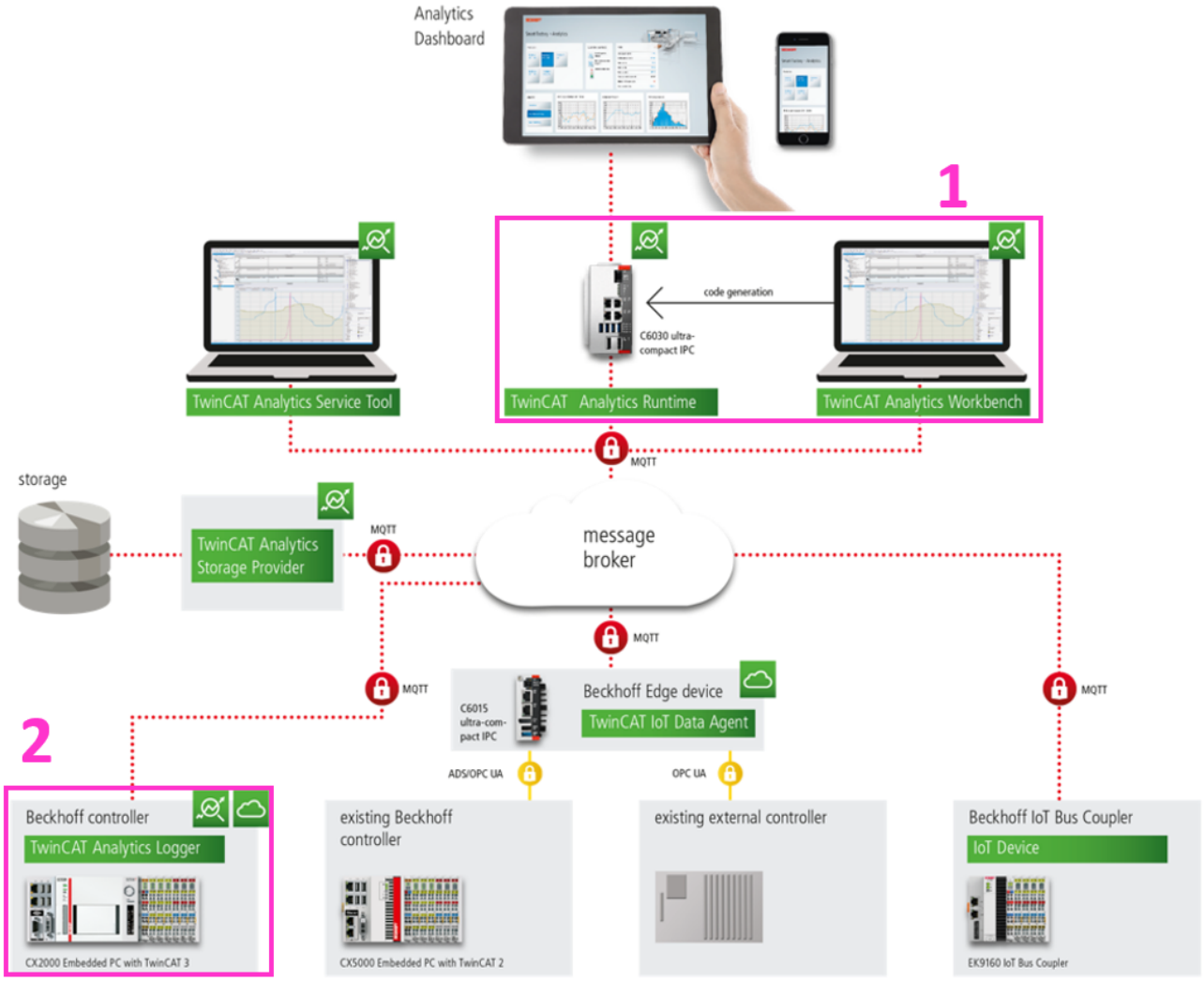
⇨ The 7-day trial version is enabled.

4 Technical Introduction

There are two main use cases for the TwinCAT Analytics Library.

1. The library is used by the automatic PLC code generation of the TwinCAT Analytics Workbench. All base algorithm of the Toolbox in the Analytics configurator are also available in the TwinCAT Analytics Library. The generated code can be downloaded to the Analytics Runtime, where it runs 24h 7 days a week parallel to the applications in the field.
2. The library can be used also as standard PLC library on the target device in the field. In fact, it is not necessary to use the library always together with MQTT communication. You can take also just local variables and put them to the algorithm of the library. This you can see in the sample code [▶ 353] of this document.

Where it works in the TwinCAT Analytics workflow is shown in the picture below.



Princip and handling

The function blocks of the algorithm provide different methods. With the exception of a few algorithms, each function block has a configuration method. This method will be called first to configure the algorithm with e.g. thresholds or operators etc. An additional “SetChannelValue” method provide the specific input variable to the algorithm. The input value is not processed until the “Call” method has been invoked. Therefore every function block has a call method which starts finally the calculation. The results are outputs at the body of the function block.

Timestamps

The timestamps and timespans in a TwinCAT analytics system are based on the DC time used in EtherCAT. Timestamps are expressed in nanoseconds since 1.01.2000 UTC. They are represented in a unsigned 64 bit integer value. Timespans are represented in a signed 64 bit integer value. For a better visualization the algorithms provide function blocks in their outputs to store timestamps or timespans. For timespans the function block FB_ALY_DateTime [▶ 315] is used. Timespans are stored in the function block FB_ALY_Timespan [▶ 318]. By using this function blocks it is easy to calculate with or to provide them as strings for an HMI application.

5 PLC API

5.1 Function blocks

5.1.1 Algorithms

5.1.1.1 Base

5.1.1.1.1 FB_ALY_BatchNShift_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_BatchNShift_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initialize the number of output channels (size of the internal buffer).
GetChannelOutputValue()	Local	Method for getting individual output values from the output array.
GetOutputArray()	Local	Method for getting the entire output array.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbBatchNShift : FB_ALY_BatchNShift_1Ch(nBufferSize := 100);
    eSampleMode : E_ALY_SampleMode := E_ALY_SampleMode.Flow;
    bConfigure : BOOL := TRUE;
    nInput : INT;
    aBuffer : ARRAY[1..100] OF LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbBatchNShift.Configure(eSampleMode);
END_IF

// Call algorithm
fbBatchNShift.SetChannelValue(nInput);
fbBatchNShift.Call();
fbBatchNShift.GetOutputArray(ADR(aBuffer), SIZEOF(aBuffer));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.1.1 FB_init

Initialization of the buffer size.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nBufferSize: UDINT := 1;
END_VAR
```

Inputs

Name	Type	Description
nBufferSize	UDINT	Indicates the size of the buffer and thus the number of values that are stored. The number of output channels equals the buffer size.

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.1.1.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.1.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
    eSampleMode : E_ALY_SampleMode;
VAR_INPUT
END_VAR
```

Inputs

Name	Type	Description
eSampleMode	E_ALY_SampleMode	The values from the buffer can be passed to the output channels in two different modes: *Flow*: The buffer is filled like a ring buffer. At the start of the analysis all output values are set to zero. Each change to the ring buffer is transferred to the output channels immediately. The New Result flag is set to TRUE, once all output channels got assigned a value and is always true, when a new value is saved in the buffer. *Wait*: At the start of the analysis or after reset all output channels are set to zero. Only when the internal buffer is full, these values are transferred to the output channels and the New Result flag is set to TRUE. These values stay as output values until all the values in the internal buffer are renewed. Only then they are transferred to the output channels.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.1.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.1.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.1.6 GetChannelOutputValue

Fetching a channel-specific output value. The output value is only updated if the Call() method was called previously.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.1.7 GetOutputArray

Getting the entire output array. The number of elements corresponds to the configured buffer size.

Syntax

Definition:

```
METHOD GetOutputArray : BOOL
VAR_INPUT
    pArrayOut : POITNER TO LREAL;
    nArrayOutSize : ANY;
END_VAR
```

Inputs

Name	Type	Description
pArrayOut	POINTER TO LREAL	Pointer to the output array: Dim: nBufferSize
nArrayOutSize	UDINT	Size of the output array

Return value

Name	Type	Description
GetOutputArray	BOOL	Returns `TRUE` if successful.

5.1.1.1.2 FB_ALY_ContinuousPieceCounter_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ContinuousPieceCounter_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    nNumIntervals: ULINT;
    nCountLastInterval: ULINT;
    nCountCurrentInterval: ULINT;
    nCountMin: ULINT;
    nCountMax: ULINT;
    fbTimeCountMin: FB_ALY_DateTime;
    fbTimeCountMax: FB_ALY_DateTime;
    fbTimeCurrentInterval: FB_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
nNumIntervals	ULINT	Number of elapsed intervals.
nCountLastInterval	ULINT	Number of events in the last interval.
nCountCurrentInterval	ULINT	Number of events in the current interval.
nCountMin	ULINT	Minimum number of events in the current interval.
nCountMax	ULINT	Maximum number of events in the current interval.
fbTimeCountMin	FB_ALY_DateTime	Timestamp of `nCountMin`.
fbTimeCountMax	FB_ALY_DateTime	Timestamp of `nCountMax`.
fbTimeCurrentInterval	FB_ALY_Timespan	Elapsed time in the current interval.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
Pause()	Local	Method to pause the execution including the internal time intervals.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbContinuousPieceCounter : FB_ALY_ContinuousPieceCounter_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_Threshold;
    tInterval : LTIME := LTIME#5M;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
```

```
    stThresholdEdge.fThreshold := 1;

    fbContinuousPieceCounter.ConfigureChannel(stThresholdEdge);
    fbContinuousPieceCounter.Configure(tInterval);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbContinuousPieceCounter.SetChannelValue(nInput);
fbContinuousPieceCounter.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.2.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.2.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    tInterval : LTIME;
END_VAR
```

Inputs

Name	Type	Description
tInterval	LTIME	Interval time in which the events are to be considered

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.2.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.2.4 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.2.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.2.6 Pause

With the Pause method it is possible to pause the execution of the function block. The internally running time interval does not continue after calling the pause function. A single call of the method is sufficient. The interval is not continued until the next call of the Call method.

Syntax

Definition:

```
METHOD Pause : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Pause	BOOL	Returns `TRUE` if successful.

5.1.1.1.3 FB_ALY_Downsampling_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_Downsampling_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fOut: LREAL;
    fbTimeLastSample: FB_ALY_DateTime;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fOut	LREAL	Output signal with the sampling rate lower by the downsampling factor.
fbTimeLastSample	FB_ALY_DateTime	Stores the timestamp of the last data point output to `fOut`.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbDownsampling : FB_ALY_Downsampling_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nDownsamplingFactor : UDINT := 10;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbDownsampling.Configure(nDownsamplingFactor);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbDownsampling.SetChannelValue(nInput);
fbDownsampling.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.3.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.3.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
    nDownsamplingFactor : UDINT;
VAR_INPUT
END_VAR
```

Inputs

Name	Type	Description
nDownsamplingFactor	UDINT	The factor used for downsampling. For example, if the downsampling factor is 100, only every 100th value is saved. The sample time is thus 100 times the original cycle time, which lies between two sampled data points. If the downsampling factor is set to one, all values are buffered.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.3.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.3.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.4 FB_ALY_EdgeCounter_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_EdgeCounter_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bEdge: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bEdge	BOOL	`TRUE` if an edge has been detected.
nCount	ULINT	Counts the number of edges detected.
fbTimeLastEvent	FB_ALY_DateTime	Stores the timestamp of the last detected edge.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInital()	Local	Method for setting initial values for the algorithms, e.g. already expired lifetime.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbEdgeCounter : FB_ALY_EdgeCounter_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_Threshold;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.fThreshold := 1;

    fbEdgeCounter.ConfigureChannel(stThresholdEdge);
    fbEdgeCounter.Configure();
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbEdgeCounter.SetChannelValue(nInput);
fbEdgeCounter.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.4.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.4.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.4.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.4.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.4.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.4.6 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    nCount : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nCount	ULINT	Initializes the number of detected edges.

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.1.5 FB_ALY_EdgeCounterOnOff_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_EdgeCounterOnOff_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bIsOn: BOOL;
    bEdgeOn: BOOL;
    bEdgeOff: BOOL;
    nCountOn: ULINT;
    nCountOff: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bIsOn	BOOL	TRUE within the timespan between the On event and the Off result, otherwise FALSE.
bEdgeOn	BOOL	TRUE if an On event has been detected.
bEdgeOff	BOOL	TRUE if an Off event has been detected.
nCountOn	ULINT	Counts the number of On events that were triggered.
nCountOff	ULINT	Counts the number of Off events that were triggered.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last event.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInital()	Local	Method for setting initial values for the algorithms, e.g. already expired lifetime.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbEdgeCounterOnOff : FB_ALY_EdgeCounterOnOff_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_ThresholdOnOff;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.On.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.On.fThreshold := 1;
    stThresholdEdge.Off.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.Off.fThreshold := 0;

    fbEdgeCounterOnOff.ConfigureChannel(stThresholdEdge);
    fbEdgeCounterOnOff.Configure();
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbEdgeCounterOnOff.SetChannelValue(nInput);
fbEdgeCounterOnOff.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.5.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.5.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.5.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_ThresholdOnOff;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_ThresholdOnOff [▶ 346]	Combination of comparison operator and threshold for on-state and off-state

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.5.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.5.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.5.6 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    nCountOn : ULINT;
    nCountOff : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nCountOn	ULINT	Initializes the number of detected On events.
nCountOff	ULINT	Initializes the number of detected Off events.

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.1.6 FB_ALY_EdgeCounterOnOff_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_EdgeCounterOnOff_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bIsOn: BOOL;
    bEdgeOn: BOOL;
    bEdgeOff: BOOL;
    nCountOn: ULINT;
    nCountOff: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bIsOn	BOOL	TRUE within the timespan between the On event and the Off result, otherwise FALSE.
bEdgeOn	BOOL	TRUE if an On event has been detected.
bEdgeOff	BOOL	TRUE if an Off event has been detected.
nCountOn	ULINT	Counts the number of On events that were triggered.
nCountOff	ULINT	Counts the number of Off events that were triggered.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last event.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInital()	Local	Method for setting initial values for the algorithms, e.g. already expired lifetime.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbEdgeCounterOnOff : FB_ALY_EdgeCounterOnOff_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_ThresholdOnOff;
    bResetOnMultipleOn : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    nInput : INT;
    fInput : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.On.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.On.fThreshold := 1;
    stThresholdEdge.Off.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.Off.fThreshold := 0;

    fbEdgeCounterOnOff.ConfigureChannel(1, stThresholdEdge.On);
    fbEdgeCounterOnOff.ConfigureChannel(2, stThresholdEdge.Off);
    fbEdgeCounterOnOff.Configure(bResetOnMultipleOn);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbEdgeCounterOnOff.SetChannelValue(1, nInput);
fbEdgeCounterOnOff.SetChannelValue(2, fInput);

fbEdgeCounterOnOff.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.6.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.6.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    bResetOnMultipleOn : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bResetOnMultipleOn	BOOL	Reset in case of multiple On event

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.6.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    nChannel : UDINT;
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *On* 2: *Off*
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.6.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.6.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *On* 2: *Off*
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.6.6 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    nCountOn : ULINT;
    nCountOff : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nCountOn	ULINT	Initializes the number of detected On events.
nCountOff	ULINT	Initializes the number of detected Off events.

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.1.7 FB_ALY_EventTimingAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_EventTimingAnalysis_1Ch
VAR_INPUT
    bPersistent: BOOL;
    bInitWithThresholdLevel: BOOL
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bIsOn: BOOL;
    nSwitchedOn: ULINT;
    fbTimeCurrentInterval: FB_ALY_Timespan;
    fbTimeOnMin: FB_ALY_Timespan;
    fbTimeOnMax: FB_ALY_Timespan;
    fbTimeOnAvg: FB_ALY_Timespan;
    fbTimeOnTotal: FB_ALY_Timespan;
    fbTimeOffMin: FB_ALY_Timespan;
    fbTimeOffMax: FB_ALY_Timespan;
    fbTimeOffAvg: FB_ALY_Timespan;
    fbTimeOffTotal: FB_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.
bInitWithThresholdLevel	BOOL	If the value is `TRUE`, the algorithm uses a threshold to initialize the internal state, instead of waiting for an edge.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bIsOn	BOOL	TRUE within the timespan between the On event and the Off result, otherwise FALSE.
nSwitchedOn	ULINT	Counts the number of On events that were triggered.
fbTimeCurrentInterval	FB_ALY_Timespan	Timespan of the current interval.
fbTimeOnMin	FB_ALY_Timespan	Minimum time between On event and Off event.
fbTimeOnMax	FB_ALY_Timespan	Maximum time between On event and Off event.
fbTimeOnAvg	FB_ALY_Timespan	Average time between On event and Off event.
fbTimeOnTotal	FB_ALY_Timespan	Total time between On events and Off events.
fbTimeOffMin	FB_ALY_Timespan	Minimum time between Off event and On event
fbTimeOffMax	FB_ALY_Timespan	Maximum time between Off event and On event
fbTimeOffAvg	FB_ALY_Timespan	Average time between Off event and On event
fbTimeOffTotal	FB_ALY_Timespan	Total time between Off events and On events.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbEventTimingAnalysis : FB_ALY_EventTimingAnalysis_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_ThresholdOnOff;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
```

```
    stThresholdEdge.On.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.On.fThreshold := 1;
    stThresholdEdge.Off.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.Off.fThreshold := 0;

    fbEventTimingAnalysis.ConfigureChannel(stThresholdEdge);
    fbEventTimingAnalysis.Configure();
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbEventTimingAnalysis.SetChannelValue(nInput);
fbEventTimingAnalysis.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.7.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.7.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.7.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_ThresholdOnOff;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_ThresholdOnOff [▶ 346]	Combination of comparison operator and threshold for on-state and off-state. It is possible to configure only the condition for the On event. In this case, only this event is taken into account when the algorithm is executed.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.7.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.7.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.8 FB_ALY_EventTimingAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_EventTimingAnalysis_2Ch
VAR_INPUT
    bPersistent: BOOL;
    bInitWithThresholdLevel: BOOL
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bIsOn: BOOL;
    nSwitchedOn: ULINT;
    fbTimeCurrentInterval: FB_ALY_Timespan;
    fbTimeOnMin: FB_ALY_Timespan;
    fbTimeOnMax: FB_ALY_Timespan;
    fbTimeOnAvg: FB_ALY_Timespan;
    fbTimeOnTotal: FB_ALY_Timespan;
    fbTimeOffMin: FB_ALY_Timespan;
    fbTimeOffMax: FB_ALY_Timespan;
    fbTimeOffAvg: FB_ALY_Timespan;
    fbTimeOffTotal: FB_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.
bInitWithThresholdLevel	BOOL	If the value is `TRUE`, the algorithm uses a threshold to initialize the internal state, instead of waiting for an edge.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bIsOn	BOOL	TRUE within the timespan between the On event and the Off result, otherwise FALSE.
nSwitchedOn	ULINT	Counts the number of On events that were triggered.
fbTimeCurrentInterval	FB_ALY_Timespan	Timespan of the current interval.
fbTimeOnMin	FB_ALY_Timespan	Minimum time between On event and Off event.
fbTimeOnMax	FB_ALY_Timespan	Maximum time between On event and Off event.
fbTimeOnAvg	FB_ALY_Timespan	Average time between On event and Off event.
fbTimeOnTotal	FB_ALY_Timespan	Total time between On events and Off events.
fbTimeOffMin	FB_ALY_Timespan	Minimum time between Off event and On event.
fbTimeOffMax	FB_ALY_Timespan	Maximum time between Off event and On event.
fbTimeOffAvg	FB_ALY_Timespan	Average time between Off event and On event.
fbTimeOffTotal	FB_ALY_Timespan	Total time between Off events and On events.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbEventTimingAnalysis : FB_ALY_EventTimingAnalysis_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_ThresholdOnOff;
    bResetOnMultipleOn : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    nInput : INT;
    fInput : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.On.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.On.fThreshold := 1;
    stThresholdEdge.Off.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.Off.fThreshold := 0;

    fbEventTimingAnalysis.ConfigureChannel(1, stThresholdEdge.On);
    fbEventTimingAnalysis.ConfigureChannel(2, stThresholdEdge.Off);
    fbEventTimingAnalysis.Configure(bResetOnMultipleOn);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbEventTimingAnalysis.SetChannelValue(1, nInput);
fbEventTimingAnalysis.SetChannelValue(2, fInput);
fbEventTimingAnalysis.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.8.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.8.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
    bResetOnMultipleOn : BOOL;
VAR_INPUT
END_VAR
```

Inputs

Name	Type	Description
bResetOnMultipleOn	BOOL	Reset in case of multiple On event

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.8.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    nChannel : UDINT;
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *On* 2: *Off*
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.8.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.8.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *On* 2: *Off*
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.9 FB_ALY_FlipFlop_2Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_FlipFlop_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bOut: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bOut	BOOL	Result of the bistable flip-flop.
nCount	ULINT	Incremented when the output value is TRUE. The behavior depends on the configuration parameter eCountMode.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last event.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInital()	Local	Method for setting initial values for the algorithms, e.g. already expired lifetime.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbFlipFlop : FB_ALY_FlipFlop_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;

    stConfigS : ST_ALY_Threshold :=(eComparisonOperator:=E_ALY_ComparisonOperator.GreaterThan, fThre
shold:= 0.5);
    stConfigR : ST_ALY_Threshold :=(eComparisonOperator:=E_ALY_ComparisonOperator.GreaterThan, fThre
shold:= 0.5);
    bSetIsDominant : BOOL := TRUE;
    eCountMode : E_ALY_CountMode := E_ALY_CountMode.Cyclic;
    bConfigure : BOOL := TRUE;

    nInS : INT;
    bInR : BOOL;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
```

```
    fbFlipFlop.ConfigureChannel(1, stConfigS);
    fbFlipFlop.ConfigureChannel(2, stConfigR);

    fbFlipFlop.Configure(bSetIsDominant, eCountMode);
END_IF

// Call algorithm
fbFlipFlop.SetChannelValue(1, nInS);
fbFlipFlop.SetChannelValue(2, bInR);
fbFlipFlop.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.9.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.9.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    bSetIsDominant : BOOL;
    eCountMode : E_ALY_CountMode;
END_VAR
```

Inputs

Name	Type	Description
bSetIsDominant	BOOL	The dominance of setting (RS) or resetting (SR) can be configured.
eCountMode	E_ALY_CountMode [▶ 348]	Mode of the result counter. *OnChange*: The counter counts every time the result changes to TRUE. *Cyclic*: The counter increments every cycle when the condition is TRUE.

Return value

Name	Type	Description
Configure	BOOL	Returns TRUE if successful.

5.1.1.1.9.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    nChannel : UDINT;
    stThresholdLevel : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Set (S) 2: Reset (R)
stThresholdLevel	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for threshold value consideration.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns TRUE if successful.

5.1.1.1.9.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns TRUE if the reset was successful.

5.1.1.1.9.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Set (S) 2: Reset (R)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.9.6 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    nCount : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nCount	ULINT	Initializes the counter value.

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.1.10 FB_ALY_IntervalPieceCounter_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IntervalPieceCounter_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bExecutingInterval: BOOL;
    nNumIntervals: ULINT;
    nCountLastInterval: ULINT;
    nCountCurrentInterval: ULINT;
    nCountMin: ULINT;
    nCountMax: ULINT;
    fbTimeCountMin: FB_ALY_DateTime;
    fbTimeCountMax: FB_ALY_DateTime;
    fbTimeCurrentInterval: FB_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bExecutingInterval	BOOL	`TRUE` when the interval is executed.
nNumIntervals	ULINT	Number of elapsed intervals.
nCountLastInterval	ULINT	Number of events in the last interval.
nCountCurrentInterval	ULINT	Number of events in the current interval.
nCountMin	ULINT	Minimum number of events in the current interval.
nCountMax	ULINT	Maximum number of events in the current interval.
fbTimeCountMin	FB_ALY_DateTime	Timestamp of `nCountMin`.
fbTimeCountMax	FB_ALY_DateTime	Timestamp of `nCountMax`.
fbTimeCurrentInterval	FB_ALY_Timespan	Elapsed time in the current interval.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
Pause()	Local	Method to pause the execution including the internal time intervals.

Sample

```
VAR
    fbIntervalPieceCounter : FB_ALY_IntervalPieceCounter_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_Threshold;
    bResetOnMultipleStart : BOOL := FALSE;
    tInterval : LTIME := LTIME#5M;
    bConfigure : BOOL := TRUE;
    bStartInterval : BOOL := FALSE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.fThreshold := 1;

    fbIntervalPieceCounter.ConfigureChannel(stThresholdEdge);
    fbIntervalPieceCounter.Configure(bResetOnMultipleStart, tInterval);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbIntervalPieceCounter.SetChannelValue(nInput);
fbIntervalPieceCounter.Call(fbSystemTime.tSystemTime, bStartInterval)
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.10.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
    bStartInterval : BOOL;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp.
bStartInterval	BOOL	Start processing the configured interval.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.10.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    bResetOnMultipleStart : BOOL;
    tInterval : LTIME
END_VAR
```

Inputs

Name	Type	Description
bResetOnMultipleStart	BOOL	Reset in case of multiple "Start" event
tInterval	LTIME	Interval time in which the events are to be considered

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.10.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.10.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.10.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.10.6 Pause

With the Pause method it is possible to pause the execution of the function block. The internally running time interval does not continue after calling the pause function. A single call of the method is sufficient. The interval is not continued until the next call of the Call method.

Syntax

Definition:

```
METHOD Pause : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Pause	BOOL	Returns `TRUE` if successful.

5.1.1.1.11 FB_ALY_LatchingSwitch_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_FlipFlop_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bOut: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bOut	BOOL	Result of the virtual impulse switch.
nCount	ULINT	Incremented when the output value changes.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last event.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInital()	Local	Method for setting initial values for the algorithms, e.g. already expired lifetime.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbLatchingSwitch : FB_ALY_LatchingSwitch_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;

    stConfig: ST_ALY_Threshold :=(eComparisonOperator:=E_ALY_ComparisonOperator.GreaterThan, fThresh
old:= 0.5);
    bConfigure : BOOL := TRUE;

    nInput : INT;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbLatchingSwitch.ConfigureChannel(stConfig);
    fbLatchingSwitch.Configure();
END_IF

// Call algorithm
fbLatchingSwitch.SetChannelValue(nInput);
fbLatchingSwitch.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.11.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.11.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.11.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.11.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.11.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.11.6 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    nCount : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nCount	ULINT	Initializes the counter value.

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.1.12 FB_ALY_LifecycleAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_LifecycleAnalysis_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    nCyclesElapsed: ULINT;
    nCyclesRemaining: LINT;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
nCyclesElapsed	ULINT	Number of cycles elapsed.
nCyclesRemaining	LINT	Remaining cycles.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInital()	Local	Method for setting initial values for the algorithms, e.g. already expired lifetime.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbLifecycleAnalysis : FB_ALY_LifecycleAnalysis_1Ch;
    stThresholdEdge : ST_ALY_Threshold;
    nCyclesEstimated : ULINT := 1_000_000;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.fThreshold := 1;

    fbLifecycleAnalysis.ConfigureChannel(stThresholdEdge);
    fbLifecycleAnalysis.Configure(nCyclesEstimated);
END_IF

// Call algorithm
fbLifecycleAnalysis.SetChannelValue(nInput);
fbLifecycleAnalysis.Call();
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.12.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.12.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nCyclesEstimated : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nCyclesEstimated	ULINT	Estimated component cycles

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.12.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.12.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.12.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.12.6 SetInitial

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    nCyclesElapsed : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nCyclesElapsed	ULINT	Initializes the number of elapsed cycles.

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.1.13 FB_ALY_LifetimeAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_LifetimeAnalysis_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fbLifetimeElapsed: FB_ALY_Timespan;
    fbLifetimeRemaining: FB_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fbLifetimeElapsed	FB_ALY_Timespan	Time elapsed from the lifetime.
fbLifetimeRemaining	FB_ALY_Timespan	Expected remaining lifetime.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInital()	Local	Method for setting initial values for the algorithms, e.g. already expired lifetime.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbLifetimeAnalysis : FB_ALY_LifetimeAnalysis_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdLevel : ST_ALY_Threshold;
    tLifetimeEstimated : LTIME := LTIME#10000H;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdLevel.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdLevel.fThreshold := 1;

    fbLifetimeAnalysis.ConfigureChannel(stThresholdLevel);
    fbLifetimeAnalysis.Configure(tLifetimeEstimated);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbLifetimeAnalysis.SetChannelValue(nInput);
fbLifetimeAnalysis.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.13.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.13.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    tLifetimeEstimated : LTIME;
END_VAR
```

Inputs

Name	Type	Description
tLifetimeEstimated	LTIME	Estimated component lifetime

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.13.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdLevel : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdLevel	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and used threshold for on-state.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.13.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.13.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.13.6 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    tLifetimeElapsed : LTIME;
END_VAR
```

Inputs

Name	Type	Description
nCount	ULINT	Initializes the lifetime that has already expired.

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.1.14 FB_ALY_MinMaxAvg_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_MinMaxAvg_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fMin: LREAL;
    fMax: LREAL;
    fAvg: LREAL;
    fbTimeMin: FB_ALY_DateTime;
    fbTimeMax: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fMin	LREAL	Minimum of the input values.
fMax	LREAL	Maximum of the input values.
fAvg	LREAL	Average of the input values.
fbTimeMin	FB_ALY_DateTime	Timestamp of `fMin`.
fbTimeMax	FB_ALY_DateTime	Timestamp of `fMax`.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbMinMaxAvg : FB_ALY_MinMaxAvg_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nInput : INT;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Call algorithm
fbMinMaxAvg.SetChannelValue(nInput);
fbMinMaxAvg.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.14.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.14.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.14.3 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.15 FB_ALY_MinMaxAvgInterval_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_MinMaxAvgInterval_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
```

```
    fMin: LREAL;
    fMax: LREAL;
    fAvg: LREAL;
    fbTimeMin: FB_ALY_DateTime;
    fbTimeMax: FB_ALY_DateTime;
    fbTimeCurrentInterval: FB_ALY_Timespan;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fMin	LREAL	Minimum of the input values in the current time interval.
fMax	LREAL	Maximum of the input values in the current time interval.
fAvg	LREAL	Average of the input values in the current time interval.
fbTimeMin	FB_ALY_DateTime	Timestamp of `fMin`.
fbTimeMax	FB_ALY_DateTime	Timestamp of `fMax`.
fbTimeCurrentInterval	FB_ALY_Timespan	Elapsed time of the current interval.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
Pause()	Local	Method to pause the execution including the internal time intervals.

Sample

```
VAR
    fbMinMaxAvgInterval : FB_ALY_MinMaxAvgInterval_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    tInterval : LTIME := LTIME#20S;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbMinMaxAvgInterval.Configure(tInterval);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbMinMaxAvgInterval.SetChannelValue(nInput);
fbMinMaxAvgInterval.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.15.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.15.2 Configure

Configure the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    tInterval : LTIME;
END_VAR
```

Inputs

Name	Type	Description
tInterval	LTIME	Interval time

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.15.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.15.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.15.5 Pause

With the Pause method it is possible to pause the execution of the function block. The internally running time interval does not continue after calling the pause function. A single call of the method is sufficient. The interval is not continued until the next call of the Call method.

Syntax

Definition:

```
METHOD Pause : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Pause	BOOL	Returns `TRUE` if successful.

5.1.1.1.16 FB_ALY_MovingAvg_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*.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_MovingAvg_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fMovingAvg: LREAL;
    fMovingMin: LREAL;
    fMovingMax: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fMovingAvg	LREAL	Moving average value of the input value.
fMovingMin	LREAL	Moving minimum of the input value.
fMovingMax	LREAL	Moving maximum of the input value.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbMovingAvg : FB_ALY_MovingAvg_1Ch;
    nNumValues : UDINT := 100;
    eStartupBehaviour : E_ALY_MovingAvgStartupBehaviour := E_ALY_MovingAvgStartupBehaviour.AvgOverEx
isting;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbMovingAvg.Configure(nNumValues, eStartupBehaviour);
END_IF
```

```
// Call algorithm
fbMovingAvg.SetChannelValue(nInput);
fbMovingAvg.Call();
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.16.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.16.2 Configure

Configure the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nNumValues : UDINT;
    eStartupBehaviour : E_ALY_MovingAvgStartupBehaviour;
END_VAR
```

Inputs

Name	Type	Description
nNumValues	ULINT	Number of values included in the calculation of the moving average.
eStartupBehaviour	E_ALY_MovingAvgStartupBehaviour [▶ 349]	Calculation behavior at the beginning of the analysis with less values than configured in Num Values. *ZeroPadding*: The missing values are filled with zeros. *UseFirstValue:* The first value is used until the number of values equals Num Values. *WaitUntilFilled:* The first result is calculated when the number of values equals Num Values. *AvgOverExisting:* The average is calculated with the already existing values until the number of values equals Num Values.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.16.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.16.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.17 FB_ALY_MovingIntervalCounter_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_MovingIntervalCounter_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bEdge: BOOL;
    bLimited: BOOL;
    nCountsInInterval: ULINT;
    fbTimeFirstCount: FB_ALY_DateTime;
    fbTimeLastCount: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bEdge	BOOL	TRUE at the time the event is triggered, otherwise FALSE.
bLimited	BOOL	TRUE if the number of edges in the current interval exceeds the set Count Limit.
nCountsInInterval	ULINT	Number of triggered events in the current interval.
fbTimeFirstCount	FB_ALY_DateTime	Timestamp of the first event in the current interval.
fbTimeLastCount	FB_ALY_DateTime	Timestamp of the last event in the current interval.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbMovingIntervalCounter : FB_ALY_MovingIntervalCounter_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_Threshold;
    tInterval : LTIME := LTIME#20S;
    nCountLimit : UDINT := 20;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
```

```
    stThresholdEdge.fThreshold := 1;

    fbMovingIntervalCounter.ConfigureChannel(stThresholdEdge);
    fbMovingIntervalCounter.Configure(tInterval, nCountLimit);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbMovingIntervalCounter.SetChannelValue(nInput);
fbMovingIntervalCounter.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.17.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.17.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    tInterval : LTIME;
    nCountLimit : UDINT;
END_VAR
```

Inputs

Name	Type	Description
tInterval	LTIME	Time interval in which the values are to be calculated.
nCountLimit	UDINT	Limit of counts in the configured interval

Return value

Name	Type	Description
Configure	BOOL	Returns TRUE if successful.

5.1.1.1.17.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns TRUE if successful.

5.1.1.1.17.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns TRUE if the reset was successful.

5.1.1.1.17.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.18 FB_ALY_OverallEquipmentEffectiveness

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_OverallEquipmentEffectiveness
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fOEE: LREAL;
    eOeeClass: E_ALY_Classification_3Cls;
    fbTimeOeeEventWarning: FB_ALY_DateTime;
    fbTimeOeeEventAlarm: FB_ALY_DateTime;
    fAvailability: LREAL;
    nPerformance: LREAL;
    fQuality: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fOEE	LREAL	Overall equipment effectiveness in percent. It is calculated by multiplying the availability factor, the performance factor and the quality factor.
eOeeClass	E_ALY_Classification_3Cls [▶ 346]	Result of the classification of the OEE.
fbTimeOeeEventWarning	FB_ALY_DateTime	Timestamp of the last classification of the OEE as a warning.
fbTimeOeeEventAlarm	FB_ALY_DateTime	Timestamp of the last classification of the OEE as an alarm.
fAvailability	LREAL	Availability factor in percent. It is calculated from the ratio between the runtime and the operating time.
fPerformance	LREAL	Performance factor in percent. It is calculated from the ratio of units actually produced and the number of units produced in the ideal case.
fQuality	LREAL	Quality factor in percent. It is calculated as the ratio of intact produced units to produced units.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbOEE : FB_ALY_OverallEquipmentEffectiveness;
    fbSystemTime : FB_ALY_GetSystemTime;

    tIdealCycleTime : LTIME := LTIME#1M30S;
    fThresholdLevelOkWarning : LREAL := 90.0;
    fThresholdLevelWarningAlarm : LREAL := 75.0;
    bConfigure : BOOL := TRUE;

    tScheduledTime : LTIME;
    tOperatingTime : LTIME;
    nUnitsProduced : ULINT;
    nDefectiveUnits : ULINT;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbOEE.Configure(tIdealCycleTime, fThresholdLevelOkWarning, fThresholdLevelWarningAlarm);
END_IF

// Call algorithm
fbOEE.SetChannelValue(1, tScheduledTime);
fbOEE.SetChannelValue(2, tOperatingTime);
fbOEE.SetChannelValue(3, nUnitsProduced);
fbOEE.SetChannelValue(4, nDefectiveUnits);
fbOEE.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.18.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.18.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    tIdealCycleTime : LTIME;
    fThresholdLevelOkWarning : LREAL;
    fThresholdLevelWarningAlarm : LREAL;
END_VAR
```

Inputs

Name	Type	Description
tIdealCycleTime	BOOL	Ideal cycle time for the production of one unit.
fThresholdLevelOkWarning	LREAL	The overall equipment effectiveness greater than the configured threshold is classified as *OK*. If the overall equipment effectiveness is less than or equal to the configured threshold, it is classified as *Warning*.
fThresholdLevelWarningAlarm	LREAL	If the overall equipment effectiveness is greater than the configured threshold, it is classified as *Warning*. If the overall equipment effectiveness is less than or equal to the configured threshold, it is classified as *Alarm* .

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.18.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.18.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

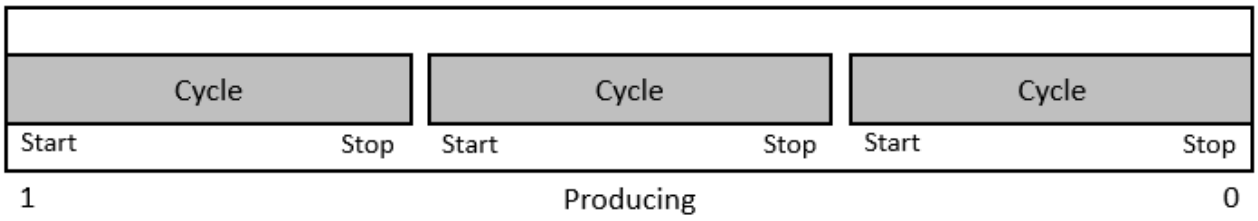
Name	Type	Description
nChannel	UDINT	Channel index 1: *Scheduled Time:* The operating time is calculated from the calendar time minus the scheduled non-production. 2: *Operating Time*: The running time is calculated from the operating time minus the downtimes. 3: *Units Produced*: Corresponds to the number of units produced including defective units. 4: *Defective Units*: Corresponds to the number of defective units.
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.19 FB_ALY_ProductivityDiagnosis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ProductivityDiagnosis_3Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
```

```
    bNewResult: BOOL;
    bConfigured: BOOL;
    bProducing: BOOL;
    bCycleFinished: BOOL;
    fProductivity: LREAL;
    fProductivityLastCycle: LREAL;
    fExpectedProductivity: LREAL;
    fbElapsedTime: FB_ALY_Timespan;
    nProductionCycles: ULINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bProducing	BOOL	Indicates that the production interval is active.
bCycleFinished	BOOL	TRUE if a production cycle has been completed, otherwise FALSE.
fProductivity	LREAL	Productivity of the entire production interval in percent.
fProductivityLastCycle	LREAL	Productivity of the last production cycle in percent.
fExpectedProductivity	LREAL	Estimates the productivity of the production interval. Specified in percent.
fbElapsedTime	FB_ALY_Timespan	Timespan since the start of the production interval.
nProductionCycles	ULINT	Number of complete production cycles in the current production interval.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbProductivityDiagnosis : FB_ALY_ProductivityDiagnosis_3Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdIsProducing : ST_ALY_Threshold;
```

```
    stThresholdStartCycle : ST_ALY_Threshold;
    stThresholdStopCycle : ST_ALY_Threshold;
    nNumProducedPieces : ULINT := 40;
    tProductionTime : LTIME := LTIME#1H;
    bConfigure : BOOL := TRUE;
    bIsProducing : BOOL;
    bStartProductionCycle : BOOL;
    bStopProductionCycle : BOOL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdIsProducing.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdIsProducing.fThreshold := 1;
    stThresholdStartCycle.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdStartCycle.fThreshold := 1;
    stThresholdStopCycle.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdStopCycle.fThreshold := 1;

    fbProductivityDiagnosis.ConfigureChannel(1, stThresholdIsProducing);
    fbProductivityDiagnosis.ConfigureChannel(2, stThresholdStartCycle);
    fbProductivityDiagnosis.ConfigureChannel(3, stThresholdStopCycle);
    fbProductivityDiagnosis.Configure(nNumProducedPieces, tProductionTime);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbProductivityDiagnosis.SetChannelValue(1, bIsProducing);
fbProductivityDiagnosis.SetChannelValue(2, bStartProductionCycle);
fbProductivityDiagnosis.SetChannelValue(3, bStopProductionCycle);
fbProductivityDiagnosis.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.19.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.19.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nNumProducedPieces : ULINT;
    tProductionTime : LTIME;
END_VAR
```

Inputs

Name	Type	Description
nNumProducedPieces	ULINT	Setpoint for pieces produced during the configured time interval (*Production Time*).
tProductionTime	LTIME	Time interval in which nNumProducedPieces are produced.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.19.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    nChannel : UDINT;
    stThreshold : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *Is producing* (level) 2: *Start production cycle* (edge) 2: Stop production cycle (level)
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold. Depending on nChannel.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.19.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.19.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Is producing (level) 2: Start production cycle (edge) 2: Stop production cycle (level)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.20 FB_ALY_ProductivityInterval_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ProductivityInterval_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bWithinInterval: BOOL;
    fbCurrentTimestamp: FB_ALY_DateTime;
    fbIntervalLength: FB_ALY_Timespan;
    fbElapsedTime: FB_ALY_Timespan;
    fbRemainingTime: FB_ALY_Timespan;
    nProducedInInterval: ULINT;
    nRemainingInInterval: ULINT;
    fCurrentProductivity: LREAL;
    fExpectedProductivity: LREAL;
    fLastFullPeriodProductivity: LREAL;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bWithinInterval	BOOL	Indicates whether the current time is within the interval.
fbCurrentTimestamp	FB_ALY_DateTime	Current timestamp.
fbIntervalLength	FB_ALY_Timespan	Length of the interval.
fbElapsedTime	FB_ALY_Timespan	Elapsed time within the interval.
fbRemainingTime	FB_ALY_Timespan	Time remaining within the interval.
nProducedInInterval	ULINT	Produced parts within the interval.
nRemainingInInterval	ULINT	Remaining pieces within the interval.
fCurrentProductivity	LREAL	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.
fExpectedProductivity	LREAL	Expected productivity of the interval in percent. To determine the possible pieces that can be produced within the remaining time, the production time of the last part is used.
fLastFullPeriodProductivity	LREAL	Productivity of the last complete interval in percent. This is only calculated if the interval was fully processed.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbProductivityInterval : FB_ALY_ProductivityInterval_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThreshold : ST_ALY_Threshold;
    nExpectedPieces : ULINT := 40;
    bConfigure : BOOL := TRUE;
    bPieceProduced : BOOL;
    tTimeStart : LTIME := LTIME#8H;
    tTimeStop : LTIME := LTIME#16H;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThreshold.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThreshold.fThreshold := 1;

    fbProductivityInterval.ConfigureChannel(stThreshold);
    fbProductivityInterval.Configure(nExpectedPieces);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbProductivityInterval.SetChannelValue(bPieceProduced);
fbProductivityInterval.Call(tTimeStart, tTimeStop, fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.20.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimeStart : LTIME;
    tTimeStop : LTIME;
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimeStart	LTIME	Start time of the specified interval. Allows the interval to be specified from process data.
tTimeStop	LTIME	End time of the specified interval. Allows the interval to be specified from process data.
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.20.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nExpectedPieces: ULINT;
END_VAR
```

Inputs

Name	Type	Description
nExpectedPieces	ULINT	Specification of the pieces to be produced within the defined timespan.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.20.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThreshold : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold. This setting determines when a produced piece is counted.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.20.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.20.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.21 FB_ALY_SignalGenerator_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_SignalGenerator_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fSignal: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fSignal	LREAL	Output of the configured signal

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.

Sample

```
VAR
    fbSignalGenerator : FB_ALY_SignalGenerator_1Ch;
    eFunctionType : E_ALY_FunctionType := E_ALY_FunctionType.Sine;
    fSampleRate : LREAL := 1000.0;
    fFrequency : LREAL := 50.0;
    fAmplitude : LREAL := 100.0;
    fOffset : LREAL := 0.0;
    bConfigure : BOOL := TRUE;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbSignalGenerator.Configure(eFunctionType, fSampleRate, fFrequency, fAmplitude, fOffset);
END_IF

// Call algorithm
fbSignalGenerator.Call();
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.21.1 Call

Calling the algorithm.

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.21.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eFunctionType : E_ALY_FunctionType;
    fSampleRate : LREAL;
```

```
    fFrequency : LREAL;
    fAmplitude : LREAL;
    fOffset : LREAL;
END_VAR
```

Inputs

Name	Type	Description
eFunctionType	E_ALY_FunctionType [▶ 351]	Function type of the generated signal. *Const:* Constant *Rectangle:* Rectangle function *Sawtooth:* Sawtooth function *Sine:* Sine function *Triangle:* Triangle function
fSampleRate	LREAL	Sample rate of the system to be analyzed.
fFrequency	LREAL	Frequency of the generated signal.
fAmplitude	LREAL	Configuration of the signal amplitude.
fOffset	LREAL	Constant offset of the generated signal.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.21.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.22 FB_ALY_TimeClock_1Ch

Time Clock 1Ch executes a time switch which can be configured with switch-on time, switch-off time and the days of the week on which the time switch should be active. A timestamp is required as a reference value because the algorithm needs a time context in which to operate.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_TimeClock_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bIsOn: BOOL;
    fbTimeUntilNextSwitch: FB_ALY_Timespan
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bIsOn	BOOL	TRUE if the current time is within the configured On time.
fbTimeUntilNextSwitch	FB_ALY_Timespan	Timespan until the next switching.

Methods

Name	Definition Location	Description
Call()	Local	method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.

Sample

```
VAR
    fbTimeClock : FB_ALY_TimeClock_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    tTimeOn : LTIME := LTIME#8H;
    tTimeOff : LTIME := LTIME#16H;
    nDayOfWeekMask : WORD := E_ALY_DayOfWeekMask.MondayToFriday;
    bConfigure : BOOL := TRUE;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbTimeClock.Configure(tTimeOn, tTimeOff, nDayOfWeekMask);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbTimeClock.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.22.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.22.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
    tTimeOn : LTIME;
    tTimeOff : LTIME;
    nDayOfWeekMask : WORD;
VAR_INPUT
END_VAR
```

Inputs

Name	Type	Description
tTimeOn	LTIME	Time at which the timer should switch on
tTimeOff	LTIME	Time at which the timer should switch off
nDayOfWeekMask	WORD	Bit mask for selecting the day of the week on which the timer program is to be executed. 1: *Sunday* 2: *Monday* 4: *Tuesday* 8: *Wednesday* 16: *Thursday* 32: *Friday* 64: *Saturday* The enumeration E_ALY_DayOfWeekMask [▶ 348] can be used. Multiple days can be combined with the operator OR.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.22.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.23 FB_ALY_Timer_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_Timer_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bOut: BOOL;
    fbTimeElapsed: FB_ALY_Timespan;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bOut	BOOL	Output value affected by the configured timer.
fbTimeElapsed	FB_ALY_Timespan	Elapsed time.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
Pause()	Local	Method to pause the execution including the internal time intervals.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbTimer : FB_ALY_Timer_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    eTimerMode : E_ALY_TimerMode := E_ALY_TimerMode.TON;
    tInterval : LTIME := LTIME#20S;
    bConfigure : BOOL := TRUE;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbTimer.Configure(eTimerMode, tInterval);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbTimer.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.23.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.23.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eTimerMOde : E_ALY_TimerMode;
    tInterval : LTIME;
END_VAR
```

Inputs

Name	Type	Description
eTimerMode	E_ALY_TimerMode [▶ 350]	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*.
tInterval	LTIME	Interval time in which the events are to be considered

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.1.23.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdLevel : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdLevel	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for the input condition.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.1.23.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.1.23.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.1.23.6 Pause

With the Pause method it is possible to pause the execution of the function block. The internally running time interval does not continue after calling the pause function. A single call of the method is sufficient. The interval is not continued until the next call of the Call method.

Syntax

Definition:

```
METHOD Pause : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Pause	BOOL	Returns `TRUE` if successful.

5.1.1.1.24 FB_ALY_TimingAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_TimingAnalysis_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bIsOn: BOOL;
    nSwitchedOn: ULINT;
    fbTimeCurrentInterval: FB_ALY_Timespan;
    fbTimeOnTotal: FB_ALY_Timespan;
    fbTimeOffTotal: FB_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bIsOn	BOOL	`TRUE` within the timespan between the On event and the Off result, otherwise `FALSE`.
nSwitchedOn	ULINT	Counts the number of On events that were triggered.
fbTimeCurrentInterval	FB_ALY_Timespan	Timespan of the current interval.
fbTimeOnTotal	FB_ALY_Timespan	Total time, for which `bIsOn=TRUE`.
fbTimeOffTotal	FB_ALY_Timespan	Total time, for which `bIsOn=FALSE`.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbTimingAnalysis : FB_ALY_TimingAnalysis_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stThresholdEdge : ST_ALY_Threshold;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.fThreshold := 1;

    fbTimingAnalysis.ConfigureChannel(stThresholdEdge);
    fbTimingAnalysis.Configure();
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbTimingAnalysis.SetChannelValue(nInput);
fbTimingAnalysis.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.1.24.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.1.24.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Configure	BOOL	Returns TRUE if successful.

5.1.1.1.24.3 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    stThresholdLevel : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
stThresholdLevel	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and used threshold for on-state.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns TRUE if successful.

5.1.1.1.24.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns TRUE if the reset was successful.

5.1.1.1.24.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2 Classification

5.1.1.2.1 FB_ALY_BandwidthClassificator_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_BandwidthClassificator_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    eClass: E_ALY_Classification_Bounds;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
eClass	E_ALY_Classification_Bounds [▶ 347]	Class to which the input values belong (*WithinBounds/ Smaller/Bigger*).
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last change of the classification result.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbBandwidthClassificator : FB_ALY_BandwidthClassificator_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    fLowerBound : LREAL := 10.0;
    fUpperBound : LREAL := 20.0;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbBandwidthClassificator.Configure(fLowerBound, fUpperBound);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbBandwidthClassificator.SetChannelValue(nInput);
fbBandwidthClassificator.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.1.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fLowerBound : LREAL;
    fUpperBound : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fLowerBound	LREAL	Lower limit for comparison.
fUpperBound	LREAL	Upper limit for comparison.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.2.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.2 FB_ALY_BandwidthClassificator_3Ch

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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_BandwidthClassificator_3Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    eClass: E_ALY_Classification_Bounds;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
eClass	E_ALY_Classification_Bounds [▶ 347]	Class to which the input values belong (*WithinBounds/ Smaller/Bigger*).
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last change of the classification result.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbBandwidthClassificator : FB_ALY_BandwidthClassificator_3Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    fInputLowerBound : LREAL;
    fInputUpperBound : LREAL;
    nInput : INT;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Call algorithm
fbBandwidthClassificator.SetChannelValue(1, nInput);
fbBandwidthClassificator.SetChannelValue(2, fInputLowerBound);
fbBandwidthClassificator.SetChannelValue(3, fInputUpperBound);
fbBandwidthClassificator.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.2.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.2.2 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *Data in* 2: Lower bound 3: *Upper bound*
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.3 FB_ALY_CurveSketcher_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_CurveSketcher_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fLastPeak: LREAL;
    fbTimeLastPeak: FB_ALY_DateTime;
    nCountPeaks: ULINT;
    fLastValley: LREAL;
    fbTimeLastValley: FB_ALY_DateTime;
    nCountValleys: ULINT;
    fValueAtMaxDelta: LREAL;
    fMaxDelta: LREAL;
    fbTimeMaxDelta: FB_ALY_DateTime;
    nCountMaxDelta: ULINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fLastPeak	LREAL	Y-value of the last identified peak.
fbTimeLastPeak	FB_ALY_DateTime	Timestamp of `fLastPeak`.
nCountPeaks	ULINT	Total number of peaks counted.
fLastValley	LREAL	Y-value of the last identified valley.
fbTimeLastValley	FB_ALY_DateTime	Timestamp of `fLastValley`.
nCountValleys	ULINT	Number of valleys detected.
fValueAtMaxDelta	LREAL	Input variable guided by the last detected maximum of Delta. The Delta value is the difference between this value and the input value from one cycle before.
fMaxDelta	LREAL	The last detected local maximum of the absolute difference between two successive values in the input current.
fbTimeMaxDelta	FB_ALY_DateTime	Timestamp of `fValueAtMaxDelta`.
nCountMaxDelta	ULINT	Total number of counted local maxima of Delta.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbCurveSketcher : FB_ALY_CurveSketcher_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    fThresholdReversal : LREAL := 0.0;
    bCalcInflection : BOOL := FALSE;
    fThresholdDelta : LREAL := 10.0;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbCurveSketcher.Configure(fThresholdReversal, bCalcInflection, fThresholdDelta);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbCurveSketcher.SetChannelValue(nInput);
fbCurveSketcher.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.3.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.3.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.2.3.3 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

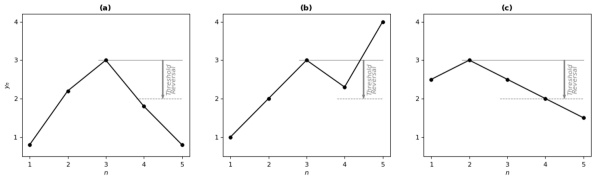
Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.3.4 Configure

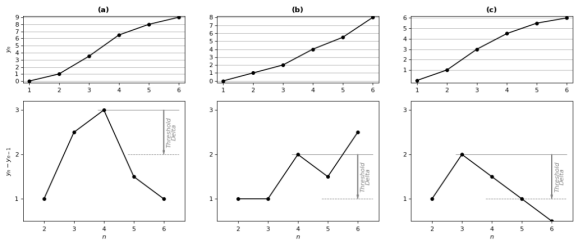
Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Configuration options

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



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



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

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fThresholdReversal : LREAL;
    bCalcInflection : BOOL;
    fThresholdDelta : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fThresholdReversal	LREAL	Threshold for peak analysis
bCalcInflection	BOOL	Max Delta evaluation
fThresholdDelta	LREAL	Threshold for Delta analysis

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.4 FB_ALY_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_Histogram_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
```

```
    bConfigured: BOOL;
    nNumValues: ULINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
nNumValues	ULINT	Number of values included in the histogram.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
GetResults()	Local	Getting the result array without adding new values.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbHistogram : FB_ALY_Histogram_1Ch;
    eHistMode : E_ALY_HistMode := E_ALY_HistMode.Absolute;
    nBins : UDINT := 20;
    fMinBinned : LREAL := 1;
    fMaxBinned : LREAL:= 200;
    bConfigure : BOOL := TRUE;
    nInput : INT;
    aHistArrayOut : ARRAY[0..21] OF ULINT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbHistogram.Configure(eHistMode, nBins, fMinBinned, fMaxBinned);
END_IF

// Call algorithm
fbHistogram.SetChannelValue(nInput);
fbHistogram.Call(ADR(aHistArrayOut), SIZEOF(aHistArrayOut));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.4.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    pHistArrayOut : PVOID;
    nHistArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pHistArrayOut	PVOID	Pointer to histogram array. Dim.: nBins + 2 The data type depends on the configured mode. *Abs*: ULINT *Rel*: LREAL
nHistArrayOutSize	UDINT	Size of the histogram array.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.4.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.2.4.3 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.4.4 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eHistMode : E_ALY_HistMode;
    nBins : UDINT;
    fMinBinned : LREAL;
    fMaxMinned : LREAL;
END_VAR
```

Inputs

Name	Type	Description
eHistMode	E_ALY_HistMode [▶ 351]	Operation mode of the histogram. *Abs*: Absolute values *Rel*: Relative values to show the percentage distribution.
nBins	UDINT	Total number of histogram classes to be calculated.
fMinBinned	LREAL	Minimum value to be analyzed.
fMaxBinned	LREAL	Maximum value to be analyzed.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.4.5 GetResults

Get result array without adding new values.

Syntax

Definition:

```
METHOD GetResults : BOOL
VAR_INPUT
    pHistArrayOut: POINTER TO ULINT;
    nHistArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pHistArrayOut	POINTER TO ULINT	Pointer to histogram array. Dim.: nBins + 2
nHistArrayOutSize	UDINT	Size of the histogram array. Dim.: nBins + 2

Return value

Name	Type	Description
GetResults	BOOL	Returns `TRUE` if successful

5.1.1.2.5 FB_ALY_SectionTimer_1Ch

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 or equal than the maximal border are stored in the last array element.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_SectionTimer_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    nSection: UDINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
nSection	UDINT	Currently classified section of the input.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
GetResults()	Local	Getting the result array without adding new values.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
SetSections()	Local	Setting individual sections.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInitial()	Local	Method for setting start section times.

Sample

```
VAR
    fbSectionTimer : FB_ALY_SectionTimer_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nNumSections : UDINT := 4;
    fFirstLowerBorder : LREAL := 10;
    aUpperBorders : ARRAY[1..4] OF LREAL := [20,30,40,50];
    bConfigure : BOOL := TRUE;
    nInput : INT;
    aTimespansOut : ARRAY [0..5] OF LINT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
```

```
    fbSectionTimer.Configure(nNumSections);
    fbSectionTimer.SetSections(fFirstLowerBorder, ADR(aUpperBorders), SIZEOF(aUpperBorders));
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbSectionTimer.SetChannelValue(nInput);
fbSectionTimer.Call(fbSystemTime.tSystemTime, ADR(aTimespansOut), SIZEOF(aTimespansOut));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.5.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
    pTimespanArrayOut : POINTER TO LINT;
    nTimespanArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp.
pTimespanArrayOut	POINTER TO LINT	Pointer to the timespan array. These values are: Dim.: nNumSections + 2
nTimespanArrayOutSize	UDINT	Size of the timespan array. Dim.: nNumSections + 2

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.5.2 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    pTimespanArrayIn : POINTER TO ULINT;
    nTimespanArrayInSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pTimespanArrayIn	POINTER TO ULINT	Pointer to timespan array to be set as output values: Dim.: nNumSections + 2
nTimespanArrayInSize	UDINT	Size of the timespan array to be set as output values. Dim.: nNumSections + 2

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.2.5.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nNumSections : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nNumSections	UDINT	Configures the number of sections.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.5.4 SetSections

Set sections after the FB has been configured. 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. The first lower border is set separately.

Syntax

Definition:

```
METHOD SetSections : BOOL
VAR_INPUT
    fFirstLowerBorder : LREAL;
    pSectionConfigArray : POINTER TO LREAL;
    nSectionConfigArraySize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
fFirstLowerBorder	LREAL	Configuration of the first lower border.
pSectionConfigArray	POINTER TO LREAL	Pointer to array of upper section limits: Dim: nNumSections
nSectionConfigArraySize	UDINT	Size of the section configuration array. Dim.: nNumSections

Return value

Name	Type	Description
SetSections	BOOL	Returns `TRUE` if successful

5.1.1.2.5.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.5.6 GetResults

Get result array without adding new values.

Syntax

Definition:

```
METHOD GetResults : BOOL
VAR_INPUT
    pTimespanArrayOut : POINTER TO LINT;
    nTimespanArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pTimespanArrayOut	POINTER TO LINT	Pointer to the timespan array. These values are: Dim.: nNumSections + 2
nTimespanArrayOutSize	UDINT	Size of the timespan array. Dim.: nNumSections + 2

Return value

Name	Type	Description
GetResults	BOOL	Returns TRUE if successful

5.1.1.2.5.7 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns TRUE if the reset was successful.

5.1.1.2.6 FB_ALY_StateHistogram_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StateHistogram_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    nNumValues: ULINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
nNumValues	ULINT	Number of values included in the histogram.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
GetResults()	Local	Get result array without adding new values.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.
SetInital()	Local	Method to set initial histogram values.

Sample

```
VAR
    fbStateHistogram : FB_ALY_StateHistogram_1Ch;
    eStateHistMode : E_ALY_StateHistMode := E_ALY_StateHistMode.Absolute;
    nMin : LINT := 1;
    nMax : LINT:= 20;
    bConfigure : BOOL := TRUE;
    nInput : INT;
    aHistArrayOut : ARRAY[0..21] OF ULINT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbStateHistogram.Configure(eStateHistMode, nMin, nMax);
END_IF

// Call algorithm
fbStateHistogram.SetChannelValue(nInput);
fbStateHistogram.Call(ADR(aHistArrayOut), SIZEOF(aHistArrayOut));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.6.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    pHistArrayOut : PVOID;
    nHistArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pHistArrayOut	PVOID	Pointer to histogram array. The data type depends on the configured mode. *Abs*: ULINT *Rel*: LREAL
nHistArrayOutSize	UDINT	Size of the histogram array. Dim.: nMax – nMin + 3

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.6.2 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.6.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eStateHistMode : E_ALY_StateHistMode;
    nMin : LINT;
    nMax : LINT
END_VAR
```

Inputs

Name	Type	Description
eStateHistMode	E_ALY_StateHistMode [▶ 349]	Operation mode of the histogram. *Abs*: Absolute values *Rel*: Relative values to show the percentage distribution.
nMin	LINT	Minimum value to be analyzed.
nMax	LINT	Maximum value to be analyzed.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.6.4 SetInital

Initialize internal values of the algorithm.

Syntax

Definition:

```
METHOD SetInitial : BOOL
VAR_INPUT
    nCountOn : ULINT;
    nCountOff : ULINT;
END_VAR
```

Inputs

Name	Type	Description
eStateHistMode	E_ALY_StateHistMode [▶ 349]	Operation mode of the histogram. Abs: Absolute values Rel: Relative values to show the percentage distribution.
nNumValues	ULINT	Number of values that will be stored in the histogram.
pHistArrayIn	PVOID	Pointer to histogram array to be set as start. The data type depends on the configured mode. Abs: ULINT Rel: LREAL
nHistArrayInSize	UDINT	Size of the histogram array to be set as the start. Dim.: nMax – nMin + 3

Return value

Name	Type	Description
SetInitial	BOOL	Returns `TRUE` if successful.

5.1.1.2.6.5 GetResults

Get result array without adding new values.

Syntax

Definition:

```
METHOD GetResults : BOOL
VAR_INPUT
    pHistArrayOut: PVOID;
    nHistArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pHistArrayOut	PVOID	Pointer to histogram array. The data type depends on the configured mode. Abs: ULINT Rel: LREAL
nHistArrayOutSize	UDINT	Size of the histogram array. Dim.: nMax – nMin + 3

Return value

Name	Type	Description
GetResults	BOOL	Returns TRUE if successful

5.1.1.2.7 FB_ALY_ThresholdClassificator_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ThresholdClassificator_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    eClass: E_ALY_Classification_3Cls;
    fbTimeLastEventWarning: FB_ALY_DateTime;
    fbTimeLastEventAlarm: FB_ALY_DateTime;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
eClass	E_ALY_Classification_3Cls [▶ 346]	Classification result.
fbTimeLastEventWarning	FB_ALY_DateTime	Timestamp of the last classification as a warning.
fbTimeLastEventAlarm	FB_ALY_DateTime	Timestamp of the last classification as an alarm.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbThresholdClassificator : FB_ALY_ThresholdClassificator_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    fThresholdLevelOkWarning : LREAL := 10;
    fThresholdLevelWarningAlarm : LREAL:= 20;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbThresholdClassificator.Configure(fThresholdLevelOkWarning, fThresholdLevelWarningAlarm);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbThresholdClassificator.SetChannelValue(nInput);
fbThresholdClassificator.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.7.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.7.2 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.7.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fThresholdLevelOkWarning : LREAL;
    fThresholdLevelWarningAlarm : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fThresholdLevelOkWarning	LREAL	The input values that are smaller than the configured threshold are classified as *OK*. The input values greater than or equal to the configured threshold are classified as *Warning*.
fThresholdLevelWarningAlarm	LREAL	The input values that are smaller than the configured threshold are classified as *Warning*. The input values greater than or equal to the configured threshold are classified as *Alarm*.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.7.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.2.8 FB_ALY_ThresholdStringClassificator_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*.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ThresholdStringClassificator_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    sResult: STRING(255) := '';
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
sResult	STRING(255)	Classification result
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last classification change.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbThresholdStringClassificator : FB_ALY_ThresholdStringClassificator_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    fThresholdLevel12 : LREAL := 10;
    fThresholdLevel23 : LREAL:= 20;
    sResult1 : STRING := 'This string is set at level 1 ( < 10)';
    sResult2 : STRING := 'This string is set at level 2 ( >= 10)';
    sResult3 : STRING := 'This string is set at level 3 ( >= 20)';
```

```
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbThresholdStringClassificator.Configure(fThresholdLevel12, fThresholdLevel23, sResult1, sResult
2, sResult3);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbThresholdStringClassificator.SetChannelValue(nInput);
fbThresholdStringClassificator.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.8.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.8.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.2.8.3 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.8.4 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fThresholdLevel12 : LREAL;
    fThresholdLevel23 : LREAL;
    sStringLevel1 : STRING(255);
    sStringLevel2 : STRING(255);
    sStringLevel3 : STRING(255);
END_VAR
```

Inputs

Name	Type	Description
fThresholdLevel12	LREAL	The input values that are smaller than the configured threshold are classified in the *first* class. The input values greater than or equal to the configured threshold are classified in the *second* class.
fUpperBound	LREAL	The input values that are smaller than the configured threshold are classified in the *second* class. The input values greater than or equal to the configured threshold are classified in the *third* class.
sStringLevel1	STRING(255)	String to be classified in class 1
sStringLevel2	STRING(255)	String to be classified in class 2
sStringLevel3	STRING(255)	String to be classified in class 3

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.9 FB_ALY_TimeBasedEnvelope_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, for example. The comparison starts when the signal of the Start Period flag is *TRUE*. It is recommended not to use *Time Based Envelope 1Ch* simultaneously with Time Based Teach Path 1Ch due to concurrent file access. Instead, a reference signal should first be taught in with Time Based Teach Path 1Ch and only then should the evaluation be carried out with the aid of the *Time Based Envelope 1Ch*.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_TimeBasedEnvelope_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bBusy: BOOL;
    eState: E_ALY_ReadState;
    bExecutingCompare: BOOL;
    nValueNumber: ULINT;
    fValueRead: LREAL;
    fBandLower: LREAL;
    fBandUpper: LREAL;
    bWithinBand: BOOL;
    eCompareResult: E_ALY_Classification_Bounds;
    nCurrentComparedCycles: ULINT;
    nCountWithinBand: ULINT;
    nCountSmaller: ULINT;
    nCountBigger: ULINT;
    stFileHeader: ST_ALY_FileHeader;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bBusy	BOOL	TRUE if the FB is active due to a file access.
eState	E_ALY_ReadState [▶ 349]	Current state of the FB. Necessary due to asynchronous file access.
bExecutingCompare	BOOL	TRUE, if the algorithm processes the envelope, otherwise FALSE. The envelope process begins when the flag is bStartPeriod=TRUE.
nValueNumber	ULINT	Value number of the data point in the file that is currently being compared.
fValueRead	LREAL	Value of the data point that is read from a file.
fBandLower	LREAL	Calculated lower limit.
bWithinBand	BOOL	Calculated upper limit.
eCompareResult	E_ALY_Classification_Bounds [▶ 347]	Class to which the input values belong (*WithinBounds/ Smaller/Bigger*).
nCurrentComparedCycles	ULINT	Number of cycles compared.
nCountWithinBand	ULINT	Counts how often the values were within the band.
nCountSmaller	ULINT	Counts how often the values were smaller than the band.
nCountBigger	ULINT	Counts how often the values were larger than the band.
stFileHeader	ST_ALY_FileHeader	Header information of the file that was read.

Methods

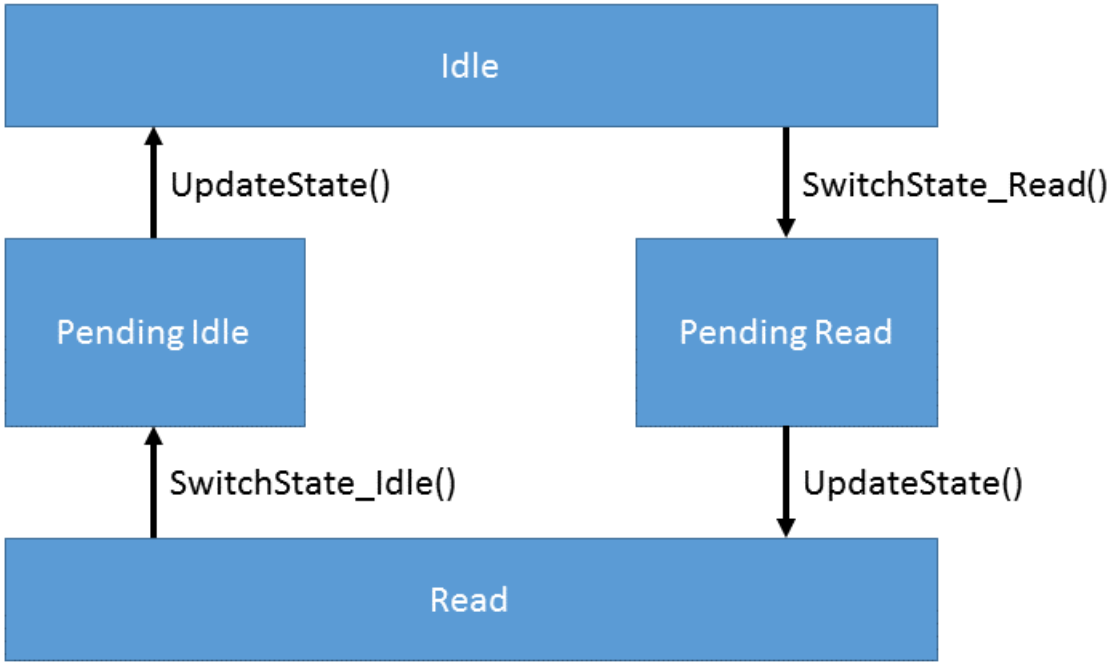
Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
GetBusyState()	Local	This method returns the Busy state of the function block.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SwitchState_Idle()	Local	Initiating the change from state *Read* to state *Idle*. See state diagram.
SwitchState_Read()	Local	Initiating the change from state *Idle* to state *Read*. See state diagram.
UpdateState()	Local	Updating the state after a state change has been initiated and while the target state has not been reached.

State diagram

Due to the asynchronous file access in real time applications this function block needs a state machine to prepare and finish file access.

At startup the function block is in *Idle* state. To compare the incoming data with data from file it has to be switched to state *Read*. Therefore the method *SwitchState_Read()* has to be called once to set the function block to the *PendingRead* state. After that the method *UpdateState()* has to be called until the function block is in state *Read*. In this state one or multiple comparison cycles can be proceeded. If the function block should not compare additional cycles it can be set to *Idle* state again. To initiate the state switch the method *SwitchState_Idle()* has to be called. After that the method *UpdateState()* has to be called until the function block is in state *Idle*.

State diagram for the read procedure of the data:



Sample

```
VAR
    fbTimeBasedEnvelope : FB_ALY_TimeBasedEnvelope_1Ch;
    eBandMode : E_ALY_BandMode := E_ALY_BandMode.Absolute;
    fBand : LREAL := 2.0;
    nSegmentSize : UDINT := 200;
    tTimeout : TIME := T#5S;
    sFilePath : STRING := 'C:\TwinCAT\3.1\Boot\TimeBasedTeach.tas';
    bNegateStartPeriod : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    eState : E_ALY_ReadState := E_ALY_ReadState.Idle;
    bRead : BOOL;
    fInput : LREAL;
    bStartPeriod : BOOL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbTimeBasedEnvelope.Configure(eBandMode, fBand, nSegmentSize, tTimeout, sFilePath, bNegateStartP
eriod);
END_IF

// Call algorithm
eState := fbTimeBasedEnvelope.eState;
CASE eState OF
E_ALY_ReadState.Idle:
    IF bRead THEN
        fbTimeBasedEnvelope.SwitchState_Read();
        fbTimeBasedEnvelope.UpdateState();
    END_IF
E_ALY_ReadState.Read:
    fbTimeBasedEnvelope.SetChannelValue(fInput);
    fbTimeBasedEnvelope.Call(bStartPeriod:=bStartPeriod);
```

```
    IF NOT bRead THEN
        fbTimeBasedEnvelope.SwitchState_Idle();
        fbTimeBasedEnvelope.UpdateState();
    END_IF
E_ALY_ReadState.Pending,
E_ALY_ReadState.PendingIdle,
E_ALY_ReadState.PendingRead:
    fbTimeBasedEnvelope.UpdateState();
    eState := fbTimeBasedEnvelope.eState;
END_CASE
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.9.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    bStartPeriod : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bStartPeriod	BOOL	Rising edge starts reading period. The FB must be in the Read state.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.9.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.2.9.3 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.9.4 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eBandMode : E_ALY_BandMode;
    fBand : LREAL;
    nSegmentSize : UDINT;
    tTimeout : TIME;
    sFilePath : STRING(255);
    bNegateStartPeriod : BOOL;
END_VAR
```

Inputs

Name	Type	Description
eBandMode	E_ALY_BandMode [▶ 347]	Configuring the band mode *Absolute*: Upper bound: read value + fBand Lower bound: read value – fBand *Relative*: Upper bound: read value + (read value * fBand/100) Lower bound: read value - (read value * fBand/100)
fBand	LREAL	Configuring the band depending on the band mode.
nSegmentSize	UDINT	Number of buffered elements for file operation.
tTimeout	TIME	Timeout for asynchronous operations.
sFilePath	STRING(255)	Path to the taught file, e.g. *C:\TwinCAT\3.1\Boot\Teach.tas*
bNegateStartPeriod	BOOL	Negates the input parameter bStartPeriod.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.2.9.5 SwitchState_Idle

Initiate switch from *Read* state to *Idle* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Idle : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Idle	BOOL	Returns TRUE if successful

5.1.1.2.9.6 SwitchState_Read

Initiate switch from *Idle* state to *Read* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Read : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Read	BOOL	Returns TRUE if successful

5.1.1.2.9.7 UpdateState

Update state after state change was initiated and until target state is not reached.

Syntax

Definition:

```
METHOD UpdateState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
UpdateState	BOOL	Returns TRUE if successful

5.1.1.2.9.8 GetBusyState

Returns TRUE if the function block is busy due to an asynchronous file access.

Syntax

Definition:

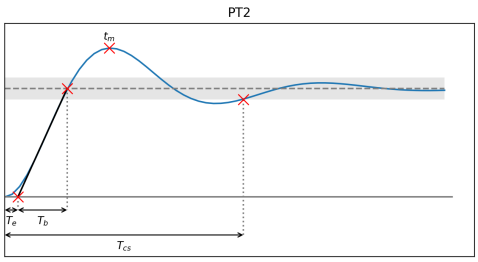
```
METHOD GetBusyState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
GetBusyState	BOOL	Returns TRUE if the FB is active.

5.1.1.2.10 FB_ALY_StepResponse_1Ch

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



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

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StepResponse_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bExecuting: BOOL;
    fbEquivalentDeadTime: FB_ALY_Timespan;
    fbEquivalentTimeConstant: FB_ALY_Timespan;
    fbSettlingTime: FB_ALY_Timespan;
    fbTimeMax: FB_ALY_DateTime;
    fError: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bExecuting	BOOL	TRUE if the parameter identification is active.
fbEquivalentDeadTime	FB_ALY_Timespan	Delay time of the track. Timespan between the start of the step response and the intersection of the inflectional tangents with the start value.
fbEquivalentTimeConstant	FB_ALY_Timespan	Compensation time of the track. Timespan between the intersection point of the inflectional tangents with the start value and the intersection point of the inflectional tangents with the setpoint.
fbSettlingTime	FB_ALY_Timespan	Settling time of the track.
fbTimeMax	FB_ALY_DateTime	Time of the maximum overshoot.
fError	LREAL	Difference between input value and setpoint.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbStepResponse : FB_ALY_StepResponse_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;

    fThresholdReversal : LREAL;
    bUseRelativeTolerance : BOOL;
    fErrorTolerance : LREAL;
    bConfigure : BOOL := TRUE;

    fInput : LREAL;
    fSetpoint : LREAL;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbStepResponse.Configure(fThresholdReversal, bUseRelativeTolerance, fErrorTolerance);
END_IF

// Call algorithm
fbStepResponse.SetChannelValue(1, fInput);
fbStepResponse.SetChannelValue(2, fSetpoint);
fbStepResponse.Call(fbSystemTime.tSystemTime)
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.2.10.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.2.10.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.2.10.3 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Data In 2: Setpoint
Input	ANY	Input value of any data type.

Return value

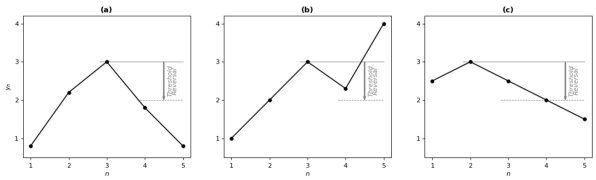
Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.2.10.4 Configure

Configuring the algorithm.

Configuration options

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



Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fThresholdReversal : LREAL;
    bUseRelativeTolerance : BOOL;
    fErrorTolerance : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fThresholdReversal	LREAL	Threshold for peak analysis
bUseRelativeTolerance	BOOL	Boolean flag. If this flag is *True*, the parameter *Error Tolerance* refers to the setpoint at the input as a percentage. Otherwise, an absolute tolerance band is taken into account.
fErrorTolerance	LREAL	Specifies the size of the tolerance band in relation to the parameter *Relative Tolerance*. Since tolerance band is updated when parameter identification is restarted.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.3 Clustering

5.1.1.3.1 FB_ALY_DenStream

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

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

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

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

Parameter setting

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

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

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

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

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

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

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

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

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

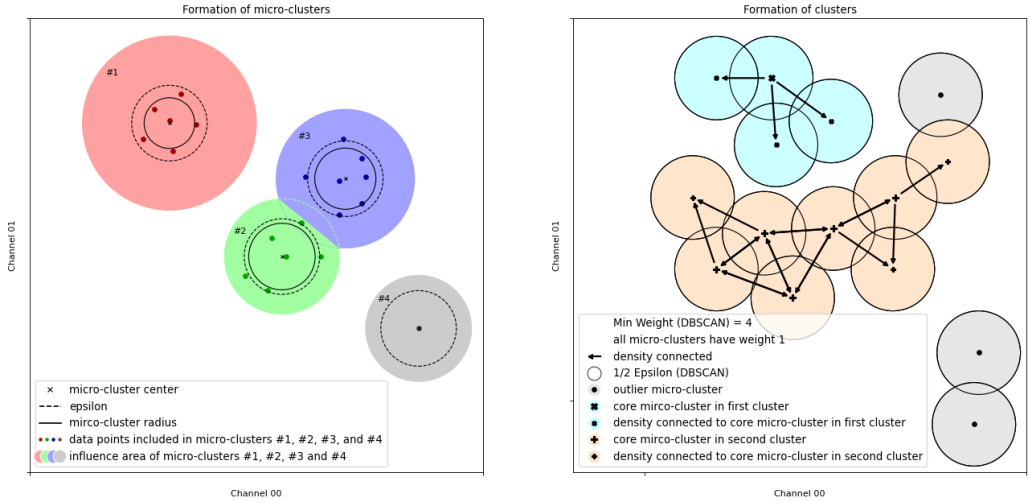
The parameters *Epsilon (DBSCAN)* and *Min Weight (DBSCAN)* affect the second phase. These parameters were adopted from the DBSCAN algorithm [3].

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

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

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

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



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

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

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

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

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

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

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_DenStream
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    nClusterIdx: DINT;
    nNumClusters: DINT;
    fbTimeLastEvent: FB_ALY_DateTime;
    fbTimeLastSwitch: FB_ALY_DateTime;
    bOverflow: BOOL;
    nNumPotMCs: UDINT;
    nNumOutMCs: UDINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
nClusterIdx	DINT	Specifies the cluster index that the DBSCAN algorithm outputs for the data point of the current cycle.
nNumClusters	DINT	Specifies the total number of clusters detected by the DBSCAN algorithm.
fbTimeLastEvent	FB_ALY_DateTime	This is the timestamp of the last cycle with a change of the cluster index
fbTimeLastSwitch	FB_ALY_DateTime	This is the timestamp of the last cycle with an alternation between updating and not updating micro-clusters (either by setting input `bUpdateMicroCluster` to `TRUE` or by internally preventing an overflow of `nPotMCsBufferSize` or `nOutMCBufferSize`)
bOverflow	BOOL	`TRUE` when the micro-cluster update is stopped to prevent `nPotMCsBufferSize` or `nOutMCsBufferSize` from overflowing.
nNumPotMCs	UDINT	Indicates the number of potential micro-clusters currently present.
nNumOutMCs	UDINT	Indicates the number of outlier micro-clusters currently present.

Sample

```
VAR
    fbDenStream : FB_ALY_DenStream(nNumChannels := 2);
    fbSystemTime : FB_ALY_GetSystemTime;

    fEps : LREAL := 0.15;
    fMuBeta : LREAL := 2;
    fLambda : LREAL := 0;
    fEps_DBSCAN : LREAL := 0.8;
    fMinWeight_DBSCAN : LREAL := 2;
    nPotMCsBufferSize : UDINT := 100;
    nOutMCsBufferSize : UDINT := 100;
    bConfigure : BOOL := TRUE;

    nInputCh1 : UDINT;
    fInputCh2 : LREAL;
    bUpdateMCs : BOOL := TRUE;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbDenStream.Configure(
        fEps := fEps,
        fMuBeta := fMuBeta,
        fLambda := fLambda,
        fEpsDBSCAN := fEps_DBSCAN,
        fMinWeightDBSCAN:= fMinWeight_DBSCAN,
        nPotMCsBufferSize := nPotMCsBufferSize,
        nOutMCsBufferSize := nOutMCsBufferSize);
END_IF
```

```
// Get current system time
fbSystemTime.Call();
```

```
// Call algorithm
fbDenStream.SetChannelValue(1, nInputCh1);
fbDenStream.SetChannelValue(2, fInputCh2);
fbDenStream.Call(tTimestamp := fbSystemTime.tSystemTime, bUpdateMCs := bUpdateMCs);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

5.1.1.3.1.1 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.3.1.2 FB_init

Initializes the number of input channels, the number of cluster centers and the size of the aggregation buffer.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
```

```
    nNumClusters : UDInT := 1;
    nAggBufferSize : UDINT := 10;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.
nNumClusters	UDINT	Defines the number of clusters.
nAggBufferSize	UDINT	Specifies the number of cycles after which the cluster centers are updated. The input values over this number of cycles are buffered internally (in the aggregation buffer).

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.3.1.3 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
    bUpdateMCs : BOOL;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.
bUpdateMCs	BOOL	TRUE: The micro-clusters are updated by the incoming data. FALSE: The existing micro-clusters remain unchanged and are only used to determine the cluster index of the incoming data points.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.3.1.4 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fEps : LREAL;
```

```
    fMuBeta : LREAL;
    fLambda : LREAL;
    fEpsDBSCAN : LREAL;
    fMinWeightDBSCAN : LREAL;
    nPotMCsBufferSize : UDINT;
    nOutMCsBufferSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nEps	LREAL	Threshold for the maximum radius of micro-clusters.
fMuBeta	LREAL	Threshold for the weight of a micro-cluster between outlier and potential micro-cluster.
fLambda	LREAL	Specifies the forgetting rate of the algorithm. The weight of each data point decreases by a factor of 2^(-*fLambda*) every second.
fEpsDBSCAN	LREAL	Specifies the epsilon parameter of the DBSCAN algorithm.
fMinWeightDBSCAN	LREAL	Threshold for the sum of weights in the epsilon neighborhood of a micro-cluster for the DBSCAN algorithm.
nPotMCsBufferSize	UDINT	Maximum number of potential micro-clusters. The memory is allocated to *potMCs Buffer Size* micro-clusters.
nOutMCsBufferSize	UDINT	Maximum number of outlier micro-clusters. The memory is allocated to *outMCs Buffer Size* micro-clusters.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.3.1.5 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.3.2 FB_ALY_SequentialKMeans

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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_SequentialKMeans
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    nClusterIdx: DINT;
    fDistance: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
nClusterIdx	DINT	Specifies the cluster index that the DBSCAN algorithm outputs for the data point of the current cycle.
fDistance	LREAL	Specifies the total number of clusters detected by the DBSCAN algorithm.

Sample

```
VAR
    fbSequentialKMeans : FB_ALY_SequentialKMeans(nNumChannels := 2, nNumClusters := 3, nAggBufferSiz
e := 10);

    nMaxIterations : UDINT :=1;
    eInitMode : E_ALY_KMeansInitMode := E_ALY_KMeansInitMode.Values;
    aInitialClusterCenters : ARRAY[1..3] OF ARRAY[1..2] OF LREAL := [[-30, 0], [10, 2], [30, 4]];

    bConfigure : BOOL := TRUE;

    nInputCh1 : UDINT;
    fInputCh2 : LREAL;
    bUpdateClusterCenters : BOOL := TRUE;

    aClusterCenters : ARRAY[1..3] OF ARRAY[1..2] OF LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbSequentialKMeans.Configure(nMaxIterations := nMaxIterations, eInitMode := eInitMode);
    fbSequentialKMeans.SetInitialClusterCenters(ADR(aInitialClusterCenters), SIZEOF(aInitialClusterC
enters));
END_IF

// Call algorithm
fbSequentialKMeans.SetChannelValue(1, nInputCh1);
fbSequentialKMeans.SetChannelValue(2, fInputCh2);
fbSequentialKMeans.Call(bUpdateClusterCenters, ADR(aClusterCenters), SIZEOF(aClusterCenters));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initializes the number of input channels.
GetResults()	Local	Getting the result matrix without adding new values
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetInitialBounds()	Local	Sets the initial boundaries. Depends on the configured initialization mode.
SetInitialClusterCenters()	Local	Sets the initial cluster centers. Depends on the configured initialization mode.

5.1.1.3.2.1 FB_init

Initializes the number of input channels, the number of cluster centers and the size of the aggregation buffer.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
    nNumClusters : UDInT := 1;
    nAggBufferSize : UDINT := 10;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.
nNumClusters	UDINT	Defines the number of clusters.
nAggBufferSize	UDINT	Specifies the number of cycles after which the cluster centers are updated. The input values over this number of cycles are buffered internally (in the aggregation buffer).

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.3.2.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
    bUpdateMCs : BOOL;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.
bUpdateMCs	BOOL	TRUE: The micro-clusters are updated by the incoming data. FALSE: The existing micro-clusters remain unchanged and are only used to determine the cluster index of the incoming data points.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.3.2.3 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nMaxIterations : UDINT;
    eInitMode : E_ALYKMeansInitMode;
END_VAR
```

Inputs

Name	Type	Description
nMaxIterations	UDINT	Specifies how often to iterate over the values in the aggregation buffer. The default is 1.
eInitMode	E_ALY_KMeansInitMode [▶ 351]	Specifies which method is used to set the initial values for the cluster centers: *Random*: The cluster centers are set randomly in the bounds set by Lower Bounds and Upper Bounds. The bounds can be set after configuration with the method SetInitialBounds [▶ 153]. *Euqidistant*: The cluster centers are distributed equidistantly in the range of values defined by the Lower Bounds and Upper Bounds. The bounds can be set after configuration with the method SetInitialBounds [▶ 153]. *Values*: The cluster centers are initialized with the values set by initial cluster centers. The cluster centers can be set after configuration using the SetInitialClusterCenters [▶ 154] method.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.3.2.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.3.2.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.3.2.6 SetInitialBounds

Sets the initial boundaries. Depends on the configured initialization mode.

Syntax

Definition:

```
METHOD SetInitialBounds : BOOL
VAR_INPUT
    pLowerBoundsConfigArray : POINTER TO LREAL;
    nLowerBoundsConfigArraySize : UDINT;
    pUpperBoundsConfigArray : POINTER TO LREAL;
    nUpperBoundsConfigArraySize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pLowerBoundsConfigArray	POINTER TO LREAL	Pointer to an array containing the lower values of the bounds to be initialized. Dim.: nNumChannels
nLowerBoundsConfigArraySize	UDINT	Size of the array with the lower values of the limits to be initialized.
pUpperBoundsConfigArray	POINTER TO LREAL	Pointer to an array containing the upper values of the bounds to be initialized. Dim.: nNumChannels
nUpperBoundsConfigArraySize	UDINT	Size of the array with the upper values of the limits to be initialized.

Return value

Name	Type	Description
SetInitialBounds	BOOL	Returns `TRUE` if successful.

5.1.1.3.2.7 SetInitialClusterCenters

Sets the initial cluster centers. Depends on the configured initialization mode.

Syntax

Definition:

```
METHOD SetInitialClusterCenters : BOOL
VAR_INPUT
    pInitialClusterCenterMatrix : POINTER TO LREAL;
    nInitialClusterCenterMatrixSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pInitialClusterCenterMatrix	POINTER TO LREAL	Pointer to a matrix with initial cluster centers. Dim.: nNumChannels*nNumClusters
nInitialClusterCenterMatrixSize	UDINT	Size of the matrix with initial cluster centers.

Return value

Name	Type	Description
SetInitialClusterCenters	BOOL	Returns `TRUE` if successful.

5.1.1.3.2.8 GetResults

Retrieves the result matrix without adding new values.

Syntax

Definition:

```
METHOD GetResults : BOOL
VAR_INPUT
    pClusterCenterMatrixOut : POINTER TO LREAL;
    nClusterCenterMatrixOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pClusterCenterMatrixOut	POINTER TO LREAL	Pointer to cluster center matrix. Dim.: nNumChannels*nNumClusters
nClusterCenterMatrixOut	UDINT	Size of the cluster center matrix

Return value

Name	Type	Description
GetResults	BOOL	Returns `TRUE` if successful.

5.1.1.4 Compare

5.1.1.4.1 FB_ALY_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_Demultiplexer
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSwitched: BOOL;
    nCurrentChannel: UDINT;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bSwitched	BOOL	`TRUE` if a channel change has taken place.
nCurrentChannel	UDINT	Indicates the number of the selected channel. The value is 0 if the selected channel is outside the configured channels.
nCount	ULINT	Starts with 1 for the channel selected at the start of the analysis and increments each time another channel is selected.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last channel change.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
GetResults()	Local	Getting the result array without adding new values
FB_init()	Local	Initializes the number of output channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing channel-specific values to the algorithm.

Sample

```
VAR
    fbDemultiplexer : FB_ALY_Demultiplexer(nNumChannels := 3);
    fbSystemTime : FB_ALY_GetSystemTime;
    nInput : INT := 1;
    aResults : ARRAY[0..3] OF BOOL;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Call algorithm
fbDemultiplexer.SetChannelValue(nInput);
fbDemultiplexer.Call(fbSystemTime.tSystemTime, ADR(aResults), SIZEOF(aResults));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.1.1 FB_init

Initializes the number of output channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.4.1.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
    pDataOut : POINTER TO BOOL;
    nDataOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.
pDataOut	POINTER TO BOOL	Pointer to the result array of the output channels. Dim.: nNumChannels+1
nDataOutSize	UDINT	Size of the result array.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.1.5 GetResults

Retrieves a result array without adding new values.

Syntax

Definition:

```
METHOD GetResults : BOOL
VAR_INPUT
    pDataOut : POINTER TO BOOL;
    nDataOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pDataOut	POINTER TO BOOL	Pointer to the result array of the output channels. Dim.: nNumChannels+1
nDataOutSize	UDINT	Size of the result array.

Return value

Name	Type	Description
GetResults	BOOL	Returns `TRUE` if successful.

5.1.1.4.2 FB_ALY_DetectStringChange_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_DetectStringChange_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bStringChanged: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bStringChanged	BOOL	`TRUE`, if a change in the input string was detected, otherwise `FALSE`.
nCount	ULINT	Count up each time, if `bStringChanged=TRUE`.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last detected string change.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbDetectStringChange : FB_ALY_DetectStringChange_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    bCaseSensitive : BOOL := TRUE;
    bConfigure : BOOL := TRUE;
    sInput : STRING := 'Modify';
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbDetectStringChange.Configure(bCaseSensitive);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbDetectStringChange.SetChannelValue(sInput);
fbDetectStringChange.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.2.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.2.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.2.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    bCaseSensitive : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bCaseSensitive	BOOL	If TRUE, upper and lower case is taken into account.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.2.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.3 FB_ALY_DynamicTimeWarping

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

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

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_DynamicTimeWarping
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bBusy: BOOL;
    eState: E_ALY_ReadState;
    bExecutingCompare: BOOL;
    nBestMatchIdx: ULINT;
    fValueRead: LREAL;
    stFileHeader: ST_ALY_FileHeader;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bBusy	BOOL	TRUE if the FB is active due to a file access.
eState	E_ALY_ReadState [▶ 349]	Current state of the FB due to asynchronous file accesses.
bExecutingCompare	BOOL	TRUE, if the algorithm processes the envelope, otherwise FALSE. The envelope process begins when the flag is bStartPeriod=TRUE.
nBestMatchIdx	UDINT	Outputs the index of the template with the smallest distance to the input channel.
stFileHeader	ST_ALY_FileHeader	Header information of the last read file.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Configures the file paths of the templates.
FB_init()	Local	Initializes the number of templates.
GetBusyState()	Local	This method returns the Busy state of the function block.
GetChannelOutputVaue()	Local	Method for getting individual output values from the output array
GetChannelOutputArray()	Local	Method for getting the entire output array.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SwitchState_Idle()	Local	Initiating the change from state *Read* to state *Idle*. See state diagram.
SwitchState_Read()	Local	Initiating the change from state *Idle* to state *Read*. See state diagram.
UpdateState()	Local	Updating the state after a state change has been initiated and while the target state has not been reached.

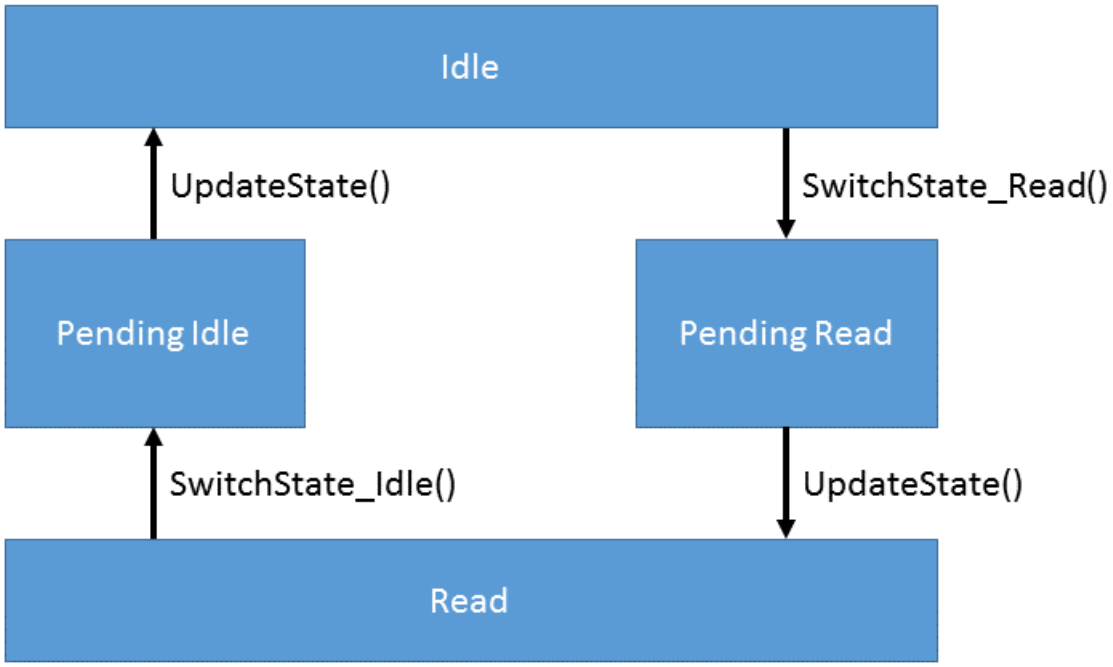
State diagram

Due to the asynchronous file access in real time applications this function block needs a state machine to prepare and finish file access.

At startup the function block is in *Idle* state. To compare the incoming data with data from file it has to be switched to state *Read*. Therefore the method *SwitchState_Read()* has to be called once to set the function block to the *PendingRead* state. After that the method *UpdateState()* has to be called until the function block is in state *Read*. In this state one or multiple comparison cycles can be proceeded. If the function block

should not compare additional cycles it can be set to *Idle* state again. To initiate the state switch the method *SwitchState_Idle()* has to be called. After that the method *UpdateState()* has to be called until the function block is in state *Idle*.

State diagram for the read procedure of the data:



Sample

```
VAR
    fbDynamicTimeWarping : FB_ALY_DynamicTimeWarping(nNumChannels := 3);
    tTimeout : TIME := T#5S;
    sFilePath1 : STRING := 'C:\TwinCAT\3.1\Boot\Template1.tas';
    sFilePath2 : STRING := 'C:\TwinCAT\3.1\Boot\Template2.tas';
    sFilePath3 : STRING := 'C:\TwinCAT\3.1\Boot\Template3.tas';

    bConfigure : BOOL := TRUE;
    eState : E_ALY_ReadState := E_ALY_ReadState.Idle;
    bRead : BOOL;
    fInput : LREAL;
    bStartPeriod : BOOL;
    bStopPeriod : BOOL;

    aDistances : ARRAY[1..3] OF LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
    fbDynamicTimeWarping.ConfigureChannel(1, sFilePath1);
    fbDynamicTimeWarping.ConfigureChannel(2, sFilePath2);
    fbDynamicTimeWarping.ConfigureChannel(3, sFilePath3);
    fbDynamicTimeWarping.Configure(tTimeout);
END_IF

// Call algorithm
eState := fbDynamicTimeWarping.eState;
CASE eState OF
E_ALY_ReadState.Idle:
    IF bRead THEN
        fbDynamicTimeWarping.SwitchState_Read();
        fbDynamicTimeWarping.UpdateState();
    END_IF
E_ALY_ReadState.Read:
    fbDynamicTimeWarping.SetChannelValue(fInput);
    fbDynamicTimeWarping.Call(bStartPeriod:=bStartPeriod, bStopPeriod:=bStopPeriod);

    IF NOT bRead THEN
        fbDynamicTimeWarping.SwitchState_Idle();
```

```
        fbDynamicTimeWarping.UpdateState();
    END_IF
E_ALY_ReadState.Pending,
E_ALY_ReadState.PendingIdle,
E_ALY_ReadState.PendingRead:
    fbDynamicTimeWarping.UpdateState();
    eState := fbDynamicTimeWarping.eState;
END_CASE

// Get results
fbDynamicTimeWarping.GetOutputArray(pArrayOut:=ADR(aDistances), nArrayOutSize:=SIZEOF(aDistances));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.3.1 FB_init

Initializes the number of comparison channels (templates).

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Number of comparison channels (templates)

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.4.3.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    bStartPeriod : BOOL;
    bStopPeriod : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bStartPeriod	BOOL	Rising edge starts the calculation. The FB must be in the Read state.
bStopPeriod	BOOL	Rising edge stops the calculation. The results are output. The FB must be in the Read state.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.3.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    tTimeout : TIME;
END_VAR
```

Inputs

Name	Type	Description
tTimeout	TIME	Timeout for asynchronous operations.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.3.4 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    nChannel : UDINT;
    sFilePath : STRING(255);
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
sFilePath	STRING(255)	Path to the taught file, e.g. *C:\TwinCAT\3.1\Boot\Teach.tas*

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.4.3.5 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns TRUE if the reset was successful.

5.1.1.4.3.6 GetBusyState

Returns TRUE if the function block is busy due to an asynchronous file access.

Syntax

Definition:

```
METHOD GetBusyState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
GetBusyState	BOOL	Returns TRUE if the FB is active.

5.1.1.4.3.7 GetChannelOutputValue

Fetching a channel-specific output value. The output value corresponds to the distance between the input signal and the respective template. The smaller the distance, the more equal the compared signals are. The value is updated only if the Call() method has been called before.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns TRUE if successful.

5.1.1.4.3.8 GetOutputArray

Getting the entire output array. The array elements correspond to the distance between the input signal and the respective template. The smaller the distance, the more equal the compared signals are. The number of elements corresponds to the configured number of comparison channels (templates). The values are updated only if the Call() method has been called before.

Syntax

Definition:

```
METHOD GetOutputArray : BOOL
VAR_INPUT
    pArrayOut : POITNER TO LREAL;
    nArrayOutSize : ANY;
END_VAR
```

Inputs

Name	Type	Description
pArrayOut	POINTER TO LREAL	Pointer to the output array: Dim: nBufferSize
nArrayOutSize	UDINT	Size of the output array

Return value

Name	Type	Description
GetOutputArray	BOOL	Returns `TRUE` if successful.

5.1.1.4.3.9 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.3.10 SwitchState_Idle

Initiate switch from *Read* state to *Idle* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Idle : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Idle	BOOL	Returns `TRUE` if successful

5.1.1.4.3.11 SwitchState_Read

Initiate switch from *Idle* state to *Read* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Read : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwithcState_Read	BOOL	Returns `TRUE` if successful

5.1.1.4.3.12 UpdateState

Update state after state change was initiated and until target state is not reached.

Syntax

Definition:

```
METHOD UpdateState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
UpdateState	BOOL	Returns `TRUE` if successful

5.1.1.4.4 FB_ALY_DynamicTimeWarpingInterval

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_DynamicTimeWarpingInterval
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    nBestMatchIdx: UDINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
nBestMatchIdx	UDINT	Outputs the index of the input channel with the smallest distance to the reference channel.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	Configuration of the algorithm.
FB_init()	Local	Initializes the number of input channels.
GetChannelOutputVaue()	Local	Method for getting individual output values from the output array
GetChannelOutputArray()	Local	Method for getting the entire output array.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbDynamicTimeWarpingInterval : FB_ALY_DynamicTimeWarpingInterval(nNumChannels := 3);

    nWindowSize : UDINT := 100;
    bConfigure : BOOL := TRUE;

    fReference : LREAL;
    fInput1 : LREAL;
    fInput2 : LREAL;
    fInput3 : LREAL;

    aDistances : ARRAY[1..3] OF LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
    fbDynamicTimeWarpingInterval.Configure(nWindowSize);
END_IF

// Call algorithm
fbDynamicTimeWarpingInterval.SetChannelValue(0, fReference);
fbDynamicTimeWarpingInterval.SetChannelValue(1, fInput1);
fbDynamicTimeWarpingInterval.SetChannelValue(2, fInput2);
fbDynamicTimeWarpingInterval.SetChannelValue(3, fInput3);
fbDynamicTimeWarpingInterval.Call();

// Get results
fbDynamicTimeWarpingInterval.GetOutputArray(pArrayOut:=ADR(aDistances), nArrayOutSize:=SIZEOF(aDista
nces));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.4.1 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.4.4.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.4.3 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nWindowSize : UDINT
END_VAR
```

Inputs

Name	Type	Description
nWindowSize	UDINT	Specifies the number of cycles over which a calculation is made. The memory requirement of the algorithm is proportional to this parameter.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.4.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.4.5 GetChannelOutputValue

Fetching a channel-specific output value. The output value corresponds to the distance between the reference channel and the input channel. The smaller the distance, the more equal the compared signals are. The value is updated only if the Call() method has been called before.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.4.6 GetOutputArray

Getting the entire output array. The array elements correspond to the distance between the reference channel and the input channels. The smaller the distance, the more equal the compared signals are. The number of elements corresponds to the configured number of input channels. The values are updated only if the Call() method has been called before.

Syntax

Definition:

```
METHOD GetOutputArray : BOOL
VAR_INPUT
    pArrayOut : POITNER TO LREAL;
    nArrayOutSize : ANY;
END_VAR
```

Inputs

Name	Type	Description
pArrayOut	POINTER TO LREAL	Pointer to the output array: Dim: nBufferSize
nArrayOutSize	UDINT	Size of the output array

Return value

Name	Type	Description
GetOutputArray	BOOL	Returns `TRUE` if successful.

5.1.1.4.4.7 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 0: reference channel 1 to nChannels: comparison channels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.5 FB_ALY_LogicOperationCounter

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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_LogicOperationCounter
VAR_INPUT
    bPersistent: BOOL;
```

```
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bOperationOut: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bOperationOut	BOOL	Result of the logical operation.
nCount	ULINT	Incremented when the output value is bOperationOut=TRUE. The behavior depends on the configuration parameter eCountMode.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of last change from bOperationOut=TRUE.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initialize the number of input channels.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbLogicOperationCounter : FB_ALY_LogicOperationCounter(nNumChannels := 3);
    fbSystemTime : FB_ALY_GetSystemTime;
    eLogicOperator : E_ALY_LogicOperator := E_ALY_LogicOperator.AND_;
    eCountMode : E_ALY_CountMode := E_ALY_CountMode.OnChange;
    stThresholdLevel : ARRAY[1..3] OF ST_ALY_Threshold :=
```

```
        [(eComparisonOperator := E_ALY_ComparisonOperator.GreaterThan, fThreshold := 10),
         (eComparisonOperator := E_ALY_ComparisonOperator.LessThan, fThreshold := 2),
         (eComparisonOperator := E_ALY_ComparisonOperator.Equals, fThreshold := 1)];
    bConfigure : BOOL := TRUE;
    nInputCh1 : INT := 11;
    fInputCh2 : LREAL := 1.5;
    bInputCh3 : BOOL := TRUE;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbLogicOperationCounter.ConfigureChannel(1, stThresholdLevel[1]);
    fbLogicOperationCounter.ConfigureChannel(2, stThresholdLevel[2]);
    fbLogicOperationCounter.ConfigureChannel(3, stThresholdLevel[3]);
    fbLogicOperationCounter.Configure(eLogicOperator, eCountMode);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbLogicOperationCounter.SetChannelValue(1, nInputCh1);
fbLogicOperationCounter.SetChannelValue(2, fInputCh2);
fbLogicOperationCounter.SetChannelValue(3, bInputCh3);

fbLogicOperationCounter.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.5.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.5.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eLogicOperator : E_ALY_LogicOperator;
    eCountMode : E_ALY_CountMode;
END_VAR
```

Inputs

Name	Type	Description
eLogicOperator	E_ALY_LogicOperator [▶ 348]	Configured logical operator.
eCountMode	E_ALY_CountMode [▶ 348]	Mode of the result counter. *OnChange*: The counter counts every time the result changes to TRUE. *Cyclic*: The counter increments every cycle when the condition is TRUE.

Return value

Name	Type	Description
Configure	BOOL	Returns TRUE if successful.

5.1.1.4.5.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns TRUE if the reset was successful.

5.1.1.4.5.4 ConfigureChannel

Configure channel specific parameters. The configuration will not be processed until the method Configure() is called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    nChannel : UDINT
    stThresholdLevel : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns TRUE if successful.

5.1.1.4.5.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.5.6 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.4.6 FB_ALY_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_Multiplexer
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fResult: LREAL;
    nCurrentChannel: UDINT;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fResult	LREAL	Returns the signal of the selected input channel.
nCurrentChannel	UDINT	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.
nCount	ULINT	Starts with 1 for the channel selected at the start of the analysis and increments each time another channel is selected.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last channel change.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValues()	Local	Method for passing channel-specific values to the algorithm.
SetDefaultChannelValue()	Local	Method to pass a value to the algorithm that will be selected as default if no channel is selected.

Sample

```
VAR
    fbMultiplexer : FB_ALY_Multiplexer(nNumChannels := 3);
    fbSystemTime : FB_ALY_GetSystemTime;
    nInputCh1 : INT := 11;
    fInputCh2 : LREAL := 1.5;
    nInputCh3 : UDINT := 123;
```

```
    fDefault : LREAL := 3.1415;
    bConditionCh1 : BOOL;
    bConditionCh2 : BOOL;
    bConditionCh3 : BOOL;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Call algorithm
fbMultiplexer.SetChannelValues(1, bConditionCh1, nInputCh1);
fbMultiplexer.SetChannelValues(2, bConditionCh2, fInputCh2);
fbMultiplexer.SetChannelValues(3, bConditionCh3, nInputCh3);
fbMultiplexer.SetDefaultChannelValue(fDefault);

fbMultiplexer.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.6.1 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.4.6.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the methods SetChannelConditionAndValue () and SetChannelDefaultValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.6.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.6.4 SetChannelValues

Set channel specific input value and condition. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValues: BOOL
VAR_INPUT
    nChannel : UDINT;
    bCondition : BOOL;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
bCondition	BOOL	Channel condition.
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValues	BOOL	Returns `TRUE` if successful

5.1.1.4.6.5 SetDefaultChannelValue

Set default value which is set if no channel condition is TRUE. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetDefaultChannelValue: BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetDefaultChannelValue	BOOL	Returns `TRUE` if successful

5.1.1.4.7 FB_ALY_NumericalCompare_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_NumericalCompare_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bOperationOut: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bOperationOut	BOOL	Result of the comparison operation.
nCount	ULINT	Incremented when the output value is `bOperationOut=TRUE`. The behavior depends on the configuration parameter `eCountMode`.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last change to `bOperationOut=TRUE`.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbNumericalCompare : FB_ALY_NumericalCompare_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    fReference : LREAL := 40;
    eComparisonOperator : E_ALY_ComparisonOperator := E_ALY_ComparisonOperator.GreaterThan;
    eCountMode : E_ALY_CountMode := E_ALY_CountMode.OnChange;
    bUseAbsValues : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbNumericalCompare.Configure(fReference, eComparisonOperator, eCountMode, bUseAbsValues);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbNumericalCompare.SetChannelValue(nInput);
fbNumericalCompare.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.7.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.7.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fReference : LREAL;
    eComparisonOperator : E_ALY_ComparisonOperator;
    eCountMode : E_ALY_CountMode;
    bUseAbsValues : BOOL;
END_VAR
```

Inputs

Name	Type	Description
fReference	LREAL	Reference value for the comparison operation.
eComparisonOperator	E_ALY_ComparisonOperator [▶ 348]	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.
eCountMode	E_ALY_CountMode [▶ 348]	Mode of the result counter. *OnChange*: The counter counts every time the result changes to `TRUE`. *Cyclic*: The counter increments every cycle when the condition is `TRUE`.
bUseAbsValues	BOOL	If `TRUE`, the absolute values of the input signals and the reference are used.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.7.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.7.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.8 FB_ALY_NumericalCompare_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_NumericalCompare_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bOperationOut: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bOperationOut	BOOL	Result of the comparison operation.
nCount	ULINT	Incremented when the output value is bOperationOut=TRUE. The behavior depends on the configuration parameter eCountMode.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last change to bOperationOut=TRUE.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbNumericalCompare : FB_ALY_NumericalCompare_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    eComparisonOperator : E_ALY_ComparisonOperator := E_ALY_ComparisonOperator.GreaterThan;
    eCountMode : E_ALY_CountMode := E_ALY_CountMode.OnChange;
    bUseAbsValues : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    nInputCh1 : INT;
    fInputCh2 : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbNumericalCompare.Configure(eComparisonOperator, eCountMode, bUseAbsValues);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbNumericalCompare.SetChannelValue(1, nInputCh1);
fbNumericalCompare.SetChannelValue(2, fInputCh2);
fbNumericalCompare.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.8.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns TRUE if successful.

5.1.1.4.8.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eComparisonOperator : E_ALY_ComparisonOperator;
    eCountMode : E_ALY_CountMode;
    bUseAbsValues : BOOL;
END_VAR
```

Inputs

Name	Type	Description
eComparisonOperator	E_ALY_ComparisonOperator [▶ 348]	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.
eCountMode	E_ALY_CountMode [▶ 348]	Mode of the result counter. *OnChange*: The counter counts every time the result changes to TRUE. *Cyclic*: The counter increments every cycle when the condition is TRUE.
bUseAbsValues	BOOL	If TRUE, the absolute values of the input signals are used.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.8.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.8.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *Channel 1* 2: *Channel 2*
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.9 FB_ALY_StringCompare_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StringCompare_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bStringMatch: BOOL;
```

```
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bStringMatch	BOOL	Result of the string comparison.
nCount	ULINT	Incremented when the output value is bStringMatch=TRUE. The behavior depends on the configuration parameter eCountMode.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last change to bStringMatch=TRUE.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbStringCompare : FB_ALY_StringCompare_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    sReference : STRING := 'String to compare with';
    eStringCompareMode : E_ALY_StringCompareMode := E_ALY_StringCompareMode.Equals;
    eCountMode : E_ALY_CountMode := E_ALY_CountMode.OnChange;
    bCaseSensitive : BOOL := TRUE;
    bConfigure : BOOL := TRUE;
    sInput : STRING := 'Input string';
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbStringCompare.Configure(sReference, eStringCompareMode, eCountMode, bCaseSensitive);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbStringCompare.SetChannelValue(sInput);
fbStringCompare.Call(fbSystemTime.tSystemTime)
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.9.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns TRUE if successful.

5.1.1.4.9.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    sReference : STRING(255);
    eStringCompareMode : E_ALY_StringCompareMode;
    eCountMode : E_ALY_CountMode;
    bCaseSensitive : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sReference	STRING(255)	Reference string for the comparison.
eStringCompareMode	E_ALY_StringCompareMode [▶ 349]	String compare mode: *Equals*: Input string corresponds to the reference string. *BeginsWith*: Input string starts with the reference string. *Contains*: Input string contains the reference string.
eCountMode	E_ALY_CountMode [▶ 348]	Mode of the result counter. *OnChange*: The counter counts every time the result changes to TRUE. *Cyclic*: The counter increments every cycle when the condition is TRUE.
bCaseSensitive	BOOL	If TRUE, upper and lower case is taken into account.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.9.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.9.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY_STRING;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY_STRING	Input value of a string data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.10 FB_ALY_StringCompare_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StringCompare_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bStringMatch: BOOL;
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bStringMatch	BOOL	Result of the string comparison.
nCount	ULINT	Incremented when the output value is bStringMatch=TRUE. The behavior depends on the configuration parameter eCountMode.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last change to bStringMatch=TRUE.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbStringCompare : FB_ALY_StringCompare_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    eStringCompareMode : E_ALY_StringCompareMode := E_ALY_StringCompareMode.Equals;
    eCountMode : E_ALY_CountMode := E_ALY_CountMode.OnChange;
    bCaseSensitive : BOOL := TRUE;
    bConfigure : BOOL := TRUE;
    sInputCh1 : STRING := 'Input string';
    sInputCh2 : STRING := 'String to compare with';
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbStringCompare.Configure(eStringCompareMode, eCountMode, bCaseSensitive);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbStringCompare.SetChannelValue(1, sInputCh1);
fbStringCompare.SetChannelValue(2, sInputCh2);
fbStringCompare.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.10.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.10.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eStringCompareMode : E_ALY_StringCompareMode;
    eCountMode : E_ALY_CountMode;
    bCaseSensitive : BOOL;
END_VAR
```

Inputs

Name	Type	Description
eStringCompareMode	E_ALY_StringCompareMode [▶ 349]	String compare mode: *Equals*: Input string corresponds to the reference string. *BeginsWith*: Input string starts with the reference string. *Contains*: Input string contains the reference string.
eCountMode	E_ALY_CountMode [▶ 348]	Mode of the result counter. *OnChange*: The counter counts every time the result changes to `TRUE`. *Cyclic*: The counter increments every cycle when the condition is `TRUE`.
bCaseSensitive	BOOL	If `TRUE`, upper and lower case is taken into account.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.10.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.10.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: *Channel 1* 2: *Channel 2*
Input	ANY_STRING	Input value of a string data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.4.11 FB_ALY_DetectValueChange_1Ch

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_DetectValueChange_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bValueChanged: BOOL;
```

```
    nCount: ULINT;
    fbTimeLastEvent: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bValueChanged	BOOL	TRUE, if a change of value was detected, otherwise FALSE.
nCount	ULINT	Incremented when the output value is bValueChanged=TRUE.
fbTimeLastEvent	FB_ALY_DateTime	Timestamp of the last detected value change.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbDetectValueChange : FB_ALY_DetectValueChange_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    fTolerance : LREAL := 0.1;
    bConfigure : BOOL := TRUE;
    fIn : LREAL;
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbDetectValueChange.Configure(fTolerance);
END_IF

// Call algorithm
fbDetectValueChange.SetChannelValue(fIn);
fbDetectValueChange.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.4.11.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.4.11.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fTolerance : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fTolerance	LREAL	The tolerance refers to the input value of the last detected value change. If the input value is outside this tolerance, a value change is detected.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.4.11.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.4.11.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.5 Math

5.1.1.5.1 FB_ALY_Integrator_1Ch

The *Integrator 1Ch* integrates the input value over time with a base unit of one second and provides the result of this integration operation. For the approximation of this integral the trapezoidal rule is used. The trapezoidal $T(t_n, t_{n+1})$ between two subsequent timestamps t_n and t_{n+1} with the values y_n and y_{n+1} is calculates as

$$T(t_n, t_{n+1}) = (t_{n+1}[s] - t_n[s]) \cdot \frac{y_n + y_{n+1}}{2}$$

.

If the integration mode "absolute" ("|x|") is chosen in the configuration, y_n and y_{n+1} are substituted by their absolute values in the above equation.

In each cycle the trapezoidal between the current and the last timestamp is calculated and added to the sum of trapezoids starting from the beginning of the analysis. Additionally, this sum can be scaled by a factor that can be configured individually.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_Integrator_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
```

```
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fResult: LREAL;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fResult	LREAL	Outputs the result of the integration.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbIntegrator : FB_ALY_Integrator_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    eIntegrationMode : E_ALY_IntegrationMode := E_ALY_IntegrationMode.Direct;
    fFactor : LREAL := 1.0;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbIntegrator.Configure(eIntegrationMode, fFactor);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbIntegrator.SetChannelValue(nInput);
fbIntegrator.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.5.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.5.1.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eIntegrationMode : E_ALY_IntegrationMode;
    fFactor : LREAL;
END_VAR
```

Inputs

Name	Type	Description
eIntegrationMode	E_ALY_IntegrationMode [▶ 348]	Integration mode *Direct*: Integration of the input values *Absolut*: Integration of the absolute input values
fFactor	LREAL	The integral is multiplied by this factor.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.5.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.5.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.5.2 FB_ALY_MathOperation

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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_MathOperation
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fResult: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fResult	LREAL	Outputs the result of the mathematical operation.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbMathOperation : FB_ALY_MathOperation(nNumChannels := 3);
    eMathOperator : E_ALY_MathOperator := E_ALY_MathOperator.Addition;
    bUseAbsValues : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    nInputCh1 : INT;
    fInputCh2 : LREAL;
    nInputCh3 : UDINT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbMathOperation.Configure(eMathOperator, bUseAbsValues);
END_IF

// Call algorithm
fbMathOperation.SetChannelValue(1, nInputCh1);
fbMathOperation.SetChannelValue(2, fInputCh2);
fbMathOperation.SetChannelValue(3, nInputCh3);
fbMathOperation.Call();
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.5.2.1 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.5.2.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.5.2.3 Configure

Configure the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eMathOperator : E_ALY_MathOperator;
    bUseAbsValues : BOOL;
END_VAR
```

Inputs

Name	Type	Description
eMathOperator	E_ALY_MathOperator [▶ 349]	Mathematical operator *Addition* *Subtraction* *Multiplication* *Division* *PowerOf* *Modulo*
bUseAbsValues	BOOL	If TRUE, the absolute values of the input signals are used.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.5.2.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.5.2.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.5.3 FB_ALY_MathOperation_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_MathOperation_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fResult: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fResult	LREAL	Outputs the result of the mathematical operation.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbMathOperation : FB_ALY_MathOperation_1Ch;
    fOperand : LREAL := 50;
    eMathOperator : E_ALY_MathOperator := E_ALY_MathOperator.Addition;
    bUseAbsValues : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbMathOperation.Configure(fOperand, eMathOperator, bUseAbsValues);
END_IF

// Call algorithm
fbMathOperation.SetChannelValue(nInput);
fbMathOperation.Call();
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.5.3.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.5.3.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    fOperand : LREAL;
    eMathOperator : E_ALY_MathOperator;
    bUseAbsValues : BOOL;
END_VAR
```

Inputs

Name	Type	Description
fOperand	LREAL	Operand for the mathematical operation
eMathOperator	E_ALY_MathOperator [▶ 349]	Mathematical operator *Addition* *Subtraction* *Multiplication* *Division* *PowerOf* *Modulo*
bUseAbsValues	BOOL	If TRUE, the absolute values of the input signals are used.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.5.3.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.5.3.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.5.4 FB_ALY_RMS_1Ch

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

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

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

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_RMS_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
SetCascades()	Local	Method for configuring the cascades. The call is made after the call to the Configure method.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Sample

```
VAR
    fbRMS_1Ch : FB_ALY_RMS_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nNumCascades : UDINT := 5;
    fSampleRate : UDINT := 1000;
    eStartupBehaviour : E_ALY_CascadeStartupBehaviour := E_ALY_CascadeStartupBehaviour.UsePreviousCa
scadeValue;
    aCascadesConfigArray : ARRAY[1..5] OF LTIME := [LTIME#20MS, LTIME#1S, LTIME#1M, LTIME#1H, LTIME#
1D];
    bConfigure : BOOL := TRUE;
    fInput : LREAL;
    aRMS : ARRAY[1..5] OF LREAL;
    aNewResult : ARRAY[1..5] OF BOOL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbRMS_1Ch.Configure(nNumCascades, fSampleRate, eStartupBehaviour);
    fbRMS_1Ch.SetCascades(ADR(aCascadesConfigArray), SIZEOF(aCascadesConfigArray));
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbRMS_1Ch.SetChannelValue(fInput);
fbRMS_1Ch.Call(fbSystemTime.tSystemTime, ADR(aRMS), SIZEOF(aRMS), ADR(aNewResult), SIZEOF(aNewResult
));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.5.4.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
    pRmsArrayOut : POINTER TO LREAL;
    nRmsArrayOutSize : UDINT;
    pNewResultArrayOut : POINTER TO BOOL;
    nNewResultArrayOutSize : UDINT
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.
pRmsArrayOut	POINTER TO LREAL	Pointer to an array in which the RMS results are to be stored. The dimension corresponds to the number of configured cascades.
nRmsArrayOutSize	UDINT	Size of the RMS array.
pNewResultArrayOut	POINTER TO BOOL	Pointer to an array in which the states of new RMS results are to be stored. The dimension corresponds to the number of configured cascades.
nNewResultArrayOutSize	UDINT	Size of the status array.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.5.4.2 Configure

Configuring the algorithm. The output cascades can be set using the SetCascades() method.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nNumCascades : UDINT;
    fSampleRate : LREAL;
    eStartupBehaviour : E_ALY_CascadeStartupBehaviour;
END_VAR
```

Inputs

Name	Type	Description
nNumCascades	UDINT	Number of output cascades
fSampleRate	LREAL	Sampling rate of the system to be analyzed in Hz
eStartupBehaviour	E_ALY_CascadeStartupBehaviour [▶ 347]	Start-up behavior *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.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.5.4.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.5.4.4 SetCascades

Configuration of the output cascades. Before that, the number of cascades must be set with the Configure() method.

Syntax

Definition:

```
METHOD SetCascades : BOOL
VAR_INPUT
    pCascadesConfigArray : POINTER TO LTIME;
    nCascadesConfigArraySize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pCascadesConfigArray	POINTER TO LTIME	Pointer to an array used to configure the time intervals of the cascades. The time interval of the configured cascade must correspond to an integer multiple of the time interval of the previous cascade.
nCascadesConfigArraySize	UDINT	Size of the configuration array.

Return value

Name	Type	Description
SetCascades	BOOL	Returns `TRUE` if successful.

5.1.1.5.4.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.5.5 FB_ALY_SlopeAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_SlopeAnalysis_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fSlope: LREAL;
    fSlopeMin: LREAL;
    fSlopeMax: LREAL;
    fDeltaValue: LREAL;
    fbDeltaTime: FB_ALY_Timespan;
    fbTimeSlope: FB_ALY_DateTime;
    fbTimeSlopeMin: FB_ALY_DateTime;
    fbTimeSlopeMax: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fDeltaValue	LREAL	Difference of the values of the first and last element of the window.
fbDeltaTime	FB_ALY_Timespan	Difference of timestamp values of the first and last element of the window. It is scaled to seconds.
fSlope	LREAL	Division of fDeltaValue and fDeltaTime.
fSlopeMin	LREAL	Minimum occurred value of fSlope.
fSlopeMax	LREAL	Maximum occurred value of fSlope.
fbTimeSlope	FB_ALY_DateTime	Timestamp of the calculated slope. It is placed in the center of the configured window.
fbTimeSlopeMin	FB_ALY_DateTime	Timestamp of the last updated fSlopeMin.
fbTimeSlopeMax	FB_ALY_DateTime	Timestamp of the last updated fSlopeMax.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbSlopeAnalysis : FB_ALY_SlopeAnalysis_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nNumValues : UDINT := 200;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbSlopeAnalysis.Configure(nNumValues);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbSlopeAnalysis.SetChannelValue(nInput);
fbSlopeAnalysis.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.5.5.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.5.5.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nNumValues : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nNumValues	UDINT	Distance between the two data sets for the calculation of the slope.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.5.5.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.5.5.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6 Statistics

5.1.1.6.1 FB_ALY_ArrayStatistics

The *Array Statistics* algorithm calculates various statistical quantities based on the input array.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ArrayStatistics
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fMin: LREAL;
    nIdxMin: UDINT
    fMax: LREAL;
    nIdxMax: UDINT;
    fMaxDelta: LREAL;
    nIdxMaxDelta: UDINT;
    nCountPeaks: ULINT;
    nCountValleys: ULINT;
    fSum: LREAL;
    fMean: LREAL;
    fStandardDeviation: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fMin	LREAL	Smallest value in the input array.
nIdxMin	UDINT	Array index from `fMin`. The indexing starts at 1.
fMax	LREAL	Largest value in the input array.
nIdxMax	UDINT	Array index from `fMax`. The indexing starts at 1.
fMaxDelta	LREAL	Maximum of the absolute difference between two consecutive values in the input array.
nIdxMaxDelta	UDINT	Array index from `fMaxDelta`. The indexing starts at 1.
nCountPeaks	ULINT	Total number of peaks identified.
nCountValleys	ULINT	Total number of valleys identified.
fSum	LREAL	Sum over the entire input array.
fMean	LREAL	Mean value over the entire input array.
fStandardDeviation	LREAL	Standard deviation over the entire input array.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions. The configuration is identical for all channels.
Reset()	Local	Resets all internal states or the calculations performed so far.

Sample

```
VAR
    fbArrayStatistics : FB_ALY_ArrayStatistics;
    bUseBesselCorrection : BOOL := TRUE;
    fThresholdReversal : LREAL := 0.5;
    fThresholdDelta : LREAL := 0.5;
    bConfigure : BOOL := TRUE;
    aInput : ARRAY[1..20] OF LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbArrayStatistics.Configure(bUseBesselCorrection := bUseBesselCorrection, fThresholdReversal :=
fThresholdReversal, fThresholdDelta := fThresholdDelta);
END_IF

// Call algorithm
fbArrayStatistics.Call(ADR(aInput), SIZEOF(aInput));
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.1.1 Call

Calling the algorithm.

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    pArrayIn : PVOID;
    nArrayInSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pArrayIn	PVOID	Pointer to the input array.
nArrayInSize	UDINT	Size of the input array

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.1.2 Configure

Configure the algorithm.

Configuration options

- **Use Bessel Correction:** if the checkbox is activated, Bessel correction will be applied. In order to obtain an expectation-true result for random samples, this parameter must be activated. The parameter is only relevant for the calculation of the standard deviation.

The empirical standard deviation, without Bessel's correction

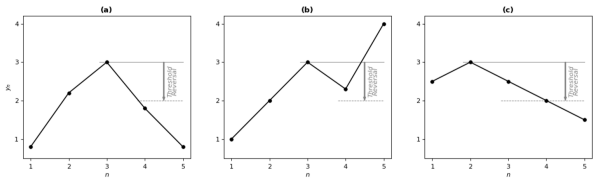
$$s' = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x[n] - \overline{x})^2}$$

The empirical standard deviation, with Bessel's correction

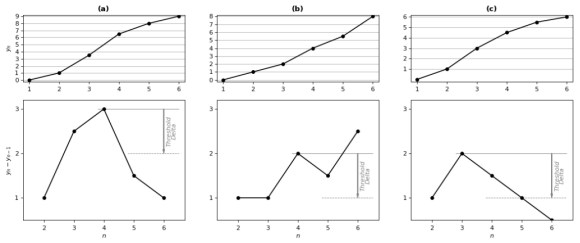
$$s = \sqrt{\frac{1}{N-1}\sum_{n=0}^{N-1}(x[n] - \overline{x})^2}$$

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

and y_5 *exceeds Threshold Reversal*. The value y_2 cannot be identified as a peak beforehand because the difference between y_2 and y_3 (y_4) is less than/equal to *Threshold Reversal* and it is not known whether the values will continue to decrease.



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



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

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    bUseBesselCorrection : BOOL;
    fThresholdReversal : LREAL;
    fThresholdDelta : LREAL;
END_VAR
```

Inputs

Name	Type	Description
bUseBesselCorrection	BOOL	Use of the Bessel correction (see above).
fThresholdReversal	LREAL	Threshold for identifying reversals (see above).
fThresholdDelta	LREAL	Threshold for identifying Delta maxima (see above).

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.2 FB_ALY_CorrelationFunction

The *Correlation Function* function block calculates the discrete correlation function between a reference signal (Channel Ref) and one or more other signals (Channel 00, ..., Channel 0n). The correlation coefficients are calculated for time shifts of *m* cycles between the two signals, with the maximum and minimum values for *m* being limited by the parameters *Minimum Lag* (negative integer) and *Maximum Lag* (positive integer).

The *Step Size* parameter determines the number of cycles by which the signals are moved to calculate two consecutive correlation coefficients. I.e. *m* is always a multiple of *StepSize*. Accordingly, for *Minimum Lag* and *Maximum Lag* only multiples of *StepSize* are allowed. If the *StepSize* is set to one and *Minimum Lag* is set to -6 and *Maximum Lag* is set to +4 -for example-, correlation coefficients are calculated for shifts of -6, -5, -4, -3, -2, -1, 0, +1, +2, +3 and +4 cycles. If the *StepSize* is set to two, coefficients are calculated for shifts by -6, -4, -2, 0, +2 and +4 cycles.

The coefficients can be calculated over different timeframes, which are set by the *Window Mode* parameter. In *Continuous* mode, all values since the beginning of the analysis are included in the calculations. In *SlidingWindow* mode, the calculation runs continuously for the last number of cycles set by the *Window Size*. In *FixWindow* mode, the calculation is also done for the number of cycles set by the *Window Size*. However, the calculation restarts after each *WindowSize* cycle and the output values are updated only when the last cycle of a window is run through.

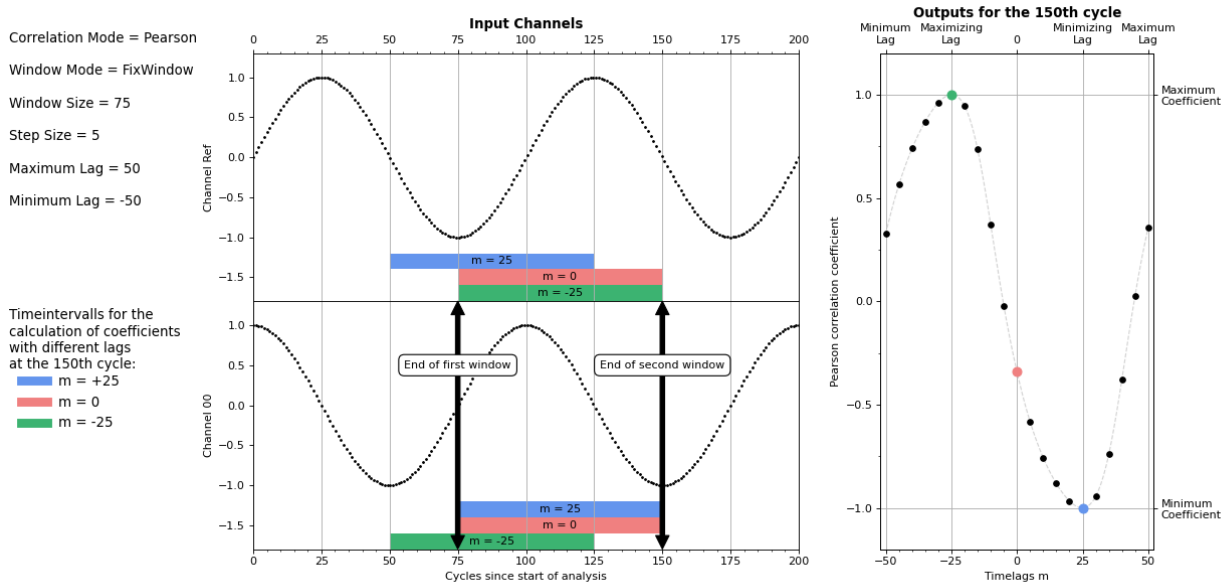
If *m* is not zero, the window for the corresponding signal shifts by *m* cycles. The number of values included in the calculation of the coefficients is the same for all values of *m*. The only exceptions to this are the results from the first cycles after the start of the analysis. If the number of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

For positive values of *m*, the values of the reference signal (Channel Ref) are stored in a ring memory, so that the respective value of the reference signal received before *m* cycles can be compared with the current values from Channel 00 to Channel 0n. This corresponds to a shift of the reference signal into the past. For negative values of *m*, the reference signal would accordingly have to be moved into the future. Since this is obviously not possible, the second signal (Channel 00, .. Channel0n) is saved and moved backward instead.

The correlation coefficients can be calculated according to different calculation rules. This is determined by the *Correlation Mode*. In the *Base* and *Normed* modes, the coefficients are calculated analogously to the definition from signal processing, which calculates the correlation over the convolution. In *Normed* mode, the coefficients are also divided by the number of summands. *Covariance* and *CovarianceBessel* calculate the covariance without and with Bessel correction. *Pearson* mode uses the definition of the Pearson correlation coefficient commonly used in statistics. The exact calculation rules are mathematically listed in the configuration options. Here, x_n denotes the value of the reference signal and y_n denotes the value of the second signal (in each case Channel00, ..., Channel0n) at the timestamp t_n (corresponding to the nth cycle since the start of the analysis or since reset, except for the cycles in which *Enable Execution* = FALSE). The value of *N* depends on the *WindowMode* you select. For *SlidingWindow* mode and *FixWindow* mode, *N* is equal to the *WindowSize*, provided a corresponding number of cycles have already elapsed since the start of the analysis, so that x_{n-N-m} (or y_{n-N+m}) has been recorded, otherwise *N* will be reduced to *n-m+1* or *n+m+1* respectively. Note that in *FixWindow* mode the output values are only updated every *WindowSize* cycle. In *Continuous* mode, *N* = *n+1* always applies.

In the illustration, the output values of the function block for two signals (Channel Ref and Channel 00) for a given cycle (n = 150) are shown as an example of a configuration (*Correlation Mode* = *Pearson*, *Window Mode* = *FixWindow*, *Window Size* = 75, *Step Size* = 5, *Maximum Lag* = 50, *Minimum Lag* = -50). In the two left plots, the input signals Channel Ref and Channel 00 are shown over time. The right plot shows the discrete correlation function (the Pearson correlation coefficients in relation to m). Coefficients are shown for the shifts m = -50, -45, -40, -35, -30, -25, -20, -15, -10, -5, 0, +5, +10, +15, +20, +25, +30, +35, +40, +45, +50. These are output as an array in the function block. In addition to the two parameters Minimum Lag and Maximum Lag, the output values Minimizing Lag and Maximizing Lag are marked on the abscissa. The corresponding coefficients Minimum Coefficient and Maximum Coefficient, which also represent outputs of

the function block, are marked on the ordinate. For the shifts m = -25, m = 0 and m =+25 in the plots of the input channels (left), the time ranges included in the calculation of the respective coefficient are highlighted in color. In the right plot, the corresponding points are colored.



Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_CorrelationFunction
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
GetChannelOutputArray()	Local	Retrieves the result array for a specific channel without adding new values.
GetChannelOutputValues()	Local	Retrieves the results (individual values) for a specific channel without adding new values.

Sample

```
VAR
    fbCorrelationFunction : FB_ALY_CorrelationFunction(nMinLag := -50, nMaxLag := 50, nStepSize := 5
, nNumChannels := 1, eWindowMode := E_ALY_WindowMode.FixWindow, eCorrelationMode := E_ALY_Correlatio
nMode.Pearson, nWindowSize := 75);

    // Configuration
    nMinLag : DINT := -50;
    nMaxLag : DINT := 50;
    nStepSize : UDINT := 5;
    nNumChannels : UDINT := 1;
    eWindowMode : E_ALY_WindowMode := E_ALY_WindowMode.FixWindow;
    eCorrelationMode : E_ALY_CorrelationMode := E_ALY_CorrelationMode.Pearson;
    nWindowSize : UDINT := 75;

    bConfigure : BOOL := TRUE;

    // Inputs
    nInputRef : INT;
    fInputCh1 : LREAL;

    // Results
    fMinCoef : LREAL;
    fMaxCoef : LREAL;
    nMinimizingLag : DINT;
    nMaximizingLag : DINT;
    aCoefficients : ARRAY[-10..10] OF LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbCorrelationFunction.Configure(
        nMinLag := nMinLag,
        nMaxLag := nMaxLag,
        nStepSize := nStepSize,
        nNumChannels := nNumChannels,
        eWindowMode := eWindowMode,
        eCorrelationMode := eCorrelationMode,
        nWindowSize := nWindowSize);
END_IF

// Call algorithm
fbCorrelationFunction.SetChannelValue(0, nInputRef);
fbCorrelationFunction.SetChannelValue(1, fInputCh1);
fbCorrelationFunction.Call();

fbCorrelationFunction.GetChannelOutputArray(1, ADR(aCoefficients), SIZEOF(aCoefficients));
fbCorrelationFunction.GetChannelOutputValues(1, fMinCoef, fMaxCoef, nMinimizingLag, nMaximizingLag);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.2.1 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used

5.1.1.6.2.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.2.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nMinLag : DINT;
    nMaxLag : DINT;
    nStepSize : UDINT;
    nNumChannels : UDINT;
    eWindowMode : E_ALY_WindowMode;
```

```
    eCorrelationMode : E_ALY_CorrelationMode;
    nWindowSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nMinLag	UDINT	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.
nMaxLag	UDINT	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.
nStepSize	UDINT	Specifies by how many cycles the signals are shifted to calculate two consecutive correlation coefficients.
nNumChannels	UDINT	The number of channels that are correlated with the reference signal.
eWindowMode	E_ALY_WindowMode [▶ 351]	Specifies the type of window used to calculate the coefficients: *Continuous*: All values since the start of the analysis are included in the analysis with equal weighting. *SlidingWindow:* The calculation is done via a window of the size Window Size. The current values are always included in the analysis and outputs are updated with each cycle. *FixWindow:* The outputs are updated every Window Size cycles and calculated via a window with the length Window Size.

Name	Type	Description
eCorrelationMode	E_ALY_CorrelationMode [▶ 350]	The coefficients are calculated based on one of the following definitions: *Base:* $$C_{xy}[m,t_n]=\begin{cases}\sum_{i=0}^{N-1} x_{n-i-m}\cdot y_{n-i}, & m\geq 0\\ \sum_{i=0}^{N-1} x_{n-i}\cdot y_{n-i+m}, & m<0\end{cases}$$ *Normed:* $$\hat{C}_{xy}[m,t_n]=\begin{cases}\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i-m}\cdot y_{n-i}, & m\geq 0\\ \frac{1}{N}\sum_{i=0}^{N-1} x_{n-i}\cdot y_{n-i+m}, & m<0\end{cases}$$ *Covariance:* $$cov_{xy}[m,t_n]=\begin{cases}\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i-m}\cdot y_{n-i}-\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i-m}\cdot\frac{1}{N}\sum_{i=0}^{N-1} y_{n-i}, & m\geq 0\\ \frac{1}{N}\sum_{i=0}^{N-1} x_{n-i}\cdot y_{n-i+m}-\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i}\cdot\frac{1}{N}\sum_{i=0}^{N-1} y_{n-i+m}, & m<0\end{cases}$$ *CovarianceBessel:* $$\widetilde{cov}_{xy}[m,t_n]=\frac{N}{N-1}cov_{xy}[m,t_n]$$ *Pearson:* $$\rho_{xy}[m,t_n]=\frac{cov(x,y)[m,t_n]}{\sigma_x[m,t_n]\sigma_y[m,t_n]},\ where\ \sigma_x^2[m,t_n]=cov_{xx}[m,t_n]$$
nWindowSize	UDINT	For the *SlidingWindow* and *FixWindow* window modes, specifies the number of cycles over which the coefficients are calculated. For the Window Mode *Continuous* the setting of nWindowSize has no effect. 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 must be taken into account when setting these parameters.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.2.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.2.5 GetChannelOutputArray

Retrieves a channel-specific output array. The output values were updated in the previously called Call() method.

Syntax

Definition:

```
METHOD GetChannelOutputArray : BOOL
VAR_INPUT
    nChannel : UDINT;
    pCorrCoefsArrayOut : PVOID;
    nCorrCoefsArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
pCorrCoefsArrayOut	PVOID	Pointer to an array to which the coefficients are to be written.
nCorrCoefsArrayOutSize	UDINT	Array size

Return value

Name	Type	Description
GetChannelOutputArray	BOOL	Returns `TRUE` if successful.

5.1.1.6.2.6 GetChannelOutputValues

Retrieves channel-specific output values. The output values were updated in the previously called Call() method.

Syntax

Definition:

```
METHOD GetChannelOutputValues : BOOL
VAR_INPUT
    nChannel : UDINT;
    pCorrCoefsArrayOut : PVOID;
    nCorrCoefsArrayOutSize : UDINT;
END_VAR
VAR_IN_OUT
    fMinCoef : LREAL;
    fMaxCoef : LREAL;
    nMinimizingLag : DINT;
    nMaximizingLag : DINT;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
fMinCoef	LREAL	Writing to variable takes place with the minimum coefficient of the output array.
fMaxCoef	LREAL	Writing to variable takes place with the maximum coefficient of the output array.
nMinimizingLag	DINT	Writing to the variable takes place with the lag at which the coefficient is smallest.
nMaximizingLag	DINT	Writing to the variable takes place with the lag at which the coefficient is greatest.

Return value

Name	Type	Description
GetChannelOutputValues	BOOL	Returns `TRUE` if successful.

5.1.1.6.2.7 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (0 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.3 FB_ALY_CorrelationFunctionReference

The *Correlation Function Reference* function block calculates the discrete correlation function between a recorded signal (referred to below as reference signal), which is read from a tcab file, and one or more input signals (Channel 00, ..., Channel 0n).

The correlation coefficients are calculated for time shifts of *m* cycles between the two signals, with the maximum and minimum values for *m* being limited by the parameters *Minimum Lag* (negative integer) and *Maximum Lag* (positive integer).

The *Step Size* parameter determines the number of cycles by which the signals are moved to calculate two consecutive correlation coefficients. I.e. *m* is always a multiple of the *Step Size*. Accordingly, only multiples of the *Step Size* are also allowed for *Minimum Lag* and *Maximum Lag*. If the *Step Size* is set to one,

Minimum Lag to -6 and *Maximum Lag* to +4, for example, coefficients are calculated for shifts by -6, -5, -4, -3, -2, -1, 0, +1, +2, +3, and +4 cycles. If the *Step Size* is set to two, coefficients are calculated for shifts by -6, -4, -2, 0, +2, and +4 cycles.

From the start of the analysis, a value is processed from the read-in signal per cycle. When the end of the file is reached, the process starts again with the first value of the file. The read-in signal is therefore assumed to be periodic. If you only want to correlate certain periods of the input signal with the reference signal, you can control this via the *Enable Execution* and *Reset* inputs as well as via the *New Result* output.

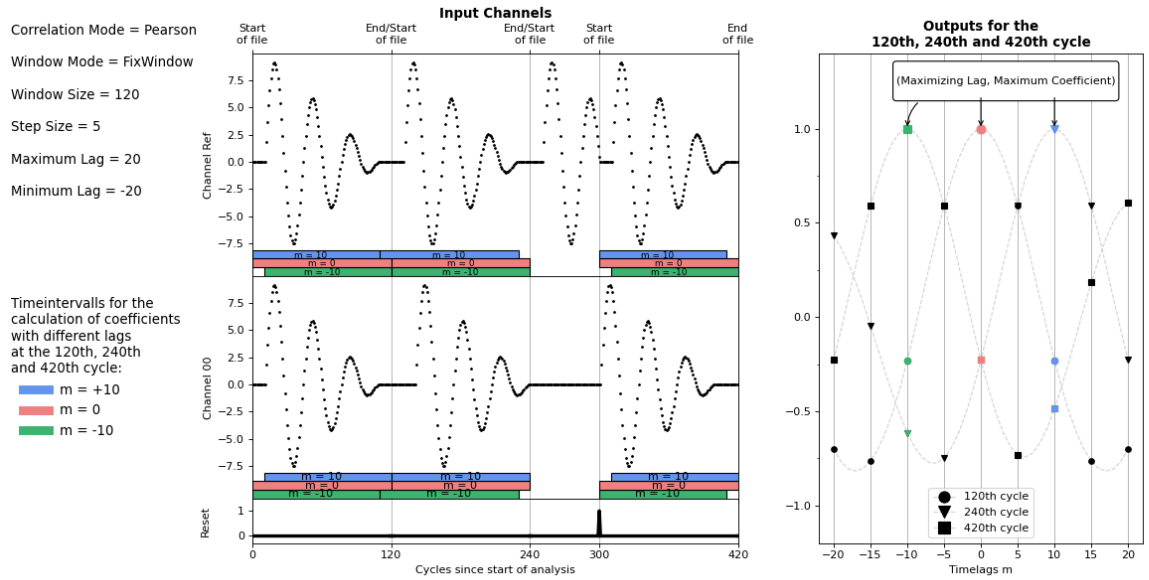
The correlation coefficients can be calculated over different timeframes. which are set by the *Window Mode* parameter. In *Continuous* mode, all values since the beginning of the analysis are included in the calculations. In *SlidingWindow* mode, the calculation runs continuously for the last number of cycles set by the *Window Size*. In *FixWindow* mode, the calculation is done over the length of the reference signal and the output values are always updated at the end of the recorded sequence.

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

For positive values of *m,* the values of the reference signal (Channel Ref) are stored in a ring memory, so that the respective value of the reference signal received before *m* cycles can be compared with the current values from Channel 00 to Channel 0n. This corresponds to a shift of the reference signal into the past. For negative values of *m,* the reference signal would accordingly have to be moved into the future. Since this is obviously not possible, the second signal (Channel 00, .. Channel0n) is saved and moved backward instead.

The correlation coefficients can be calculated according to different calculation rules. This is determined by the *Correlation Mode*. In the *Base* and *Normed* modes, the coefficients are calculated analogously to the definition from signal processing, which calculates the correlation over the convolution. In *Normed* mode, the coefficients are also divided by the number of summands. *Covariance* and *CovarianceBessel* calculate the covariance without and with Bessel correction. *Pearson* mode uses the definition of the Pearson correlation coefficient commonly used in statistics. The exact calculation rules are mathematically listed in the configuration options. Here, x_n denotes the value of the reference signal and y_n denotes the value of the input signal (in each case Channel00, ..., Channel0n) at the timestamp t_n (corresponding to the n^{th} cycle since the start of the analysis or since reset, except for the cycles in which *Enable Execution* = FALSE). The value of *N* depends on the *WindowMode* you select. For *SlidingWindow* mode and *FixWindow* mode, *N* is equal to the *WindowSize,* provided a corresponding number of cycles have already elapsed since the start of the analysis, so that x_{n-N-m} (or y_{n-N+m}) has been recorded, otherwise *N* will be reduced to *n-m+1* or *n+m+1* respectively. In FixWindow mode, the window size is not to be set manually, but corresponds to the length of the read-in signal section (reference signal) and the output values are updated at the end of the signal section. In *Continuous* mode, *N* = *n+1* always applies.

The illustration shows the different input and output values of the function block for a configuration (*Correlation Mode* = *Pearson*, *Window Mode* = *FixWindow*, *Window Size* = 120 (= number of values in the file), *Step Size* = 5, *Maximum Lag* = 20, *Minimum Lag* = -20) and an input channel. On the left side, the input signals Channel 00 and *Reset* are shown in the two lower plots, above which the read-in sequence is shown on the same timeline, according to its processing. The signal starts to be read in at the beginning of the analysis. If the last value from the file is processed in the 120^{th} (or 240^{th}) cycle, the process starts again with the first one. In the 300^{th} 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 120^{th}, 240^{th} and 420^{th} 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 420^{th} 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.



Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_CorrelationFunctionReference
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bBusy: BOOL;
    eState: E_ALY_ReadState;
    fValueRead: LREAL;
    stFileHeader: ST_ALY_FileHeader;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bBusy	BOOL	TRUE if the FB is active due to file accesses.
eState	E_ALY_ReadState [▶ 349]	Current state of the FB due to asynchronous file access.
fValueRead	LREAL	Value of the data point that is read from a file.
stFileHeader	ST_ALY_FileHeader	Header information of the file that was read.

Methods

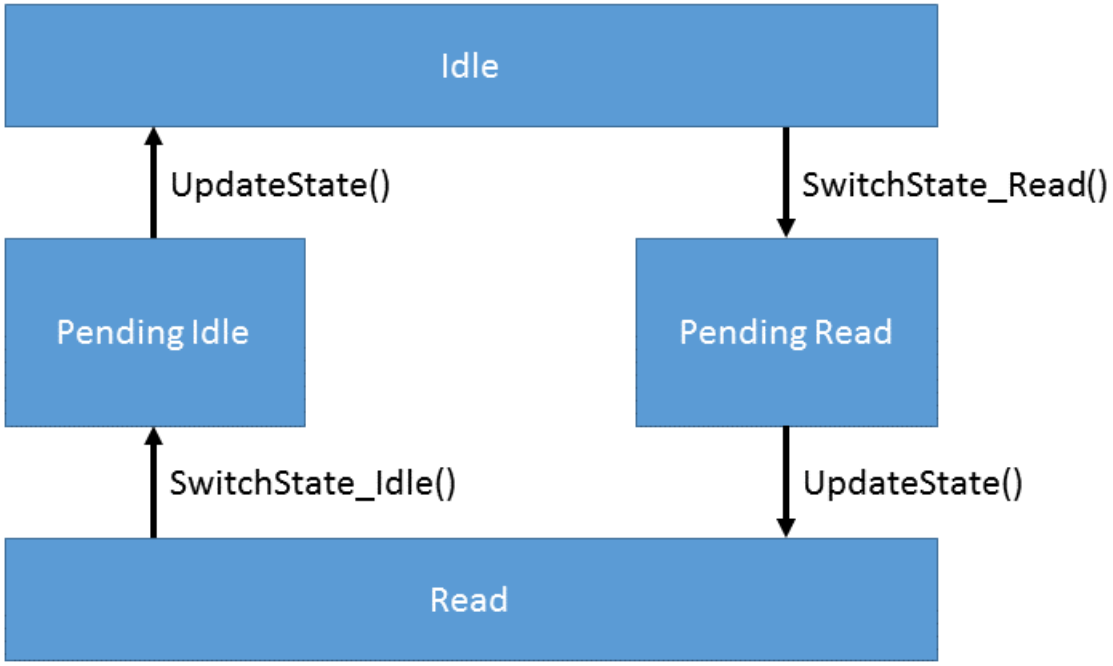
Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
GetChannelOutputArray()	Local	Retrieves the result array for a specific channel without adding new values.
GetChannelOutputValues()	Local	Retrieves the results (individual values) for a specific channel without adding new values.
SwitchState_Idle()	Local	Initiating the change from state Read to state Idle. See state diagram.
SwitchState_Read()	Local	Initiating the change from state Idle to state Read. See state diagram.
UpdateState()	Local	Updating the state after a state change has been initiated and while the target state has not been reached.
GetBusyState()	Local	Returns TRUE if the FB is active.

State diagram

Due to the asynchronous file access in real-time applications, this function block requires a state machine to prepare and complete the file access.

At startup the function block is in *Idle* state. To read the data from the file, the state must switch to *Read*. Therefore, the method *SwitchState_Read()* must be called once to set the function block to state *PendingRead*. Then the method *UpdateState()* must be called until the function block is in state *Read*. In this state the function block can run through one or more cycles. If the function block is not to compare any further cycles, it can be returned to the *Idle* state. To initiate the state switch, the method *SwitchState_Idle()* must be called. Then the method *UpdateState()* must be called until the function block is in state *Idle*.

State diagram for the data read operation:



Sample

```
VAR
    fbCorrFctRef : FB_ALY_CorrelationFunctionReference(
        nNumChannels := 1,
        nMinLag := 0,
        nMaxLag := 10,
        nStepSize := 1,
        eWindowMode := E_ALY_WindowMode.FixWindow,
        eCorrelationMode := E_ALY_CorrelationMode.Normed,
        nWindowSize := 120,
        nSegmentSize := 400,
        tTimeout := TIME#5S,
        sFilePath := 'C:\TwinCAT\3.1\Boot\UnitTest_CorrelationFunction.tas');

    // State
    eState : E_ALY_ReadState := E_ALY_ReadState.Idle;
    bRead : BOOL;

    // Inputs
    fInputCh1 : LREAL;

    // Results
    fMinCoef : LREAL;
    fMaxCoef : LREAL;
    nMinimizingLag : DINT;
    nMaximizingLag : DINT;
    aCoefficients : ARRAY[-10..10] OF LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbCorrFctRef.Configure(eCorrelationMode := eCorrelationMode, sFilePath := sFilePath );
END_IF

// Call algorithm
eState := fbCorrFctRef.eState;
CASE eState OF
E_ALY_ReadState.Idle:
    IF bRead THEN
        fbCorrFctRef.SwitchState_Read();
        fbCorrFctRef.UpdateState();
    END_IF
E_ALY_ReadState.Read:
    fbCorrFctRef.SetChannelValue(1, fInputCh1);
    fbCorrFctRef.Call();
    fbCorrFctRef.GetChannelOutputArray(1, ADR(aCoefficients), SIZEOF(aCoefficients));
```

```
    fbCorrFctRef.GetChannelOutputValues(1, fMinCoef, fMaxCoef, nMinimizingLag, nMaximizingLag);
    IF NOT bRead THEN
        fbCorrFctRef.SwitchState_Idle();
        fbCorrFctRef.UpdateState();
    END_IF
E_ALY_ReadState.Pending,
E_ALY_ReadState.PendingIdle,
E_ALY_ReadState.PendingRead:
    fbCorrFctRef.UpdateState();
    eState := fbCorrFctRef.eState;
END_CASE
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.3.1 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (0 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.3.2 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 2;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used.

5.1.1.6.3.3 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.3.4 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nMinLag : DINT;
    nMaxLag : DINT;
    nStepSize : UDINT;
    nNumChannels : UDINT;
    eWindowMode : E_ALY_WindowMode;
    eCorrelationMode : E_ALY_CorrelationMode;
    nWindowSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nMinLag	UDINT	Specifies the minimum number by which the two signals are shifted relative to each other for the calculation of the correlation. This is a negative integer.
nMaxLag	UDINT	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.
nStepSize	UDINT	Specifies by how many cycles the signals are shifted to calculate two consecutive correlation coefficients.
nNumChannels	UDINT	The number of channels that are correlated with the reference signal.
eWindowMode	E_ALY_WindowMode [▶ 351]	Specifies the type of window used to calculate the coefficients: *Continuous*: All values since the start of the analysis are included in the analysis with equal weighting. *SlidingWindow:* The calculation is done via a window of the size Window Size. The current values are always included in the analysis and outputs are updated with each cycle. *FixWindow:* The outputs are updated every Window Size cycles and calculated via a window with the length Window Size.

Name	Type	Description
eCorrelationMode	E_ALY_CorrelationMode [▶ 350]	The coefficients are calculated based on one of the following definitions: *Base:* $$C_{xy}[m,t_n]=\begin{cases}\sum_{i=0}^{N-1} x_{n-i-m}\cdot y_{n-i}, & m\geq 0\\ \sum_{i=0}^{N-1} x_{n-i}\cdot y_{n-i+m}, & m<0\end{cases}$$ *Normed:* $$\hat{C}_{xy}[m,t_n]=\begin{cases}\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i-m}\cdot y_{n-i}, & m\geq 0\\ \frac{1}{N}\sum_{i=0}^{N-1} x_{n-i}\cdot y_{n-i+m}, & m<0\end{cases}$$ *Covariance:* $$cov_{xy}[m,t_n]=\begin{cases}\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i-m}\cdot y_{n-i}-\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i-m}\cdot\frac{1}{N}\sum_{i=0}^{N-1} y_{n-i}, & m\geq 0\\ \frac{1}{N}\sum_{i=0}^{N-1} x_{n-i}\cdot y_{n-i+m}-\frac{1}{N}\sum_{i=0}^{N-1} x_{n-i}\cdot\frac{1}{N}\sum_{i=0}^{N-1} y_{n-i+m}, & m<0\end{cases}$$ *CovarianceBessel:* $$\widetilde{cov}_{xy}[m,t_n]=\frac{N}{N-1}cov_{xy}[m,t_n]$$ *Pearson:* $$\rho_{xy}[m,t_n]=\frac{cov(x,y)[m,t_n]}{\sigma_x[m,t_n]\sigma_y[m,t_n]},\ where\ \sigma_x^2[m,t_n]=cov_{xx}[m,t_n]$$
nWindowSize	UDINT	For the *SlidingWindow* and *FixWindow* window modes, specifies the number of cycles over which the coefficients are calculated. For the Window Mode *Continuous* the setting of nWindowSize has no effect. 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 must be taken into account when setting these parameters.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.3.5 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.3.6 GetBusyState

Returns TRUE if the function block is busy due to an asynchronous file access.

Syntax

Definition:

```
METHOD GetBusyState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
GetBusyState	BOOL	Returns `TRUE` if the FB is active.

5.1.1.6.3.7 GetChannelOutputArray

Retrieves a channel-specific output array. The output values were updated in the previously called Call() method.

Syntax

Definition:

```
METHOD GetChannelOutputArray : BOOL
VAR_INPUT
    nChannel : UDINT;
    pCorrCoefsArrayOut : PVOID;
    nCorrCoefsArrayOutSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
pCorrCoefsArrayOut	PVOID	Pointer to an array to which the coefficients are to be written.
nCorrCoefsArrayOutSize	UDINT	Array size

Return value

Name	Type	Description
GetChannelOutputArray	BOOL	Returns `TRUE` if successful.

5.1.1.6.3.8 GetChannelOutputValues

Retrieves channel-specific output values. The output values were updated in the previously called Call() method.

Syntax

Definition:

```
METHOD GetChannelOutputValues : BOOL
VAR_INPUT
    nChannel : UDINT;
    pCorrCoefsArrayOut : PVOID;
    nCorrCoefsArrayOutSize : UDINT;
END_VAR
VAR_IN_OUT
    fMinCoef : LREAL;
    fMaxCoef : LREAL;
    nMinimizingLag : DINT;
    nMaximizingLag : DINT;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
fMinCoef	LREAL	Writing to variable takes place with the minimum coefficient of the output array.
fMaxCoef	LREAL	Writing to variable takes place with the maximum coefficient of the output array.
nMinimizingLag	DINT	Writing to the variable takes place with the lag at which the coefficient is smallest.
nMaximizingLag	DINT	Writing to the variable takes place with the lag at which the coefficient is greatest.

Return value

Name	Type	Description
GetChannelOutputValues	BOOL	Returns `TRUE` if successful.

5.1.1.6.3.9 SwitchState_Idle

Initiate switch from *Read* state to *Idle* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Idle : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Idle	BOOL	Returns `TRUE` if successful

5.1.1.6.3.10 SwitchState_Read

Initiate switch from *Idle* state to *Read* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Read : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Read	BOOL	Returns `TRUE` if successful

5.1.1.6.3.11 UpdateState

Update state after state change was initiated and until target state is not reached.

Syntax

Definition:

```
METHOD UpdateState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
UpdateState	BOOL	Returns `TRUE` if successful.

5.1.1.6.4 FB_ALY_LinearRegressionFitting

The Linear Regression Fitting function block approximates one variable (the Dependent input) by linear combination of several other variables (Input 01 ... Input 0n). This is done by the incremental stochastic gradient method. At the end of the analysis, the calculated coefficients are written to a file.

The linear combination is given by the following equation:

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i \times \text{Input 0i}$$

In each cycle, the values for β_0 to β_n are recalculated using the following rule:

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

This corresponds to the minimization of the squared deviation of the calculated values y (output by the function block as result) from the corresponding input value Dependent. The parameter γ corresponds to the step size and specifies how strongly the parameters are adjusted. The larger the value, the faster the coefficients approach a local optimum. However, if the value is too large, the algorithm may not converge.

Typically, the Linear Regression Fitting function block is first used to fit the weights for the regression of a target variable. Then, using the Linear Regression Inference function block and the fitted weights, the target variable can be predicted based on the input variables.

Syntax

Definition:

```
FUNCTION_BLOCK FB_LinearRegressionFitting
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fResult: LREAL;
    fMSE: LREAL;
    bBusy: BOOL;
    eState: E_TeachState;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fResult	LREAL	Outputs the approximated value for the inputs of the current cycle with the coefficients updated from them.
fMSE	LREAL	Specifies the MSE (mean squared error) between the calculated Result value and the Dependent input value.
bBusy	BOOL	TRUE if the FB is active due to a file access.
eState	E_ALY_TeachState [▶ 350]	Current state of the function block. See state diagram.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
FB_init()	Local	Initializes the number of input channels.
GetBusyState()	Local	This method returns the Busy state of the function block.
GetChannelOutputValue()	Local	Method for receiving values from different output channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SwitchState_Idle()	Local	Method for setting the function block to Idle state when the teach-in process is complete.
SwitchState_Teach()	Local	Method for putting the function block into teach-in mode.
UpdateState()	Local	Method to be used for the transition from one state to another state.

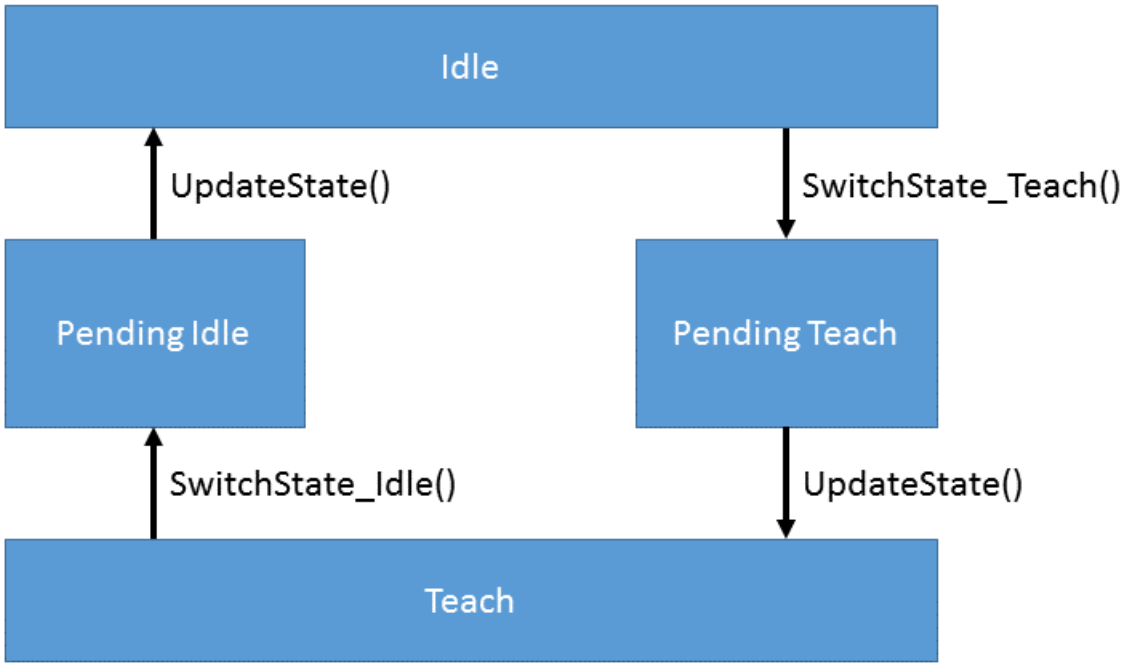
State diagram

Due to the asynchronous file access in real time applications this function block needs a state machine to prepare and finish file access.

At startup the function block is in *Idle* state. To write the incoming data to a file it has to be switched to state *Teach*. Therefore the method *SwitchState_Teach()* has to be called once to set the function block to the *PendingTeach* state. After that the method *UpdateState()* has to be called until the function block is in state

Teach. In this state one or multiple teach cycles can be proceeded. If the function block should not teach additional cycles it can be set to *Idle* state again. To initiate the state switch the method *SwitchState_Idle()* has to be called. After that the method *UpdateState()* has to be called until the function block is in state *Idle*.

State diagram for the teach procedure of the data:



Sample

```
VAR
    fbLinearRegressionFitting : FB_ALY_LinearRegressionFitting(nNumChannels := 1);
    bBias : BOOL := TRUE;
    fStepSize : LREAL := 0.01;
    nMiniBatchSize : UDINT := 1;
    tTimeout : TIME := T#5S;
    bInvolveExistingFile : BOOL := TRUE;
    sFilePath : STRING := 'C:\TwinCAT\3.1\Boot\LinearRegressionFitting.tas';
    bConfigure : BOOL := TRUE;
    eState : E_ALY_TeachState := E_ALY_TeachState.Idle;
    bTeach : BOOL;
    bFit : BOOL := TRUE;
    fInputX : LREAL;
    fInputY : LREAL;
    fOut0 : LREAL;
    fOut1 : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbLinearRegressionFitting.Configure(bBias, fStepSize, nMiniBatchSize, tTimeout, bInvolveExisting
File, sFilePath);
END_IF

// Call algorithm
eState := fbLinearRegressionFitting.eState;
CASE eState OF
E_ALY_TeachState.Idle:
    IF bTeach THEN
        fbLinearRegressionFitting.SwitchState_Teach();
        fbLinearRegressionFitting.UpdateState();
    END_IF
E_ALY_TeachState.Teach:
    fbLinearRegressionFitting.SetChannelValue(1, fInputX);
    fbLinearRegressionFitting.Call(bFit := bFit, dependent := fInputY);
    fbLinearRegressionFitting.GetChannelOutputValue(0, fOut0);
    fbLinearRegressionFitting.GetChannelOutputValue(1, fOut1);
    IF NOT bTeach THEN
```

```
        fbLinearRegressionFitting.SwitchState_Idle();
        fbLinearRegressionFitting.UpdateState();
    END_IF
E_ALY_TeachState.Pending,
E_ALY_TeachState.PendingIdle,
E_ALY_TeachState.PendingTeach:
    fbLinearRegressionFitting.UpdateState();
    eState := fbLinearRegressionFitting.eState;
END_CASE
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.4.1 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 1;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used

5.1.1.6.4.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    bFit : BOOL;
    dependent : ANY;
END_VAR
```

Inputs

Name	Type	Description
bFit	BOOL	Parameter adjustment can be enabled or disabled.
dependent	ANY	Dependent value resulting from the linear combination of the input values.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.4.3 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    bBias : BOOL;
    fStepSize : LREAL;
    nMiniBatchSize : UDINT
    tTimeout : TIME;
    nSetNumTeaches : UDINT;
    bInvolveExistingFile : BOOL;
    sFilePath : STRING(255);
END_VAR
```

Inputs

Name	Type	Description
bBias	BOOL	If FALSE, the Bias Output 00 is set to zero and is not approximated further.
fStepSize	LREAL	Specifies how much the coefficients are adjusted after each new calculation.
nMiniBatchsize	UDINT	Specifies over how many cycles the MSE is to be calculated before the coefficients are adjusted based on it.
tTimeout	TIME	Timeout for asynchronous operations.
bInvolveExistingFile	BOOL	If TRUE, include existing file (if any). If FALSE, create new file.
sFilePath	STRING(255)	Path to the taught file, e.g. *C:\TwinCAT\3.1\Boot\Teach.tas*

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.4.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.4.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.4.6 GetChannelOutputValue

Fetching a channel-specific output value. The output value is only updated if the Call() method was called previously.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (0 (bias) to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.4.7 SwitchState_Idle

Initiate switch from *Teach* state to *Idle* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Idle : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Idle	BOOL	Returns `TRUE` if successful

5.1.1.6.4.8 SwitchState_Teach

Initiate switch from *Idle* state to *Teach* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Teach : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Teach	BOOL	Returns `TRUE` if successful

5.1.1.6.4.9 GetBusyState

Returns TRUE if the function block is busy due to an asynchronous file access.

Syntax

Definition:

```
METHOD GetBusyState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
GetBusyState	BOOL	Returns `TRUE` if the FB is active.

5.1.1.6.4.10 UpdateState

Update state after state change was initiated and until target state is not reached.

Syntax

Definition:

```
METHOD UpdateState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
UpdateState	BOOL	Returns `TRUE` if successful

5.1.1.6.5 FB_ALY_LinearRegressionInference

The Linear Regression Inference function block calculates the linear combination of the inputs (Input 01 .. Input 0n) with the coefficients (Weights 00.. Weights 0n).

$$\text{Result} = \text{Weights 00} + \sum_{i=1}^{n} \text{Weights 0i} \times \text{Input 0i}$$

The parameters Weights 00 to Weights 0n can either be set manually or automatically via a file generated by the Linear Regression Fitting function block by dragging / dropping onto the parameter field. Typically, the Linear Regression Fitting function block is first used to fit the weights for the regression of a target variable. Then, using the Linear Regression Inference function block and the fitted weights, the target variable can be predicted based on the input variables.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_LinearRegressionInference
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bBusy: BOOL;
    eState: E_ALY_ConfigState;
    stFileHeader: ST_ALY_FileHeader;
    fResult : LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bBusy	BOOL	TRUE if the FB is active due to a file access.
eState	E_ALY_ConfigState [▶ 351]	Configuration status of the function block.
stFileHeader	ST_ALY_FileHeader	Header information of the file that was read.
fResult	LREAL	Specifies the value calculated from the linear combination.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm after the individual coefficients have been set.
ConfigureChannel()	Local	Configuration of the coefficients.
ConfigureFromFile()	Local	Configuration of the algorithm via a previously created file.
FB_init	Local	Initializes the number of input channels.
GetBusyState()	Local	This method returns the Busy state of the function block.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
UpdateState()	Local	Updating the state after a state change has been initiated and while the target state has not been reached.

Example – Configuration by parameters

```
VAR
    fbLinearRegressionInference : FB_ALY_LinearRegressionInference(nNumChannels := 1);
    fCoefficient0 : LREAL := 10.0;
    fCoefficient1 : LREAL := 1.0;
    bConfigure : BOOL := TRUE;
    fInputX : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbLinearRegressionInference.ConfigureChannel(0, fCoefficient0);
    fbLinearRegressionInference.ConfigureChannel(1, fCoefficient1);
    fbLinearRegressionInference.Configure();
END_IF

// Call algorithm
fbLinearRegressionInference.SetChannelValue(1, fInputX);
fbLinearRegressionInference.Call();
END_CASE
```

Sample – Configuration by file

```
VAR
    fbLinearRegressionInference : FB_ALY_LinearRegressionInference(nNumChannels := 1);
    tTimeout : TIME := T#5S;
    sFilePath : STRING := 'C:\TwinCAT\3.1\Boot\LinearRegressionFitting.tas';
    bConfigure : BOOL := TRUE;
    fInputX : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
    fbLinearRegressionInference.ConfigureFromFile(tTimeout, sFilePath);
END_IF

// Update pending state
IF fbLinearRegressionInference.eState = E_ALY_ConfigState.Pending THEN
    fbLinearRegressionInference.UpdateState();
END_IF

// Call algorithm
IF fbLinearRegressionInference.eState = E_ALY_ConfigState.Configured THEN
    fbLinearRegressionInference.SetChannelValue(1, fInputX);
    fbLinearRegressionInference.Call();
END_IF
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.5.1 FB_init

Initialize the number of input channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 1;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used

5.1.1.6.5.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.5.3 Configure

Configuring the algorithm. The coefficients must be set beforehand with the ConfigureChannel() method.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.5.4 ConfigureChannel

Configuration of the coefficients for the calculation of the linear combination. The coefficients are only applied after the Configure() method has been called.

Syntax

Definition:

```
METHOD ConfigureChannel : BOOL
VAR_INPUT
    nChannel : UDINT;
    fCoefficient : LREAL;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index for the coefficients. Starting at zero.
fCoefficient	LREAL	Coefficient for the calculation of the linear combination.

Return value

Name	Type	Description
ConfigureChannel	BOOL	Returns `TRUE` if successful.

5.1.1.6.5.5 ConfigureFromFile

Configures the algorithm based on a previously created file. After calling this method, the method UpdateState() must be called until the status of the function block is no longer Pending.

Syntax

Definition:

```
METHOD ConfigureFromFile : BOOL
VAR_INPUT
    tTimeout : TIME;
    sFilePath : STRING(255);
END_VAR
```

Inputs

Name	Type	Description
tTimeout	TIME	Timeout for asynchronous operations.
sFilePath	STRING(255)	Path to the taught file, e.g. *C:\TwinCAT\3.1\Boot\Teach.tas*

Return value

Name	Type	Description
ConfigureFromFile	BOOL	Returns `TRUE` if successful.

5.1.1.6.5.6 UpdateState

Updates the state after a state change has been initiated by calling the ConfigureFromFile() method. The method must be called cyclically until the state of the function block is no longer Pending.

Syntax

Definition:

```
METHOD UpdateState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
UpdateState	BOOL	Returns `TRUE` if successful

5.1.1.6.5.7 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.5.8 GetBusyState

Returns TRUE if the function block is busy due to an asynchronous file access.

Syntax

Definition:

```
METHOD GetBusyState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
GetBusyState	BOOL	Returns `TRUE` if the FB is active.

5.1.1.6.5.9 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.6 FB_ALY_StandardDeviation

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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StandardDeviation
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions. The configuration is identical for all channels.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
GetChannelOutputValue()	Local	Method for receiving values from different output channels.

Sample

```
VAR
    fbStandardDeviation : FB_ALY_StandardDeviation(nNumChannels := 1);
    bUseBesselCorrection : BOOL :=  TRUE;
    eWindowMode : E_ALY_WindowMode := E_ALY_WindowMode.FixWindow;
```

```
    nWindowSize : UDINT := 100;
    bConfigure : BOOL := TRUE;
    fInput : LREAL;
    fStandardDeviation : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
    fbStandardDeviation.Configure(bUseBesselCorrection, eWindowMode, nWindowSize);
END_IF

// Call algorithm
fbStandardDeviation.SetChannelValue(1, fInput);
fbStandardDeviation.Call();
fbStandardDeviation.GetChannelOutputValue(1, fStandardDeviation);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.6.1 FB_init

Initializes the number of independent input and output channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 1;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used

5.1.1.6.6.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.6.3 Configure

Configure the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    bUseBesselCorrection : BOOL;
    eWindowMode : E_ALY_WindowMode;
    nWindowSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
bUseBesselCorrection	BOOL	If the parameter bUseBesselCorrection is set to *TRUE*, 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 $s = \sqrt{\frac{1}{N-1}\sum_{n=0}^{N-1}(x[n]-\overline{x})^2}$ The empirical standard deviation, with Bessel's correction $s' = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x[n]-\overline{x})^2}$
eWindowMode	E_ALY_WindowMode [▶ 351]	Used window mode. 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 done every *N* calls. • *Sliding Window*: Only the last *N* input values are included in the calculation. The calculation is performed cyclically.
nWindowSize	UDINT	Depending on the window mode used, you can configure the number of values *N* that will be included in the calculation. In the window mode *Continuous* this parameter is ignored.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.6.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns TRUE if the reset was successful.

5.1.1.6.6.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns TRUE if successful.

5.1.1.6.6.6 GetChannelOutputValue

Fetching a channel-specific output value. The output value is only updated if the Call() method was called previously.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns TRUE if successful.

5.1.1.6.7 FB_ALY_ClearanceFactor

The algorithm *Clearance Factor* calculates the signal feature of the same name from the input values. The output value is calculated from the ratio of the peak value of the input signal to the squared mean of the square roots of the absolute input signal.

$$\text{Clearance Factor} = \frac{\max(|x|)}{\left(\frac{1}{N}\sum_{n=1}^{N}\sqrt{|x[n]|}\right)^2}$$

The number of input data to be included in the calculation and the type of calculation can be configured.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ClearanceFactor
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions. The configuration is identical for all channels.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
GetChannelOutputValue()	Local	Method for receiving values from different output channels.

Sample

```
VAR
    fbClearanceFactor : FB_ALY_ClearanceFactor(nNumChannels := 1);
    eWindowMode : E_ALY_WindowMode := E_ALY_WindowMode.FixWindow;
    nWindowSize : UDINT := 100;
    bConfigure : BOOL := TRUE;
```

```
    fInput : LREAL;
    fClearanceFactor : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
    fbClearanceFactor.Configure(eWindowMode, nWindowSize);
END_IF

// Call algorithm
fbClearanceFactor.SetChannelValue(1, fInput);
fbClearanceFactor.Call();
fbClearanceFactor.GetChannelOutputValue(1, fClearanceFactor);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.7.1 FB_init

Initializes the number of independent input and output channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 1;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used

5.1.1.6.7.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.7.3 Configure

Configure the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eWindowMode : E_ALY_WindowMode;
    nWindowSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
eWindowMode	E_ALY_WindowMode [▶ 351]	Used window mode. 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 done every *N* calls. • *Sliding Window*: Only the last *N* input values are included in the calculation. The calculation is performed cyclically.
nWindowSize	UDINT	Depending on the window mode used, you can configure the number of values *N* that will be included in the calculation. In the window mode *Continuous* this parameter is ignored.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.7.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.7.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.7.6 GetChannelOutputValue

Fetching a channel-specific output value. The output value is only updated if the Call() method was called previously.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.8 FB_ALY_ImpulseFactor

The algorithm *Impulse Factor* calculates the signal feature of the same name from the input values. The output value is calculated from the ratio of the peak value of the input signal to the mean value of the input signal.

$$\text{Impulse Factor} = \frac{\max(|x|)}{\frac{1}{N}\sum_{n=1}^{N}|x[n]|}$$

The number of input data to be included in the calculation and the type of calculation can be configured.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ImpulseFactor
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
```

```
    bNewResult: BOOL;
    bConfigured: BOOL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions. The configuration is identical for all channels.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
GetChannelOutputValue()	Local	Method for receiving values from different output channels.

Sample

```
VAR
    fbImpulseFactor : FB_ALY_ImpulseFactor(nNumChannels := 1);
    eWindowMode : E_ALY_WindowMode := E_ALY_WindowMode.FixWindow;
    nWindowSize : UDINT := 100;
    bConfigure : BOOL := TRUE;
    fInput : LREAL;
    fImpulseFactor : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
    fbImpulseFactor.Configure(eWindowMode, nWindowSize);
END_IF

// Call algorithm
fbImpulseFactor.SetChannelValue(1, fInput);
fbImpulseFactor.Call();
fbImpulseFactor.GetChannelOutputValue(1, fImpulseFactor);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.8.1 FB_init

Initializes the number of independent input and output channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 1;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used

5.1.1.6.8.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.8.3 Configure

Configure the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eWindowMode : E_ALY_WindowMode;
    nWindowSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
eWindowMode	E_ALY_WindowMode [▶ 351]	Used window mode. 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 done every *N* calls. • *Sliding Window*: Only the last *N* input values are included in the calculation. The calculation is performed cyclically.
nWindowSize	UDINT	Depending on the window mode used, you can configure the number of values *N* that will be included in the calculation. In the window mode *Continuous* this parameter is ignored.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.8.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.8.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.8.6 GetChannelOutputValue

Fetching a channel-specific output value. The output value is only updated if the Call() method was called previously.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.9 FB_ALY_ShapeFactor

The algorithm *Shape Factor* calculates the signal feature of the same name from the input values. The output value is calculated from the ratio of the root mean square (RMS) of the input signal to the mean of the absolute values of the input signal.

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

The number of input data to be included in the calculation and the type of calculation can be configured.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_ShapeFactor
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions. The configuration is identical for all channels.
FB_init()	Local	Initializes the number of input channels.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
GetChannelOutputValue()	Local	Method for receiving values from different output channels.

Sample

```
VAR
    fbShapeFactor : FB_ALY_ShapeFactor(nNumChannels := 1);
    eWindowMode : E_ALY_WindowMode := E_ALY_WindowMode.FixWindow;
    nWindowSize : UDINT := 100;
    bConfigure : BOOL := TRUE;
    fInput : LREAL;
    fShapeFactor : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
    fbShapeFactor.Configure(eWindowMode, nWindowSize);
END_IF

// Call algorithm
fbShapeFactor.SetChannelValue(1, fInput);
fbShapeFactor.Call();
fbShapeFactor.GetChannelOutputValue(1, fShapeFactor);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.6.9.1 FB_init

Initializes the number of independent input and output channels.

Syntax

Definition:

```
METHOD FB_init : BOOL
VAR_INPUT
    nNumChannels : UDINT := 1;
END_VAR
```

Inputs

Name	Type	Description
nNumChannels	UDINT	Initializes the number of input channels.

Return value

Name	Type	Description
FB_init	BOOL	Not used

5.1.1.6.9.2 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.6.9.3 Configure

Configure the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eWindowMode : E_ALY_WindowMode;
    nWindowSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
eWindowMode	E_ALY_WindowMode [▶ 351]	Used window mode. 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 done every *N* calls. • *Sliding Window*: Only the last *N* input values are included in the calculation. The calculation is performed cyclically.
nWindowSize	UDINT	Depending on the window mode used, you can configure the number of values *N* that will be included in the calculation. In the window mode *Continuous* this parameter is ignored.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.6.9.4 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.6.9.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.6.9.6 GetChannelOutputValue

Fetching a channel-specific output value. The output value is only updated if the Call() method was called previously.

Syntax

Definition:

```
METHOD GetChannelOutputValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    output : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nNumChannels)
output	ANY	Output value of any data type.

Return value

Name	Type	Description
GetChannelOutputValue	BOOL	Returns `TRUE` if successful.

5.1.1.7 Traning Base

5.1.1.7.1 FB_ALY_TimeBasedTeachPath_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 algorithm, for example.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TimeBasedTeachPath_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bBusy: BOOL;
    bTeaching: BOOL;
    eState: E_TeachState;
    nWrittenValues: ULINT;
    nValuesInFile: ULINT;
    nCurrentTeachCycles: ULINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bBusy	BOOL	TRUE if the FB is active due to a file access.
bTeaching	BOOL	TRUE, if the algorithm processes the teach-in, otherwise FALSE. The teach-in process starts when the Start Period flag switches to TRUE and stops when the Stop Period flag switches to TRUE.
eState	E_ALY_TeachState [▶ 350]	Current state of the function block. See state diagram.
nWrittenValues	ULIINT	Total number of values written during the teach-in process. Not to be confused with the number of values in File, which are overwritten in each teach-in cycle.
nValuesInFile	ULINT	Number of values currently being written to the file.
nCurrentTeachCycles	ULINT	Number of teach-in cycles in the file.

Methods

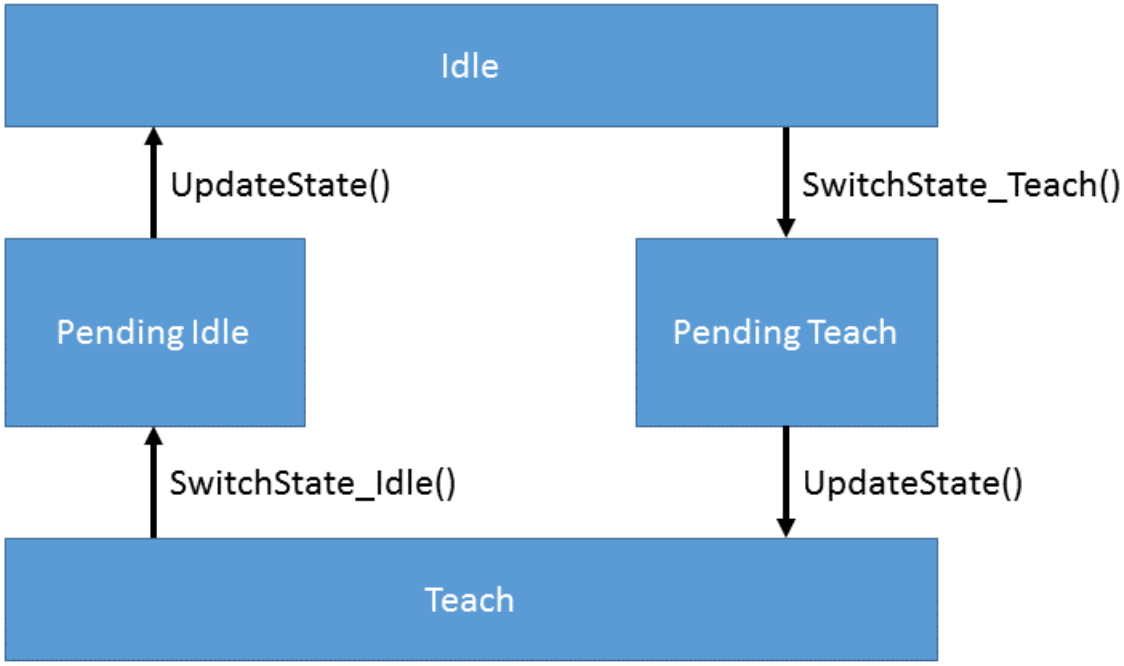
Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
GetBusyState()	Local	This method provide the busy state of the function block.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.
SwitchState_Idle()	Local	Method to set the function block into an idle state if the teach procedure is finished.
SwitchState_Teach()	Local	Method to set the function block into the teach mode.
UpdateState()	Local	Method to use for the transaction from one state to another state.

State diagram

Due to the asynchronous file access in real time applications this function block needs a state machine to prepare and finish file access.

At startup the function block is in *Idle* state. To write the incoming data to a file it has to be switched to state *Teach*. Therefore the method *SwitchState_Teach()* has to be called once to set the function block to the *PendingTeach* state. After that the method *UpdateState()* has to be called until the function block is in state *Teach*. In this state one or multiple teach cycles can be proceeded. If the function block should not teach additional cycles it can be set to *Idle* state again. To initiate the state switch the method *SwitchState_Idle()* has to be called. After that the method *UpdateState()* has to be called until the function block is in state *Idle*.

State diagram for the teach procedure of the data:



Sample

```
VAR
    fbTimeBasedTeach : FB_ALY_TimeBasedTeachPath_1Ch;
    eTeachMode : E_ALY_TeachMode := E_ALY_TeachMode.Mean;
    nSegmentSize : UDINT := 200;
    tTimeout : TIME := T#5S;
    nSetNumTeaches : UDINT := 10;
    bInvolveExistingFile : BOOL := TRUE;
    sFilePath : STRING := 'C:\TwinCAT\3.1\Boot\TimeBasedTeach.tas';
    bNegateStartPeriod : BOOL := FALSE;
    bNegateStopPeriod : BOOL := FALSE;
    bConfigure : BOOL := TRUE;
    eState : E_ALY_TeachState := E_ALY_TeachState.Idle;
    bTeach : BOOL;
    fInput : LREAL;
    bStartPeriod : BOOL;
    bStopPeriod : BOOL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbTimeBasedTeach.Configure(eTeachMode, nSegmentSize, tTimeout, nSetNumTeaches, bInvolveExistingF
ile, sFilePath, bNegateStartPeriod, bNegateStopPeriod);
END_IF

// Call algorithm
eState := fbTimeBasedTeach.eState;
CASE eState OF
E_ALY_TeachState.Idle:
    IF bTeach THEN
        fbTimeBasedTeach.SwitchState_Teach();
        fbTimeBasedTeach.UpdateState();
    END_IF
E_ALY_TeachState.Teach:
    fbTimeBasedTeach.SetChannelValue(fInput);
    fbTimeBasedTeach.Call(bStartPeriod := bStartPeriod, bStopPeriod := bStopPeriod);

    IF NOT bTeach THEN
        fbTimeBasedTeach.SwitchState_Idle();
        fbTimeBasedTeach.UpdateState();
    END_IF
E_ALY_TeachState.Pending,
E_ALY_TeachState.PendingIdle,
E_ALY_TeachState.PendingTeach:
```

```
    fbTimeBasedTeach.UpdateState();
    eState := fbTimeBasedTeach.eState;
END_CASE
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.7.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    bStartPeriod : ULINT;
    bStopPeriod : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bStartPeriod	BOOL	Rising edge starts teach-in process.
bStopPeriod	BOOL	Rising edge stops teach-in process.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.7.1.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eTeachMode : E_ALY_TeachMode;
    nSegmentSize : UDINT;
    tTimeout : TIME;
    nSetNumTeaches : UDINT;
    bInvolveExistingFile : BOOL;
    sFilePath : STRING(255);
    bNegateStartPeriod : BOOL;
    bNegateStopPeriod : BOOL;
END_VAR
```

Inputs

Name	Type	Description
eTeachMode	E_ALY_TeachMode [▶ 350]	Configuring the teach-in mode *Minimum*: Minimum of the stored data point and input data point. *Maximum*: Maximum of the stored data point and input data point. *Mean:* Mean value of the stored data point and input data point in relation to nCurrentTeachCycles.
nSegmentSize	UDINT	Number of buffered elements for file operation.
tTimeout	TIME	Timeout for asynchronous operations.
nSetNumTeaches	UDINT	Teach-in number to be added to the file in Read state. "0" = not limited.
bInvolveExistingFile	BOOL	If TRUE, include existing file (if any). If FALSE, create new file.
sFilePath	STRING(255)	Path to the taught file, e.g. *C:\TwinCAT\3.1\Boot\Teach.tas*
bNegateStartPeriod	BOOL	Negates the input parameter bStartPeriod.
bNegateStopPeriod	BOOL	Negates the input parameter bStopPeriod.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.7.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.7.1.4 GetBusyState

Returns TRUE if the function block is busy due to an asynchronous file access.

Syntax

Definition:

```
METHOD GetBusyState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
GetBusyState	BOOL	Returns `TRUE` if the FB is active.

5.1.1.7.1.5 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.7.1.6 SwitchState_Idle

Initiate switch from *Teach* state to *Idle* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Idle : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Idle	BOOL	Returns `TRUE` if successful

5.1.1.7.1.7 SwitchState_Teach

Initiate switch from *Idle* state to *Teach* state. See State diagram.

Syntax

Definition:

```
METHOD SwitchState_Teach : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
SwitchState_Teach	BOOL	Returns `TRUE` if successful

5.1.1.7.1.8 UpdateState

Update state after state change was initiated and until target state is not reached.

Syntax

Definition:

```
METHOD UpdateState : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
UpdateState	BOOL	Returns `TRUE` if successful

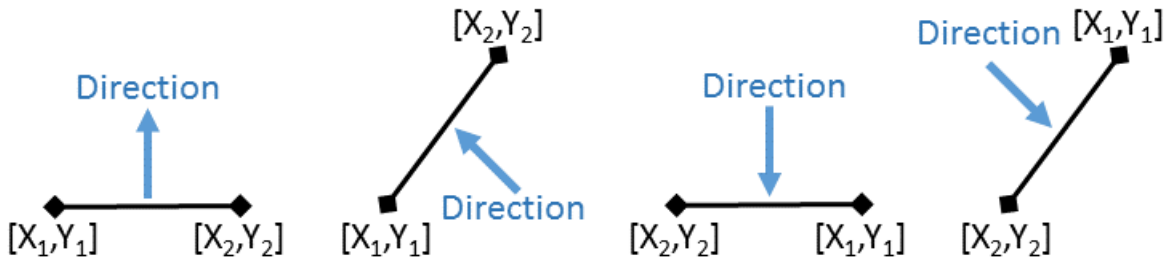
5.1.1.8 XY Path

5.1.1.8.1 FB_ALY_XyGateMonitor_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.

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).



Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_XyGateMonitor_2Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bExecuting: BOOL;
    bGateIntersection: BOOL;
    bOutlierIntersection: BOOL;
    fPosIntersectionX: LREAL;
    fPosIntersectionY: LREAL;
    nCountGateIntersections: ULINT;
    nCountOutlierIntersections: ULINT;
    fbTimeLastIntersection: FB_ALY_DateTime;
    eClassification: E_ALY_Classification_2Cls;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bExecuting	BOOL	TRUE if the Call() method was called with bStart:=TRUE. FALSE if the call method was called with bStop:=TRUE.
bGateIntersection	BOOL	TRUE if a gate intersection has been detected.
bOutlierIntersection	BOOL	TRUE if an outlier intersection has been detected.
fPosIntersectionX	LREAL	X coordinate of the last intersection.
fPosIntersectionY	LREAL	Y coordinate of the last intersection.
nCountGateIntersections	ULINT	Number of intersections detected.
nCountOutlierIntersections	ULINT	Number of outliers detected.
fbTimeLastIntersection	FB_ALY_DateTime	Timestamp of the last intersection.
eClassification	E_ALY_Classification_2Cls	Classification result. Depending on the configured gate mode.

Methods

Name	Definition Location	Description
Call()	Local	method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbXyGateMonitor : FB_ALY_XyGateMonitor_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    eGateMode : E_ALY_GateMode := E_ALY_GateMode.IntersectGate;
    stGatePos1 : ST_ALY_XyPosition := ( X:= 0.0, Y := 0.0);
    stGatePos2 : ST_ALY_XyPosition := ( X:= 0.0, Y := 4.0);
    bConfigure : BOOL := TRUE;
    fInputChX : LREAL;
    fInputChY : LREAL;
    bStart : BOOL;
    bStop : BOOL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbXyGateMonitor.Configure(eGateMode, stGatePos1, stGatePos2);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
```

```
fbXyGateMonitor.SetChannelValue(1, fInputChX);
fbXyGateMonitor.SetChannelValue(2, fInputChY);
fbXyGateMonitor.Call(bStart, bStop, fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.8.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    bStart : BOOL;
    bStop : BOOL;
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
bStart	BOOL	Start gate observe.
bStop	BOOL	Stop gate observe.
tTimestamp	ULINT	Current timestamp.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.8.1.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.8.1.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    eGateMode : E_ALY_GateMode;
    stGatePos1 : ST_ALY_XyPosition;
    stGatePos2 : ST_ALY_XyPosition;
END_VAR
```

Inputs

Name	Type	Description
eGateMode	E_ALY_GateMode [▶ 350]	Configured gate mode *Intersect Gate:* Determines whether the XY signal crosses the gate in the configured direction. If there is an intersection during the analysis period, this is classified as OK, otherwise NOK. *Not Intersect Gate:* Monitors whether the XY signal does not intersect with the gate in the configured direction during the analysis period. Then this is classified as OK, otherwise NOK. *Intersect Projection:* Determines whether the XY signal crosses the projection of the gate in the configured direction. If there is an intersection during the analysis period, this is classified as OK, otherwise NOK. *Not Intersect Projection:* Monitors whether the XY signal does not intersect with the projection of the gate in the configured direction during the analysis period. Then this is classified as OK, otherwise NOK. *Intersect Gate Or Projection:* Determines whether the XY signal intersects the gate or its projection in the configured direction. If there is an intersection during the analysis period, this is classified as OK, otherwise NOK.
stCorner1	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the first corner in XY coordinates.
stCorner2	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the second corner in XY coordinates.
stCorner3	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the third corner in XY coordinates.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.8.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Value of X-coordinate 2: Value of Y-coordinate
Input	ANY	Input value of arbitrary data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns TRUE if successful.

5.1.1.8.2 FB_ALY_XyShapeMonitor_Circle_2Ch

The *XY Shape Monitor Circle 2Ch* count the amount of intersections of an XY input with a specified circle shape.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_XyShapeMonitor_Circle_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bWithinShape: BOOL;
    bIntersection: BOOL;
    nCountIntersections: ULINT;
    fbTimeLastIntersection: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bWithinShape	BOOL	TRUE if the input signal is currently within the specified shape.
bIntersection	BOOL	TRUE if the input signal currently crosses the specified shape.
nCountIntersections	ULINT	Total number of intersections of input signal and shape.
fbTimeLastIntersection	FB_ALY_DateTime	Timestamp of the last intersection.

Methods

Name	Definition Location	Description
Call()	Local	method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbXyShapeMonitor_Circle : FB_ALY_XyShapeMonitor_Circle_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stCentre : ST_ALY_XyPosition := ( X:= 4.0, Y := 2.0);
    fRadius : LREAL := 1.5;
    bConfigure : BOOL := TRUE;
    fInputChX : LREAL;
    fInputChY : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbXyShapeMonitor_Circle.Configure(stCentre, fRadius);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbXyShapeMonitor_Circle.SetChannelValue(1, fInputChX);
fbXyShapeMonitor_Circle.SetChannelValue(2, fInputChY);
fbXyShapeMonitor_Circle.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.8.2.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.8.2.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.8.2.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    stCentre : ST_ALY_XyPosition;
    fRadius : LREAL;
END_VAR
```

Inputs

Name	Type	Description
stCentre	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the circle center in XY coordinates.
fRadius	LREAL	Radius of the circle.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.8.2.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Value of the X coordinate 2: Value of the Y coordinate
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns TRUE if successful.

5.1.1.8.3 FB_ALY_XyShapeMonitor_Rectangle_2Ch

The *XY Shape Monitor Rectangle 2Ch* count the amount of intersections of an XY input with a specified rectangle shape.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_XyShapeMonitor_Rectangle_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bWithinShape: BOOL;
    bIntersection: BOOL;
    nCountIntersections: ULINT;
    fbTimeLastIntersection: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
bWithinShape	BOOL	TRUE if the input signal is currently within the specified shape.
bIntersection	BOOL	TRUE if the input signal currently crosses the specified shape.
nCountIntersections	ULINT	Total number of intersections of input signal and shape.
fbTimeLastIntersection	FB_ALY_DateTime	Timestamp of the last intersection.

Methods

Name	Definition Location	Description
Call()	Local	method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbXyShapeMonitor_Rectangle : FB_ALY_XyShapeMonitor_Rectangle_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stLowerLeftCorner : ST_ALY_XyPosition := ( X:= 0.0, Y := 0.0);
    fLength_X : LREAL := 10;
    fLength_Y : LREAL := 5;
    bConfigure : BOOL := TRUE;
    fInputChX : LREAL;
    fInputChY : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbXyShapeMonitor_Rectangle.Configure(stLowerLeftCorner, fLength_X, fLength_Y );
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbXyShapeMonitor_Rectangle.SetChannelValue(1, fInputChX);
fbXyShapeMonitor_Rectangle.SetChannelValue(2, fInputChY);
fbXyShapeMonitor_Rectangle.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.8.3.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.8.3.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.8.3.3 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    stLowerLeftCorner : ST_ALY_XyPosition;
    fLength_X : LREAL;
    fLength_Y : LREAL;
END_VAR
```

Inputs

Name	Type	Description
stLowerLeftCorner	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the lower left corner of the rectangle in XY coordinates.
fLenght_X	LREAL	Length of the rectangle in positive X direction.
fLenght_Y	LREAL	Length of the rectangle in positive Y direction.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.8.3.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Value of the X coordinate 2: Value of the Y coordinate
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.8.4 FB_ALY_XyShapeMonitor_Triangle_2Ch

The *XY Shape Monitor Triangle 2Ch* counts the amount of intersections of an XY input with a specified triangle shape.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_XyShapeMonitor_Triangle_2Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bWithinShape: BOOL;
    bIntersection: BOOL;
    nCountIntersections: ULINT;
    fbTimeLastIntersection: FB_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is `TRUE`, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
bWithinShape	BOOL	`TRUE` if the input signal is currently within the specified shape.
bIntersection	BOOL	`TRUE` if the input signal currently crosses the specified shape.
nCountIntersections	ULINT	Total number of intersections of input signal and shape.
fbTimeLastIntersection	FB_ALY_DateTime	Timestamp of the last intersection.

Methods

Name	Definition Location	Description
Call()	Local	method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
ConfigureChannel()	Local	Channel-specific configuration for the respective algorithm.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbXyShapeMonitor_Triangle : FB_ALY_XyShapeMonitor_Triangle_2Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    stCorner1 : ST_ALY_XyPosition := ( X:= 0.0, Y := 0.0);
    stCorner2 : ST_ALY_XyPosition := ( X:= 0.0, Y := 6.0);
    stCorner3 : ST_ALY_XyPosition := ( X:= 4.0, Y := 3.0);
    bConfigure : BOOL := TRUE;
    fInputChX : LREAL;
    fInputChY : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbXyShapeMonitor_Triangle.Configure(stCorner1, stCorner2, stCorner3);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbXyShapeMonitor_Triangle.SetChannelValue(1, fInputChX);
fbXyShapeMonitor_Triangle.SetChannelValue(2, fInputChY);
fbXyShapeMonitor_Triangle.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.8.4.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.8.4.2 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.8.4.3 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index 1: Value of the X coordinate 2: Value of the Y coordinate
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.8.4.4 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    stCorner1 : ST_ALY_XyPosition;
    stCorner2 : ST_ALY_XyPosition;
    stCorner3 : ST_ALY_XyPosition;
END_VAR
```

Inputs

Name	Type	Description
stCorner1	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the first corner in XY coordinates.
stCorner2	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the second corner in XY coordinates.
stCorner3	ST_ALY_XyPosition [▶ 346]	Structure for saving the position of the third corner in XY coordinates.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.9 Specific

5.1.1.9.1 XTS

5.1.1.9.1.1 FB_ALY_XtsAccelerationAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_XtsAccelerationAnalysis_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fAcceleration: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fAcceleration	LREAL	Current acceleration of the XTS mover in m/s².

Methods

Name	Definition Location	Description
Call()	Local	method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbXtsAcceleration : FB_ALY_XtsAccelerationAnalysis_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nXtsLenght : UDINT := 4000;
    bConfigure : BOOL := TRUE;
    fPosition : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

   fbXtsAcceleration.Configure(nXtsLenght);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbXtsAcceleration.SetChannelValue(fPosition);
fbXtsAcceleration.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.9.1.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.1.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nXtsLength : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nXtsLength	UDINT	Length of the given XTS system in millimeters.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.9.1.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.2 FB_ALY_XtsDistanceIntegrator_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_XtsDistanceIntegrator_1Ch
VAR_INPUT
    bPersistent: BOOL;
END_VAR
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fDistance: LREAL;
    fDistancePos: LREAL;
    fDistanceNeg: LREAL;
END_VAR
```

Inputs

Name	Type	Description
bPersistent	BOOL	If the value is TRUE, the internal data is stored persistently.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fDistance	LREAL	Total distance covered by the XTS mover. In m.
fDistancePos	LREAL	Positive distance covered by the XTS mover (direction: forward). In m.
fDistanceNeg	LREAL	Negative distance covered by the XTS mover (direction: backward). In m.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbXtsDistance : FB_ALY_XtsDistanceIntegrator_1Ch;
    nXtsLenght : UDINT := 4000;
```

```
    bConfigure : BOOL := TRUE;
    fPosition : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

   fbXtsDistance.Configure(nXtsLenght);
END_IF

// Call algorithm
fbXtsDistance.SetChannelValue(fPosition);
fbXtsDistance.Call();
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.9.1.2.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.2.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nXtsLength : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nXtsLength	UDINT	Length of the given XTS system in millimeters.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.2.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.9.1.2.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.3 FB_ALY_XtsVelocityAnalysis_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_XtsVelocityAnalysis_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fVelocity: LRAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fVelocity	LREAL	Current velocity of the XTS mover in m/s.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbXtsVelocity : FB_ALY_XtsVelocityAnalysis_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nXtsLenght : UDINT := 4000;
    bConfigure : BOOL := TRUE;
    fPosition : LREAL;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

   fbXtsVelocity.Configure(nXtsLenght);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbXtsVelocity.SetChannelValue(fPosition);
fbXtsVelocity.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.9.1.3.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.3.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nXtsLength : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nXtsLength	UDINT	Length of the given XTS system in millimeters.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.9.1.3.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.9.1.3.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.9.2 Wind Turbine

5.1.1.9.2.1 FB_ALY_WtTurbulence_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.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_WtTurbulence_1Ch
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fMean: LREAL;
    fTurbulence: LREAL;
    fTurbulenceIntensity: LREAL;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is `TRUE` if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is `TRUE`.
bConfigured	BOOL	Displays `TRUE` when the function block is successfully configured.
fMean	LREAL	Mean value of the wind speed.
fTurbulence	LREAL	Turbulence of the wind. According to the EN standard, this is the standard deviation of the wind speed during a time interval of 10 minutes.
fTurbulenceIntensity	LREAL	Intensity of wind turbulence.

Methods

Name	Definition Location	Description
Call()	Local	Method calculates the outputs for a given configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations made so far.
SetChannelValue()	Local	Method to pass values to the algorithm.

Sample

```
VAR
    fbWtTurbulence : FB_ALY_WtTurbulence_1Ch;
    fbSystemTime : FB_ALY_GetSystemTime;
    nNumCycles : UDINT;
    bConfigure : BOOL := TRUE;
    nInput : INT;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    fbWtTurbulence.Configure(nNumCycles);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbWtTurbulence.SetChannelValue(nInput);
fbWtTurbulence.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.9.2.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.9.2.1.2 Configure

Configure the algorithm. Channel specific parameter are set using the method ConfigueChannel().

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    nNumCycles : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nNumCycles	UDINT	Indicates the number of PLC cycles that fit into the time interval for the calculation, according to EN 61400-1 this is an interval of ten minutes.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.9.2.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.9.2.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.10 Reporting

24/7 reporting can be implemented in TwinCAT Analytics using the algorithms in the *Reporting* category. The reporting collectors collect the data and send it to the reporting server. The reporting triggers trigger the creation of a report.

5.1.1.10.1 Reporting Collector

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

5.1.1.10.1.1 Reporting Collector Edge

The Reporting Collector Edge collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the signal of the input channel passes the configured edge at a certain threshold.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Rpt_CollectorEdge
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fbTimeLastEvent: FB_ALY_DateTime;
    nBufferCount: UDINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fbTimeLastEvent	FB_ALY_DateTime	Stores the timestamp of the last message sent to the reporting server.
nBufferCount	UDINT	Indicates the number of elements in the buffer.

Methods

Name	Type	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetDisplayName()	Local	Method for describing the input channel.
SetEdgeValue()	Local	Method for passing values to the edge channel of the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbRpt_CollectorEdge: FB_Rpt_CollectorEdge(sNetID:='', nNumChannels:=3);
    fbSystemTime : FB_ALY_GetSystemTime;
    bEdge: BOOL;
    fDataInCh1: LREAL;
    fDataInCh2: LREAL;
    fDataInCh3: LREAL;

    stThresholdEdge: ST_ALY_Threshold;
    sReportName: STRING(255);
    fTolerance: LREAL;
    sDataKey: STRING(255);
    nBufferSize: UDINT;
    bIncludeTimestamps: BOOL;
    aChannelNames: ARRAY [1..3] OF STRING(255);
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    sReportName:= 'Beckhoff Report Template';
    fTolerance:= 0;
    sDataKey:= 'DataKey101';
    nBufferSize:= 1;
    bIncludeTimestamps:= TRUE;
    aChannelNames[1]:= 'Min';
    aChannelNames[2]:= 'Max';
    aChannelNames[3]:= 'Avg';
    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.fThreshold := 1;

    fbRpt_CollectorEdge.Configure(
        sConfigId:= sReportName,
        sDataKey:= sDataKey,
        nBufferSize:= nBufferSize,
        bBufferOnEvent:= bBufferOnEvent,
        bIncludeTimestamps:= bIncludeTimestamps,
        stThresholdEdge:= stThresholdEdge);

    fbRpt_CollectorEdge.fTolerance:= fTolerance;
    fbRpt_CollectorEdge.SetDisplayName(nChannel:= 1, sDisplayName:= aChannelNames[1]);
    fbRpt_CollectorEdge.SetDisplayName(nChannel:= 2, sDisplayName:= aChannelNames[2]);
    fbRpt_CollectorEdge.SetDisplayName(nChannel:= 3, sDisplayName:= aChannelNames[3]);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbRpt_CollectorEdge.SetChannelValue(nChannel:= 1, Input:= fDataInCh1);
fbRpt_CollectorEdge.SetChannelValue(nChannel:= 2, Input:= fDataInCh2);
fbRpt_CollectorEdge.SetChannelValue(nChannel:= 3, Input:= fDataInCh3);

fbRpt_CollectorEdge.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.10.1.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.1.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    sConfigId : STRING;
    sDataKey : STRING;
    stThresholdEdge : ST_ALY_Threshold;
    bIncludeTimestamps : BOOL;
    nBufferSize : UDINT;
    bBufferOnEvent : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sConfigId	STRING	Indicates the name of the report. The name must correspond to the name of the configuration file at the Reporting Server.
sDataKey	STRING	The data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.
bIncludeTimestamps	BOOL	Inserts a column with the timestamps.
nBufferSize	UDINT	Indicates the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
bBufferOnEvent	BOOL	Indicates how the data is to be collected and buffered. If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer. If the parameter is False, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the TcReportingServer.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.10.1.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.1.5 SetDisplayName

Setting a channel-specific description. In reporting, the description serves as a table heading for a table and as a key value for a key-value pair.

Syntax

Definition:

```
METHOD SetDisplayName : BOOL
VAR_INPUT
    nChannel : UDINT;
    sDisplayName : STRING(255);
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
sDisplayName	STRING(255)	Input value of type string.

Return value

Name	Type	Description
SetDisplayName	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.1.6 SetEdgeValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetEdgeValue : BOOL
VAR_INPUT
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns TRUE if successful.

5.1.1.10.1.2 Reporting Collector Interval

The Reporting Collector Interval collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the timespan of the configured interval has expired.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Rpt_CollectorInterval
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fbTimeLastEvent: FB_ALY_DateTime;
    fbTimeCurrentInterval: FB_ALY_Timespan;
    nBufferCount: UDINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fbTimeLastEvent	FB_ALY_DateTime	Stores the timestamp of the last message sent to the reporting server.
fbTimeCurrentInterval	FB_ALY_Timespan	Indicates the timespan until the next event.
nBufferCount	UDINT	Indicates the number of elements in the buffer.

Methods

Name	Type	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetDisplayName()	Local	Method for describing the input channel.
Pause()	Local	Method to pause the execution including the internal time intervals.

Sample

```
VAR
    fbRpt_CollectorInterval: FB_Rpt_CollectorInterval(sNetID:='', nNumChannels:=3);
    fbSystemTime : FB_ALY_GetSystemTime;
    fDataInCh1: LREAL;
    fDataInCh2: LREAL;
    fDataInCh3: LREAL;

    tInterval : LTIME;
    sReportName: STRING(255);
    sDataKey: STRING(255);
    nBufferSize: UDINT;
    bIncludeTimestamps: BOOL;
    aChannelNames: ARRAY [1..3] OF STRING(255);
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    sReportName:= 'Beckhoff Report Template';
    sDataKey:= 'DataKey101';
    nBufferSize:= 1;
    bIncludeTimestamps:= TRUE;
    aChannelNames[1]:= 'Min';
    aChannelNames[2]:= 'Max';
    aChannelNames[3]:= 'Avg';
    tInterval := LTIME#5S;

    fbRpt_CollectorInterval.Configure(
        sConfigId:= sReportName,
        sDataKey:= sDataKey,
        tInterval:= tInterval,
        nBufferSize:= nBufferSize,
        bBufferOnEvent:= bBufferOnEvent,
        bIncludeTimestamps:= bIncludeTimestamps);

    fbRpt_CollectorInterval.SetDisplayName(nChannel:= 1, sDisplayName:= aChannelNames[1]);
    fbRpt_CollectorInterval.SetDisplayName(nChannel:= 2, sDisplayName:= aChannelNames[2]);
    fbRpt_CollectorInterval.SetDisplayName(nChannel:= 3, sDisplayName:= aChannelNames[3]);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbRpt_CollectorInterval.SetChannelValue(nChannel:= 1, Input:= fDataInCh1);
fbRpt_CollectorInterval.SetChannelValue(nChannel:= 2, Input:= fDataInCh2);
fbRpt_CollectorInterval.SetChannelValue(nChannel:= 3, Input:= fDataInCh3);

fbRpt_CollectorInterval.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.10.1.2.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.2.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    sConfigId : STRING;
    sDataKey : STRING;
    tInterval : LTIME;
    bIncludeTimestamps : BOOL;
    nBufferSize : UDINT;
    bBufferOnEvent : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sConfigId	STRING	Indicates the name of the report. The name must correspond to the name of the configuration file at the Reporting Server.
sDataKey	STRING	The data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
tInterval	LTIME	Time interval at which the data should be sent to the Reporting Server.
bIncludeTimestamps	BOOL	Inserts a column with the timestamps.
nBufferSize	UDINT	Indicates the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
bBufferOnEvent	BOOL	Indicates how the data is to be collected and buffered. If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer. If the parameter is False, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the TcReportingServer.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.2.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.10.1.2.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.2.5 SetDisplayName

Setting a channel-specific description. In reporting, the description serves as a table heading for a table and as a key value for a key-value pair.

Syntax

Definition:

```
METHOD SetDisplayName : BOOL
VAR_INPUT
    nChannel : UDINT;
    sDisplayName : STRING(255);
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
sDisplayName	STRING(255)	Input value of type string.

Return value

Name	Type	Description
SetDisplayName	BOOL	Returns TRUE if successful.

5.1.1.10.1.2.6 Pause

With the Pause method it is possible to pause the execution of the function block. The internally running time interval does not continue after calling the pause function. A single call of the method is sufficient. The interval is not continued until the next call of the Call method.

Syntax

Definition:

```
METHOD Pause : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Pause	BOOL	Returns TRUE if successful.

5.1.1.10.1.3 Reporting Collector Time

The Reporting Collector Time collects the data from the input channels and sends the data to the Reporting Server after an event or after the buffer is filled, depending on the configuration. An event is triggered when the configured switch-on time is reached. The switch-on time and the days of the weeks can be configured.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Rpt_CollectorTime
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fbTimeLastEvent: FB_ALY_DateTime;
    fbTimeUntilNextSwitch: FB_ALY_Timespan;
    nBufferCount: UDINT;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fbTimeLastEvent	FB_ALY_DateTime	Stores the timestamp of the last message sent to the reporting server.
fbTimeUntilNextSwitch	FB_ALY_Timespan	Indicates the timespan until the next event.
nBufferCount	UDINT	Indicates the number of elements in the buffer.

Methods

Name	Type	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
SetDisplayName()	Local	Method for describing the input channel.

Sample

```
VAR
    fbRpt_CollectorTime: FB_Rpt_CollectorTime(sNetID:='', nNumChannels:=3);
    fbSystemTime : FB_ALY_GetSystemTime;
    fDataInCh1: LREAL;
    fDataInCh2: LREAL;
    fDataInCh3: LREAL;

    tTimeOn : LTIME;
    nDayOfWeekMask : WORD;
    sReportName: STRING(255);
    sDataKey: STRING(255);
    nBufferSize: UDINT;
    bIncludeTimestamps: BOOL;
    aChannelNames: ARRAY [1..3] OF STRING(255);
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    sReportName:= 'Beckhoff Report Template';
    sDataKey:= 'DataKey101';
    nBufferSize:= 1;
    bIncludeTimestamps:= TRUE;
    aChannelNames[1]:= 'Min';
    aChannelNames[2]:= 'Max';
    aChannelNames[3]:= 'Avg';
    tTimeOn := LTIME#5H;
    nDayOfWeekMask := 2#0011_0101;

    fbRpt_CollectorTime.Configure(
        sConfigId:= sReportName,
        sDataKey:= sDataKey,
        tTimeOn:= tTimeOn,
        nDayOfWeekMask:= nDayOfWeekMask,
```

```
        nBufferSize:= nBufferSize,
        bBufferOnEvent:= bBufferOnEvent
        bIncludeTimestamps:= bIncludeTimestamps);

    fbRpt_CollectorTime.SetDisplayName(nChannel:= 1, sDisplayName:= aChannelNames[1]);
    fbRpt_CollectorTime.SetDisplayName(nChannel:= 2, sDisplayName:= aChannelNames[2]);
    fbRpt_CollectorTime.SetDisplayName(nChannel:= 3, sDisplayName:= aChannelNames[3]);
END_IF

// Get current system time
fbSystemTime.Call();

// Call algorithm
fbRpt_CollectorTime.SetChannelValue(nChannel:= 1, Input:= fDataInCh1);
fbRpt_CollectorTime.SetChannelValue(nChannel:= 2, Input:= fDataInCh2);
fbRpt_CollectorTime.SetChannelValue(nChannel:= 3, Input:= fDataInCh3);

fbRpt_CollectorTime.Call(fbSystemTime.tSystemTime);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.10.1.3.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.3.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    sConfigId : STRING;
    sDataKey : STRING;
    tTimeOn : LTIME;
    nDayOfWeekMask : WORD;
    bIncludeTimestamps : BOOL;
    nBufferSize : UDINT;
    bBufferOnEvent : BOOL;
END_VAR
```

Inputs

Name	Type	Description
sConfigId	STRING	Indicates the name of the report. The name must correspond to the name of the configuration file at the Reporting Server.
sDataKey	STRING	The data key should be unique within a report. The data object can be uniquely assigned to the report via the data key.
tTimeOn	LTIME	Indicates the switch-on time.
nDayOfWeekMask	WORD	Indicates the days of the week when the timer should be active.
bIncludeTimestamps	BOOL	Inserts a column with the timestamps.
nBufferSize	UDINT	Indicates the size of the buffered data. If the buffer size is equal to one, a key-value structure is built. If the buffer size is greater than one, the data is presented in a table.
bBufferOnEvent	BOOL	Indicates how the data is to be collected and buffered. If the parameter is True, the data of the inputs is buffered at each edge signal at the input. As soon as the Buffer Count output has the same value as the Buffer Size parameter, the data is sent to the TcReportingServer. If the parameter is False, the data of the inputs are buffered in the buffer at each cycle. Once the buffer size is reached, the new data replaces the previous data. If there is an edge signal at the input, the data is sent to the TcReportingServer.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.3.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.10.1.3.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.10.1.3.5 SetDisplayName

Setting a channel-specific description. In reporting, the description serves as a table heading for a table and as a key value for a key-value pair.

Syntax

Definition:

```
METHOD SetDisplayName : BOOL
VAR_INPUT
    nChannel : UDINT;
    sDisplayName : STRING(255);
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
sDisplayName	STRING(255)	Input value of type string.

Return value

Name	Type	Description
SetDisplayName	BOOL	Returns `TRUE` if successful.

5.1.1.10.2 Reporting Trigger

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

5.1.1.10.2.1 Reporting Trigger Edge

The Reporting Trigger Edge triggers the creation of a report after an event is triggered. An event is triggered when the input channel signal exceeds the configured edge at a specified threshold. Internally, the inputs that were once True remain True. The inputs are only reset to False as soon as all inputs were True at least once. This allows the output bNewResult to be used as one input by multiple Reporting Collectors and once all Reporting Collectors have sent a data message, a trigger message is sent.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Rpt_TriggerEdge
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
```

```
    fbTimeLastEvent: FB_ALY_DateTime;
    sOverview: STRING(255) := '';
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fbTimeLastEvent	FB_ALY_DateTime	Stores the timestamp of the last message sent to the reporting server.
sOverview	STRING	Indicates which input channels were TRUE at least once.

Methods

Name	Definition location	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
SetChannelValue()	Local	Method for passing values to the algorithm.

Properties

Name	Type	Access	Definition location	Initial value	Description
fTolerance	LREAL	Get, Set	Local	0.0	Tolerance value for the Equal / NotEqual comparisons

Sample

```
VAR
    fbRptTriggerEdge: FB_Rpt_TriggerEdge(sNetId:= '', nNumChannels:=2);
    fbSystemTime : FB_ALY_GetSystemTime;
    fEdgeCh1: BOOL;
    fEdgeCh2: BOOL;
    stThresholdEdge: ST_ALY_Threshold;
    fTolerance: LREAL;
    sReportName: STRING(255);
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;
```

```
    sReportName:= 'Beckhoff Report Template';
    fTolerance:= 0;
    stThresholdEdge.eComparisonOperator := E_ALY_ComparisonOperator.Equals;
    stThresholdEdge.fThreshold := 1;

    fbRptTriggerEdge.Configure(sConfigId:= sReportName, stThresholdEdge := stThresholdEdge);
END_IF

fbRptTriggerEdge.SetChannelValue(nChannel:= 1, input:= fEdgeCh1);
fbRptTriggerEdge.SetChannelValue(nChannel:= 2, input:= fEdgeCh1);

fbRptTriggerEdge.Call(tTimestamp);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.10.2.1.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.1.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    sConfigId : STRING;
    stThresholdEdge : ST_ALY_Threshold;
END_VAR
```

Inputs

Name	Type	Description
sConfigId	STRING	Indicates the name of the report. The name must correspond to the name of the configuration file at the Reporting Server.
stThresholdEdge	ST_ALY_Threshold [▶ 346]	Combination of comparison operator and threshold for edge detection.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.1.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.10.2.1.4 SetChannelValue

Set channel specific input value. The input value is not processed until the Call() method has been invoked.

Syntax

Definition:

```
METHOD SetChannelValue : BOOL
VAR_INPUT
    nChannel : UDINT;
    input : ANY;
END_VAR
```

Inputs

Name	Type	Description
nChannel	UDINT	Channel index (1 to nChannels)
Input	ANY	Input value of any data type.

Return value

Name	Type	Description
SetChannelValue	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.2 Reporting Trigger Interval

The Reporting Trigger Interval triggers the creation of a report after an event has been triggered. An event is triggered when the timespan of the configured interval has expired.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Rpt_TriggerInterval
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fbTimeLastEvent: FB_ALY_DateTime;
    fbTimeCurrentInterval: FB_ALY_Timespan;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fbTimeLastEvent	FB_ALY_DateTime	Stores the timestamp of the last message sent to the reporting server.
fbTimeCurrentInterval	FB_ALY_Timespan	Indicates the timespan until the next event.

Methods

Name	Type	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
Pause()	Local	Method to pause the execution including the internal time intervals.

Sample

```
VAR
    fbRptTriggerInterval: FB_Rpt_TriggerInterval(sNetId:= '', nNumChannels:=2);
    fbSystemTime : FB_ALY_GetSystemTime;
    sReportName: STRING(255);
    tInterval: LTIME;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    sReportName:= 'Beckhoff Report Template';
    tInterval := LTIME#5S;

    fbRptTriggerInterval.Configure(sConfigId:= sReportName, tInterval:= tInterval);
END_IF


fbRptTriggerInterval.Call(tTimestamp);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.10.2.2.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.2.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    sConfigId : STRING;
    tInterval : LTIME;
END_VAR
```

Inputs

Name	Type	Description
sConfigId	STRING	Indicates the name of the report. The name must correspond to the name of the configuration file at the Reporting Server.
tInterval	LTIME	Time interval at which the data should be sent to the Reporting Server.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.2.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.1.10.2.2.4 Pause

With the Pause method it is possible to pause the execution of the function block. The internally running time interval does not continue after calling the pause function. A single call of the method is sufficient. The interval is not continued until the next call of the Call method.

Syntax

Definition:

```
METHOD Pause : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Pause	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.3 Reporting Trigger Time

The Reporting Trigger Time triggers the creation of a report after an event has been triggered. An event is triggered when the configured switch-on time is reached. The switch-on time and the days of the weeks can be configured.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Rpt_TriggerTime
VAR_OUTPUT
    ipResultMessage: Tc3_EventLogger.I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    fbTimeLastEvent: FB_ALY_DateTime;
    fbTimeUntilNextSwitch: FB_ALY_Timespan;
END_VAR
```

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Contains more detailed information on the current return value. This special interface pointer is internally secured so that it is always valid/assigned.
bError	BOOL	This output is TRUE if an error occurs.
bNewResult	BOOL	When a new result has been calculated, the output is TRUE.
bConfigured	BOOL	Displays TRUE when the function block is successfully configured.
fbTimeLastEvent	FB_ALY_DateTime	Stores the timestamp of the last message sent to the reporting server.
fbTimeUntilNextSwitch	FB_ALY_Timespan	Indicates the timespan until the next event.

Methods

Name	Type	Description
Call()	Local	Method for calculating the outputs for a specific configuration.
Configure()	Local	General configuration of the algorithm with its parameterized conditions.
Reset()	Local	Resets all internal states or the calculations performed so far.
Pause()	Local	Method to pause the execution including the internal time intervals.

Sample

```
VAR
    fbRptTriggerTime: FB_Rpt_TriggerTime(sNetId:= '', nNumChannels:=2);
    fbSystemTime : FB_ALY_GetSystemTime;
    sReportName: STRING(255);
    nDayOfWeekMask : WORD;
END_VAR
```

```
// Configure algorithm
IF bConfigure THEN
    bConfigure := FALSE;

    sReportName:= 'Beckhoff Report Template';
    tTimeOn := LTIME#5H;
    nDayOfWeekMask := 2#0011_0101;

    fbRptTriggerTime.Configure(sConfigId:= sReportName, tTimeOn:= tTimeOn, nDayOfWeekMask:= nDayOfWe
ekMask);
END_IF

fbRptTriggerTime.Call(tTimestamp);
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.1.10.2.3.1 Call

Call the algorithm after a new input value is set. A new input can be assigned by using the method SetChannelValue().

Syntax

Definition:

```
METHOD Call : BOOL
VAR_INPUT
    tTimestamp : ULINT;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Current timestamp with a resolution of 1 ns.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.3.2 Configure

Configuring the algorithm.

Syntax

Definition:

```
METHOD Configure : BOOL
VAR_INPUT
    sConfigId : STRING;
    tTimeOn : LTIME;
    nDayOfWeekMask : WORD;
END_VAR
```

Inputs

Name	Type	Description
sConfigId	STRING	Indicates the name of the report. The name must correspond to the name of the configuration file at the Reporting Server.
tTimeOn	LTIME	Indicates the switch-on time.
nDayOfWeekMask	WORD	Indicates the days of the week when the timer should be active.

Return value

Name	Type	Description
Configure	BOOL	Returns `TRUE` if successful.

5.1.1.10.2.3.3 Reset

Resetting the algorithm.

Return value

Name	Type	Description
Reset	BOOL	Returns `TRUE` if the reset was successful.

5.1.2 System

5.1.2.1 FB_ALY_DateTime

Function block to store and process timestamps. Timestamps are stored as raw values in accuracy of 1ns. Using the methods of this FB the timespan can be modified, compared or formatted.

Methods

Name	Definition Location	Description
AddRaw	Local	Method to add a timespan in raw format.
AddTimespan	Local	Method to add a timespan stored in FB_ALY_Timespan [▶ 318]
SubRaw	Local	Method to subtract a timespan in raw format.
SubTimespan	Local	Method to subtract a timespan stored in FB_ALY_Timespan [▶ 318]
EqualsTo	Local	Method to compare two timestamps.
ToString	Local	Method to format the current raw value as string.

Sample

```
VAR
    fbDateTime : FB_ALY_DateTime;
    fbSystemTime : FB_ALY_GetSystemTime;
    sFormattedDateTime : STRING(29);
END_VAR
```

```
// Get current system time
fbSystemTime.Call();

fbDateTime.nRaw := fbSystemTime.tSystemTime;
fbDateTime.AddRaw(TO_LINT(LTIME#1H));
sFormattedDateTime := fbDateTime.ToString()
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.2.1.1 AddRaw

Call to add a timespan in raw format to the timestamp value.

Syntax

Definition:

```
METHOD AddRaw : ULINT
VAR_INPUT
    nRaw : LINT;
END_VAR
```

Inputs

Name	Type	Description
nRaw	LINT	Raw value to be added to the timestamp.

Return value

Name	Type	Description
AddRaw	ULINT	Returns the resulting timestamp value.

5.1.2.1.2 AddTimespan

Call to add a timespan to the timestamp value.

Syntax

Definition:

```
METHOD AddTimespan : ULINT
VAR_INPUT
    ipTimespan : I_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
ipTimespan	I_ALY_Timespan	Interface pointer to a timespan instance to be added to the timestamp.

Return value

Name	Type	Description
AddTimespan	ULINT	Returns the resulting timestamp value.

5.1.2.1.3 SubRaw

Call to subtract a timespan in raw format from the timestamp value.

Syntax

Definition:

```
METHOD SubRaw : BOOL
VAR_INPUT
    nRaw : LINT;
END_VAR
```

Inputs

Name	Type	Description
nRaw	LINT	Raw value to be subtracted from the timestamp.

Return value

Name	Type	Description
SubRaw	ULINT	Returns resulting timestamp value.

5.1.2.1.4 SubTimespan

Call to subtract a timespan from the timestamp value.

Syntax

Definition:

```
METHOD SubTimespan : ULINT
VAR_INPUT
    ipTimespan : I_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
ipTimespan	I_ALY_Timespan	Interface pointer to a timespan instance to be added to the timestamp.

Return value

Name	Type	Description
SubTimespan	ULINT	Returns the resulting timestamp value.

5.1.2.1.5 EqualsTo

Call to compare the current timestamp with the input timestamp.

Syntax

Definition:

```
METHOD EqualsTo : BOOL
VAR_INPUT
    ipDateTime: I_ALY_DateTime;
END_VAR
```

Inputs

Name	Type	Description
ipDateTime	I_ALY_DateTime	Interface pointer to a DateTime instance for comparison.

Return value

Name	Type	Description
EqualsTo	BOOL	Returns TRUE if the timestamps are equal.

5.1.2.1.6 ToString

Method to format the current raw value as string. The output format is YYYY-MM-DD-hh:mm:ss.nnnnnnnnn.

Return value

Name	Type	Description
ToString	STRING(29)	Returns the formatted string.

5.1.2.2 FB_ALY_GetSystemTime

Gets the local system time in resolution of 1ns. The accuracy of this time is 100ns. The value is updated by the Call()-method.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_DateTime
    tSystemTime : ULINT;
END_VAR
```

Outputs

Name	Type	Description
tSystemTime	ULINT	Current system time.

Methods

Name	Definition location	Description
Call()	Local	Method for recording the current system time.

Sample

```
VAR
    fbSystemTime : FB_ALY_GetSystemTime;
    tSystemTime : ULINT;
END_VAR
```

```
// Gather current system time
fbSystemTime.Call();

// Allocate system time
tSystemTime := fbSystemTime.tSystemTime;
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.2.2.1 Call

Call to gather the current system time. The system time is stored in the output data of the FB.

Return value

Name	Type	Description
Call	BOOL	Returns `TRUE` if successful.

5.1.2.3 FB_ALY_Timespan

Function block to store and process timespans. Timespans are stored as raw values in accuracy of 1ns. Using the methods of this FB the timespan can be modified, compared or formatted.

Methods

Name	Definition location	Description
AddRaw	Local	Method for adding a timespan in raw format.
AddTimespan	Local	Method for adding a timespan stored in FB_ALY_Timespan [▶ 318].
SubRaw	Local	Method for subtracting a timespan in raw format.
SubTimespan	Local	Method for subtracting a timespan stored in FB_ALY_Timespan [▶ 318].
EqualsTo	Local	Method for comparing two timestamps.
ToFormatString	Local	Method for formatting the current raw value as a string.
ToString	Local	Method for formatting the current raw value as a string with configurable precision.
TotalDays	Local	Method to get the total number of days.
TotalHours	Local	Method to get the total number of hours.
TotalMinutes	Local	Method to get the total number of minutes.
TotalSeconds	Local	Method to get the total number of seconds.

Sample

```
VAR
    fbTimespan : FB_ALY_Timespan;
    sTimespan : STRING;
    sFormattedTimespan : STRING;
END_VAR
```

```
fbTimespan.AddRaw(TO_LINT(LTIME#1S));
sTimespan := fbTimespan.ToString(eAccuracy := E_ALY_TimestampAccuracy.Second);
sFormattedTimespan := fbTimespan.ToFormatString();
```

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.2.3.1 AddRaw

Call to add a timespan in raw format to the timespan value.

Syntax

Definition:

```
METHOD AddRaw : ULINT
VAR_INPUT
    nRaw : LINT;
END_VAR
```

Inputs

Name	Type	Description
nRaw	LINT	Raw value to be added to the timespan.

Return value

Name	Type	Description
AddRaw	ULINT	Returns the resulting timespan value.

5.1.2.3.2 AddTimespan

Call to add a timespan to the timespan value.

Syntax

Definition:

```
METHOD AddTimespan : ULINT
VAR_INPUT
    ipTimespan : I_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
ipTimespan	I_ALY_Timespan	Interface pointer to an Timespan instance to be added to the timespan.

Return value

Name	Type	Description
AddTimespan	ULINT	Returns resulting timespan value.

5.1.2.3.3 SubRaw

Call to subtract a timespan in raw format from the timespan value.

Syntax

Definition:

```
METHOD SubRaw : BOOL
VAR_INPUT
    nRaw : LINT;
END_VAR
```

Inputs

Name	Type	Description
nRaw	LINT	Raw value to be subtracted from the timespan.

Return value

Name	Type	Description
SubRaw	ULINT	Returns resulting timespan value.

5.1.2.3.4 SubTimespan

Call to subtract a timespan from the timespan value.

Syntax

Definition:

```
METHOD SubTimespan : ULINT
VAR_INPUT
    ipTimespan : I_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
ipTimespan	I_ALY_Timespan	Interface pointer to an Timespan instance to be added to the timespan.

Return value

Name	Type	Description
SubTimespan	ULINT	Returns resulting timespan value.

5.1.2.3.5 EqualsTo

Call to compare the current timespan with the input timespan.

Syntax

Definition:

```
METHOD EqualsTo : BOOL
VAR_INPUT
    ipTimespan : I_ALY_Timespan;
END_VAR
```

Inputs

Name	Type	Description
ipTimespan	I_ALY_Timespan	Interface pointer to an Timespan instance to compare with.

Return value

Name	Type	Description
EqualsTo	BOOL	Returns TRUE if timespans are equal.

5.1.2.3.6 ToString

Method to format the current raw value as string with configurable accuracy. An example output is 1d2m3s4ms5ns.

Inputs

Name	Type	Description
eAccuracy	E_ALY_TimestampAccuracy [▶ 352]	Define accuracy of output string.

Return value

Name	Type	Description
ToString	STRING	Returns formatted string.

5.1.2.3.7 ToFormatString

Method to format the current raw value as string. The output format is dd:hh:mm:ss,f.

Return value

Name	Type	Description
ToFormatString	STRING	Returns formatted string.

5.1.2.3.8 TotalDays

Method to get the amount of total days.

Return value

Name	Type	Description
TotalDays	DINT	Returns amount of total days.

5.1.2.3.9 TotalHours

Method to get the amount of total hours.

Return value

Name	Type	Description
TotalHours	DINT	Returns amount of total hours.

5.1.2.3.10 TotalMinutes

Mehod to get the amount of total minutes.

Return value

Name	Type	Description
TotalMinutes	DINT	Returns amount of total minutes.

5.1.2.3.11 TotalSeconds

Method to get the amount of total seconds.

Return value

Name	Type	Description
TotalSeconds	LINT	Returns amount of total seconds.

5.1.3 IoT

5.1.3.1 IoT Symbol

5.1.3.1.1 FB_ALY_IotSymbol_BOOL

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_BOOL
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
```

```
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.2 FB_ALY_IotSymbol_BYTE

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_BYTE
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
```

```
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.3 FB_ALY_IotSymbol_DINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_DINT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
```

```
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.4 FB_ALY_IotSymbol_DWORD

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_DWORD
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
```

```
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.5 FB_ALY_IotSymbol_INT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_INT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
```

```
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.6 FB_ALY_IotSymbol_LINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_LINT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.7 FB_ALY_IotSymbol_LREAL

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_LREAL
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.8 FB_ALY_IotSymbol_LWORD

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_LWORD
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.9 FB_ALY_IotSymbol_REAL

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_REAL
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.10 FB_ALY_IotSymbol_SINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_SINT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.11 FB_ALY_IotSymbol_STRING

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_STRING
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.12 FB_ALY_IotSymbol_UDINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_UDINT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.13 FB_ALY_IotSymbol_UINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_UINT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.14 FB_ALY_IotSymbol_ULINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_ULINT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.15 FB_ALY_IotSymbol_USINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_USINT
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.1.16 FB_ALY_IotSymbol_WORD

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_IotSymbol_WORD
VAR_INPUT
    stConfig : ST_ALY_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bSymbolHandlerAssigned	BOOL	TRUE if symbol handler assigned
bVariableFound	BOOL	TRUE if variable found in stream
sSymbolPath	STRING(255)	Topic/Stream.Symbol
tCycleTime	LTIME	Cycle time of publishing system
nMaxNumElements	UDINT	Maximum number of Symbols stored affected by StreamHelper
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
GetValue	Local	Get the value of the specified element
GetOversamplingValues	Local	Get the oversampling values of the specified element
GetArrayValues	Local	Get the array values of the specified element

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.2 FB_ALY_StreamHelper

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StreamHelper
VAR_INPUT
    stConfig : ST_ALY_StreamHelper_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bConnected: BOOL;
    sStream: STRING(255);
    nNumIotSymbolsRegistered: UDINT;
    tCycleTime: LTIME;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_IotSymbol_Config	Struct to configure the FB

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	Eventlogger
bError	BOOL	TRUE if an error has been occurred
bNewResult	BOOL	TRUE if a new result has been calculated
bConfigured	BOOL	TRUE if FB is configured successfully
bConnected	BOOL	TRUE if StreamHelper connected to message broker
sStream	STRING(255)	Topic/Stream
nNumIotSymbolsRegistered	UDINT	Amount of registered IoT symbols
tCycleTime	LTIME	Cycle time of publishing system
nNumElements	UDINT	Number of elements received last call

Methods

Name	Definition Location	Description
Call	Local	
Configure	Local	
GetTimestampElement	Local	
AddIotSymbol	Local	
ContainsIotSymbol	Local	
ReleaseAllIotSymbols	Local	
ReleaseIotSymbol	Local	

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.3 BinaryStream

5.1.3.3.1 FB_ALY_BinaryStream_File

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StreamHelper
VAR_INPUT
    nCycleTime: UDINT;
    stSystemID: GUID;
    sPath: T_MaxString;
    nMaxFileSize: UINT;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipResultMessage: I_TcMessage;
END_VAR
```

Inputs

Name	Type	Description
nCycleTime	UDINT	Logging cycle time
stSystemID	GUID	Twincat System ID
sPath	T_MaxString	Folder path to log in
nMaxFileSize	UINT	Max file size

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error has been occurred
ipResultMessage	I_TcMessage	Eventlogger

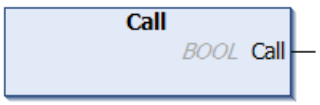
Methods

Name	Definition Location	Description
Call [▶ 340]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Close [▶ 340]	Local	Close the opened tay file
CreateDescriptionFile [▶ 341]	Local	Create *.tad file for stream desciption
CreateSymbolsFile [▶ 341]	Local	Create *.tas file for symbol description
WriteData [▶ 342]	Local	Write data to *.tay file

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.3.1.1 Call



Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	

5.1.3.3.1.2 Close

Syntax

```
METHOD Close : BOOL
```

Return value

Name	Type	Description
Close	BOOL	TRUE if done

5.1.3.3.1.3 CreateDescriptionFile

Syntax

```
METHOD CreateDescriptionFile : BOOL
VAR_INPUT
    nDataSize: UDINT;
    sLayout: GUID;
END_VAR
```

Inputs

Name	Type	Description
nDataSize	UDINT	Size of one Sample
sLayout	GUID	Layout Guid created by CreateSymbolic

Return value

Name	Type	Description
CreateDescriptionFile	BOOL	TRUE if done

5.1.3.3.1.4 CreateSymbolsFile

Syntax

```
METHOD CreateSymbolsFile : BOOL
VAR_INPUT
    pSymbol: PVOID;
    nSymbolSize: UDINT;
    sDataType: STRING;
END_VAR
VAR_OUTPUT
    sLayout: GUID;
END_VAR
```

Inputs

Name	Type	Description
pSymbol	PVOID	Pointer to ADS Symbol
nSymbolSize	UDINT	Size of ADS Symbol
sDataType	STRING	Name of ADS Symbol Datatype

Outputs

Name	Type	Description
sLayout	GUID	Layout hash of symbol description

Return value

Name	Type	Description
CreateSymbolsFile	BOOL	TRUE if done

5.1.3.3.1.5 WriteData

Syntax

```
METHOD WriteData : BOOL
VAR_INPUT
    pData: PVOID;
    nDataSize: UDINT;
    nDataCount: INT;
    sLayout: GUID;
END_VAR
```

Inputs

Name	Type	Description
pData	PVOID	Pointer to sample array
nDataSize	UDINT	Size of sample array
nDataCount	INT	Count of sample array elements
sLayout	GUID	Symbolic layout

Return value

Name	Type	Description
WriteData	BOOL	TRUE if done

5.1.3.3.2 FB_ALY_BinaryStream_Mqtt

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_BinaryStream_Mqtt
VAR_INPUT
    nCycleTime: UDINT;
    stSystemID: GUID;
    sStreamTopic: T_MaxString;
    stMqttConnSettings: ST_MqttConnectionSettings;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipResultMessage: I_TcMessage;
    eConnectionState: ETcIotMqttClientState;
END_VAR
```

Inputs

Name	Type	Description
nCycleTime	UDINT	Logging cycle time
stSystemID	GUID	Twincat System ID
sStreamTopic	T_MaxString	Topic to stream to
stMqttConnSettings	ST_MqttConnectionSettings	Settings for the mqtt connection to the message broker

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error has been occurred
ipResultMessage	I_TcMessage	Eventlogger
eConnectionState	ETcIotMqttClientState	Indicates the state of the connection between client and broker as enumeration ETcIotMqttClientState.

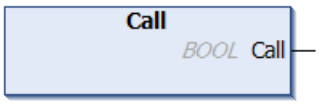
Methods

Name	Definition Location	Description
Call [▶ 343]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
CloseStream [▶ 343]	Local	Close the analytics binary stream
CreateDescription [▶ 344]	Local	Create binary stream description
CreateSymbolic [▶ 344]	Local	Create binary stream symbolic
SendData [▶ 345]	Local	Send binary stream data

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

5.1.3.3.2.1 Call



Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	

5.1.3.3.2.2 CloseStream

Syntax

```
METHOD CloseStream : BOOL
```

Return value

Name	Type	Description
CloseStream	BOOL	TRUE if done

5.1.3.3.2.3 CreateDescription

Syntax

```
METHOD CreateDescription : BOOL
VAR_INPUT
    nDataSize: UDINT;
    sLayout: GUID;
END_VAR
```

Inputs

Name	Type	Description
nDataSize	UDINT	Size of one Sample
sLayout	GUID	Layout Guid created by CreateSymbolic

Return value

Name	Type	Description
CreateDescription	BOOL	TRUE if done

5.1.3.3.2.4 CreateSymbolic

Syntax

```
METHOD CreateSymbolic : BOOL
VAR_INPUT
    pSymbol: PVOID;
    nSymbolSize: UDINT;
    sDataType: STRING;
END_VAR
VAR_OUTPUT
    sLayout: GUID;
END_VAR
```

Inputs

Name	Type	Description
pSymbol	PVOID	Pointer to ADS Symbol
nSymbolSize	UDINT	Size of ADS Symbol
sDataType	STRING	Name of ADS Symbol Datatype

Outputs

Name	Type	Description
sLayout	GUID	Layout hash of symbol description

Return value

Name	Type	Description
CreateSymbolic	BOOL	TRUE if done

5.1.3.3.2.5 SendData

Syntax

```
METHOD SendData : BOOL
VAR_INPUT
    pData: PVOID;
    nDataSize: UDINT;
    nDataCount: INT;
    sLayout: GUID;
END_VAR
```

Inputs

Name	Type	Description
pData	PVOID	Pointer to sample array
nDataSize	UDINT	Size of sample array
nDataCount	INT	Count of sample array elements
sLayout	GUID	Symbolic layout

Return value

Name	Type	Description
SendData	BOOL	TRUE if done

5.2 Functions

5.2.1 F_RawTimespan_TO_Structured

Function to convert an raw timespan input to an output of type ST_ALY_Timespan [▶ 352].

Inputs

Name	Type	Description
tTimespan	LINT	Timespan to be converted.

Return value

Name	Type	Description
F_RawTimespan_TO_Structured	ST_ALY_Timespan [▶ 352]	Returns structured timespan.

5.2.2 F_StructuredTimespan_TO_Raw

Function to convert an input of type ST_ALY_Timespan [▶ 352] to a raw timespan.

Inputs

Name	Type	Description
stTimespan	ST_ALY_Timespan [▶ 352]	Structured timespan to be converted.

Return value

Name	Type	Description
F_StructuredTimespan_TO_Raw	LINT	Returns raw timespan.

5.3 Data types

5.3.1 General

5.3.1.1 ST_ALY_Threshold

```
TYPE ST_ALY_Threshold :
STRUCT
    eComparisonOperator : E_ALY_ComparisonOperator;
    fThreshold          : LREAL;
END_STRUCT
END_TYPE
```

5.3.1.2 ST_ALY_ThresholdOnOff

```
TYPE ST_ALY_ThresholdOnOff :
STRUCT
    On  : ST_ALY_Threshold;
    Off : ST_ALY_Threshold;
END_STRUCT
END_TYPE
```

5.3.1.3 ST_ALY_XyPosition

```
TYPE ST_ALY_XyPosition :
STRUCT
    X : LREAL;
    Y : LREAL;
END_STRUCT
END_TYPE
```

5.3.1.4 E_ALY_Classification_2Cls

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_Classification_2Cls :
(
    NotInitialized := 0,
    OK : =1,
    NOK := 2
) := NotInitialized;
END_TYPE
```

5.3.1.5 E_ALY_Classification_3Cls

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_Classification_3Cls :
(
    NotInitialized := 0,
    OK := 1,
    Warning := 2,
    Alarm := 3
) := NotInitialized;
END_TYPE
```

5.3.1.6 E_ALY_Classification_Bounds

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_Classification_Bounds :
(
    NotInitialized := 0,
    WithinBounds := 1,
    Smaller := 2,
    Bigger := 3
) := NotInitialized;
END_TYPE
```

5.3.2 Config

5.3.2.1 ST_ALY_IotSymbolString_Config

```
TYPE ST_ALY_IotSymbolString_Config :
STRUCT
    sSymbolName         : STRING(255) := '';       // Symbol name (e.g. MAIN.bOn)
    nStringSize         : UDINT := SIZEOF(STRING); // String lenth plus null termination
    nOversamplingFactor : UDINT := 1;              // Oversampling factor
    nArrayElements      : UDINT := 1;              // Number of array elements
END_STRUCT
END_TYPE
```

5.3.2.2 ST_ALY_IotSymbol_Config

```
TYPE ST_ALY_IotSymbol_Config :
STRUCT
    sSymbolName         : STRING(255) := ''; // Symbol name (e.g. MAIN.bOn)
    nOversamplingFactor : UDINT := 1;        // Oversampling factor
    nArrayElements      : UDINT := 1;        // Number of array elements
END_STRUCT
END_TYPE
```

5.3.2.3 ST_ALY_StreamHelper_Config

```
TYPE ST_ALY_StreamHelper_Config :
STRUCT
    nObjectID       : OTCID := 0;  // Object ID of referenced StreamHelper
    nNumInputBuffer : UDINT := 20; // Number of input buffer to be reserved for every symbol
END_STRUCT
END_TYPE
```

5.3.2.4 E_ALY_BandMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_BandMode :
(
    NotInitialized := 0,
    Absolute := 1,
    Relative := 2,
) := NotInitialized;
END_TYPE
```

5.3.2.5 E_ALY_CascadeStartupBehaviour

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_CascadeStartupBehaviour :
(
    NotInitialized := 0,
    WaitUntilFilled := 1,
    UsePreviousCascadeValue := 2,
) := NotInitialized;
END_TYPE
```

5.3.2.6 E_ALY_ComparisonOperator

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_ComparisonOperator :
(
    NotInitialized := 0,
    GreaterThan := 1,
    GreaterThanOrEqualTo := 2,
    LessThan := 3,
    LessThanOrEqualTo := 4,
    Equals := 5,
    NotEqualTo := 6
) := NotInitialized;
END_TYPE
```

5.3.2.7 E_ALY_CountMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_CountMode :
(
    NotInitialized := 0,
    Cyclic := 1,
    OnChange := 2,
) := NotInitialized;
END_TYPE
```

5.3.2.8 E_ALY_DayOfWeekMask

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_DayOfWeekMask :
(
    None := 0,
    Sunday := 1,
    Monday := 2,
    Tuesday := 4,
    Wednesday := 8,
    Thursday := 16,
    Friday := 32,
    Saturday := 64,
    EveryDay := 127,
    MondayToFriday := 62,
    Weekend := 65
) := NotInitialized;
END_TYPE
```

5.3.2.9 E_ALY_IntegrationMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_IntegrationMode :
(
    NotInitialized := 0,
    Direct := 1,
    Absolute := 2,
) := NotInitialized;
END_TYPE
```

5.3.2.10 E_ALY_LogicOperator

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_LogicOperator :
(
    NotInitialized := 0,
    AND_ := 1,
    OR_ := 2,
    NAND_ := 3,
    NOR_ := 4,
    XOR_ := 5
```

```
) := NotInitialized;
END_TYPE
```

5.3.2.11 E_ALY_MathOperator

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_MathOperator :
(
    NotInitialized := 0,
    Addition := 1,
    Subtraction := 2,
    Multiplication := 3,
    Division := 4,
    PowerOf := 5,
    Modulo := 6,
) := NotInitialized;
END_TYPE
```

5.3.2.12 E_ALY_MovingAvgStartupBehaviour

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_MovingAvgStartupBehaviour :
(
    NotInitialized := 0, (* Not initialized *)
    ZeroPadding := ,     (* All buffers are initialized with zero. *)
    UseFirstValue := 2,  (* Use first occuring value to initialize buffers. *)
    WaitUntilFilled := 3,(* Wait with calculation until filled. *)
    AvgOverExisting := 4 (* Average over existing elements. *)
) := NotInitialized;
END_TYPE
```

5.3.2.13 E_ALY_ReadState

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_ReadState :
(
    Idle := 0,
    Read := 1,
    Pending := 2,
    PendingRead := 3,
    PendingIdle := 4
) := Idle;
END_TYPE
```

5.3.2.14 E_ALY_StateHistMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_StateHistMode :
(
    NotInitialized := 0,
    Absolute := 1,
    Relative := 2
) := NotInitialized ;
END_TYPE
```

5.3.2.15 E_ALY_StringCompareMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_StringCompareMode :
(
    NotInitialized := 0,
    Equals := 1,
    BeginsWith := 2,
    Contains := 3
```

```
) := NotInitialized;
END_TYPE
```

5.3.2.16 E_ALY_TeachMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_TeachMode :
(
    NotInitialized := 0,
    Minimum := 1,
    Maximum := 2,
    Mean := 3,
) := NotInitialized;
END_TYPE
```

5.3.2.17 E_ALY_TeachState

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_TeachState :
(
    Idle := 0,
    Teach := 1,
    Pending := 2,
    PendingTeach := 3,
    PendingIdle := 4
) := Idle;
END_TYPE
```

5.3.2.18 E_ALY_TimerMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_TimerMode :
(
    NotInitialized := 0, (* Not initialized *)
    TON := 1,            (* Switch-on delay *)
    TOF := 2,            (* Switch-off delay *)
    TP := 3              (* Pulse *)
) := NotInitialized;
END_TYPE
```

5.3.2.19 E_ALY_GateMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_LogicOperator :
(
    NotInitialized:=0,
    IntersectGate:=1,
    NotIntersectGate:=2,
    IntersectProjection:=3,
    NotIntersectProjection:=4,
    IntersectGateOrProjection:=5
) := NotInitialized;
END_TYPE
```

5.3.2.20 E_ALY_CorrelationMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_CorrelationMode :
(
    NotInitialized := 0,
    Base := 1,
    Normed := 2,
    Covariance := 3,
    CovarianceBesel :=4,
    Pearson :=5
```

```
) := NotInitialized;
END_TYPE
```

5.3.2.21 E_ALY_WindowMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_WindowMode :
(
    NotInitialized := 0,
    SlidingWindow := 1,
    FixWindow := 2,
    Continuous := 3
) := NotInitialized;
END_TYPE
```

5.3.2.22 E_ALY_KMeansInitMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_KMeansInitMode :
(
    NotInitialized := 0,
    Random := 1,
    Equidistant := 2,
    Values := 3,
) := NotInitialized;
END_TYPE
```

5.3.2.23 E_ALY_FunctionType

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_FunctionType :
(
    NotInitialized := 0,
    Constt := 1,
    Sine := 2,
    Triangle := 3,
    Rectangle := 4,
    Sawtooth := 5,
) := NotInitialized;
END_TYPE
```

5.3.2.24 E_ALY_ConfigState

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_ConfigState :
(
    NotConfigured := 0,
    Pending := 1,
    Configured := 2,
) := NotConfigured ;
END_TYPE
```

5.3.2.25 E_ALY_HistMode

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_HistMode :
(
    NotInitialized := 0,
    Absolute := 1,
    Relative := 2
) := NotInitialized ;
END_TYPE
```

5.3.3 Time

5.3.3.1 ST_ALY_Timespan

```
TYPE ST_ALY_Timespan :
STRUCT
    bIsNegative   : BOOL;
    wDays         : WORD;
    wHours        : WORD;
    wMinutes      : WORD;
    wSeconds      : WORD;
    wMilliseconds : WORD;
    wMicroseconds : WORD;
    wNanoseconds  : WORD;
END_STRUCT
END_TYPE
```

5.3.3.2 E_ALY_TimestampAccuracy

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ALY_TimestampAccuracy :
(
    Day,
    Hour,
    Minute,
    Second,
    Millisecond,
    Microsecond,
    Nanosecond
);
END_TYPE
```

6 Samples

6.1 Combination of algorithms with local inputs

The sample shows how to include and combine analytics algorithms. Algorithms of Type FB_ALY_EdgeCounter_1Ch [▶ 31], FB_ALY_MathOperation_1Ch [▶ 201] and FB_ALY_MinMaxAvg_1Ch [▶ 71] are used.

The sample is available for download from here:

https://infosys.beckhoff.com/content/1033/tf3500_tc3_analytics_library/Resources/6917579531/.zip

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7 Appendix

7.1 Return codes

See the following list with return codes of the TwinCAT Analytics library.

Event ID (Hex)	Event text
16#1001	Error during initialization. No router memory available. Check size of router memory.
16#1002	Error in Transition PREOP->SAFEOP
16#1003	Error in Transition SAFEOP->OP
16#1004	Error in Transition SAFEOP->OP: No Task assigned. Module will not be executed cyclically.
16#1005	Error in Transition OP->SAFEOP
16#1006	Error in Transition SAFEOP->PREOP
16#1007	Error during initialization. Invalid object ID.
16#1008	Error during initialization. Invalid symbol name.
16#1009	Error during initialization. Invalid symbol size.
16#100A	Error during initialization. Invalid number of buffers.
16#100B	Error during initialization. Invalid data area number.
16#100C	Error during initialization. Invalid number of channels.
16#2001	Error during configuration. No router memory available. Check size of router memory.
16#2002	Error during configuration. A null pointer has been allocated.
16#2003	The configured DataType is not valid.
16#2004	The configured count mode is not valid.
16#2005	The configured integration mode is not valid.
16#2006	The configured state histogram mode is not valid.
16#2007	The configured comparison operator is not valid.
16#2008	The configured teach mode is not valid
16#2009	The configured logic operator is not valid.
16#200A	The configured mathematical operator is not valid.
16#200B	The configured startup behavior is not valid.
16#200C	The configured bounds are not valid.
16#200D	The amount of configured values is not valid.
16#200E	The configured sample rate is not valid.
16#200F	The configured interval is not valid.
16#2010	The configured interval is too short.
16#2011	The amount of configured subsets is not valid.
16#2012	The amount of configured bins is not valid.
16#2013	The configured file path is not valid.
16#2014	The configured segment size is not valid.
16#2015	The configured timeout is not valid.
16#2016	The configured band mode is not valid.
16#2017	The configured length is not valid
16#2018	The configured timer mode is not valid.
16#2019	The configured days of week are not valid.
16#201A	The configured threshold is not valid.
16#201B	The configured string compare mode is not valid.
16#201C	The configured gate mode is not valid.
16#201D	Error during configuration. Invalid symbol name.
16#201E	Error during configuration. Invalid symbol size.
16#201F	Error during configuration. Invalid number of buffers.
16#2020	Error during configuration. Invalid number of bands.
16#2021	Error during configuration. Illegal threshold order.

Event ID (Hex)	Event text
16#2022	Error during configuration. Missing initialization.
16#3001	Error during runtime. No router memory available. Check size of router memory.
16#3002	Error during runtime. A null pointer has been allocated.
16#3003	Error during runtime. Invalid input size.
16#3004	Error during runtime. Invalid output size.
16#3005	Error during runtime. Division by zero.
16#3006	Error during runtime. Timestamp cannot be null.
16#3007	Error during runtime. Missing configuration.
16#3008	Error during runtime. Invalid state.
16#3009	Error during runtime. Cyclic caller is assigned. Methods can not be called.
16#300A	Error during runtime. Out of range.
16#300B	Error during runtime. Invalid header size.
16#300C	Error during runtime. Invalid file.
16#300D	Error during runtime. Invalid interval.
16#300E	Error during runtime. Symbol already assigned.
16#300F	Error during runtime. No stream helper assigned.
16#3010	Error during runtime. Invalid input channel.
16#3011	Error during runtime. Invalid data type.
16#3012	Error during runtime. Invalid unit.

7.2 FAQ - frequently asked questions and answers

In this section frequently asked questions are answered in order to make your work with TwinCAT Analytics library easier. If you have further questions, please contact our support team support@beckhoff.com.

Can I use the Analytics library independent from the TwinCAT Analytics workflow? [▶ 356]

Is it possible to use TE3500 Analytics Workbench with automatic code generation, if I like to use the Analytics library in my local application? [▶ 356]

?Can I use the Analytics library independent from the TwinCAT Analytics workflow?

!Yes, for sure. You can use the library as a standard PLC library for your local machine application without any MQTT communication and other Analytics products.

?Is it possible to use TE3500 Analytics Workbench with automatic code generation, if I like to use the Analytics library in my local application?

!Yes, you can choose in TE3500 Analytics Workbench configurator the option “Add to existing Solution” in the Deploy Analytics Runtime dialog. This adds the Analytics part to your local application code by taking the right Solution in the path control. Also it is possible to add variables into the configurator from an ADS route. In the code generation is then a place holder where you have to do the mapping by hand.

More Information:
www.beckhoff.com/tf3510

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