Manual

TC3 Power Monitoring

TwinCAT 3

Version:	1.1
Date:	2019-01-03
Order No.:	TF3650

BECKHOFF

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 the 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®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® 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, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

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

Copyright

1.2 Safety instructions

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Description of symbols

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

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

2 Overview

Beckhoff offers various terminals for implementing supply network analysis and monitoring in its I/O portfolio. In a single- or three-phase grid, power measurement terminals such as the EL3403 or EL3413 directly provide RMS values for current and voltage, as well as active, reactive and apparent power. The EL3773 and EL3783 power monitoring terminals can also be used to log the raw data for current and voltage, so that grid events can be detected even more accurately, and various values can be calculated in the controller itself.

The TC3 Power Monitoring function is a PLC library for evaluating raw current and voltage data provided by the power monitoring terminals.

The library provides function blocks for calculating RMS values for current, voltage and power. These can be output as instantaneous or average values. The function block also offers maximum and minimum values. Frequency and frequency spectra can be determined, such as e.g. harmonics in the network and their load in the form of the Total Harmonic Distortion (THD). Furthermore, the library offers function blocks which can be used to determine frequencies and rotary fields.

Product information

The current version of the power monitoring library is available for download from the Beckhoff website. The PLC library offers various algorithms for analyzing current and voltage in a single- or three-phase grid. Some algorithms are based on implementations of the TwinCAT Condition Monitoring library.

Product components

The TF3650 power monitoring product consists of the following components:

PLC libraries	Tc3_PowerMonitoring.compiled-library Tc3_CM_Base.compiled-library Tc3_Filter.compiled-library Tc3_MultiArray.compiled-library
Drivers	TcCM.sys TcFilter.sys TcMultiArray.sys

3 Installation

3.1 System requirements

Engineering system

An engineering system describes a computer that is used for the development of program code and does not execute program code. An engineering system must meet the following requirements:

- TwinCAT 3 XAE (engineering installation) build 4022 or higher
- Installation of TF3650 power monitoring
- A 7-day trial license can be activated repeatedly for the engineering system (see also Licensing [▸ 11])

Runtime system

A runtime system describes an Industrial or Embedded PC on which program code is executed. An runtime system must meet the following requirements:

- TwinCAT 3 XAR (runtime installation) build 4022 or higher
- 32-bit and 64-bit systems are supported
- License for TC1200 PLC and for TF3650 Power Monitoring
- A 7-day trial license can be activated repeatedly for testing purposes.

Engineering and runtime on the same system

To use engineering and runtime on the same system, the following system requirements have to be met:

- TwinCAT3 XAE (engineering installation) build 4022 or higher
- License for TC1200 PLC and for TF3650 Power Monitoring
- A 7-day trial license can be activated repeatedly for testing purposes.

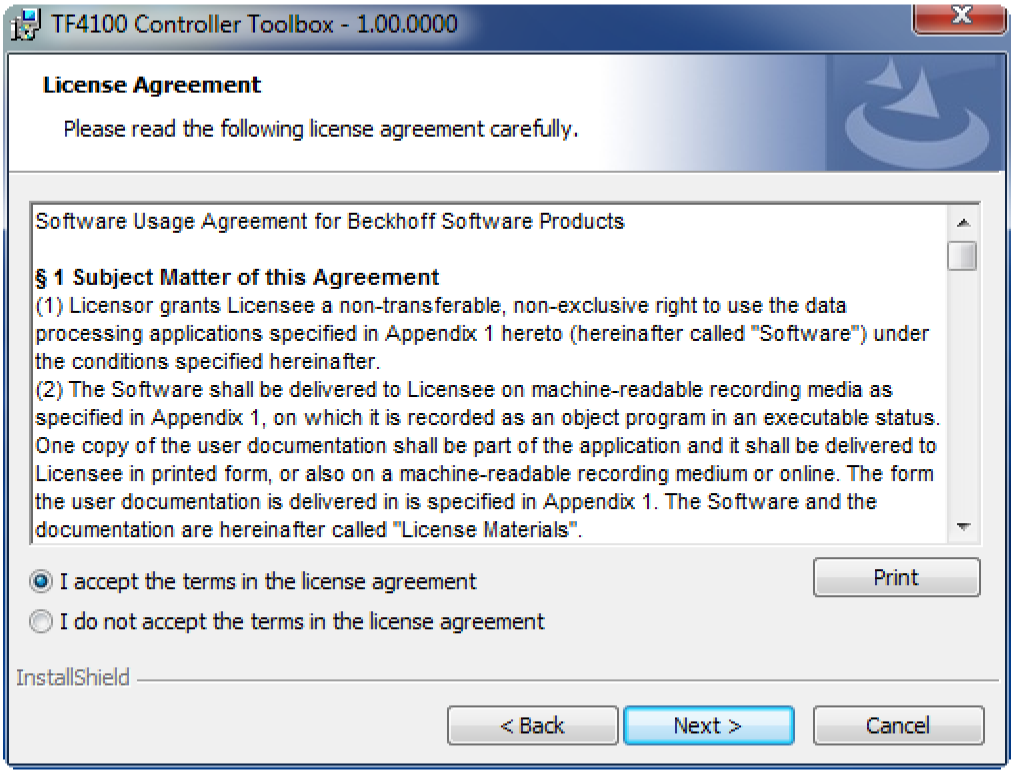
3.2 Installation

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

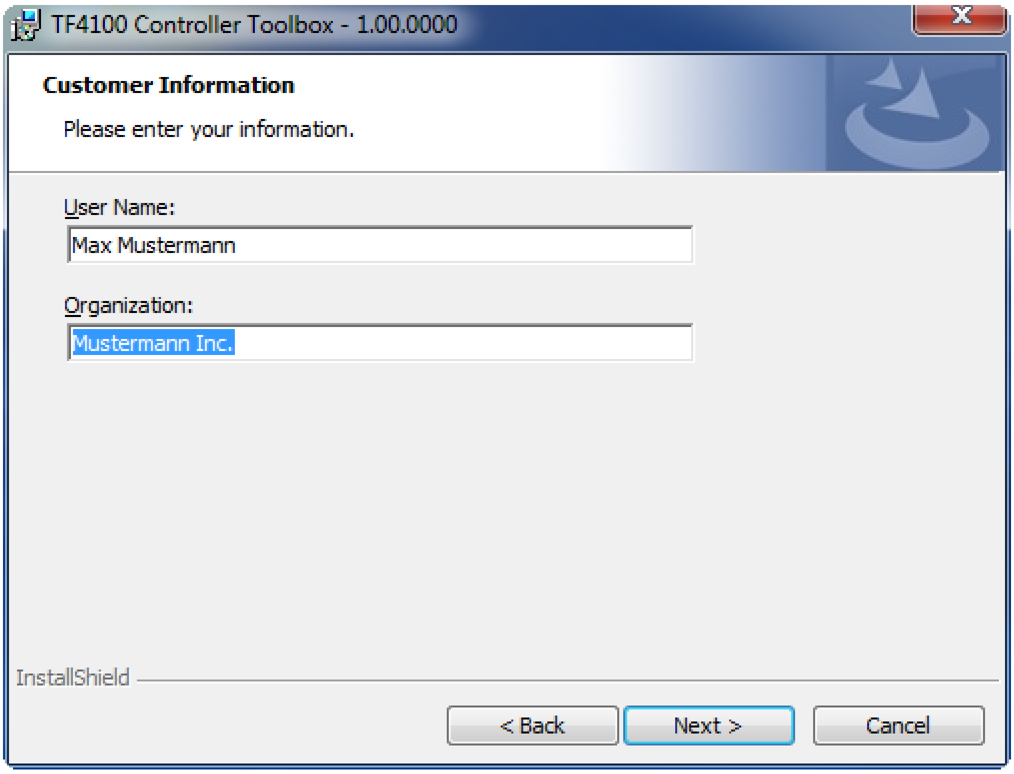
✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.

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.

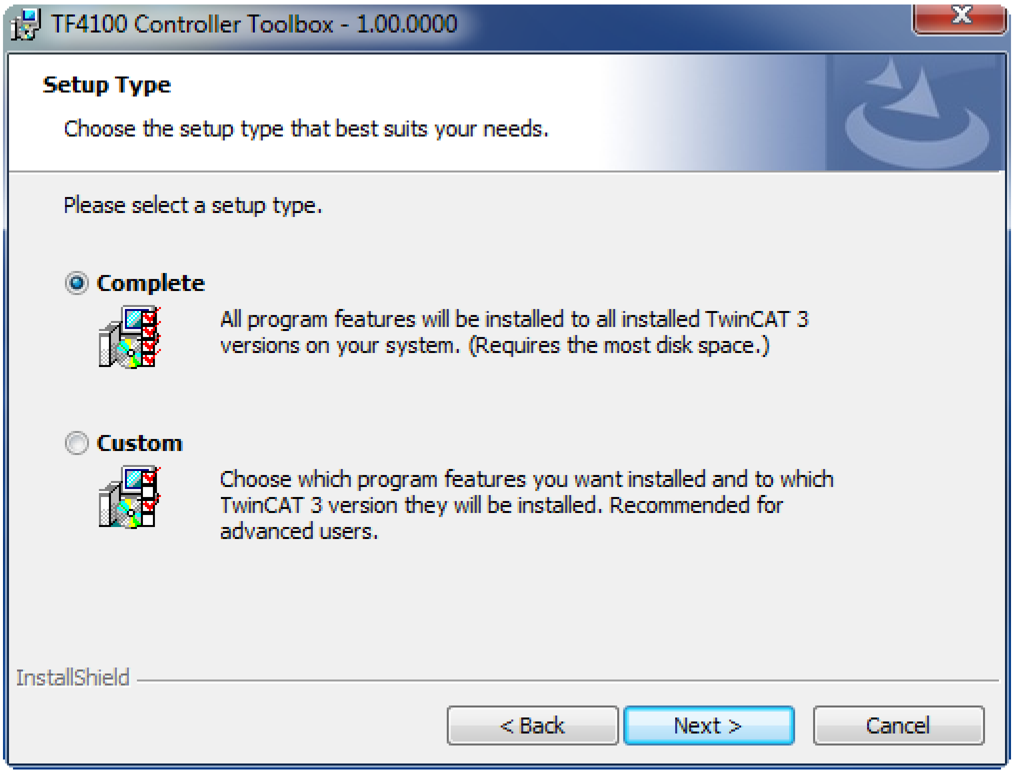
2. Accept the end user licensing agreement and click **Next**.



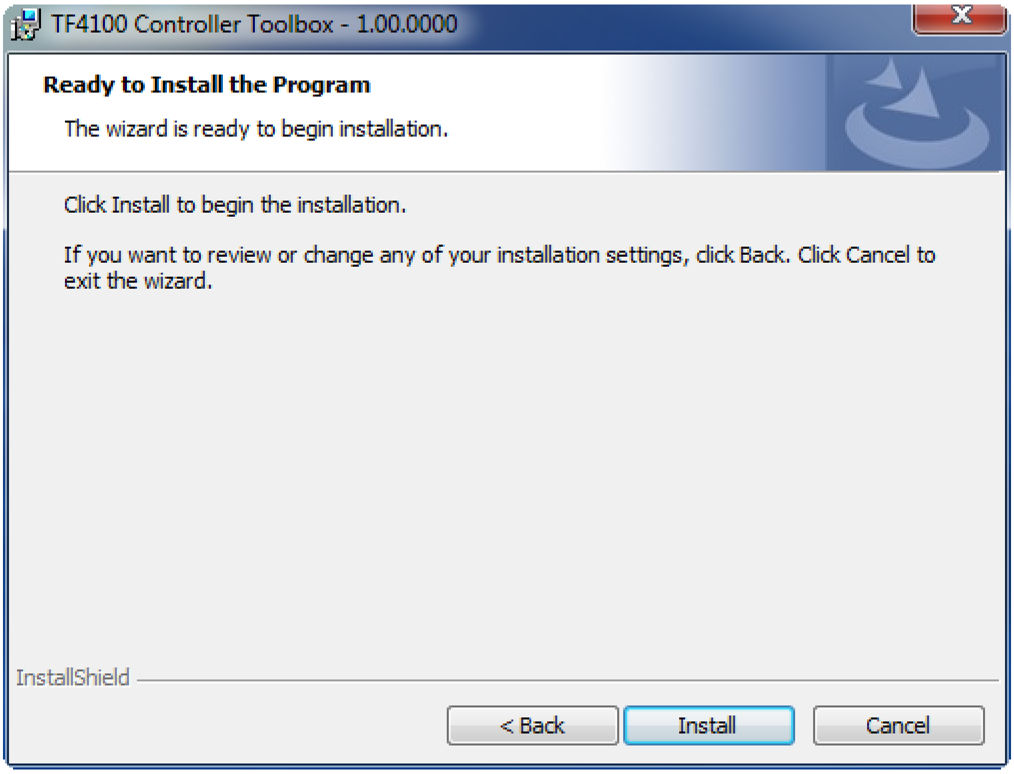
3. Enter your user data.



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**.

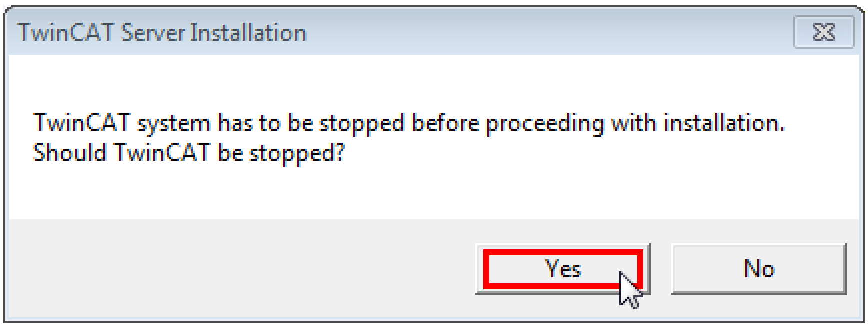


5. Select **Next**, then **Install** to start the installation.

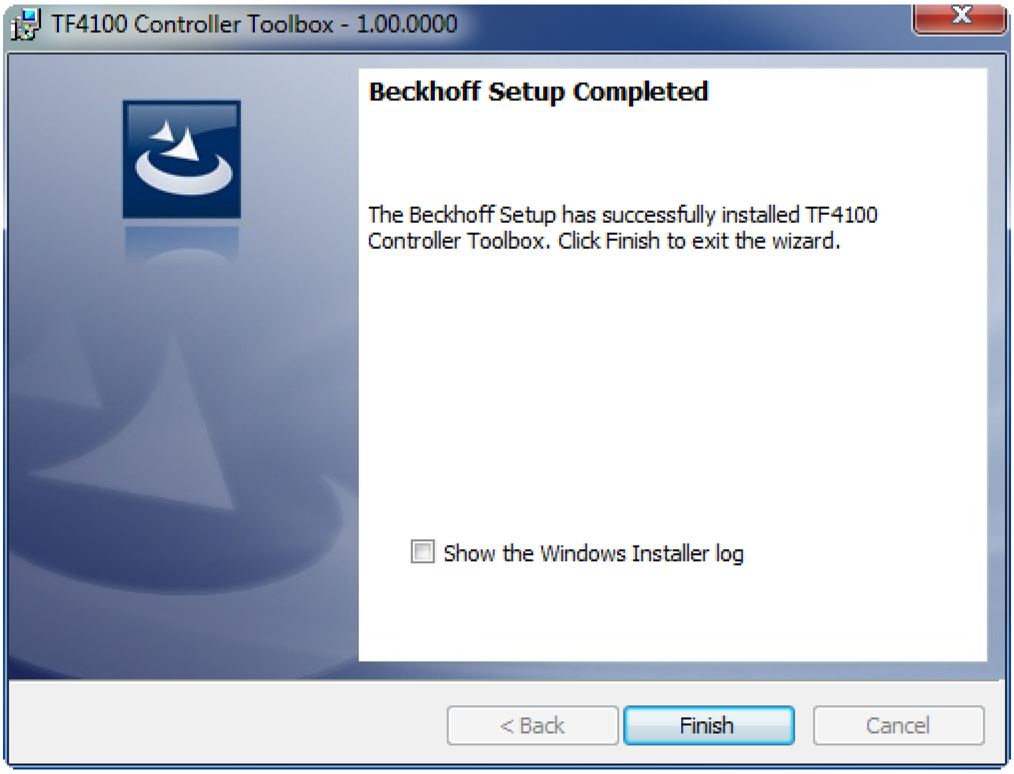


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

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 [▸ 11]).

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).

The licensing of a TwinCAT 3 Function is described below. The description is divided into the following sections:

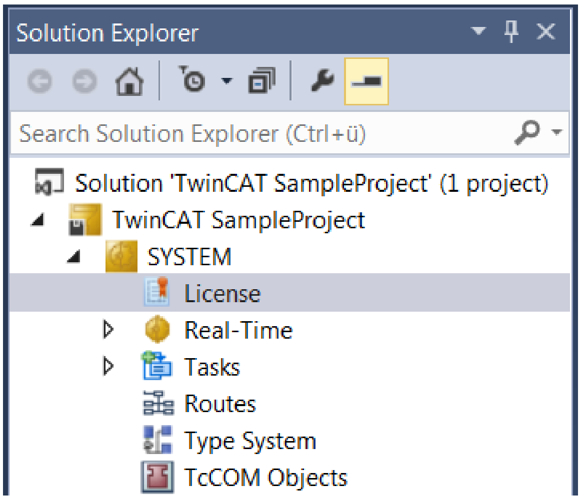
- Licensing a 7-day test version [▸ 11]
- Licensing a full version [▸ 13]

Further information on TwinCAT 3 licensing can be found in the “Licensing” documentation in the Beckhoff Information System (TwinCAT 3 > Licensing).

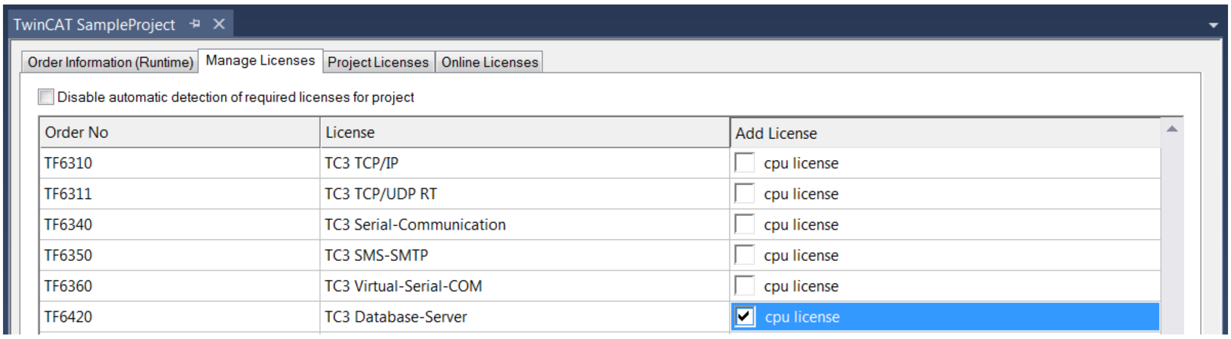
Licensing a 7-day test version

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.

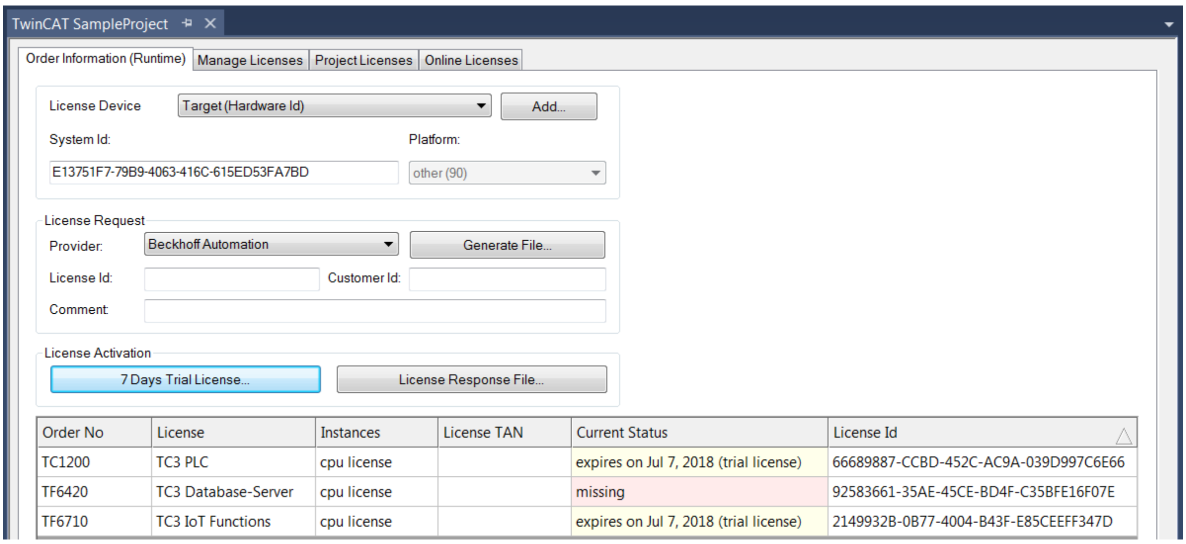
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the 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.



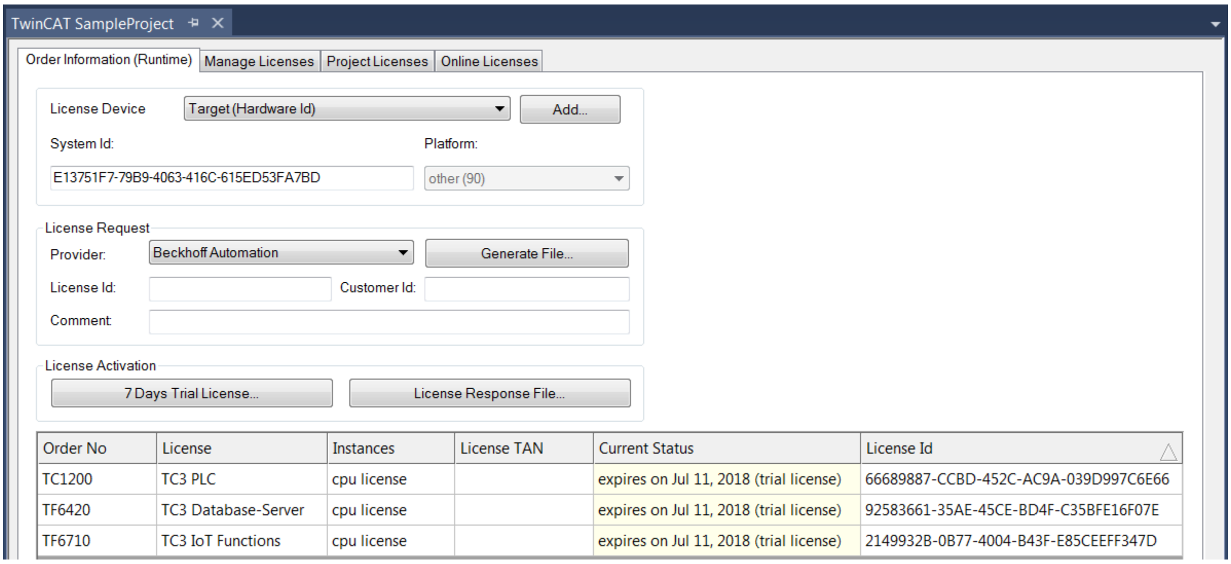
   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. “TF6420: TC3 Database Server“).



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 appears, confirm it and acknowledge the subsequent dialog indicating successful activation.

   ⇨ In the tabular overview of licenses, the license status now indicates the expiration date of the license.

9. Restart the TwinCAT system.

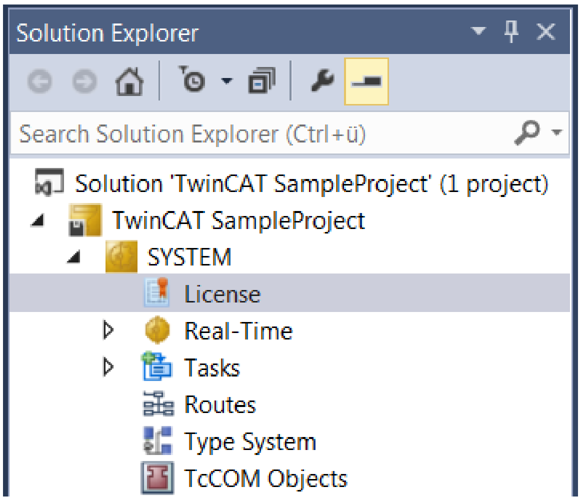
⇨ The 7-day trial version is enabled.

Licensing a full version

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.

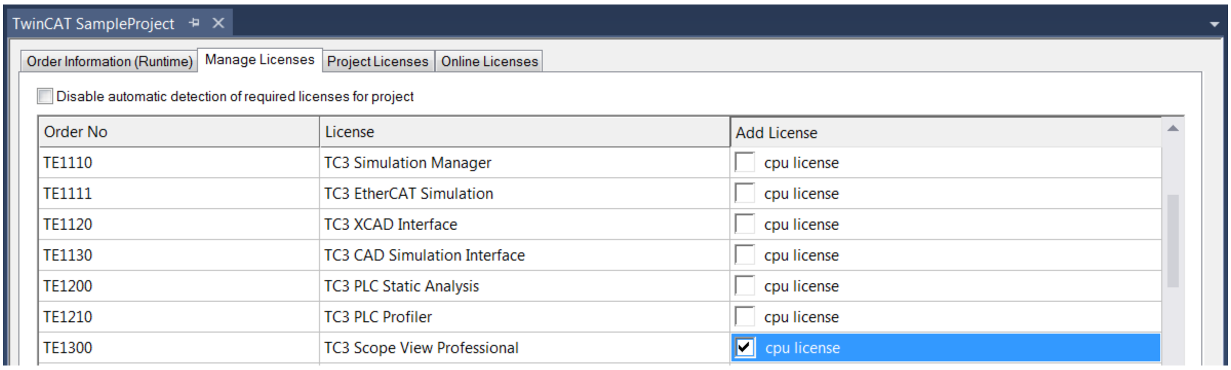
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

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

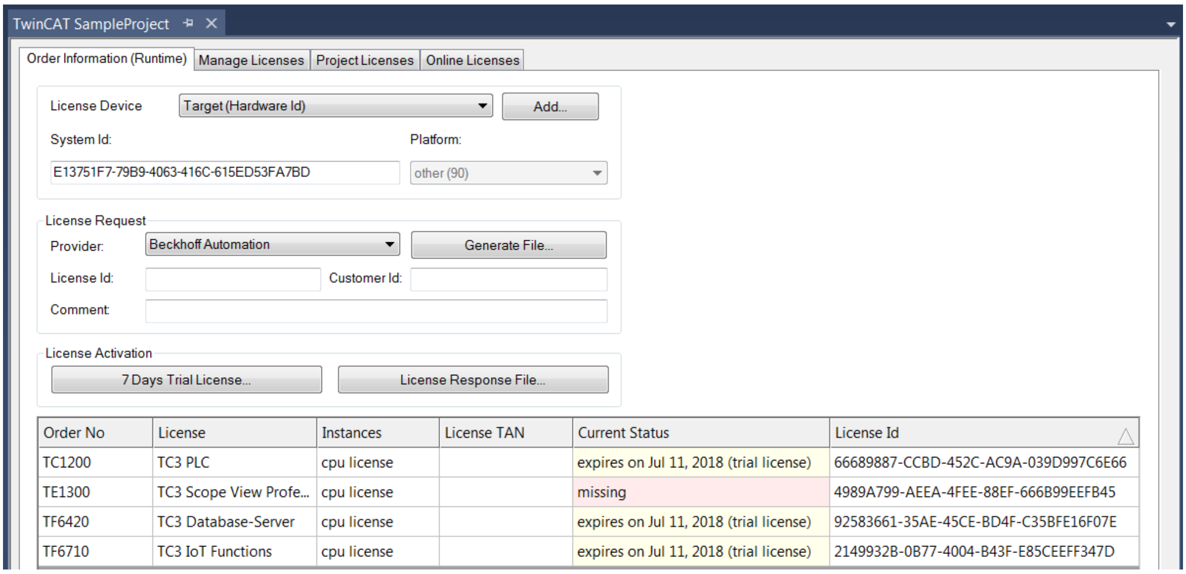


⇨ The TwinCAT 3 license manager opens.

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. “TE1300: TC3 Scope View Professional”).

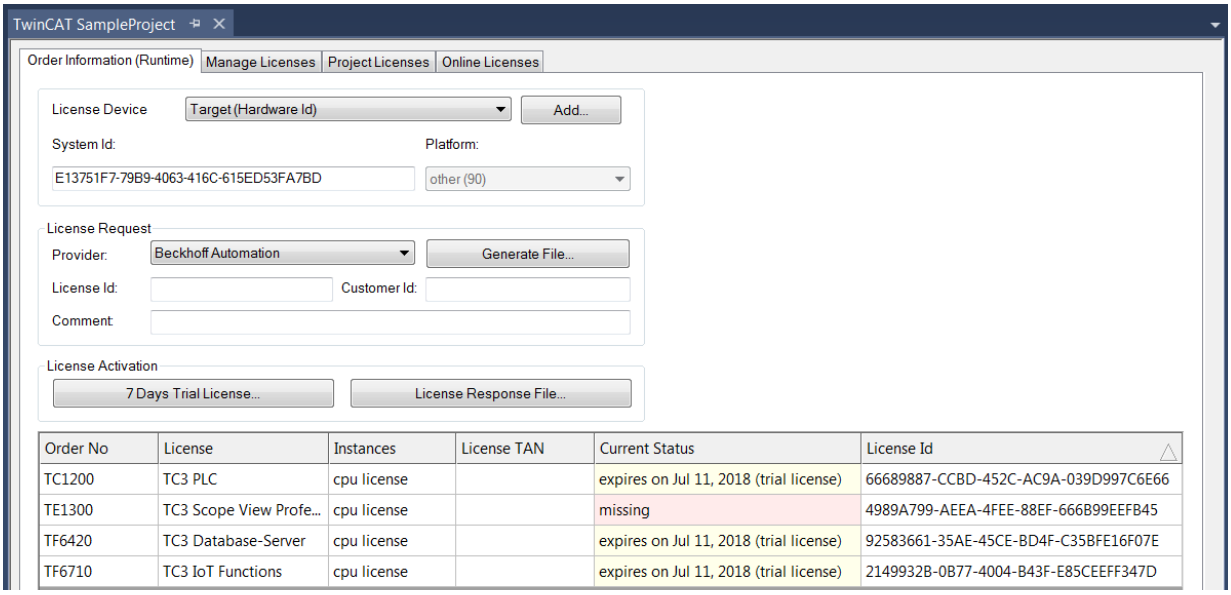


6. Open the **Order Information** tab.
   ⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”**.**

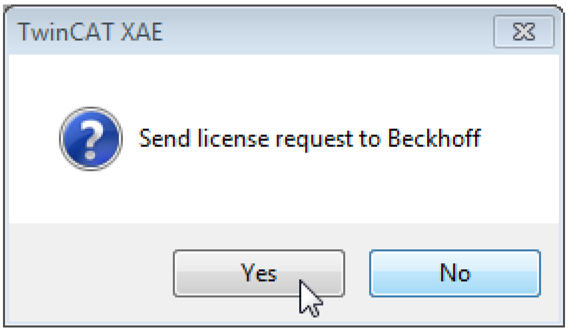


A TwinCAT 3 license is generally linked to two indices describing the platform to be licensed:
System ID: Uniquely identifies the device
Platform level: Defines the performance of the device
The corresponding **System Id** and **Platform** fields cannot be changed.

7. Enter the order number (**License Id**) for the license to be activated and optionally a separate order number (**Customer Id**), plus an optional comment for your own purposes (**Comment**). If you do not know your Beckhoff order number, please contact your Beckhoff sales contact.



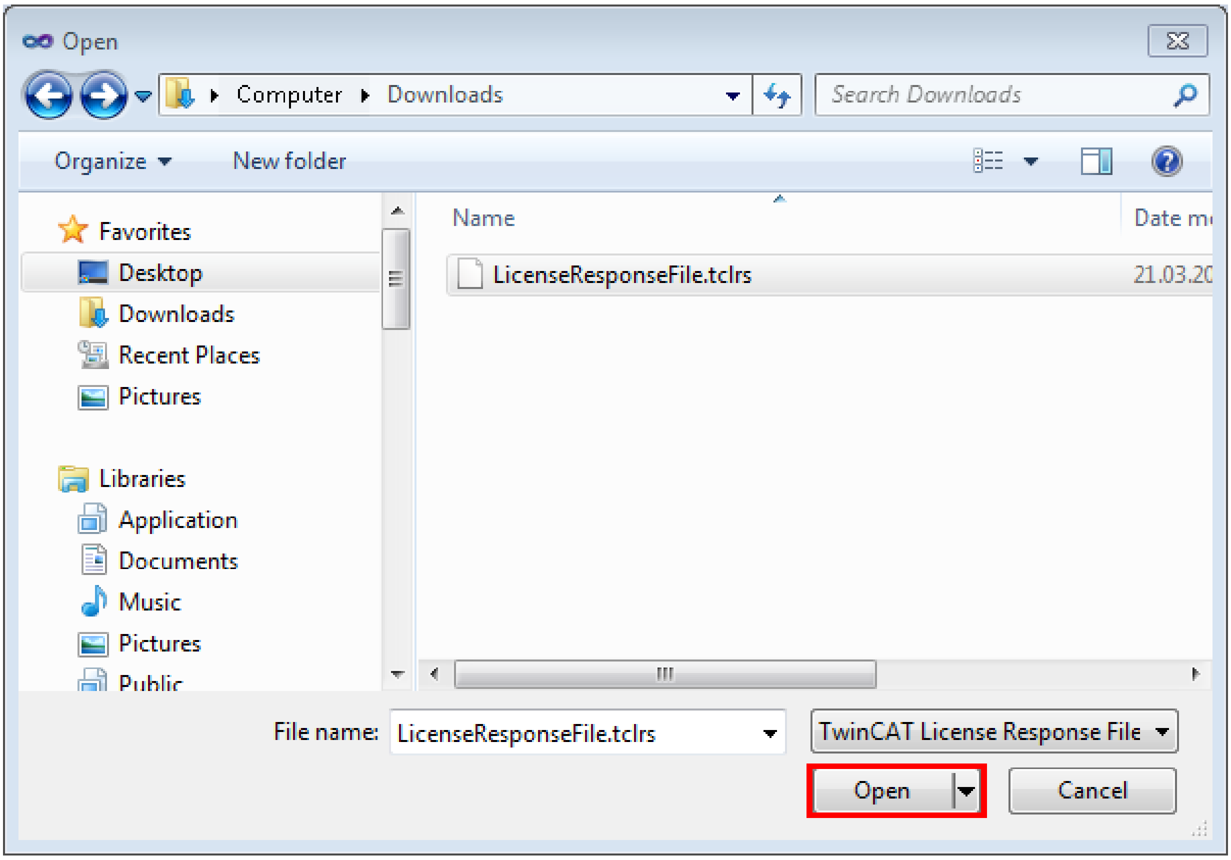
8. Click the **Generate File**... button to create a License Request File for the listed missing license.
   ⇨ A window opens, in which you can specify where the License Request File is to be stored. (We recommend accepting the default settings.)
9. Select a location and click **Save**.
   ⇨ A prompt appears asking whether you want to send the License Request File to the Beckhoff license server for verification:



- Click **Yes** to send the License Request File. A prerequisite is that an email program is installed on your computer and that your computer is connected to the internet. When you click **Yes**, the system automatically generates a draft email containing the License Request File with all the necessary information.
- Click **No** if your computer does not have an email program installed on it or is not connected to the internet. Copy the License Request File onto a data storage device (e.g. a USB stick) and send the file from a computer with internet access and an email program to the Beckhoff license server (tclicense@beckhoff.com) by email.

10. Send the License Request File.
    ⇨ The License Request File is sent to the Beckhoff license server. After receiving the email, the server compares your license request with the specified order number and returns a License Response File by email. The Beckhoff license server returns the License Response File to the same email address from which the License Request File was sent. The License Response File differs from the License Request File only by a signature that documents the validity of the license file content. You can view the contents of the License Response File with an editor suitable for XML files (e.g. “XML Notepad”). The contents of the License Response File must not be changed, otherwise the license file becomes invalid.
11. Save the License Response File.

12. To import the license file and activate the license, click **License Response File...** in the **Order Information** tab.
13. Select the License Response File in your file directory and confirm the dialog.



⇨ The License Response File is imported and the license it contains is activated.
Existing demo licenses will be removed.

14. Restart the TwinCAT system.

⇨ The license becomes active when TwinCAT is restarted. The product can be used as a full version. During the TwinCAT restart the license file is automatically copied to the directory ...*\TwinCAT\3.1\Target\License* on the respective target system.

4 Technical introduction

4.1 Memory Management

The power monitoring library uses parts of the Condition Monitoring library, which internally uses TcCOM objects. The TcCOM objects are provided by the installed drivers. The instances are created dynamically in the TwinCAT AMS router memory.

Necessity for dynamic memory management

All memory requirements and initializations are implemented or performed during the initialization phase. Since the number of elements of the input data and the internal structures depend on the configuration of the respective function blocks, the memory space for them is allocated dynamically as a matter of principle. This happens automatically when the Condition Monitoring library is used.

Since all memory allocations occur during initialization, and therefore the initialization of function blocks may require a relatively large amount of memory under certain circumstances, the initialization at this point may fail due to lack of memory, but not later.
The allocated memory is released again once the object is deleted.

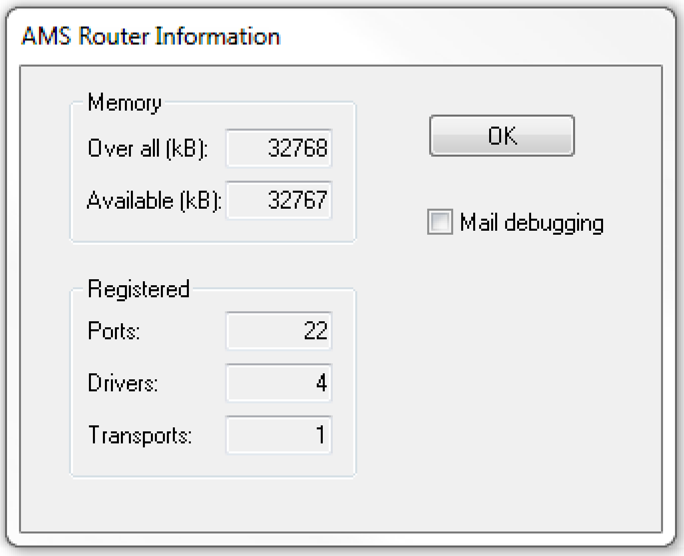
TwinCAT router memory for dynamically generated objects

The buffers reserved by the Condition Monitoring library are created in the TwinCAT AMS router memory when function blocks are initialized, so that they are available for execution under real-time conditions. Certain functions, such as high-resolution histograms and quantiles as well as the calculation of spectra with very high resolutions, require considerably more router memory than conventional control programs. Therefore, the router memory may need to be increased.

Adapting the router memory

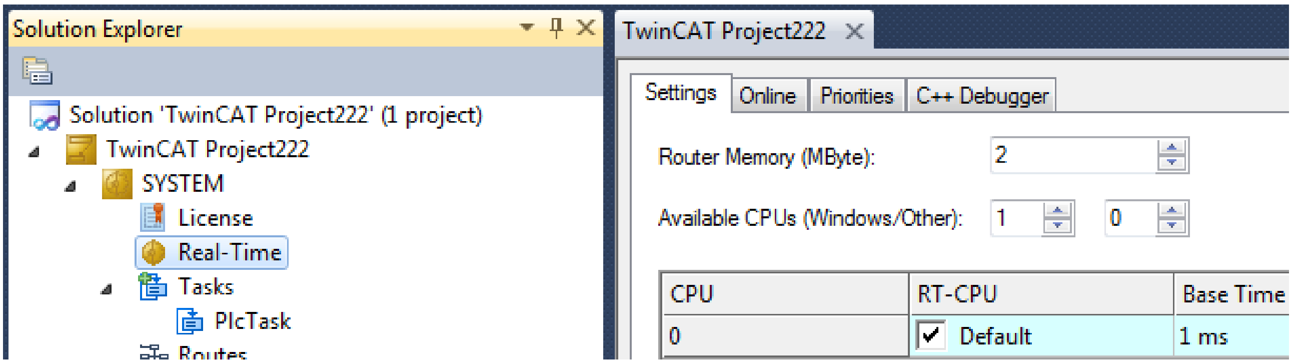
The standard size of the router memory is 32 MB. The current setting is shown in the **AMS Router Information** dialog.

To open the dialog, right-click on the TwinCAT System Service icon in the system tray and select the command **Router > Info** in the system menu that opens.



To increase the router memory, open the real-time settings in TwinCAT Engineering and enter a value in MB in the TwinCAT configuration (**TwinCAT project tree > SYSTEM > Real-Time > Settings tab > Router Memory**). Then activate the configuration.

The adjustment of the router memory necessitates a reboot of the target device.



5 PLC API

5.1 General function blocks

5.1.1 FB_PMA_Scaling

The function block FB_PMA_Scaling is used for scaling raw values. The raw values can be scaled individually or as an array, for example as oversampling values. In addition, it is possible to use single-phase and three-phase input signals.

Alternatively, the specialized function blocks FB_PMA_Scaling_EL3773 [▸ 24] can be used for inputs of the EL3773 EtherCAT Terminal and FB_PMA_Scaling_EL3783 [▸ 27] for inputs of the EL3783 EtherCAT Terminal.

Syntax

Definition:

```
FUNCTION_BLOCK FB_PMA_Scaling
VAR_INPUT
    stInitPars        : ST_PMA_Scaling_InitPars
END_VAR
VAR_OUTPUT
    bError            : BOOL;
    ipResultMessage   : I_TcMessage;
    bNewResult        : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 23] method). They may only be assigned once. A change at runtime is possible by calling the Init [▸ 23] method again.

Name	Type	Description
stInitPars	ST_PMA_Scaling_InitPars [▸ 114]	Function block-specific structure with initialization parameters

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE when new results were calculated.

Methods

Methods can be used to scale different systems (single-phase or three-phase) and different resolutions (16-bit or 32-bit).

Name	Description
Call_1Ph_16Bit [▸ 20]	The method is called to scale the 16-bit input data of type INT according to the configured parameters.
Call_1Ph_32Bit [▸ 20]	The method is called to scale the 32-bit input data of type DINT according to the configured parameters.

Name	Description
Call_3Ph_16Bit [▸ 21]	The method is called to scale the 16-bit input data of type INT according to the configured parameters.
Call_3Ph_32Bit [▸ 22]	The method is called to scale the 32-bit input data of type DINT according to the configured parameters.
Init [▸ 23]	Alternative to the function block initialization
Reconfigure [▸ 23]	The method is called in order to reconfigure the function block during the runtime.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.1.1.1 Call_1Ph_16Bit

The method is called to scale the 16-bit input data of type INT according to the configured parameters.

Syntax

```
METHOD Call_1Ph_16Bit : BOOL
VAR_INPUT
    pInputBuffer_U     : POINTER TO INT;
    pInputBuffer_I     : POINTER TO INT;
    nInputBufferSize   : UDINT;
    pOutputBuffer_U    : POINTER TO LREAL;
    pOutputBuffer_I    : POINTER TO LREAL;
    nOutputBufferSize  : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_U	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_I	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_U	POINTER TO LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_I	POINTER TO LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.

Return value

Name	Type	Description
Call_1Ph_16Bit	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.1.2 Call_1Ph_32Bit

The method is called to scale the 32-bit input data of type DINT according to the configured parameters.

Syntax

```
METHOD Call_1Ph_32Bit : BOOL
VAR_INPUT
    pInputBuffer_U      : POINTER TO DINT;
    pInputBuffer_I      : POINTER TO DINT
    nInputBufferSize    : UDINT;
    pOutputBuffer_U     : POINTER TO LREAL;
    pOutputBuffer_I     : POINTER TO LREAL;
    nOutputBufferSize   : UDINT;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_U	POINTER TO DINT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_I	POINTER TO DINT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_U	POINTER TO LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_I	POINTER TO LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.

Return value

Name	Type	Description
Call_1Ph_32Bit	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.1.3 Call_3Ph_16Bit

The method is called to scale the 16-bit input data of type INT according to the configured parameters.

Syntax

```
METHOD Call_3Ph_16Bit : BOOL
VAR_INPUT
    pInputBuffer_UL1    : POINTER TO INT;
    pInputBuffer_UL2    : POINTER TO INT;
    pInputBuffer_UL3    : POINTER TO INT;
    pInputBuffer_IL1    : POINTER TO INT
    pInputBuffer_IL2    : POINTER TO INT
    pInputBuffer_IL3    : POINTER TO INT
    nInputBufferSize    : UDINT;
    pOutputBuffer_UL1   : POINTER TO LREAL;
    pOutputBuffer_UL2   : POINTER TO LREAL;
    pOutputBuffer_UL3   : POINTER TO LREAL;
    pOutputBuffer_IL1   : POINTER TO LREAL;
    pOutputBuffer_IL2   : POINTER TO LREAL;
    pOutputBuffer_IL3   : POINTER TO LREAL;
    nOutputBufferSize   : UDINT;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_UL1 .. UL3	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_IL1 .. IL3	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_UL1 .. UL3	POINTER TO LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_IL1 .. IL3	POINTER TO LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.

Return value

Name	Type	Description
Call_3Ph_16Bit	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.1.4 Call_3Ph_32Bit

The method is called to scale the 32-bit input data of type DINT according to the configured parameters.

Syntax

```
METHOD Call_3Ph_32Bit : BOOL
VAR_INPUT
    pInputBuffer_UL1    : POINTER TO DINT;
    pInputBuffer_UL2    : POINTER TO DINT;
    pInputBuffer_UL3    : POINTER TO DINT;
    pInputBuffer_IL1    : POINTER TO DINT
    pInputBuffer_IL2    : POINTER TO DINT
    pInputBuffer_IL3    : POINTER TO DINT
    nInputBufferSize    : UDINT;
    pOutputBuffer_UL1   : POINTER TO LREAL;
    pOutputBuffer_UL2   : POINTER TO LREAL;
    pOutputBuffer_UL3   : POINTER TO LREAL;
    pOutputBuffer_IL1   : POINTER TO LREAL;
    pOutputBuffer_IL2   : POINTER TO LREAL;
    pOutputBuffer_IL3   : POINTER TO LREAL;
    nOutputBufferSize   : UDINT;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_UL1 .. UL3	POINTER TO DINT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_IL1 .. IL3	POINTER TO DINT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.

Name	Type	Description
pOutputBuffer_UL1 .. UL3	POINTER TO LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_IL1 .. IL3	POINTER TO LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.

Return value

Name	Type	Description
Call_3Ph_32Bit	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.1.5 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference).

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    stInitPars   : ST_PMA_Scaling_InitPars;
END_VAR
```

Inputs

Name	Type	Description
stInitPars	ST_PMA_Scaling_InitPars [▸ 114]	Function block-specific structure with initialization parameters

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.1.6 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fOffsetVoltage  : LREAL  := 0.0;
    fGainVoltage    : LREAL  := 1.0;
    fOffsetCurrent  : LREAL  := 0.0;
    fGainCurrent    : LREAL  := 1.0;
END_VAR
```

Inputs

Name	Type	Description
fOffsetVoltage	LREAL	Indicates a user-defined offset for the voltage scaling.
fGainVoltage	LREAL	Indicates a user-defined gain factor for the voltage scaling.
fOffsetCurrent	LREAL	Indicates a user-defined offset for the current scaling.
fGainCurrent	LREAL	Specifies a user-defined gain factor for the current scaling.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.2 FB_PMA_Scaling_EL3773

The function block FB_PMA_Scaling_EL3773 is a special version of the function block FB_PMA_Scaling [▶ 19]. It is used for scaling raw values provided by the EL3773 EtherCAT Terminal. The raw values can be scaled individually or as an array, for example as oversampling values. In addition, it is possible to use single-phase and three-phase input signals.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Scaling_EL3773
VAR_INPUT
    stInitPars        : ST_PMA_Scaling_EL3773_InitPars;
END_VAR
VAR_OUTPUT
    bError            : BOOL;
    ipResultMessage   : I_TcMessage;
    bNewResult        : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▶ 26] method). They may only be assigned once. A change at runtime is possible by calling the Init [▶ 26] method again.

Name	Type	Description
stInitPars	ST_PMA_Scaling_EL3773_InitPars [▶ 113]	Function block-specific structure with initialization parameters

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE when new results were calculated.

Methods

Methods can be used to scale different systems (single-phase or three-phase).

Name	Description
Call_1Ph [▸ 25]	The method is called to scale the input data in a single-phase system of type INT according to the configured parameters.
Call_3Ph [▸ 26]	The method is called to scale the input data in a three-phase system of type INT according to the configured parameters.
Init [▸ 26]	Alternative to the function block initialization
Reconfigure [▸ 27]	The method is called in order to reconfigure the function block during the runtime.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.1.2.1 Call_1Ph

The method is called to scale the input data in a single-phase system of type INT according to the configured parameters. LREAL is output.

Syntax

```
METHOD Call_1Ph : BOOL
VAR_INPUT
    pInputBuffer_U     : POINTER TO INT;
    pInputBuffer_I     : POINTER TO INT;
    nInputBufferSize   : UDINT;
    pOutputBuffer_U    : POINTER TO LREAL;
    pOutputBuffer_I    : POINTER TO LREAL;
    nOutputBufferSize  : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_U	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_I	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_U	POINTER_TO_LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_I	POINTER_TO_LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.

Return value

Name	Type	Description
Call_1Ph	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.2.2 Call_3Ph

The method is called to scale the input data in a three-phase system of type INT according to the configured parameters. LREAL is output.

Syntax

```
METHOD Call_3Ph : BOOL
VAR_INPUT
    pInputBuffer_UL1    : POINTER TO INT;
    pInputBuffer_UL2    : POINTER TO INT;
    pInputBuffer_UL3    : POINTER TO INT;
    pInputBuffer_IL1    : POINTER TO INT
    pInputBuffer_IL2    : POINTER TO INT
    pInputBuffer_IL3    : POINTER TO INT
    nInputBufferSize    : UDINT;
    pOutputBuffer_UL1   : POINTER TO LREAL;
    pOutputBuffer_UL2   : POINTER TO LREAL;
    pOutputBuffer_UL3   : POINTER TO LREAL;
    pOutputBuffer_IL1   : POINTER TO LREAL;
    pOutputBuffer_IL2   : POINTER TO LREAL;
    pOutputBuffer_IL3   : POINTER TO LREAL;
    nOutputBufferSize   : UDINT;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_UL1 .. UL3	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_IL1 .. IL3	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_UL1 .. UL3	POINTER_TO_LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_IL1 .. IL3	POINTER_TO_LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.

Return value

Name	Type	Description
Call_3Ph	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.2.3 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference).

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    stInitPars   : ST_PMA_Scaling_EL3773_InitPars;
END_VAR
```

Inputs

Name	Type	Description
stInitPars	ST_PMA_Scaling_EL3773_InitPars [▸ 113]	Function block-specific structure with initialization parameters

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.2.4 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fOffsetVoltage   : LREAL  := 0.0;
    fGainVoltage     : LREAL  := 1.0;
    fOffsetCurrent   : LREAL  := 0.0;
    fGainCurrent     : LREAL  := 1.0;
END_VAR
```

Inputs

Name	Type	Description
fOffsetVoltage	LREAL	Indicates a user-defined offset for the voltage scaling.
fGainVoltage	LREAL	Indicates a user-defined gain factor for the voltage scaling.
fOffsetCurrent	LREAL	Indicates a user-defined offset for the current scaling.
fGainCurrent	LREAL	Specifies a user-defined gain factor for the current scaling.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.3 FB_PMA_Scaling_EL3783

The function block FB_PMA_Scaling_EL3783 is a special version of the function block FB_PMA_Scaling [▸ 19]. It is used for scaling raw values provided by the EL3783 EtherCAT Terminal. The raw values can be scaled individually or as an array, for example as oversampling values. In addition, it is possible to use single-phase and three-phase input signals. The function block supports the Autorange functionality of the EL3783, which can operate in two current measurement ranges.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Scaling_EL3783
VAR_INPUT
    stInitPars       : ST_PMA_Scaling_EL3783_InitPars;
VAR_OUTPUT
    bError           : BOOL;
    ipResultMessage  : I_TcMessage;
    bNewResult       : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▶ 32] method). They may only be assigned once. A change at runtime is possible by calling the Init [▶ 32] method again.

Name	Type	Description
stInitPars	ST_PMA_Scaling_EL3783_InitPars [▶ 113]	Function block-specific structure with initialization parameters

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE when new results were calculated.

Methods

Methods can be used to scale different systems (single-phase or three-phase).

Name	Description
Call_1Ph [▶ 28]	The method is called to scale the input data in a single-phase system of type INT according to the configured parameters.
Call_1Ph_Autorange [▶ 29]	The method is called to scale the input data in a single-phase system of type INT according to the configured parameters.
Call_3Ph [▶ 30]	The method is called to scale the input data in a three-phase system of type INT according to the configured parameters.
Call_3Ph_Autorange [▶ 31]	The method is called to scale the input data in a three-phase system of type INT according to the configured parameters.
Init [▶ 32]	Alternative to the function block initialization
Reconfigure [▶ 33]	The method is called in order to reconfigure the function block during the runtime.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.1.3.1 Call_1Ph

The method is called to scale the input data in a single-phase system of type INT according to the configured parameters. LREAL is output.

Syntax

```
METHOD Call_1Ph : BOOL
VAR_INPUT
    pInputBuffer_U     : POINTER TO INT;
    pInputBuffer_I     : POINTER TO INT;
    nInputBufferSize   : UDINT;
    pOutputBuffer_U    : POINTER TO LREAL;
    pOutputBuffer_I    : POINTER TO LREAL;
    nOutputBufferSize  : UDINT;
    bUse_5A_Range      : BOOL;
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_U	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_I	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_U	POINTER_TO_LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_I	POINTER_TO_LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.
bUse_5A_Range	BOOL	If the value is TRUE, the 5 A measuring range of the EL3783 is used. If it is FALSE, the 1 A measurement range is used.

Return value

Name	Type	Description
Call_1Ph	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.3.2 Call_1Ph_Autorange

The method is called to scale the input data in a single-phase system of type INT according to the configured parameters. LREAL is output. The EL3783 operates in Autorange mode.

Syntax

```
METHOD Call_1Ph_Autorange : BOOL
VAR_INPUT
    pInputBuffer_U          : POINTER TO INT;
    pInputBuffer_I          : POINTER TO INT;
    nInputBufferSize        : UDINT;
    pOutputBuffer_U         : POINTER TO LREAL;
    pOutputBuffer_I         : POINTER TO LREAL;
    nOutputBufferSize       : UDINT;
    bEL3783_HcRangeActive   : BOOL;
    aEL3783_HcRange         : ARRAY [0..3] OF USINT;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_U	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_I	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_U	POINTER_TO_LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_I	POINTER_TO_LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.
bEL3783_HcRangeActive	BOOL	Autorange mode is active at the terminal.
aEL3783_HcRange	ARRAY [0..3] OF USINT	The current measuring range information for the EL3783.

Return value

Name	Type	Description
Call_1Ph_Autorange	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.3.3 Call_3Ph

The method is called to scale the input data in a three-phase system of type INT according to the configured parameters. LREAL is output.

Syntax

```
METHOD Call_3Ph : BOOL
VAR_INPUT
    pInputBuffer_UL1    : POINTER TO INT;
    pInputBuffer_UL2    : POINTER TO INT;
    pInputBuffer_UL3    : POINTER TO INT;
    pInputBuffer_IL1    : POINTER TO INT
    pInputBuffer_IL2    : POINTER TO INT
    pInputBuffer_IL3    : POINTER TO INT
    nInputBufferSize    : UDINT;
    pOutputBuffer_UL1   : POINTER TO LREAL;
    pOutputBuffer_UL2   : POINTER TO LREAL;
    pOutputBuffer_UL3   : POINTER TO LREAL;
    pOutputBuffer_IL1   : POINTER TO LREAL;
    pOutputBuffer_IL2   : POINTER TO LREAL;
    pOutputBuffer_IL3   : POINTER TO LREAL;
    nOutputBufferSize   : UDINT;
    bUse_5A_Range       : BOOL;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_UL1 .. UL3	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_IL1 .. IL3	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_UL1 .. UL3	POINTER_TO_LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_IL1 .. IL3	POINTER_TO_LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.
bUse_5A_Range	BOOL	If the value is TRUE, the 5 A measuring range of the EL3783 is used. If it is FALSE, the 1 A measurement range is used.

Return value

Name	Type	Description
Call_3Ph	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.3.4 Call_3Ph_Autorange

The method is called to scale the input data in a three-phase system of type INT according to the configured parameters. LREAL is output. The EL3783 operates in Autorange mode.

Syntax

```
METHOD Call_3Ph : BOOL
VAR_INPUT
    pInputBuffer_UL1        : POINTER TO INT;
    pInputBuffer_UL2        : POINTER TO INT;
    pInputBuffer_UL3        : POINTER TO INT;
    pInputBuffer_IL1        : POINTER TO INT
    pInputBuffer_IL2        : POINTER TO INT
    pInputBuffer_IL3        : POINTER TO INT
    nInputBufferSize        : UDINT;
    pOutputBuffer_UL1       : POINTER TO LREAL;
    pOutputBuffer_UL2       : POINTER TO LREAL;
    pOutputBuffer_UL3       : POINTER TO LREAL;
    pOutputBuffer_IL1       : POINTER TO LREAL;
    pOutputBuffer_IL2       : POINTER TO LREAL;
    pOutputBuffer_IL3       : POINTER TO LREAL;
    nOutputBufferSize       : UDINT;
    bEL3783_HcRangeActive   : BOOL;
    aEL3783_HcRange         : ARRAY [0..3] OF USINT;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
pInputBuffer_UL1 .. UL3	POINTER TO INT	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pInputBuffer_IL1 .. IL3	POINTER TO INT	Pointer to an array of current values. These can be added individually or as an oversampling array.
nInputBufferSize	UDINT	Indicates the size of a single input buffer in bytes.
pOutputBuffer_UL1 .. UL3	POINTER_TO_LREAL	Pointer to an array in which the scaled voltage values are to be stored.
pOutputBuffer_IL1 .. IL3	POINTER_TO_LREAL	Pointer to an array in which the scaled current values are to be stored.
nOutputBufferSize	UDINT	Indicates the size of a single output buffer in bytes.
bEL3783_HcRangeActive	BOOL	Autorange mode is active at the terminal.
aEL3783_HcRange	ARRAY [0..3] OF USINT	The current measuring range information for the EL3783.

Return value

Name	Type	Description
Call_3Ph_Autorange	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.3.5 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference).

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    stInitPars   : ST_PMA_Scaling_EL3783_InitPars;
END_VAR
```

Inputs

Name	Type	Description
stInitPars	ST_PMA_Scaling_EL3783_InitPars [▶ 113]	Function block-specific structure with initialization parameters

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.3.6 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fOffsetVoltage  : LREAL  := 0.0;
    fGainVoltage    : LREAL  := 1.0;
    fOffsetCurrent  : LREAL  := 0.0;
    fGainCurrent    : LREAL  := 1.0;
END_VAR
```

Inputs

Name	Type	Description
fOffsetVoltage	LREAL	Indicates a user-defined offset for the voltage scaling.
fGainVoltage	LREAL	Indicates a user-defined gain factor for the voltage scaling.
fOffsetCurrent	LREAL	Indicates a user-defined offset for the current scaling.
fGainCurrent	LREAL	Specifies a user-defined gain factor for the current scaling.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.4 FB_PMA_TaskTransfer_Send

The function block FB_PMA_TaskTransfer_Send can be used to transfer data from a slow task back to a fast task. This may be necessary if computationally intensive algorithms were swapped out to a slow task and the results are required in another task.

Together with the function block FB_PMA_TaskTransfer_Receive [▶ 36], it forms a function block pair for sending and receiving data. The data is passed to a function block with the target analysis ID after each call of the Call [▶ 35] method.

The output buffers are provided for the function blocks whose ID is entered in the array of target IDs. The buffer to be initialized must be sufficient to cover the data to be transmitted.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_TaskTransfer_Send
VAR_INPUT
    nOwnID              : UDINT;
    aDestIDs            : ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT;
    nResultBuffers      : UDINT := 4;
    tTransferTimeout    : LTIME := LTIME#40US;
    stInitPars          : ST_PMA_TaskTransfer_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
```

```
    bNewResult          : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 35] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
aDestIDs	ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT	The data of the function block is sent to the IDs of the function block instances of the type specified here as an array FB_PMA_TaskTransfer_Receive [▸ 36].
nResultBuffers	UDINT	Number of multi-array buffers that are initialized for the results.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_TaskTransfer_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been provided.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▸ 35]	The method is called in each cycle to write the values to the output buffer.
Init [▸ 35]	Alternative to the function block initialization

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_TaskTransfer_Send is set, the first value in the buffer is marked as NaN (Not a Number) in this case. The function block FB_PMA_TaskTransfer_Receive queries the status of this first value at the start of the execution. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.1.4.1 Call

The method is called in each cycle to write the values to the output buffer. The output buffer is sent cyclically.

Syntax

```
METHOD CALL : HRESULT
VAR_INPUT
    pInputData   : POINTER TO LREAL,
    nDataInSize  : UDINT;
    nOptionPars  : DWORD;
END_VAR
```

Inputs

Name	Type	Description
pInputData	POINTER TO LREAL	Pointer to an input buffer of the data to be sent.
nDataInSize	UDINT	Indicates the size of the input buffer in bytes.
nOptionPars	DWORD	
• cCMA_Option_MarkInterruption		Several errors can occur and cause interruptions of the time series collection. If the flag is set and the element type is LREAL, the first data buffer element is marked as invalid (NaN). This can be used to detect an interruption in the result data sets, because it is not possible to calculate correct spectra based on fragmented time series. See NaN values.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.4.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : HRESULT
VAR_INPUT
    nOwnID          : UDINT;
    aDestIDs        : ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT;
    nResultBuffers  : UDINT := 4;
    stInitPars      : ST_PMA_TaskTransfer_InitPars,
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
aDestIDs	ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT	The data of the function block is sent to the IDs of the function block instances of the type specified here as an array FB_PMA_TaskTransfer_Receive [▸ 36].
nResultBuffers	UDINT	Number of available multi-arrays
stInitPars	ST_PMA_TaskTransfer_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.5 FB_PMA_TaskTransfer_Receive

The function block FB_PMA_TaskTransfer_Receive can be used to transfer data from a slow task back to a fast task. This may be necessary if computationally intensive algorithms were swapped out to a slow task and the results are required in another task.

The input buffer is provided via the function block FB_PMA_TaskTransfer_Send [▸ 33]. The buffer to be initialized must be sufficient to cover the data to be transmitted.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_TaskTransfer_Receive
VAR_INPUT
    nOwnID              : UDINT
    tTransferTimeout    : LTIME := LTIME#500US
    stInitPars          : ST_PMA_TaskTransfer_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResult          : BOOL;
    bNaNValueOccured    : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 38] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
aDestIDs	ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
nResultBuffers	UDINT	Number of multi-array buffers that are initialized for the results.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_TaskTransfer_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been provided.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▸ 37]	The method is called in each cycle to read the values from the input buffer when new data is present.
Init [▸ 38]	Alternative to the function block initialization

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_TaskTransfer_Send is set, the first value in the buffer is marked as NaN (Not a Number) in this case. The function block FB_PMA_TaskTransfer_Receive queries the status of this first value at the start of the execution. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

5.1.5.1 Call

The method is called in each cycle to read the values from the input buffer when new data is present. The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD CALL : HRESULT
VAR_INPUT
    pOutputData    : POINTER TO LREAL,
    nDataOutSize   : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pOutputData	POINTER TO LREAL	Pointer to the output buffer.
nDataOutSize	UDINT	Indicates the size of the output buffer in bytes.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.1.5.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : HRESULT
VAR_INPUT
    nOwnID       : UDINT;
    stInitPars   : ST_PMA_Source_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_TaskTransfer_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2 Function blocks, single-phase

5.2.1 FB_PMA_Source_1Ph

The function block FB_PMA_Source_1Ph writes data from an external PLC data buffer into a multi-array buffer.

It accumulates input data continuously, until the maximum size of the multi-array is reached. When the multi-array is completely filled, it is passed to a function block with the target analysis ID.

The output buffers are provided for the function blocks whose ID is entered in the array of target IDs. They contain the current and voltage values. The required size of the output buffer may vary depending on the analysis block used. It depends either on the FFT length used or the buffer size.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Source_1Ph
VAR_INPUT
    nOwnID              : UDINT;
    aDestIDs            : ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT;
    nResultBuffers      : UDINT := 4;
    tTransferTimeout    : LTIME := LTIME#40US;
    stInitPars          : ST_PMA_Source_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResults         : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 41] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
aDestIDs	ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
nResultBuffers	UDINT	Number of multi-array buffers that are initialized for the results.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings.
stInitPars	ST_PMA_Source_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.

Name	Type	Description
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been provided.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▸ 40]	The method is called in each cycle to write the values to the output buffer.
ResetData [▸ 41]	This method can be used to reset the data currently in the buffer.
Init [▸ 41]	Alternative to the function block initialization

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.1.1 Call

The method is called in each cycle to write the values to the output buffer. The output buffer is sent as soon as it is filled.

Syntax

```
METHOD CALL : BOOL
VAR_INPUT
    pBuffer_U          : POINTER TO LREAL;
    pBuffer_I          : POINTER TO LREAL;
    nDataInSizePerCh   : UDINT;
    nOptionPars        : DWORD;
END_VAR
```

Inputs

Name	Type	Description
pBuffer_U	POINTER TO LREAL	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pBuffer_I	POINTER TO LREAL	Pointer to an array of current values. These can be added individually or as an oversampling array.
nDataInSizePerCh	UDINT	Indicates the size of a single input buffer in bytes.
nOptionPars	DWORD	

Name	Type	Description
• cCMA_Option_MarkInterruption		Several errors can occur and cause interruptions of the time series collection. If the flag is set, the first data buffer element is marked invalid (NaN). This can be used to detect an interruption in the result data sets, because it is not possible to calculate correct spectra based on fragmented time series. See NaN values.

5.2.1.2 ResetData

This method can be used to reset the data currently in the buffer.

Syntax

```
METHOD ResetData : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
ResetData	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.1.3 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID          : UDINT;
    aDestIDs        : ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT;
    nResultBuffers  : UDINT := 4;
    stInitPars      : ST_PMA_Source_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
aDestIDs	ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT	The result data is forwarded to the IDs of other function block instances specified here as an array.
nResultBuffers	UDINT	Number of available multi-arrays.
stInitPars	ST_PMA_Source_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2 Time-based

5.2.2.1 FB_PMA_BasicValues_Period_1Ph

The function block FB_PMA_BasicValues_Period_1Ph calculates analysis values for the signal sequence of current and voltage in a single-phase system. These include the mean value, the RMS value, the peak value, the rectified value, the crest factor and the form factor for current and voltage. The results refer to a configurable number of signal periods. The period duration refers to the frequency specified at the start of the period at the input of the Call [▸ 45] method. The statistical results refer to the entire runtime or the time at which the statistical results were last reset.

The input buffer is provided via the function block FB_PMA_Source_1Ph [▸ 39]. This can include one or more signal periods or individual fragments of oversampling arrays.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_BasicValues_Period_1Ph
VAR_INPUT
    nOwnID                  : UDINT;
    tTransferTimeout        : LTIME := LTIME#500US;
    stInitPars              : ST_PMA_BasicValues_Period_InitPars;
END_VAR
VAR_OUTPUT
    bError                  : BOOL;
    ipResultMessage         : I_TcMessage;
    bNewResult              : BOOL;
    bNaNValueOccured        : BOOL;
    nCntResults             : ULINT;
    fMeanValue_U            : LREAL;
    fRMS_U                  : LREAL;
    fRMS_U_Min              : LREAL;
    fRMS_U_Max              : LREAL;
    fPeakValue_U            : LREAL;
    fPeakHold_U             : LREAL;
    fRectifiedValue_U       : LREAL;
    fCrestFactor_U          : LREAL;
    fFormFactor_U           : LREAL;
    fMeanValue_I            : LREAL,
    fRMS_I                  : LREAL,
    fRMS_I_Min              : LREAL,
    fRMS_I_Max              : LREAL,
    fPeakValue_I            : LREAL,
    fPeakHold_I             : LREAL;
    fRectifiedValue_I       : LREAL;
    fCrestFactor_I          : LREAL;
    fFormFactor_I           : LREAL;
    bValidStatistics        : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 81] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_BasicValues_Period_InitPars [▸ 116]	Function block-specific structure with initialization parameters. The parameters must correlate to the above definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
fMeanValue_U	LREAL	Mean voltage value over n periods
fRMS_U	LREAL	RMS value of the voltage over n periods
fRMS_U_Min	LREAL	Smallest value of fRMS_U that has occurred. Can be reset via bResetStatistics of the Call method.
fRMS_U_Max	LREAL	Largest value of fRMS_U that has occurred. Can be reset via bResetStatistics of the Call method.
fPeakValue_U	LREAL	Peak voltage value over n periods
fPeakHold_U	LREAL	All-time peak voltage value over n periods. Can be reset via bResetStatistics of the Call method.
fRectifiedValue_U	LREAL	Rectified voltage value over n periods
fCrestFactor_U	LREAL	Crest factor of the voltage (peak/RMS value)
fFormFactor_U	LREAL	Form factor of the voltage (RMS/rectified value)
fMeanValue_I	LREAL	Mean value of the current over n periods
fRMS_I	LREAL	RMS value of the current over n periods
fRMS_I_Min	LREAL	Smallest value of fRMS_I that has occurred. Can be reset via bResetStatistics of the Call method.
fRMS_I_Max	LREAL	Largest value of fRMS_I that has occurred. Can be reset via bResetStatistics of the Call method.
fPeakValue_I	LREAL	Peak value of the current over n periods
fPeakHold_I	LREAL	All-time peak current value. Can be reset via bResetStatistics of the Call method.
fRectifiedValue_I	LREAL	Rectified value of the current over n periods
fCrestFactor_I	LREAL	Crest factor of current (peak/RMS value)
fFormFactor_I	LREAL	Form factor of the current (RMS/rectified value)
bValidStatistics	BOOL	TRUE if the minimum, maximum and hold value calculation has been performed. These values are valid.

Methods

Name	Description
Call [▶ 45]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▶ 81]	Alternative to the function block initialization
PassInputs [▶ 82]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▶ 46]	The method is called in order to reconfigure the function block during the runtime.
Reset [▶ 46]	The current calculations are reset with the method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.2.1.1 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID       : UDINT;
    stInitPars   : ST_PMA_BasicValues_Period_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_BasicValues_Period_InitPars [▶ 116]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.1.2 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    fFreq               : LREAL;
    bResetStatistics    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
fFreq	LREAL	Current frequency of the input signal. Is used to determine the length of the period at the beginning of a period. The output of the function block FB_CMA_Frequency_Period_1Ph [▸ 51] can be used.
bResetStatistics	BOOL	TRUE resets the minimum, maximum and hold values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.1.3 PassInputs

As long as an instance of the function block FB_PMA_Source_1Ph [▸ 39] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.1.4 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinInputCurrent    : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.1.5 Reset

The current calculations are reset with the method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.2 FB_PMA_PowerValues_Period_1Ph

The function block FB_PMA_PowerValues_Period_1Ph calculates the power values of the connected consumer. These include the fundamental components and the phase shift angle. For this purpose, only the first harmonics of the input signal are used for the calculation in addition to the signal sequence in the time domain. The advantage of these algorithms is the high dynamics of the calculations. The results refer to a configurable number of signal periods. The period duration refers to the frequency specified at the start of the period at the input of the Call [▸ 49] method. The statistical results refer to the entire runtime or the time at which the statistical results were last reset.

Alternatively, the function block FB_PMA_PowerValues_1Ph [▸ 60] can be used. This uses the individual harmonics internally to calculate the power values.

The input buffer is provided via the function block FB_PMA_Source_1Ph [▸ 39]. This can include one or more signal periods or individual fragments of oversampling values.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_PowerValues_Period_1Ph
VAR_INPUT
    nOwnID                  : UDINT;
    tTransferTimeout        : LTIME := LTIME#500US;
    stInitPars              : ST_PMA_PowerValues_Period_InitPars;
END_VAR
VAR_OUTPUT
    bError                  : BOOL;
    ipResultMessage         : I_TcMessage;
    bNewResult              : BOOL;
    bNaNValueOccured        : BOOL;
    nCntResults             : ULINT;
    fApparentPower          : LREAL;
    fApparentPower_1        : LREAL;
    fApparentPower_1_Min    : LREAL;
    fApparentPower_1_Max    : LREAL;
    fActivePower            : LREAL;
    fActivePower_Min        : LREAL;
    fActivePower_Max        : LREAL;
    fReactivePower_d        : LREAL;
    fReactivePower_1        : LREAL;
    fReactivePower_1_Min    : LREAL;
    fReactivePower_1_Max    : LREAL;
    fTotalReactivePower     : LREAL;
    fPhi                    : LREAL;
    fCosPhi                 : LREAL;
    fPowerFactor            : LREAL;
    bValidStatistics        : BOOL;
END_VAR
VAR_OUTPUT PERSISTENT
    stEnergy_Pos            : ST_PMA_Energy;
    stEnergy_Neg            : ST_PMA_Energy;
    stEnergy_Res            : ST_PMA_Energy;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 50] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_PowerValues_Period_InitPars [▸ 117]	Function block-specific structure with initialization parameters. The parameters must correlate to the above definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
fApparentPower	LREAL	Total apparent power
fApparentPower_1	LREAL	Fundamental apparent power
fApparentPower_1_Min	LREAL	Smallest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fApparentPower_1_Max	LREAL	Largest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fActivePower	LREAL	Active power
fActivePower_Min	LREAL	Smallest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.
fActivePower_Max	LREAL	Largest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.
fReactivePower_1	LREAL	Fundamental shift reactive power
fReactivePower_1_Min	LREAL	Smallest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fReactivePower_1_Max	LREAL	Largest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fTotalReactivePower	LREAL	Total reactive power
fPhi	LREAL	Phase shift angle
fCosPhi	LREAL	CosPhi (active power/fundamental apparent power)
fPowerFactor	LREAL	Power factor (active power/total apparent power)
bValidStatistics	BOOL	TRUE if the minimum and maximum value calculation has been performed. These values are valid.
stEnergy_Pos	ST_PMA_Energy [▸ 128]	Energy in the positive direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
stEnergy_Neg	ST_PMA_Energy [▸ 128]	Energy in the negative direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
stEnergy_Res	ST_PMA_Energy [▸ 128]	Resulting energy. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.

Methods

Name	Description
Call [▸ 49]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.

Name	Description
Init [▶ 50]	Alternative to the function block initialization
PassInputs [▶ 50]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▶ 51]	The method is called in order to reconfigure the function block during the runtime.
Reset [▶ 51]	The current calculations are reset with the method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.2.2.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    fFreq                : LREAL;
    bResetEnergyCalc     : LREAL;
    bResetStatistics     : BOOL;
END_VAR
```

Inputs

Name	Type	Description
fFreq	LREAL	Current frequency of the input signal. Is used to determine the length of the period at the beginning of a period. The output of the function block FB_CMA_Frequency_Period_1Ph [▶ 51] can be used.
bResetEnergyCalc	LREAL	TRUE resets the calculated values of the energy measurement.
bResetStatistics	BOOL	TRUE resets the minimum and maximum values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.2.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID       : UDINT
    stInitPars   : ST_PMA_PowerValues_Period_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_PowerValues_Period_InitPars [▸ 117]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.2.3 PassInputs

As long as an instance of the function block FB_PMA_Source_1Ph [▸ 39] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.2.4 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinInputCurrent    : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.2.5 Reset

The current calculations are reset with the method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.3 FB_PMA_Frequency_Period_1Ph

The function block FB_PMA_Frequency_Period_1Ph calculates the base frequency of the given input signal. To do this, the signal is first filtered with a Butterworth low-pass filter. The zero crossings of the input signal are then determined from the filtered values, and the frequency is calculated from their difference. The results refer to one or more periods, depending on the configuration. The statistical results refer to the entire runtime or the time at which the statistical results were last reset.

The input buffer is provided via the function block FB_PMA_Source_1Ph [▶ 39]. This can include one or more signal periods or individual fragments of oversampling arrays.

Syntax

Definition:

```
FUNCTION_BLOCK FB_PMA_Frequency_Period_1Ph
VAR_INPUT
    nOwnID              : UDINT;
```

```
    tTransferTimeout     : LTIME := LTIME#500US;
    stInitPars           : ST_PMA_Frequency_Period_InitPars;
END_VAR
VAR_OUTPUT
    bError               : BOOL;
    ipResultMessage      : I_TcMessage;
    bNewResult           : BOOL;
    bNaNValueOccured     : BOOL;
    nCntResults          : ULINT;
    fFreq                : LREAL;
    fFreq_Min            : LREAL;
    fFreq_Max            : LREAL;
    bValidStatistics     : BOOL;
    bOutOfRange          : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 54] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_Frequency_Period_InitPars [▸ 116]	Function block-specific structure with initialization parameters. The parameters must correlate to the above definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
fFreq	LREAL	Frequency determined by two or more zero crossings.
fFreq_Min	LREAL	Smallest value of fFreq that has occurred. Can be reset via bResetStatistics of the Call method.
fFreq_Max	LREAL	Largest value of fFreq that has occurred. Can be reset via bResetStatistics of the Call method.
bValidStatistics	BOOL	TRUE if the minimum and maximum value calculation has been performed. These values are valid.
bOutOfRange	BOOL	TRUE as soon as the input value or the frequency is not within the configured limits.

Methods

Name	Description
Call [▸ 53]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 54]	Alternative to the function block initialization
PassInputs [▸ 54]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 55]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 55]	The current calculations are reset with the method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.2.3.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    bResetStatistics   : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bResetStatistics	BOOL	TRUE resets the minimum and maximum values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.3.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID      : UDINT;
    stInitPars  : ST_PMA_Frequency_Period_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Frequency_Period_InitPars [▶ 116]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.3.3 PassInputs

As long as an instance of the function block FB_PMA_Source_1Ph [▶ 39] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.3.4 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinFreq        : LREAL;
    fMaxFreq        : LREAL;
    nPeriods        : UDINT;
    nFilterOrder    : UINT;
    fCutoff         : LREAL;
    eInputSelect    : E_PMA_InputSelect;
    fMinInput       : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinFreq	LREAL	Minimum expected measuring frequency
fMaxFreq	LREAL	Maximum expected measuring frequency
nPeriods	UDINT	Number of periods that influence the calculation. (Period length = sample rate/frequency)
nFilterOrder	UINT	Indicates the order of the low-pass filter. The stability of the filter must be considered for the setting. Only values up to the tenth order are allowed.
fCutoff	LREAL	Specifies the limit frequency of the low-pass filter.
eInputSelect	E_PMA_InputSelect [▶ 127]	Here you can configure whether the frequency of the voltage or the current should be calculated.
fMinInput	LREAL	Minimum input value (RMS) over one period. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.2.3.5 Reset

The current calculations are reset with the method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3 Frequency-based

5.2.3.1 FB_PMA_Harmonics_1Ph

The function block FB_PMA_Harmonics_1Ph calculates the RMS bands of the individual current and voltage harmonics. In addition, the calculated RMS bands are used to calculate the THD of the input variables.

The input buffer is provided via the function block FB_PMA_Source_1Ph [▶ 39]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^{9}	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

Since the Welch method is used, in each case the current input buffer together with the last transferred buffer is used for the calculation.

The frequency analysis takes step changes in the time series into account. In order to achieve a correct result, the last two input buffers should therefore be consecutive without step changes.

Syntax

Definition:

```
FUCNTION BLOCK FB_PMA_Harmonics_1Ph
VAR_INPUT
    nOwnID                : UDINT;
    tTransferTimeout      : LTIME := LTIME#500US;
    stInitPars            : ST_PMA_Harmonics_InitPars;
END_VAR
VAR_OUTPUT
    bError                : BOOL;
    ipResultMessage       : I_TcMessage;
    bNewResult            : BOOL;
    bNaNValueOccured      : BOOL;
    nCntResults           : ULINT;
    fTHD_U                : LREAL;
    fTHD_U_Min            : LREAL;
    fTHD_U_Max            : LREAL;
    fTHD_I                : LREAL;
    fTHD_I_Min            : LREAL;
    fTHD_I_Max            : LREAL;
    bValidStatistics      : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▶ 58] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.

Name	Type	Description
stInitPars	ST_PMA_Harmonics_InitPars [▸ 118]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
fTHD_U	LREAL	THD of the voltage. The output is in percent.
fTHD_U_Min	LREAL	Smallest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.
fTHD_U_Max	LREAL	Largest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.
fTHD_I	LREAL	THD of the current. The output is in percent.
fTHD_U_Min	LREAL	Smallest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.
fTHD_U_Max	LREAL	Largest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.
bValidStatistics	BOOL	TRUE if the minimum, maximum and hold value calculation has been performed. These values are valid.

Methods

Name	Description
Call [▸ 58]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 58]	Alternative to the function block initialization
PassInputs [▸ 59]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 59]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 60]	This method deletes all the data sets already added. In addition, the calculated output values are reset.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.3.1.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pHarmonicsRMS_U   : POINTER TO LREAL;
    pHarmonicsRMS_I   : POINTER TO LREAL;
    nHarmonicsRMSSize : UDINT;
    bResetStatistics  : BOOL;
END_VAR
```

Inputs

Name	Type	Description
pHarmonicsRMS_U	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: Number of harmonics. If the individual harmonics are not to be output, the input can be set to 0.
pHarmonicsRMS_I	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: Number of harmonics. If the individual harmonics are not to be output, the input can be set to 0.
nHarmonicsRMSSize	UDINT	Indicates the size of an output array for the harmonics.
bResetStatistics	BOOL	TRUE resets the minimum, maximum and hold values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.1.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID      : UDINT;
    stInitPars  : ST_PMA_Harmonics_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Harmonics_InitPars [▸ 118]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.1.3 PassInputs

As long as an instance of the function block FB_PMA_Source_1Ph [▸ 39] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.1.4 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fBaseFreq    : LREAL;
    fBandwidth   : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fBaseFreq	LREAL	Frequency of the first harmonic.
fBandwidth	LREAL	Total bandwidth of each RMS band.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.1.5 Reset

This method deletes all the data sets already added. In addition, the calculated output values are reset.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.2 FB_PMA_PowerValues_1Ph

The function block FB_PMA_PowerValues_1Ph calculates the power values of the connected consumer. These include the fundamental components and the phase shift angle. Internally, the individual harmonics and their phase angle are determined for the calculations.

Alternatively, the function block FB_PMA_PowerValues_Period_1Ph [▸ 46] can be used for the calculation. It uses simpler calculation methods for enhanced dynamics.

The input buffer is provided via the function block FB_PMA_Source_1Ph [▸ 39]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^9	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

Since the Welch method is used, in each case the current input buffer together with the last transferred buffer is used for the calculation.

The frequency analysis takes step changes in the time series into account. In order to achieve a correct result, the last two input buffers should therefore be consecutive without step changes.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_PowerValues_1Ph
VAR_INPUT
    nOwnID                  : UDINT;
    tTransferTimeout        : LTIME := LTIME#500US;
    stInitPars              : ST_PMA_PowerValues_InitPars;
END_VAR
VAR_OUTPUT
    bError                  : BOOL;
    ipResultMessage         : I_TcMessage;
    bNewResult              : BOOL;
    bNaNValueOccured        : BOOL;
    nCntResults             : ULINT;
    fApparentPower          : LREAL;
    fApparentPower_1        : LREAL;
    fApparentPower_1_Min    : LREAL;
    fApparentPower_1_Max    : LREAL;
    fActivePower            : LREAL;
    fActivePower_Min        : LREAL;
    fActivePower_Max        : LREAL;
    fReactivePower_d        : LREAL;
    fReactivePower_1        : LREAL;
    fReactivePower_1_Min    : LREAL;
    fReactivePower_1_Max    : LREAL;
    fTotalReactivePower     : LREAL;
    fPhi                    : LREAL;
    fCosPhi                 : LREAL;
    fPowerFactor            : LREAL;
    bValidStatistics        : BOOL;
END_VAR
VAR_OUTPUT PERSISTENT
    stEnergy_Pos            : ST_PMA_Energy;
    stEnergy_Neg            : ST_PMA_Energy;
    stEnergy_Res            : ST_PMA_Energy;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▶ 63] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_PowerValues_InitPars [▶ 119]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.

Name	Type	Description
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
fApparentPower	LREAL	Total apparent power
fApparentPower_1	LREAL	Fundamental apparent power
fApparentPower_1_Min	LREAL	Smallest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fApparentPower_1_Max	LREAL	Largest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fActivePower	LREAL	Active power
fActivePower_Min	LREAL	Smallest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.
fActivePower_Max	LREAL	Largest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.
fReactivePower_1	LREAL	Fundamental shift reactive power
fReactivePower_1_Min	LREAL	Smallest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fReactivePower_1_Max	LREAL	Largest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
fTotalReactivePower	LREAL	Total reactive power
fPhi	LREAL	Phase shift angle
fCosPhi	LREAL	CosPhi (active power/fundamental apparent power)
fPowerFactor	LREAL	Power factor (active power/total apparent power)
bValidStatistics	BOOL	TRUE if the minimum and maximum value calculation has been performed. These values are valid.
stEnergy_Pos	ST_PMA_Energy [▸ 128]	Energy in the positive direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
stEnergy_Neg	ST_PMA_Energy [▸ 128]	Energy in the negative direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
stEnergy_Res	ST_PMA_Energy [▸ 128]	Resulting energy. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.

Methods

Name	Description
Call [▸ 63]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 63]	Alternative to the function block initialization
PassInputs [▸ 64]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 65]	The method is called in order to reconfigure the function block during the runtime.

Name	Description
Reset [▶ 64]	This method deletes all the data sets already added. In addition, the calculated output values are reset.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.3.2.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    bResetEnergyCalc   : BOOL;
    bResetStatistics   : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bResetEnergyCalc	BOOL	TRUE resets the calculated values of the energy measurement.
bResetStatistics	BOOL	TRUE resets the minimum and maximum values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.2.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID      : UDINT;
    stInitPars  : ST_PMA_PowerValues_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_PowerValues_InitPars [▸ 119]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.2.3 PassInputs

As long as an instance of the function block FB_PMA_Source_1Ph [▸ 39] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.2.4 Reset

This method deletes all the data sets already added. In addition, the calculated output values are reset.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.2.5 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fBaseFreq        : LREAL;
    fBandwidth       : LREAL;
    fMinInputCurrent : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fBaseFreq	LREAL	Frequency of the first harmonic.
fBandwidth	LREAL	Total bandwidth of each RMS band.
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.3 FB_PMA_Spectrum_1Ph

The function block FB_PMA_Spectrum_1Ph calculates the magnitude spectra of the current and voltage values. These are suitable for analyzing the input signals in the frequency range.

The input buffer is provided via the function block FB_PMA_Source_1Ph [▸ 39]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^9	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

Since the Welch method is used, in each case the current input buffer together with the last transferred buffer is used for the calculation.

The frequency analysis takes step changes in the time series into account. In order to achieve a correct result, the last two input buffers should therefore be consecutive without step changes.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Spectrum_1Ph
VAR_INPUT
    nOwnID              : UDINT;
    tTransferTimeout    : LTIME := LTIME#500US;
    stInitPars          : ST_PMA_Spectrum_InitPars;
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResult          : BOOL;
    bNaNValueOccured    : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▶ 68] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_Spectrum_InitPars [▶ 120]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▶ 67]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▶ 68]	Alternative to the function block initialization

Name	Description
PassInputs [▸ 68]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reset [▸ 69]	This method deletes all the data sets already added.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.3.3.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pMagnitudeSpectrum_U    : POINTER TO LREAL;
    pMagnitudeSpectrum_I    : POINTER TO LREAL;
    nMagnitudeSpectrumSize  : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pMagnitudeSpectrum_U	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
pMagnitudeSpectrum_I	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
nMagnitudeSpectrumSize	UDINT	Indicates the size of the output array of a spectrum.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.3.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID      : UDINT;
    stInitPars  : ST_PMA_Spectrum_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Spectrum_InitPars [▸ 120]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.3.3 PassInputs

As long as an instance of the function block FB_PMA_Source_1Ph [▸ 39] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.3.4 Reset

This method deletes all the data sets already added.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.4 FB_PMA_Spectrum_Quantiles_1Ph

The function block FB_PMA_Spectrum_Quantiles_1Ph calculates the magnitude spectra of the current and voltage values like the function block FB_PMA_Spectrum_1Ph [▸ 65]. In addition, p-quantiles of the spectrum distribution can be calculated. The quantiles and their number can be configured individually.

The input buffer is provided via the function block FB_PMA_Source_1Ph [▸ 39]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^9	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

The function block takes into account all input values since the instantiation. If the Reset [▸ 74] method has been called since the start, all input values since its last call will be taken into account.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Spectrum_Quantiles_1Ph
VAR_INPUT
    nOwnID              : UDINT;
    tTransferTimeout    : LTIME := LTIME#500US;
    stInitPars          : ST_PMA_Spectrum_Quantiles_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResult          : BOOL;
    bNaNValueOccured    : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 72] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_Spectrum_Quantiles_InitPars [▸ 121]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▸ 71]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
CallEx [▸ 73]	To minimize CPU usage, it may be necessary to use the CallEx method. In contrast to the Call method, the quantiles are not calculated after each spectrum calculation, but only after a configurable number of calculations.
Init [▸ 72]	Alternative to the function block initialization
PassInputs [▸ 72]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 74]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 74]	This method deletes all the data sets already added. Alternatively, automatic resetting can be used on the CallEx method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_1Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.2.3.4.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

The CallEx [▸ 73] method may be suitable as an alternative. It calculates the quantiles only after a defined number of spectrum calculation results, in order to minimize CPU usage.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pMagnitudeSpectrum_U    : POINTER TO LREAL;
    pMagnitudeSpectrum_I    : POINTER TO LREAL;
    nMagnitudeSpectrumSize  : UDINT
    pSpectrumQuantiles_U    : POINTER TO LREAL;
    pSpectrumQuantiles_I    : POINTER TO LREAL;
    nSpectrumQuantilesSize  : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pMagnitudeSpectrum_U	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
pMagnitudeSpectrum_I	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
nMagnitudeSpectrumSize	UDINT	Indicates the size of the output array of a spectrum.
pSpectrumQuantiles_U	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
pSpectrumQuantiles_I	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
nSpectrumQuantilesSize	UDINT	Indicates the size of the output array for a quantile calculation.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.4.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID      : UDINT;
    stInitPars  : ST_PMA_Spectrum_Quantiles_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Spectrum_Quantiles_InitPars [▸ 121]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.4.3 PassInputs

As long as an instance of the function block FB_PMA_Source_1Ph [▸ 39] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.4.4 CallEx

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

In contrast to the Call [▸ 71] method, the CallEx method collects a variable number of results from the spectrum calculation and calculates the quantiles only afterwards. This may be required to minimize CPU load.

Syntax

```
METHOD CallEx : BOOL
VAR_INPUT
    nAppendData              : UDINT;
    bResetData               : BOOL;
    pMagnitudeSpectrum_U     : POINTER TO LREAL;
    pMagnitudeSpectrum_I     : POINTER TO LREAL;
    nMagnitudeSpectrumSize : UDINT
    pSpectrumQuantiles_U     : POINTER TO LREAL;
    pSpectrumQuantiles_I     : POINTER TO LREAL;
    nSpectrumQuantilesSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nAppendData	UDINT	Number of spectra to be calculated until the quantile is calculated. A value of 1 means that the quantiles are calculated after each result of the spectrum calculation.
bResetData	BOOL	Automatic resetting of records after each quantile calculation
pMagnitudeSpectrum_U	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
pMagnitudeSpectrum_I	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
nMagnitudeSpectrumSize	UDINT	Indicates the size of the output array of a spectrum.
pSpectrumQuantiles_U	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
pSpectrumQuantiles_I	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
nSpectrumQuantilesSize	UDINT	Indicates the size of the output array for a quantile calculation.

Return value

Name	Type	Description
CallEx	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.4.5 Reset

This method deletes all the data sets already added. Alternatively, automatic resetting can be used on the CallEx [▶ 73] method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.2.3.4.6 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinBinned      : LREAL;
    fMaxBinned      : LREAL;
    nBins           : UDINT;
    nNumQuantiles   : UDINT;
    aQuantiles      : ARRAY[0..GVL_PMA.cMaxQuantiles - 1] OF LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinBinned	LREAL	Lower limit value for which the output data of the range calculation in the regular histogram bins are counted.
fMaxBinned	LREAL	Upper limit value for which the output data of the range calculation are counted in the regular histogram bins. fMaxBinned must be larger than the fMinBinned
nBins	UDINT	Number of bins in a histogram. The minimum number is 1. In many cases, values between ten and twenty are a sensible choice. The two special bins for values that lie below fMinBinned or above fMaxBinned are not included in this value.
nNumQuantiles	UDINT	Number of quantiles to calculate for each channel. This must be an integer greater than zero.
aQuantiles	ARRAY[0..GVL_PMA.cMaxQuantiles-1] OF LREAL	Indicates the quantile limit. It must be between 0.0 and 1.0. For example, 0.2 corresponds to the 20% quantile.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3 Function blocks, three-phase

5.3.1 FB_PMA_Source_3Ph

The function block FB_PMA_Source_3Ph writes data from an external PLC data buffer into a multi-array buffer.

It accumulates input data continuously, until the maximum size of the multi-array is reached. When the multi-array is completely filled, it is passed to a function block with the target analysis ID.

The output buffers are provided for the function blocks whose ID is entered in the array of target IDs. They contain the current and voltage values. The requirements for the size of the output buffers may vary depending on the analysis function block used. They depends either on the FFT-length used or the buffer size.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Source_3Ph
VAR_INPUT
    nOwnID              : UDINT;
    aDestIDs            : ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT;
    nResultBuffers      : UDINT := 4;
    tTransferTimeout    : LTIME := LTIME#40US;
    stInitPars          : ST_PMA_Source_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResults         : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 77] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
aDestIDs	ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
nResultBuffers	UDINT	Number of multi-array buffers that are initialized for the results.

Name	Type	Description
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray).
stInitPars	ST_PMA_Source_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been provided.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▸ 76]	The method is called in each cycle to write the values to the output buffer.
ResetData [▸ 77]	This method can be used to reset the data currently in the buffer.
Init [▸ 77]	Alternative to the function block initialization

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.1.1 Call

The method is called in each cycle to write the values to the output buffer. The output buffer is sent as soon as it is filled.

Syntax

```
METHOD CALL : BOOL
VAR_INPUT
    pBuffer_UL1         : POINTER TO LREAL;
    pBuffer_UL2         : POINTER TO LREAL;
    pBuffer_UL3         : POINTER TO LREAL;
    pBuffer_IL1         : POINTER TO LREAL;
    pBuffer_IL2         : POINTER TO LREAL;
    pBuffer_IL3         : POINTER TO LREAL;
    nDataInSizePerCh    : UDINT;
    nOptionPars         : DWORD;
END_VAR
```

Inputs

Name	Type	Description
pBuffer_UL1 .. UL3	POINTER TO LREAL	Pointer to an array of voltage values. These can be added individually or as an oversampling array.
pBuffer_IL1 .. IL3	POINTER TO LREAL	Pointer to an array of current values. These can be added individually or as an oversampling array.
nDataInSizePerCh	UDINT	Indicates the size of a single input buffer in bytes.
nOptionPars	DWORD	
• cCMA_Option_MarkInterruption		Several errors can occur and cause interruptions of the time series collection. If the flag is set and the element type is LREAL, the first data buffer element is marked as invalid (NaN). This can be used to detect an interruption in the result data sets, because it is not possible to calculate correct spectra based on fragmented time series. See NaN values.

5.3.1.2 ResetData

This method can be used to reset the data currently in the buffer.

Syntax

```
METHOD ResetData : BOOL
VAR_INPUT
ENC_VAR
```

Return value

Name	Type	Description
ResetData	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.1.3 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID          : UDINT;
    aDestIDs        : ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT;
    nResultBuffers  : UDINT := 4;
    stInitPars      : ST_PMA_Source_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
aDestIDs	ARRAY[1..GVL_PMA.cMA_MaxDest] OF UDINT	The result data is forwarded to the IDs of other function block instances specified here as an array.
nResultBuffers	UDINT	Number of available multi-arrays.
stInitPars	ST_PMA_Source_InitPars [▸ 115]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2 Time-based

5.3.2.1 FB_PMA_BasicValues_Period_3Ph

The function block FB_PMA_BasicValues_Period_3Ph calculates analysis values for the signal sequence of current and voltage in a three-phase system. These include the mean value, the RMS value, the peak value, the rectified value, the crest factor and the form factor for the individual currents and voltages. In addition, the voltage values between the individual phases are calculated. The results refer to a configurable number of signal periods. The period duration refers to the frequency specified at the start of the period at the input of the Call [▸ 80] method. The statistical results refer to the entire runtime or the time at which the statistical results were last reset.

The input buffer is provided via the function block FB_PMA_Source_3Ph [▸ 75]. This can include one or more signal periods or individual fragments of oversampling arrays.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_BasicValues_Period_3Ph
VAR_INPUT
    nOwnID                  : UDINT;
    tTransferTimeout        : LTIME := LTIME#500US;
    stInitPars              : ST_PMA_BasicValues_Period_InitPars;
END_VAR
VAR_OUTPUT
    bError                  : BOOL;
    ipResultMessage         : I_TcMessage;
    bNewResult              : BOOL;
    bNaNValueOccured        : BOOL;
    nCntResults             : ULINT;
    aMeanValue_U            : ARRAY[0..2] OF LREAL;
    aRMS_U                  : ARRAY[0..2] OF LREAL;
    aRMS_U_Min              : ARRAY[0..2] OF LREAL;
    aRMS_U_Max              : ARRAY[0..2] OF LREAL;
    aRMS_U_PP               : ARRAY[0..2] OF LREAL
    aPeakValue_U            : ARRAY[0..2] OF LREAL;
    aPeakHold_U             : ARRAY[0..2] OF LREAL;
    aRectifiedValue_U       : ARRAY[0..2] OF LREAL;
    aCrestFactor_U          : ARRAY[0..2] OF LREAL;
```

```
    aFormFactor_U          : ARRAY[0..2] OF LREAL;
    aMeanValue_I           : ARRAY[0..2] OF LREAL,
    aRMS_I                 : ARRAY[0..2] OF LREAL,
    aRMS_I_Min             : ARRAY[0..2] OF LREAL,
    aRMS_I_Max             : ARRAY[0..2] OF LREAL,
    aPeakValue_I           : ARRAY[0..2] OF LREAL,
    aPeakHold_I            : ARRAY[0..2] OF LREAL;
    aRectifiedValue_I      : ARRAY[0..2] OF LREAL;
    aCrestFactor_I         : ARRAY[0..2] OF LREAL;
    aFormFactor_I          : ARRAY[0..2] OF LREAL;
    bValidStatistics       : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 81] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_BasicValues_Period_InitPars [▸ 116]	Function block-specific structure with initialization parameters. The parameters must correlate to the above definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
aMeanValue_U	ARRAY[0..2] OF LREAL	Mean voltage value over n periods
aRMS_U	ARRAY[0..2] OF LREAL	RMS value of the voltage over n periods
aRMS_U_Min	ARRAY[0..2] OF LREAL	Smallest value of fRMS_U that has occurred. Can be reset via bResetStatistics of the Call method.
aRMS_U_Max	ARRAY[0..2] OF LREAL	Largest value of fRMS_U that has occurred. Can be reset via bResetStatistics of the Call method.
aRMS_UPP	ARRAY[0..2] OF LREAL	RMS value of the voltage between two phases. Index 0: L1-L2, Index 1: L2-L3, Index 2: L3-L1
aPeakValue_U	ARRAY[0..2] OF LREAL	Peak voltage value over n periods
aPeakHold_U	ARRAY[0..2] OF LREAL	All-time peak voltage value over n periods. Can be reset via bResetStatistics of the Call method.
aRectifiedValue_U	ARRAY[0..2] OF LREAL	Rectified voltage value over n periods
aCrestFactor_U	ARRAY[0..2] OF LREAL	Crest factor of the voltage (peak/RMS value)
aFormFactor_U	ARRAY[0..2] OF LREAL	Form factor of the voltage (RMS/rectified value)
aMeanValue_I	ARRAY[0..2] OF LREAL	Mean value of the current over n periods
aRMS_I	ARRAY[0..2] OF LREAL	RMS value of the current over n periods

Name	Type	Description
aRMS_I_Min	ARRAY[0..2] OF LREAL	Smallest value of fRMS_I that has occurred. Can be reset via bResetStatistics of the Call method.
aRMS_I_Max	ARRAY[0..2] OF LREAL	Largest value of fRMS_I that has occurred. Can be reset via bResetStatistics of the Call method.
aPeakValue_I	ARRAY[0..2] OF LREAL	Peak value of the current over n periods
aPeakHold_I	ARRAY[0..2] OF LREAL	All-time peak current value. Can be reset via bResetStatistics of the Call method.
aRectifiedValue_I	ARRAY[0..2] OF LREAL	Rectified value of the current over n periods
aCrestFactor_I	ARRAY[0..2] OF LREAL	Crest factor of current (peak/RMS value)
aFormFactor_I	ARRAY[0..2] OF LREAL	Form factor of the current (RMS/rectified value)
bValidStatistics	BOOL	TRUE if the minimum, maximum and hold value calculation has been performed. These values are valid.

Methods

Name	Description
Call [▶ 80]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▶ 81]	Alternative to the function block initialization
PassInputs [▶ 82]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▶ 82]	The method is called in order to reconfigure the function block during the runtime.
Reset [▶ 82]	The current calculations are reset with the method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.2.1.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    fFreq              : LREAL;
    bResetStatistics   : BOOL;
END_VAR
```

Inputs

Name	Type	Description
fFreq	LREAL	Current frequency of the input signal. Is used to determine the length of the period at the beginning of a period. The output of the function block FB_PMA_Frequency_Period_3Ph [▶ 83] can be used.
bResetStatistics	BOOL	TRUE resets the minimum, maximum and hold values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.1.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID      : UDINT;
    stInitPars  : ST_PMA_BasicValues_Period_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_BasicValues_Period_InitPars [▶ 116]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.1.3 PassInputs

As long as an instance of the function block FB_PMA_Source_3Ph [▸ 75] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.1.4 Reset

The current calculations are reset with the method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.1.5 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinInputCurrent   : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.2 FB_PMA_Frequency_Period_3Ph

The function block FB_PMA_Frequency_Period_3Ph calculates the base frequency of the given input signal for all three phases. To do this, the signal is first filtered with a Butterworth low-pass filter. The zero crossings of the input signal are then determined from the filtered values, and the frequency is calculated from their difference. The results refer to one or more periods, depending on the configuration. The statistical results refer to the entire runtime or the time at which the statistical results were last reset.

The input buffer is provided via the function block FB_PMA_Source_3Ph [▸ 75]. This can include one or more signal periods or individual fragments of oversampling arrays.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Frequency_Period_3Ph
VAR_INPUT
    nOwnID              : UDINT;
    tTransferTimeout    : LTIME := LTIME#500US;
    stInitPars          : ST_PMA_Frequency_Period_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResult          : BOOL;
    bNaNValueOccured    : BOOL;
    nCntResults         : ULINT;
    aFreq               : ARRAY[0..2] OF LREAL;
    aFreq_Min           : ARRAY[0..2] OF LREAL;
    aFreq_Max           : ARRAY[0..2] OF LREAL;
    bValidStatistics    : BOOL;
    aOutOfRange         : ARRAY[0..2] OF  BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 85] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_Frequency_Period_InitPars [▸ 116]	Function block-specific structure with initialization parameters. The parameters must correlate to the above definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
aFreq	ARRAY[0..2] OF LREAL	Frequency determined by two or more zero crossings.
aFreq_Min	ARRAY[0..2] OF LREAL	Smallest value of fFreq that has occurred. Can be reset via bResetStatistics of the Call method.
aFreq_Max	ARRAY[0..2] OF LREAL	Largest value of fFreq that has occurred. Can be reset via bResetStatistics of the Call method.
bValidStatistics	BOOL	TRUE if the minimum and maximum value calculation has been performed. These values are valid.
aOutOfRange	ARRAY[0..2] OF	TRUE as soon as the input value or the frequency is not within the configured limits.

Methods

Name	Description
Call [▸ 84]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 85]	Alternative to the function block initialization
PassInputs [▸ 86]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 86]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 86]	The current calculations are reset with the method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.2.2.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    bResetStatistics   : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bResetStatistics	BOOL	TRUE resets the minimum and maximum values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.2.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID      : UDINT;
    stInitPars  : ST_PMA_Frequency_Period_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Frequency_Period_InitPars [▶ 116]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.2.3 PassInputs

As long as an instance of the function block FB_PMA_Source_3Ph [▸ 75] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.2.4 Reset

The current calculations are reset with the method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.2.5 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinFreq       : LREAL;
    fMaxFreq       : LREAL;
    nPeriods       : UDINT;
    nFilterOrder   : UINT;
    fCutoff        : LREAL;
    eInputSelect   : E_PMA_InputSelect;
    fMinInput      : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinFreq	LREAL	Minimum expected measuring frequency

Name	Type	Description
fMaxFreq	LREAL	Maximum expected measuring frequency
nPeriods	UDINT	Number of periods that influence the calculation. (Period length = sample rate/frequency)
nFilterOrder	UINT	Indicates the order of the low-pass filter. The stability of the filter must be considered for the setting. Only values up to the tenth order are allowed.
fCutoff	LREAL	Specifies the limit frequency of the low-pass filter.
eInputSelect	E_PMA_InputSelect [▸ 127]	Here you can configure whether the frequency of the voltage or the current should be calculated.
fMinInput	LREAL	Minimum input value (RMS) over one period. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.3 FB_PMA_PowerValues_Period_3Ph

The function block FB_PMA_PowerValues_Period_3Ph calculates the power values of the connected consumer in a three-phase grid. These include the fundamental components and the phase shift angle. For this purpose, only the first harmonics of the input signal are used for the calculation in addition to the signal sequence in the time domain. The advantage of these algorithms is the high dynamics of the calculations. The results refer to a configurable number of signal periods. The period duration refers to the frequency specified at the start of the period at the input of the Call [▸ 84] method. The statistical results refer to the entire runtime or the time at which the statistical results were last reset.

Alternatively, the function block FB_PMA_PowerValues_3Ph [▸ 97] can be used. This uses the individual harmonics internally to calculate the power values.

The input buffer is provided via the function block FB_PMA_Source_3Ph [▸ 75]. This can include one or more signal periods or individual fragments of oversampling values.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_PowerValues_Period_3Ph
VAR_INPUT
    nOwnID                  : UDINT;
    tTransferTimeout        : LTIME := LTIME#500US;
    stInitPars              : ST_PMA_PowerValues_Period_InitPars;
END_VAR
VAR_OUTPUT
    bError                  : BOOL;
    ipResultMessage         : I_TcMessage;
    bNewResult              : BOOL;
    bNaNValueOccured        : BOOL;
    nCntResults             : ULINT;
    aApparentPower          : ARRAY[0..2] OF LREAL;
    aApparentPower_1        : ARRAY[0..2] OF LREAL;
    aApparentPower_1_Min    : ARRAY[0..2] OF LREAL;
    aApparentPower_1_Max    : ARRAY[0..2] OF LREAL;
    aActivePower            : ARRAY[0..2] OF LREAL;
    aActivePower_Min        : ARRAY[0..2] OF LREAL;
    aActivePower_Max        : ARRAY[0..2] OF LREAL;
    aReactivePower_d        : ARRAY[0..2] OF LREAL;
    aReactivePower_1        : ARRAY[0..2] OF LREAL;
    aReactivePower_1_Min    : ARRAY[0..2] OF LREAL;
```

```
    aReactivePower_1_Max    : ARRAY[0..2] OF LREAL;
    aTotalReactivePower     : ARRAY[0..2] OF LREAL;
    aPhi                    : ARRAY[0..2] OF LREAL;
    aCosPhi                 : ARRAY[0..2] OF LREAL;
    aPowerFactor            : ARRAY[0..2] OF LREAL;
    fSumApparentPower       : LREAL;
    fSumActivePower         : LREAL;
    fSumTotalReactivePower  : LREAL;
    fSumReactivePower_1     : LREAL;
    bValidStatistics        : BOOL;
END_VAR
VAR_OUTPUT PERSISTENT
    aEnergy_Pos             : ARRAY[0..2] OF ST_PMA_Energy;
    aEnergy_Neg             : ARRAY[0..2] OF ST_PMA_Energy;
    aEnergy_Res             : ARRAY[0..2] OF ST_PMA_Energy;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 90] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_PowerValues_Period_InitPars [▸ 117]	Function block-specific structure with initialization parameters. The parameters must correlate to the above definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
aApparentPower	ARRAY[0..2] OF LREAL	Total apparent power
aApparentPower_1	ARRAY[0..2] OF LREAL	Fundamental apparent power
aApparentPower_1_Min	ARRAY[0..2] OF LREAL	Smallest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aApparentPower_1_Max	ARRAY[0..2] OF LREAL	Largest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aActivePower	ARRAY[0..2] OF LREAL	Active power
aActivePower_Min	ARRAY[0..2] OF LREAL	Smallest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.
aActivePower_Max	ARRAY[0..2] OF LREAL	Largest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.

Name	Type	Description
aReactivePower_1	ARRAY[0..2] OF LREAL	Fundamental shift reactive power
aReactivePower_1_Min	ARRAY[0..2] OF LREAL	Smallest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aReactivePower_1_Max	LREAL	Largest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aTotalReactivePower	ARRAY[0..2] OF LREAL	Total reactive power
aPhi	ARRAY[0..2] OF LREAL	Phase shift angle
aCosPhi	ARRAY[0..2] OF LREAL	CosPhi (active power/fundamental apparent power)
aPowerFactor	ARRAY[0..2] OF LREAL	Power factor (active power/total apparent power)
fSumApparentPower	LREAL	Sum of the total apparent power of all phases.
fSumActivePower	LREAL	Sum of the active power of all phases.
fSumTotalReactivePower	LREAL	Sum of the total reactive power of all phases.
fSumReactivePower_1	LREAL	Sum of the values of the fundamental shift reactive power of all phases.
bValidStatistics	BOOL	TRUE if the minimum and maximum value calculation has been performed. These values are valid.
aEnergy_Pos	ARRAY[0..2] OF ST_PMA_Energy [▸ 128]	Energy in the positive direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
aEnergy_Neg	ARRAY[0..2] OF ST_PMA_Energy [▸ 128]	Energy in the negative direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
aEnergy_Res	ARRAY[0..2] OF ST_PMA_Energy [▸ 128]	Resulting energy. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.

Methods

Name	Description
Call [▸ 90]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 90]	Alternative to the function block initialization
PassInputs [▸ 91]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 92]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 91]	The current calculations are reset with the method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.2.3.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    fFreq               : LREAL;
    bResetEnergyCalc    : LREAL;
    bResetStatistics    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
fFreq	LREAL	Current frequency of the input signal. Is used to determine the length of the period at the beginning of a period. The output of the function block FB_PMA_Frequency_Period_3Ph [▸ 83] can be used.
bResetEnergyCalc	LREAL	TRUE resets the calculated values of the energy measurement.
bResetStatistics	BOOL	TRUE resets the minimum and maximum values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.3.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID       : UDINT
    stInitPars   : ST_PMA_PowerValues_Period_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_PowerValues_Period_InitPars [▸ 117]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.3.3 PassInputs

As long as an instance of the function block FB_PMA_Source_3Ph [▸ 75] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.3.4 Reset

The current calculations are reset with the method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.2.3.5 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinInputCurrent    : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3 Frequency-based

5.3.3.1 FB_PMA_Harmonics_3Ph

The function block FB_PMA_Harmonics_3Ph calculates the RMS bands of the individual current and voltage harmonics in a three-phase system. In addition, the calculated RMS bands are used to calculate the THD of the input variables.

The input buffer is provided via the function block FB_PMA_Source_3Ph [▸ 75]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^9	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

Since the Welch method is used, in each case the current input buffer together with the last transferred buffer is used for the calculation.

The frequency analysis takes step changes in the time series into account. In order to achieve a correct result, the last two input buffers should therefore be consecutive without step changes.

Syntax

Definition:

```
FUCNTION BLOCK FB_PMA_Harmonics_3Ph
VAR_INPUT
    nOwnID              : UDINT;
    tTransferTimeout    : LTIME := LTIME#500US;
    stInitPars          : ST_PMA_Harmonics_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResult          : BOOL;
    bNaNValueOccured    : BOOL;
    nCntResults         : ULINT;
    aTHD_U              : ARRAY[0..2] OF LREAL;
    aTHD_U_Min          : ARRAY[0..2] OF LREAL;
    aTHD_U_Max          : ARRAY[0..2] OF LREAL;
    aTHD_I              : ARRAY[0..2] OF LREAL;
    aTHD_I_Min          : ARRAY[0..2] OF LREAL;
    aTHD_I_Max          : ARRAY[0..2] OF LREAL;
    bValidStatistics    : BOOL;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▶ 95] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray).
stInitPars	ST_PMA_Harmonics_InitPars [▶ 118]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
aTHD_U	ARRAY[0..2] OF LREAL	THD of the voltage. The output is in percent.
aTHD_U_Min	ARRAY[0..2] OF LREAL	Smallest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.
aTHD_U_Max	ARRAY[0..2] OF LREAL	Largest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.
aTHD_I	ARRAY[0..2] OF LREAL	THD of the current. The output is in percent.
aTHD_U_Min	ARRAY[0..2] OF LREAL	Smallest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.
aTHD_U_Max	ARRAY[0..2] OF LREAL	Largest value of fTHD_U that has occurred. Can be reset via bResetStatistics of the Call method.

Name	Type	Description
bValidStatistics	BOOL	TRUE if the minimum, maximum and hold value calculation has been performed. These values are valid.

Methods

Name	Description
Call [▸ 94]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 95]	Alternative to the function block initialization
PassInputs [▸ 96]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 96]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 96]	This method deletes all the data sets already added. In addition, the calculated output values are reset.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.3.1.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pHarmonicsRMS_UL1   : POINTER TO LREAL;
    pHarmonicsRMS_UL2   : POINTER TO LREAL;
    pHarmonicsRMS_UL3   : POINTER TO LREAL;
    pHarmonicsRMS_IL1   : POINTER TO LREAL;
    pHarmonicsRMS_IL2   : POINTER TO LREAL;
    pHarmonicsRMS_IL3   : POINTER TO LREAL;
    nHarmonicsRMSSize   : UDINT;
    bResetStatistics    : BOOL;
END_VAR
```

Inputs

Name	Type	Description
pHarmonicsRMS_UL1 .. UL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: Number of harmonics. If the individual harmonics are not to be output, the input can be set to 0.

Name	Type	Description
pHarmonicsRMS_IL1 .. IL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: Number of harmonics. If the individual harmonics are not to be output, the input can be set to 0.
nHarmonicsRMSSize	UDINT	Indicates the size of an output array for the harmonics.
bResetStatistics	BOOL	TRUE resets the minimum, maximum and hold values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.1.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID       : UDINT;
    stInitPars   : ST_PMA_Harmonics_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Harmonics_InitPars [▸ 118]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.1.3 PassInputs

As long as an instance of the function block FB_PMA_Source_3Ph [▶ 75] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.1.4 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fBaseFreq  : LREAL;
    fBandwidth : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fBaseFreq	LREAL	Frequency of the first harmonic.
fBandwidth	LREAL	Total bandwidth of each RMS band.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.1.5 Reset

This method deletes all the data sets already added. In addition, the calculated output values are reset.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.2 FB_PMA_PowerValues_3Ph

The function block FB_PMA_PowerValues_3Ph calculates the power values of the connected consumer in a three-phase grid. These include the fundamental components and the phase shift angle. Internally, the individual harmonics and their phase angle are determined for the calculations.

Alternatively, the function block FB_PMA_PowerValues_Period_3Ph [▸ 87] can be used for the calculation. It uses simpler calculation methods for enhanced dynamics.

The input buffer is provided via the function block FB_PMA_Source_3Ph [▸ 75]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^9	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

Since the Welch method is used, in each case the current input buffer together with the last transferred buffer is used for the calculation.

The frequency analysis takes step changes in the time series into account. In order to achieve a correct result, the last two input buffers should therefore be consecutive without step changes.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_PowerValues_3Ph
VAR_INPUT
    nOwnID                    : UDINT;
    tTransferTimeout          : LTIME := LTIME#500US;
    stInitPars                : ST_PMA_PowerValues_InitPars;
END_VAR
VAR_OUTPUT
    bError                    : BOOL;
    ipResultMessage           : I_TcMessage;
    bNewResult                : BOOL;
    bNaNValueOccured          : BOOL;
    nCntResults               : ULINT;
    aApparentPower            : ARRAY[0..2] OF LREAL;
    aApparentPower_1          : ARRAY[0..2] OF LREAL;
    aApparentPower_1_Min      : ARRAY[0..2] OF LREAL;
    aApparentPower_1_Max      : ARRAY[0..2] OF LREAL;
    aActivePower              : ARRAY[0..2] OF LREAL;
    aActivePower_Min          : ARRAY[0..2] OF LREAL;
    aActivePower_Max          : ARRAY[0..2] OF LREAL;
    aReactivePower_d          : ARRAY[0..2] OF LREAL;
    aReactivePower_1          : ARRAY[0..2] OF LREAL;
    aReactivePower_1_Min      : ARRAY[0..2] OF LREAL;
    aReactivePower_1_Max      : ARRAY[0..2] OF LREAL;
    aTotalReactivePower       : ARRAY[0..2] OF LREAL;
    aPhi                      : ARRAY[0..2] OF LREAL;
```

```
    aCosPhi                  : ARRAY[0..2] OF LREAL;
    aPowerFactor             : ARRAY[0..2] OF LREAL;
    fSumApparentPower        : LREAL;
    fSumActivePower          : LREAL;
    fSumTotalReactivePower   : LREAL;
    fSumReactivePower_1      : LREAL;
    bValidStatistics         : BOOL;
END_VAR
VAR_OUTPUT PERSISTENT
    aEnergy_Pos              : ARRAY[0..2] OF ST_PMA_Energy;
    aEnergy_Neg              : ARRAY[0..2] OF ST_PMA_Energy;
    aEnergy_Res              : ARRAY[0..2] OF ST_PMA_Energy;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 100] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_PowerValues_InitPars [▸ 119]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.
aApparentPower	ARRAY[0..2] OF LREAL	Total apparent power
aApparentPower_1	ARRAY[0..2] OF LREAL	Fundamental apparent power
aApparentPower_1_Min	ARRAY[0..2] OF LREAL	Smallest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aApparentPower_1_Max	ARRAY[0..2] OF LREAL	Largest value of fApparentPower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aActivePower	ARRAY[0..2] OF LREAL	Active power
aActivePower_Min	ARRAY[0..2] OF LREAL	Smallest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.
aActivePower_Max	ARRAY[0..2] OF LREAL	Largest value of fActivePower that has occurred. Can be reset via bResetStatistics of the Call method.
aReactivePower_1	LREAL	Fundamental shift reactive power

Name	Type	Description
aReactivePower_1_Min	ARRAY[0..2] OF LREAL	Smallest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aReactivePower_1_Max	ARRAY[0..2] OF LREAL	Largest value of fReactivePower_1 that has occurred. Can be reset via bResetStatistics of the Call method.
aTotalReactivePower	ARRAY[0..2] OF LREAL	Total reactive power
aPhi	ARRAY[0..2] OF LREAL	Phase shift angle
aCosPhi	ARRAY[0..2] OF LREAL	CosPhi (active power/fundamental apparent power)
aPowerFactor	ARRAY[0..2] OF LREAL	Power factor (active power/total apparent power)
fSumApparentPower	LREAL	Sum of the total apparent power of all phases.
fSumActivePower	LREAL	Sum of the active power of all phases.
fSumTotalReactivePower	LREAL	Sum of the total reactive power of all phases.
fSumReactivePower_1	LREAL	Sum of the values of the fundamental shift reactive power of all phases.
bValidStatistics	BOOL	TRUE if the minimum and maximum value calculation has been performed. These values are valid.
aEnergy_Pos	ARRAY[0..2] OF ST_PMA_Energy [▸ 128]	Energy in the positive direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
aEnergy_Neg	ARRAY[0..2] OF ST_PMA_Energy [▸ 128]	Energy in the negative direction. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.
aEnergy_Res	ARRAY[0..2] OF ST_PMA_Energy [▸ 128]	Resulting energy. The output is saved persistently and can be reset via bResetEnergyCalc of the Call method.

Methods

Name	Description
Call [▸ 100]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 100]	Alternative to the function block initialization
PassInputs [▸ 101]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 101]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 102]	This method deletes all the data sets already added. In addition, the calculated output values are reset.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.3.2.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    bResetEnergyCalc   : BOOL;
    bResetStatistics   : BOOL;
END_VAR
```

Inputs

Name	Type	Description
bResetEnergyCalc	BOOL	TRUE resets the calculated values of the energy measurement.
bResetStatistics	BOOL	TRUE resets the minimum and maximum values of the outputs.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.2.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID        : UDINT;
    stInitPars    : ST_PMA_PowerValues_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_PowerValues_InitPars [▶ 119]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.2.3 PassInputs

As long as an instance of the function block FB_PMA_Source_3Ph [▸ 75] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.2.4 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fBaseFreq        : LREAL;
    fBandwidth       : LREAL;
    fMinInputCurrent : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fBaseFreq	LREAL	Frequency of the first harmonic.
fBandwidth	LREAL	Total bandwidth of each RMS band.
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.2.5 Reset

This method deletes all the data sets already added. In addition, the calculated output values are reset.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.3 FB_PMA_Spectrum_3Ph

The function block FB_PMA_Spectrum_3Ph calculates the magnitude spectra of the current and voltage values. These are suitable for analyzing the input signals in the frequency range.

The input buffer is provided via the function block FB_PMA_Source_3Ph [▶ 75]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^9	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

Since the Welch method is used, in each case the current input buffer together with the last transferred buffer is used for the calculation.

The frequency analysis takes step changes in the time series into account. In order to achieve a correct result, the last two input buffers should therefore be consecutive without step changes.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Spectrum_3Ph
VAR_INPUT
    nOwnID              : UDINT;
    tTransferTimeout    : LTIME := LTIME#500US;
    stInitPars          : ST_PMA_Spectrum_InitPars;
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResult          : BOOL;
    bNaNValueOccured    : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 104] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_Spectrum_InitPars [▸ 120]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▸ 104]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
Init [▸ 104]	Alternative to the function block initialization
PassInputs [▸ 105]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reset [▸ 105]	This method deletes all the data sets already added.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.3.3.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pMagnitudeSpectrum_UL1   : POINTER TO LREAL;
    pMagnitudeSpectrum_UL2   : POINTER TO LREAL;
    pMagnitudeSpectrum_UL3   : POINTER TO LREAL;
    pMagnitudeSpectrum_IL1   : POINTER TO LREAL;
    pMagnitudeSpectrum_IL2   : POINTER TO LREAL;
    pMagnitudeSpectrum_IL3   : POINTER TO LREAL;
    nMagnitudeSpectrumSize   : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pMagnitudeSpectrum_UL1 .. UL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
pMagnitudeSpectrum_IL1 .. IL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
nMagnitudeSpectrumSize	UDINT	Indicates the size of the output array of a spectrum.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.3.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID       : UDINT;
    stInitPars   : ST_PMA_Spectrum_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Spectrum_InitPars [▶ 120]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.3.3 PassInputs

As long as an instance of the function block FB_PMA_Source_3Ph [▶ 75] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.3.4 Reset

This method deletes all the data sets already added.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.4 FB_PMA_Spectrum_Quantiles_3Ph

The function block FB_PMA_Spectrum_Quantiles_1Ph calculates the magnitude spectra of the current and voltage values like the function block FB_PMA_Spectrum_3Ph [▸ 102]. In addition, p-quantiles of the spectrum distribution can be calculated. The quantiles and their number can be configured individually.

The input buffer is provided via the function block FB_PMA_Source_3Ph [▸ 75]. The size of the input buffer is half the window length.

By way of example, possible FFT and window lengths are shown in the following table:

FFT length		Window length	Buffer length
512	2^{9}	400	200
1024	2^{10}	800	400
2048	2^{11}	1600	800
4096	2^{12}	3200	1600
8192	2^{13}	6400	3200
16384	2^{14}	12800	6400

Memory properties

The function block takes into account all input values since the instantiation. If the Reset [▸ 110] method has been called since the start, all input values since its last call will be taken into account.

Syntax

Definition:

```
FUNCTION BLOCK FB_PMA_Spectrum_Quantiles_3Ph
VAR_INPUT
    nOwnID              : UDINT;
    tTransferTimeout    : LTIME := LTIME#500US;
    stInitPars          : ST_PMA_Spectrum_Quantiles_InitPars;
END_VAR
VAR_OUTPUT
    bError              : BOOL;
    ipResultMessage     : I_TcMessage;
    bNewResult          : BOOL;
    bNaNValueOccured    : BOOL;
    nCntResults         : ULINT;
END_VAR
```

Inputs

The input parameters of this function block represent initialization parameters and have to be assigned during the declaration of the function block instance (alternatively: Init [▸ 108] method). They may only be assigned once. A change at runtime is not possible.

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
tTransferTimeout	LTIME	Setting of the synchronous timeout for internal multi-array forwardings. See Parallel Processing in Transfer Tray.
stInitPars	ST_PMA_Spectrum_Quantiles_InitPars [▸ 121]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Outputs

Name	Type	Description
bError	BOOL	TRUE if an error occurs.
ipResultMessage	I_TcMessage	The interface offers detailed information about the return value.
bNewResult	BOOL	TRUE once new results have been calculated.
bNaNValueOccured	BOOL	Indicates whether NaN values have occurred. The results may then be unusable.
nCntResults	ULINT	Count value is incremented with new output data.

Methods

Name	Description
Call [▸ 107]	The method is called in each cycle to execute the calculations from the input buffer when new data is present.
CallEx [▸ 109]	To minimize CPU usage, it may be necessary to use the CallEx method. In contrast to the Call method, the quantiles are not calculated after each spectrum calculation, but only after a configurable number of calculations.
Init [▸ 108]	Alternative to the function block initialization
PassInputs [▸ 109]	As an alternative to the Call method, the method can be called in each cycle if no calculation is to take place. The incoming input buffer is then forwarded accordingly.
Reconfigure [▸ 111]	The method is called in order to reconfigure the function block during the runtime.
Reset [▸ 110]	This method deletes all the data sets already added. Alternatively, automatic resetting can be used on the CallEx method.

NaN incident

In rare cases, interrupts may occur in the communication via the transfer tray. If the optional parameter GVL_PMA.cOption_MarkInterruption in the function block FB_PMA_Source_3Ph is set, the first value in the buffer is marked as NaN (Not a Number) in this case. All analysis function blocks query the status of this first value before the calculation. If the status is NaN, the output bNaNValueOccured in the body of the function block is set to TRUE. No calculation takes place.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.3.3.4.1 Call

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

The CallEx [▸ 109] method may be suitable as an alternative. It calculates the quantiles only after a defined number of spectrum calculation results, in order to minimize CPU usage.

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pMagnitudeSpectrum_UL1 : POINTER TO LREAL;
    pMagnitudeSpectrum_UL2 : POINTER TO LREAL;
    pMagnitudeSpectrum_UL3 : POINTER TO LREAL;
    pMagnitudeSpectrum_IL1 : POINTER TO LREAL;
```

```
    pMagnitudeSpectrum_IL2 : POINTER TO LREAL;
    pMagnitudeSpectrum_IL3 : POINTER TO LREAL;
    nMagnitudeSpectrumSize : UDINT
    pSpectrumQuantiles_UL1 : POINTER TO LREAL;
    pSpectrumQuantiles_UL2 : POINTER TO LREAL;
    pSpectrumQuantiles_UL3 : POINTER TO LREAL;
    pSpectrumQuantiles_IL1 : POINTER TO LREAL;
    pSpectrumQuantiles_IL2 : POINTER TO LREAL;
    pSpectrumQuantiles_IL3 : POINTER TO LREAL;
    nSpectrumQuantilesSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
pMagnitudeSpectrum_UL1 .. UL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
pMagnitudeSpectrum_IL1 .. IL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
nMagnitudeSpectrumSize	UDINT	Indicates the size of the output array of a spectrum.
pSpectrumQuantiles_UL1 .. UL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
pSpectrumQuantiles_IL1 .. IL3	POINTER TO LREAL	Pointer to an array of type LREAL with the dimension: FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
nSpectrumQuantilesSize	UDINT	Indicates the size of the output array for a quantile calculation.

Return value

Name	Type	Description
Call	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.4.2 Init

The Init method is usually not required in a Power Monitoring application. It offers an alternative to function block initialization, which enables encapsulation of the function block. The method FB_init or the attribute 'call_after_init' must be used for this (see TwinCAT 3 PLC > Programming Reference). The Init method may only be called during the initialization phase of the PLC. It cannot be used at runtime.

The input parameters of the function block instance may not be assigned in the declaration if the initialization is to take place using the Init method.

Syntax

```
METHOD Init : BOOL
VAR_INPUT
    nOwnID       : UDINT;
    stInitPars   : ST_PMA_Spectrum_Quantiles_InitPars;
END_VAR
```

Inputs

Name	Type	Description
nOwnID	UDINT	Identifies the function block instance with a unique ID. This must always be greater than zero. A proven approach is to define an enumeration for this purpose.
stInitPars	ST_PMA_Spectrum_Quantiles_InitPars [▶ 121]	Function block-specific structure with initialization parameters. The parameters must match the definition of the input and output buffers.

Return value

Name	Type	Description
Init	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.4.3 PassInputs

As long as an instance of the function block FB_PMA_Source_3Ph [▶ 75] is called and thus signal data is transferred to a target block, all other function blocks of the analysis chain must be called cyclically (see Parallel Processing in Transfer Tray).

Sometimes it is useful not to be execute an algorithm for a certain time. Although the function block must nevertheless be called cyclically, it is sufficient to forward the incoming input data. This is done using the PassInputs method instead of the Call method. No result is generated.

Syntax

```
METHOD PassInputs : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
PassInputs	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.4.4 CallEx

The method is called in each cycle to execute the calculations from the input buffer when new data is present.

The function block waits for input data if the method outputs neither new results nor an error. This is a regular behavior in the process of the analysis chain.

In contrast to the Call [▶ 107] method, the CallEx method collects a variable number of results from the spectrum calculation and calculates the quantiles only afterwards. This may be required to minimize CPU load.

Syntax

```
METHOD CallEx : BOOL
VAR_INPUT
    nAppendData             : UDINT;
    bResetData              : BOOL;
    pMagnitudeSpectrum_UL1 : POINTER TO LREAL;
```

```
    pMagnitudeSpectrum_UL2 : POINTER TO LREAL;
    pMagnitudeSpectrum_UL3 : POINTER TO LREAL;
    pMagnitudeSpectrum_IL1 : POINTER TO LREAL;
    pMagnitudeSpectrum_IL2 : POINTER TO LREAL;
    pMagnitudeSpectrum_IL3 : POINTER TO LREAL;
    nMagnitudeSpectrumSize : UDINT
    pSpectrumQuantiles_UL1 : POINTER TO LREAL;
    pSpectrumQuantiles_UL2 : POINTER TO LREAL;
    pSpectrumQuantiles_UL3 : POINTER TO LREAL;
    pSpectrumQuantiles_IL1 : POINTER TO LREAL;
    pSpectrumQuantiles_IL2 : POINTER TO LREAL;
    pSpectrumQuantiles_IL3 : POINTER TO LREAL;
    nSpectrumQuantilesSize : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nAppendData	UDINT	Number of spectra to be calculated until the quantile is calculated. A value of 1 means that the quantiles are calculated after each result of the spectrum calculation.
bResetData	BOOL	Automatic resetting of records after each quantile calculation
pMagnitudeSpectrum_UL1 .. UL3	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
pMagnitudeSpectrum_IL1 .. IL3	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1. If the spectrum is not to be output, the input can be set to 0.
nMagnitudeSpectrumSize	UDINT	Indicates the size of the output array of a spectrum.
pSpectrumQuantiles_UL1 .. UL3	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
pSpectrumQuantiles_IL1 .. IL3	POINTER TO LREAL	Pointer to an array of the type LREAL with the dimension FFT length/2+1 x quantile. If these values are not to be output, the input can be set to 0.
nSpectrumQuantilesSize	UDINT	Indicates the size of the output array for a quantile calculation.

Return value

Name	Type	Description
CallEx	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.4.5 Reset

This method deletes all the data sets already added. Alternatively, automatic resetting can be used on the CallEx [▶ 109] method.

Syntax

```
METHOD Reset : BOOL
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.3.3.4.6 Reconfigure

The method is called in order to reconfigure the function block during the runtime.

Syntax

```
METHOD Reconfigure : BOOL
VAR_INPUT
    fMinBinned     : LREAL;
    fMaxBinned     : LREAL;
    nBins          : UDINT;
    nNumQuantiles  : UDINT;
    aQuantiles     : ARRAY[0..GVL_PMA.cMaxQuantiles - 1] OF LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinBinned	LREAL	Lower limit value for which the output data of the range calculation in the regular histogram bins are counted.
fMaxBinned	LREAL	Upper limit value for which the output data of the range calculation are counted in the regular histogram bins. fMaxBinned must be larger than the fMinBinned
nBins	UDINT	number of bins in a histogram. The minimum number is 1. In many cases, values between ten and twenty are a sensible choice. The two special bins for values that lie below fMinBinned or above fMaxBinned are not included in this value.
nNumQuantiles	UDINT	Number of quantiles to calculate for each channel. This must be an integer greater than zero.
aQuantiles	ARRAY[0..GVL_PMA.cMaxQuantiles-1] OF LREAL	Indicates the quantile limit. It must be between 0.0 and 1.0. For example, 0.2 corresponds to the 20% quantile.

Return value

Name	Type	Description
Reconfigure	BOOL	Indicates whether the method was executed successfully. Further information is provided in the Event interface of the function block.

5.4 Data types

5.4.1 E_PMA_ScalingType

The enumeration E_PMA_ScalingType lists all scaling options for spectral calculations. It is used in the structures of the initialization parameters. Further details can be found in the appendix to the "TF3600 Condition Monitoring" documentation.

Syntax

Definition:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_PMA_ScalingType :
(
     NoScaling             := 0,
     DiracScaling          := 1,
     PeakAmplitude         := 2,
     ROOT_POWER_SUM        := 3,
     RMS                   := 4,
     GainCorrection        := 5,
     PowerSpectralDensity  := 6,
     UnitaryScaling        := 7
) UDINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.2 E_PMA_WindowType

The enumeration E_PMA_WindowType can be used to select the window type for the spectral calculations. It is used in the structures of the initialization parameters. Further details can be found in the appendix to the "TF3600 Condition Monitoring" documentation.

Syntax

Definition:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_PMA_WindowType :
(
    HannWindow          := 16#05300901,
    RectangularWindow   := 16#05300902
) UDINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3 InitParameters

5.4.3.1 General

5.4.3.1.1 ST_PMA_Scaling_EL3773_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function block FB_PMA_Scaling_EL3773 [▶ 24] is initialized.

Syntax

Definition:

```
TYPE ST_PMA_Scaling_EL3773_InitPars :
STRUCT
    nOversamples                : UDINT  := 10;   // Oversampling factor
    fOffsetVoltage              : LREAL  := 0.0;  // Output = (input * gain) + offset
    fGainVoltage                : LREAL  := 1.0;  //  ||
    fOffsetCurrent              : LREAL  := 0.0;  // Output = (input * gain) + offset
    fGainCurrent                : LREAL  := 1.0;  //  ||
    fFactorCurrentTransformer   : LREAL  := 1.0;  // Factor of current tranformer (input / output)
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Description
nOversamples	UDINT	Oversampling factor
fOffsetVoltage	LREAL	Indicates a user-defined offset for the voltage scaling.
fGainVoltage	LREAL	Indicates a user-defined gain factor for the voltage scaling.
fOffsetCurrent	LREAL	Indicates a user-defined offset for the current scaling.
fGainCurrent	LREAL	Specifies a user-defined gain factor for the current scaling.
fFactorCurrentTransformer	LREAL	Current transformer factor. This is calculated from the input current and output current.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.1.2 ST_PMA_Scaling_EL3783_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function block FB_PMA_Scaling_EL3783 [▶ 27] is initialized.

Syntax

Definition:

```
TYPE ST_PMA_Scaling_EL3783_InitPars :
STRUCT
    nOversamples                : UDINT  := 20;   // Oversampling factor
    fOffsetVoltage              : LREAL  := 0.0;  // Output = (input * gain) + offset
    fGainVoltage                : LREAL  := 1.0;  //  ||
    fOffsetCurrent              : LREAL  := 0.0;  // Output = (input * gain) + offset
    fGainCurrent                : LREAL  := 1.0;  //  ||
```

```
    fFactorCurrentTransformer   : LREAL  := 1.0;  // Factor of current tranformer (input / output)
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Description
nOversamples	UDINT	Oversampling factor
fOffsetVoltage	LREAL	Indicates a user-defined offset for the voltage scaling.
fGainVoltage	LREAL	Indicates a user-defined gain factor for the voltage scaling.
fOffsetCurrent	LREAL	Indicates a user-defined offset for the current scaling.
fGainCurrent	LREAL	Specifies a user-defined gain factor for the current scaling.
fFactorCurrentTransformer	LREAL	Current transformer factor. This is calculated from the input current and output current.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.1.3 ST_PMA_Scaling_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function block FB_PMA_Scaling [▸ 19] is initialized.

Syntax

Definition:

```
TYPE ST_PMA_Scaling_InitPars :
STRUCT
    nOversamples              : UDINT := 10;      // Oversampling factor
    nResolutionVoltage        : UDINT := (1..32); // in bit
    fMaxVoltage               : LREAL;            // Max input amplitude
    fOffsetVoltage            : LREAL := 0;       // Output = (input * gain) + offset
    fGainVoltage              : LREAL := 1;       //  ||
    nResolutionCurrent        : UDINT := (1..32); // in bit
    fMaxCurrent               : LREAL;            // Max input amplitude
    fOffsetCurrent            : LREAL := 0;       // Output = (input * gain) + offset
    fGainCurrent              : LREAL := 1;       //  ||
    fFactorCurrentTransformer : LREAL := 1;       // Factor of current transformer (input / output)
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nOversamples	UDINT	Oversampling factor
nResolutionVoltage	UDINT	Resolution of the voltage in bits. Range of values: 0-32
fMaxVoltage	LREAL	Indicated the maximum amplitude of the input voltage.
fOffsetVoltage	LREAL	Indicates a user-defined offset for the voltage scaling.
fGainVoltage	UDINT	Indicates a user-defined gain factor for the voltage scaling.
nResolutionCurrent	LREAL	Resolution of the current in bits. Range of values: 0-32

Name	Type	Parameter
fMaxCurrent	LREAL	Indicates the maximum amplitude of the input current.
fOffsetCurrent	LREAL	Indicates a user-defined offset for the current scaling.
fGainCurrent	LREAL	Specifies a user-defined gain factor for the current scaling.
fFactorCurrentTransformer	LREAL	Current transformer factor. This is calculated from the input current and output current.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.1.4 ST_PMA_Source_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function blocks FB_PMA_Source_1Ph [▶ 39] and FB_PMA_Source_3Ph [▶ 75] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_Source_InitPars :
STRUCT
    nBufferLength  : UDINT := 200;  // Length of output buffer
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nBufferlength	UDINT	Length of the output buffer

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.1.5 ST_PMA_TaskTransfer_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function blocks FB_PMA_TaskTransfer_Send [▶ 33] and FB_PMA_TaskTransfer_Receive [▶ 36] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_TaskTransfer_InitPars :
STRUCT
    nInputSize  : UDINT;  // Size of input data
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nInputSize	UDINT	Number of elements which are exchanged via the function blocks for task transfer

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.2 Time-based

5.4.3.2.1 ST_PMA_BasicValues_Period_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function blocks FB_PMA_BasicValues_Period_1Ph [▸ 42] and FB_PMA_BasicValues_Period_3Ph [▸ 78] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_BasicValues_Period_InitPars :
STRUCT
    nBufferLength    : UDINT := 200;   // Length of input buffer
    fSampleRate      : LREAL := 1000;  // in Hz
    fMinInputCurrent : LREAL := 0.0;   // Minimal input of current (RMS) to calculate outputs
    nPeriods         : UDINT := 1;     // Amount of signal periods to calculate outputs
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nBufferLength	UDINT	Length of the input buffer
fSampleRate	LREAL	Indicates the sampling rate (samples per second) of the input signal.
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.
nPeriods	UDINT	Number of signal periods for calculating the RMS

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.2.2 ST_PMA_Frequency_Period_InitPars

Function block-specific structure with initialization parameters that are evaluated when the function blocks FB_PMA_Frequency_Period_1Ph [▸ 51] and FB_PMA_Frequency_Period_3Ph [▸ 83] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_Frequency_Period_InitPars :
STRUCT
    nBufferLength      : UDINT := 200;   // Length of input buffer
    fSampleRate        : LREAL := 1_000; // in Hz
    fMinFreq           : LREAL := 45;    // Min measured Freq
    fMaxFreq           : LREAL := 65;    // Max measured Freq
    nPeriods           : UDINT := 2;     // Number of periods to be considered
    nFilterOrder       : UINT  := 3;     // Filter order of butterworth lowpass filter
    fCutoff            : LREAL := 70.0;  // Cutoff frequency of filter
    eInputSelect       : E_PMA_InputSelect := E_PMA_InputSelect.Voltage; // Select input: Voltage |
 Current
```

```
    fMinInput           : LREAL := 200.0; // Minimal input (RMS) over one period to calculate outputs
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nBufferLength	UDINT	Length of the input buffer
fSampleRate	LREAL	Indicates the sampling rate (samples per second) of the input signal.
fMinFreq	LREAL	Minimum expected measuring frequency
fMaxFreq	LREAL	Maximum expected measuring frequency
nPeriods	UDINT	Number of periods that influence the calculation. (Period length = sample rate/frequency)
nFilterOrder	UINT	Indicates the order of the low-pass filter. The stability of the filter must be considered for the setting. Only values up to the tenth order are allowed.
fCutoff	LREAL	Specifies the limit frequency of the low-pass filter.
eInputSelect	E_PMA_InputSelect [▶ 127]	Here you can configure whether the frequency of the voltage or the current should be calculated.
fMinInput	LREAL	Minimum input value (RMS) over one period. This prevents the calculation of input values that are too small.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.2.3 ST_PMA_PowerValues_Period_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function blocks FB_PMA_PowerValues_Period_1Ph [▶ 46] and FB_PMA_PowerValues_Period_3Ph [▶ 87] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_PowerValues_Period_InitPars :
STRUCT
    nBufferLength      : UDINT := 200;  // Length of input buffer
    fSampleRate        : LREAL := 1000; // in Hz
    fMinInputCurrent   : LREAL := 0.0;  // Minimal input of current (RMS) to calculate outputs
    nPeriods           : UDINT := 1;    // Amount of signal periods to calculate outputs
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nBufferLength	UDINT	Length of the input buffer
fSampleRate	LREAL	Indicates the sampling rate (samples per second) of the input signal.
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.
nPeriods	UDINT	Number of signal periods for calculating the initial values

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.3 Frequency-based

5.4.3.3.1 ST_PMA_Harmonics_InitPars

Function block-specific structure with initialization parameters that are evaluated when the function blocks FB_PMA_Harmonics_1Ph [▶ 56] and FB_PMA_Harmonics_3Ph [▶ 92] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_HarmonicAnalysis_InitPars :
STRUCT
    fSampleRate          : LREAL := 1000;    // Sample rate
    fBaseFreq            : LREAL := 50.0;    // Multiple of base frequence is used for band limit de
finition
    nFFT_Length          : UDINT := 512;     // Length of FFT
    nWindowLength        : UDINT := 400;     // Length of FFT window
    nNumBands            : UDINT := 10;      // Number of bands
    fBandwidth           : LREAL := 4.0;     // Whole bandwidth for each frequency band, min 1.0
    eWindowType          : E_PMA_WindowType := E_PMA_WindowType.HannWindow; // Window  function used
    bTransformToDecibel  : BOOL  := TRUE;    // Transform result to decibel
    fDecibelThreshold    : LREAL := 1.3E-308;// Logarithm threshold for decibel transformation
    bTransformToPercent  : BOOL := FALSE;    // Transform result to percent (1st harmonic=100%)
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
fSampleRate	LREAL	Indicates the sampling rate (samples per second) of the input signal.
fBaseFreq	LREAL	Frequency of the first harmonic.
nFFT_Length	UDINT	Length of the FFT. It must be greater than one and an integral power of two.
nWindowLength	UDINT	Length of the analysis window in samples. The length must be greater than one and an even number.
nNumBands	UDINT	Indicates the number of bands for which the RMS is calculated.
fBandwidth	LREAL	Total bandwidth of each RMS band
eWindowType	E_PMA_WindowType [▶ 112]	Defines the window function that is used. The window type "„HannWindow" is a good default value. The windowing can be switched off by the use of the window type "RectangularWindow". Further explanations and the list of possible window functions can be found in the introductory section Window functions.
bTransformToDecibel	BOOL	Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x → 20 * log10(x).

Name	Type	Parameter
fDecibelThreshold	LREAL	Very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale, since the logarithm of zero is not defined mathematically. The smallest possible value is 3.75e-324.
bTransformToPercent	BOOL	Boolean value indicating whether the result should be output in percent. The first harmonic corresponds to 100%.

Window length

The value of nWindowLength must be less or equal the value of nFFT_Length. The length of the FFT can orient itself to the required frequency resolution. Typically a value of about 3/4 of the FFT length is used as the window length.

If nFFT_Length is greater than nWindowLength, the frequency resolution of the FFT (and therefore also the length of the return values vector) is increased. The length difference is filled with zeros before the Fourier transform . This can be useful for achieving a higher frequency resolution, or for avoiding circular aliasing in calculations with inverse transformation in the time domain . Despite the higher frequency resolution, however, the result contains no more information.

5.4.3.3.2 ST_PMA_PowerValues_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function blocks FB_PMA_PowerValues_1Ph [▶ 60] and FB_PMA_PowerValues_3Ph [▶ 97] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_PowerValues_InitPars :
STRUCT
    fSampleRate                     : LREAL := 1000;     // in Hz
    fFreq                           : LREAL := 50.0;     // in HZ
    fTimeLagCurrentTransformer      : UDINT := 1;        // Number of periods to be considered |
 period length := samplerate/frequency
    nFFT_Length                     : UDINT := 512;      // Length of FFT
    nWindowLength                   : UDINT := 400;      // Length of FFT window
    nNumBands                       : UDINT := 10,       // Number of bands
    fBandwidth                      : LREAL := 4.0       // Whole bandwidth of each frequency band
    eWindowType                     : E_PMA_WindowType := E_PMA_WindowType.HannWindow; // Window funct
ion used
    fMinInputCurrent                : LREAL := 0.0;      // Minimal input of current (RMS) to calculate
 outputs
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
fSampleRate	LREAL	Indicates the sampling rate (samples per second) of the input signal.
fFreq	LREAL	Frequency of the first harmonic (fundamental)
fTimeLagCurrentTransformer	UDINT	Here, the possible delay due to the inductance of the current transformer can be specified in seconds.
nFFT_Length	UDINT	Length of the FFT. It must be greater than one and an integral power of two.
nWindowLength	UDINT	Length of the analysis window in samples. The length must be greater than one and an even number. It must be not greater than nFFT_Length.

Name	Type	Parameter
nNumBands	UDINT	Indicates the number of bands for which the RMS is calculated.
fBandwidth	LREAL	Total bandwidth of a single RMS band
eWindowType	E_PMA_WindowType [▸ 112]	Defines the window function that is used. The window type "„HannWindow" is a good default value. The windowing can be switched off by the use of the window type "RectangularWindow". Further explanations and the list of possible window functions can be found in the introductory section Window functions.
fMinInputCurrent	LREAL	Minimum input value (RMS) of the current. This prevents the calculation of input values that are too small.

Window length

The value of nWindowLength must be less or equal the value of nFFT_Length. The length of the FFT can orient itself to the required frequency resolution. Typically a value of about 3/4 of the FFT-length is used as the window length.

If nFFT_Length is greater than nWindowLength, the frequency resolution of the FFT (and therefore also the length of the return values vector) is increased. The difference in the length is filled with zeros by the Fourier transformation. This can be useful for achieving a higher frequency resolution, or for avoiding circular aliasing in calculations with inverse transformation in the time domain. Despite the higher frequency resolution, however, the result contains no more information.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.3.3 ST_PMA_Spectrum_InitPars

Function block-specific structure with initialization parameters that are evaluated when the function blocks FB_PMA_Spectrum_1Ph [▸ 65] and FB_PMA_Spectrum_3Ph [▸ 102] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_Spectrum_InitPars :
STRUCT
    nFFT_Length          : UDINT := 512;      // Length of FFT
    nWindowLength        : UDINT := 400;      // Length of FFT window
    fSampleRate          : LREAL := 1000;     // in Hz
    eScalingType         : E_PMA_ScalingType := E_PMA_ScalingType.NoScaling; // Scaling type used
    eWindowType          : E_PMA_WindowType  := E_PMA_WindowType.HannWindow; // Window function used
    bTransformToDecibel : BOOL := TRUE;      // Transform result to decibel
    fDecibelThreshold    : LREAL := 1.3E-308; // Logarithm threshold for decibel transformation
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nFFT_Length	UDINT	Length of the FFT. It must be greater than one and an integral power of two.
nWindowLength	UDINT	Length of the analysis window in samples. The length must be greater than one and an even number. It must be not greater than nFFT_Length.

Name	Type	Parameter
fSampleRate	LREAL	Indicates the sampling rate (samples per second) of the input signal.
eScalingType	E_PMA_ScalingType [▸ 112]	Allows selection of the scaling used if absolute scaling is required.
eWindowType	E_PMA_WindowType [▸ 112]	Defines the window function that is used. The window type "„HannWindow" is a good default value. The windowing can be switched off by the use of the window type "RectangularWindow". Further explanations and the list of possible window functions can be found in the introductory section Window functions.
bTransformToDecibel	BOOL	Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x → 20 * log10(x).
fDecibelThreshold	LREAL	Very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale, since the logarithm of zero is not defined mathematically. The smallest possible value is 3.75e-324, which is equivalent to the constant `cCM_MinArgLog10`.

Window length

The value of nWindowLength must be less or equal the value of nFFT_Length. The length of the FFT can orient itself to the required frequency resolution. Typically a value of about 3/4 of the FFT-length is used as the window length.

If nFFT_Length is greater than nWindowLength, the frequency resolution of the FFT (and therefore also the length of the return values vector) is increased. The difference in the length is filled with zeros by the Fourier transformation. This can be useful for achieving a higher frequency resolution, or for avoiding circular aliasing in calculations with inverse transformation in the time domain. Despite the higher frequency resolution, however, the result contains no more information.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.3.3.4 ST_PMA_Spectrum_Quantiles_InitPars

Function block-specific structure with initialization parameters which are evaluated when the function blocks FB_PMA_Spectrum_Quantiles_1Ph [▸ 69] and FB_PMA_Spectrum_Quantiles_3Ph [▸ 106] are initialized.

Syntax

Definition:

```
TYPE ST_PMA_Spectrum_Quantiles_InitPars :
STRUCT
    nFFT_Length          : UDINT := 512;       // Length of FFT
    nWindowLength        : UDINT := 400;       // Length of FFT window
    fSampleRate          : LREAL := 1000;      // in Hz
    eScalingType         : E_PMA_ScalingType  := E_PMA_ScalingType.NoScaling; // Scaling type used
    eWindowType          : E_PMA_WindowType   := E_PMA_WindowType.HannWindow; // Window function used
    bTransformToDecibel  : BOOL := TRUE;       // Transform result to decibel
    fDecibelThreshold    : LREAL := 1.3E-308; // Log threshold for decibel transformation
    fMinBinned           : LREAL;              // Minimum binned value
    fMaxBinned           : LREAL;              // Maximum binned value
    nBins                : UDINT := 1;         // Number of bins in interval
    nNumQuantiles        : UDINT := 1;         // Maximum number of quantile values
```

```
    aQuantiles          : ARRAY[0..GVL_PMA.cMaxQuantiles-1] OF LREAL; // 0.0 < aQuantiles[x] <1.0
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Parameter
nFFT_Length	UDINT	Length of the FFT. It must be greater than one and an integral power of two.
nWindowLength	UDINT	Length of the analysis window in samples. The length must be greater than one and an even number. It must be not greater than nFFT_Length.
fSampleRate	LREAL	Indicates the sampling rate (samples per second) of the input signal.
eScalingType	E_PMA_ScalingType [▸ 112]	Allows selection of the scaling used if absolute scaling is required.
eWindowType	E_PMA_WindowType [▸ 112]	Defines the window function that is used. The window type "„HannWindow" is a good default value. The windowing can be switched off by the use of the window type "RectangularWindow". Further explanations and the list of possible window functions can be found in the introductory section Window functions.
bTransformToDecibel	BOOL	Boolean value that indicates whether the result of the FFT is to be transformed to the decibel scale, according to transformation x → 20 * log10(x).
fDecibelThreshold	LREAL	Very small floating point value greater than zero. Values that are less than this number are replaced with this value before any transformation to the decibel scale, since the logarithm of zero is not defined mathematically. The smallest possible value is 3.75e-324.
fMinBinned	LREAL	Lower limit value for which the output data of the range calculation in the regular histogram bins are counted.
fMaxBinned	LREAL	Upper limit value for which the output data of the range calculation in the regular histogram bins are counted. fMaxBinned must be greater than fMinBinned.
nBins	UDINT	number of bins in a histogram. The minimum number is 1. In many cases, values between ten and twenty are a sensible choice. The two special bins for values that lie below fMinBinned or above fMaxBinned are not included in this value.
nNumQuantiles	UDINT	Number of quantiles to calculate for each channel. This must be an integer greater than zero.
aQuantiles	ARRAY[0..GVL_PMA.cMaxQuantiles-1] OF LREAL	Indicates the quantile limit. It must be between 0.0 and 1.0. For example, 0.2 corresponds to the 20% quantile.

Window length

The value of nWindowLength must be less or equal the value of nFFT_Length. The length of the FFT can orient itself to the required frequency resolution. Typically a value of about 3/4 of the FFT-length is used as the window length.

If nFFT_Length is greater than nWindowLength, the frequency resolution of the FFT (and therefore also the length of the return values vector) is increased. The difference in the length is filled with zeros by the Fourier transformation. This can be useful for achieving a higher frequency resolution, or for avoiding circular aliasing in calculations with inverse transformation in the time domain. Despite the higher frequency resolution, however, the result contains no more information.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.4 Single-phase

5.4.4.1 ST_PMA_BasicValues_Period_1Ph

Structure that summarizes the results of the corresponding function block and which can be queried via the property stResults. Details on the values contained can be found in the comment or in the description of the function block FB_PMA_BasicValues_Period_1Ph [▸ 42].

Syntax

Definition:

```
TYPE ST_PMA_BasicValues_Period_1Ph :
STRUCT
    fMeanValue_U       : LREAL; // [V] | Mean value over n periods
    fRMS_U             : LREAL; // [V] | Root mean square over n periods
    fRMS_U_Min         : LREAL; // [V] | Min value of fRMS_U
    fRMS_U_Max         : LREAL; // [V] | Max value of fRMS_U
    fPeakValue_U       : LREAL; // [V] | Peak value of U over n periods
    fPeakHold_U        : LREAL; // [V] | All time peak of U
    fRectifiedValue_U  : LREAL; // [V] | Rectified value over n periods
    fCrestFactor_U     : LREAL; // []  | Peak_value / RMS
    fFormFactor_U      : LREAL; // []  | RMS / rectified_value
    fMeanValue_I       : LREAL; // [A] | Mean value over n periods
    fRMS_I             : LREAL; // [A] | Root mean square over n periods
    fRMS_I_Min         : LREAL; // [A] | Min value of fRMS_I
    fRMS_I_Max         : LREAL; // [A] | Max value of fRMS_I
    fPeakValue_I       : LREAL; // [A] | Peak value of I over n periods
    fPeakHold_I        : LREAL; // [A] | All time peak of I
    fRectifiedValue_I  : LREAL; // [A] | Rectified value over n periods
    fCrestFactor_I     : LREAL; // []  | Peak_value / RMS
    fFormFactor_I      : LREAL; // []  | RMS / rectified_value
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.4.2 ST_PMA_Frequency_Period_1Ph

Structure that summarizes the results of the corresponding function block. It can be queried via the property stResults. Details on the values contained can be found in the comment or in the description of the function block FB_PMA_Frequency_Period_1Ph [▸ 51].

Syntax

Definition:

```
TYPE ST_PMA_Frequency_Period_1Ph :
STRUCT
    fFreq     : LREAL; // [Hz] | f      | Frequency calculated by zero crossings
```

```
    fFreq_Min : LREAL; // [Hz] | f_min | Min value of fFreq
    fFreq_Max : LREAL; // [Hz] | f_max | Max value of fFreq
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.4.3 ST_PMA_PowerValues_1Ph

Structure that summarizes the results of the corresponding function block. It can be queried via the property stResults. Details on the values contained can be found in the comment or in the descriptions of the function blocks FB_PMA_PowerValues_Period_1Ph [▸ 46] or FB_PMA_PowerValues_1Ph [▸ 60].

Syntax

Definition:

```
Type ST_PMA_PowerValues_1Ph
STRUCT
    fApparentPower       : LREAL;         // [VA]  | S        | apparent power
    fApparentPower_1     : LREAL;         // [VA]  | S_1      | fundamental apparent power
    fApparentPower_1_Min : LREAL;         // [VA]  | S_1_min  | Min value of S_1
    fApparentPower_1_Max : LREAL;         // [VA]  | S_1_max  | Max value of S_1
    fActivePower         : LREAL;         // [W]   | P        | active power
    fActivePower_Min     : LREAL;         // [W]   | P_min    | Min value of P
    fActivePower_Max     : LREAL;         // [W]   | P_max    | Max value of P
    fReactivePower_d     : LREAL;         // [var] | Q_d      | distortion reactive power
    fReactivePower_1     : LREAL;         // [var] | Q_1      | fundamental reactive power
    fReactivePower_1_Min : LREAL;         // [var] | Q_1_min  | Min value of Q_1
    fReactivePower_1_Max : LREAL;         // [var] | Q1_max   | Max value of Q_1
    fTotalReactivePower  : LREAL;         // [var] | Q_tot    | total reactive power
    fPhi                 : LREAL;         // [°]   | phi      | phase difference
    fCosPhi              : LREAL;         // []    | COS(phi)|(P / S_1)
    fPowerFactor         : LREAL;         // []    | PF       | power factor (P / S)
    stEnergy_Pos         : ST_PMA_Energy; // [kWh] | W_pos    | Energy in positive direction
    stEnergy_Neg         : ST_PMA_Energy; // [kWh] | W_neg    | Energy in negative direction
    stEnergy_Res         : ST_PMA_Energy; // [kWh] | W        | Resulting energy
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.4.4 ST_PMA_THD_1Ph

Structure that summarizes the results of the corresponding function block. It can be queried via the property stResults. Details on the values contained can be found in the comment or in the description of the function block FB_PMA_Harmonics_1Ph [▸ 56].

Syntax

Definition:

```
TYPE ST_PMA_THD_1Ph :
STRUCT
    fTHD_U     : LREAL; // [] | THD_U     | Total harmonic distortion of voltage | in percent
    fTHD_U_Min : LREAL; // [] | THD_U_min | Min value of THD_U
    fTHD_U_Max : LREAL; // [] | THD_U_max | Max value of THD_U
    fTHD_I     : LREAL; // [] | THD_I     | Total harmonic distortion of current | in percent
    fTHD_I_Min : LREAL; // [] | THD_I_min | Min value of THD_I
    fTHD_I_Max : LREAL; // [] | THD_I_max | Max value of THD_I
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.5 Three-phase

5.4.5.1 E_PMA_RotationalDirection_3Ph

Syntax

Definition:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_PMA_RotationalDirection_3Ph :
(
    No_Direction := 0; // no rotational direction detected
    Right        := 1;
    Left         := 2;
) UDINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.5.2 ST_PMA_BasicValues_Period_3Ph

Structure that summarizes the results of the corresponding function block. It can be queried via the property stResults. Details on the values contained can be found in the comment or in the description of the function block FB_PMA_BasicValues_Period_3Ph [▸ 78].

Syntax

Definition:

```
TYPE ST_PMA_BasicValues_Period_3Ph :
STRUCT
    aMeanValue_U      : ARRAY[0..2] OF LREAL; // [V] | Mean value over n periods
    aRMS_U            : ARRAY[0..2] OF LREAL; // [V] | Root mean square over n periods
    aRMS_U_Min        : ARRAY[0..2] OF LREAL; // [V] | Min value of aRMS_U
    aRMS_U_Max        : ARRAY[0..2] OF LREAL; // [V] | Max value of aRMS_U
    aRMS_UPP          : ARRAY[0..2] OF LREAL; // [V] | 0: L1 - L2 | 1: L2 - L3 | 2: L3 - L1
    aPeakValue_U      : ARRAY[0..2] OF LREAL; // [V] | Peak value of U over n periods
    aPeakHold_U       : ARRAY[0..2] OF LREAL; // [V] | All time peak of U
    aRectifiedValue_U : ARRAY[0..2] OF LREAL; // [V] | Rectified value over n periods
    aCrestFactor_U    : ARRAY[0..2] OF LREAL; // []  | Peak_value / RMS
    aFormFactor_U     : ARRAY[0..2] OF LREAL; // []  | RMS / rectified_value
    aMeanValue_I      : ARRAY[0..2] OF LREAL; // [A] | Mean value over n periods [A]
    aRMS_I            : ARRAY[0..2] OF LREAL; // [A] | Root mean square over n periods [A]
    aRMS_I_Min        : ARRAY[0..2] OF LREAL; // [A] | Min value of aRMS_I [A]
    aRMS_I_Max        : ARRAY[0..2] OF LREAL; // [A] | Max value of aRMS_I [A]
    aPeakValue_I      : ARRAY[0..2] OF LREAL; // [A] | Peak value of I over n periods [A]
    aPeakHold_I       : ARRAY[0..2] OF LREAL; // [A] | All time peak of I [A]
    aRectifiedValue_I : ARRAY[0..2] OF LREAL; // [A] | Rectified value over n periods [A]
    aCrestFactor_I    : ARRAY[0..2] OF LREAL; // []  | Peak_value / RMS
    aFormFactor_I     : ARRAY[0..2] OF LREAL; // []  | RMS / rectified_value
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.5.3 ST_PMA_Frequency_Period_3Ph

Structure that summarizes the results of the corresponding function block. It can be queried via the property stResults. Details on the values contained can be found in the comment or in the description of the function block FB_PMA_Frequency_Period_3Ph [▶ 83].

Syntax

Definition:

```
TYPE ST_PMA_Frequency_Period_3Ph :
STRUCT
    aFreq     : ARRAY[0..2] OF LREAL; // [Hz] | f     | Frequency calculated by zero crossings
    aFreq_Min : ARRAY[0..2] OF LREAL; // [Hz] | f_min | Min value of aFreq
    aFreq_Max : ARRAY[0..2] OF LREAL; // [Hz] | f_max | Max value of aFreq
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.5.4 ST_PMA_PowerValues_3Ph

Structure that summarizes the results of the corresponding function block. It can be queried via the property stResults. Details on the values contained can be found in the comment or in the descriptions of the function blocks FB_PMA_PowerValues_Period_3Ph [▶ 87] or FB_PMA_PowerValues_3Ph [▶ 97].

Syntax

Definition:

```
TYPE ST_PMA_PowerValues_3Ph :
STRUCT
    aApparentPower          : ARRAY[0..2] OF LREAL;         // [VA]   | S        | apparent power
    aApparentPower_1        : ARRAY[0..2] OF LREAL;         // [VA]   | S_1      |
 fundamental apparent power
    aApparentPower_1_Min    : ARRAY[0..2] OF LREAL;         // [VA]   | S_1_min  | Min value of S_1
    aApparentPower_1_Max    : ARRAY[0..2] OF LREAL;         // [VA]   | S_1_max  | Max value of S_1
    aActivePower            : ARRAY[0..2] OF LREAL;         // [W]    | P        | active power
    aActivePower_Min        : ARRAY[0..2] OF LREAL;         // [W]    | P_min    | Min value of P
    aActivePower_Max        : ARRAY[0..2] OF LREAL;         // [W]    | P_max    | Max value of P
    aReactivePower_d        : ARRAY[0..2] OF LREAL;         // [var]  | Q_d      |
 distortion reactive power
    aReactivePower_1        : ARRAY[0..2] OF LREAL;         // [var]  | Q_1      |
 fundamental reactive power
    aReactivePower_1_Min    : ARRAY[0..2] OF LREAL;         // [var]  | Q_1_min  | Min value of Q_1
    aReactivePower_1_Max    : ARRAY[0..2] OF LREAL;         // [var]  | Q1_max   | Max value of Q_1
    aTotalReactivePower     : ARRAY[0..2] OF LREAL;         // [var]  | Q_tot    |
 total reactive power
    aPhi                    : ARRAY[0..2] OF LREAL;         // [°]    | phi      | phase difference
    aCosPhi                 : ARRAY[0..2] OF LREAL;         // []     | COS(phi) |(P / S_1)
    aPowerFactor            : ARRAY[0..2] OF LREAL;         // []     | PF       |
 power factor (P / S)
    fSumApparentPower       : LREAL;                        // [VA]   | S_sum    |
 Sum of apparent power 1..3
    fSumActivePower         : LREAL;                        // [W]    | P_sum    |
 Sum of active power 1..3
    fSumTotalReactivePower  : LREAL;                        // [var]  | Qtot_sum |
 Sum of total reactive power 1..3
    fSumReactivePower_1     : LREAL;                        // [var]  | Q1_sum   |
 Sum of abs fundamental reactive power 1..3
    aEnergy_Pos             : ARRAY[0..2] OF ST_PMA_Energy; // [kWh]  | W_pos    |
```

```
 Energy in positive direction
    aEnergy_Neg            : ARRAY[0..2] OF ST_PMA_Energy; // [kWh]  | W_neg    |
 Energy in negative direction
    aEnergy_Res            : ARRAY[0..2] OF ST_PMA_Energy; // [kWh]  | W        | Resulting energy
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.5.5 ST_PMA_THD_3Ph

Structure that summarizes the results of the corresponding function block. It can be queried via the property stResults. Details on the values contained can be found in the comment or in the description of the function block FB_PMA_Harmonics_Ph3 [▶ 92].

Syntax

Definition:

```
TYPE ST_PMA_THD_3Ph :
STRUCT
    aTHD_U      : ARRAY[0..2] OF LREAL; // [] | THD_U     | Total harmonic distortion of voltage |
 in percent
    aTHD_U_Min  : ARRAY[0..2] OF LREAL; // [] | THD_U_min | Min value of THD_U
    aTHD_U_Max  : ARRAY[0..2] OF LREAL; // [] | THD_U_max | Max value of THD_U
    aTHD_I      : ARRAY[0..2] OF LREAL; // [] | THD_I     | Total harmonic distortion of current |
 in percent
    aTHD_I_Min  : ARRAY[0..2] OF LREAL; // [] | THD_I_min | Min value of THD_I
    aTHD_I_Max  : ARRAY[0..2] OF LREAL; // [] | THD_I_max | Max value of THD_I
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.6 E_PMA_InputSelect

The enumeration E_PMA_InputSelect can be used to select whether the results to be calculated should relate to the current or voltage values.

Syntax

Definition:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_PMA_InputSelect :
(
     Voltage    := 0,     // Voltage selected
     Current    := 1      // Current selected
) UDINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.4.7 ST_PMA_Energy

Structure for the output of energy values. The full kilowatt hours (kWh) and the decimal places are output.

Syntax

Definition:

```
TYPE ST_PMA_Energy :
STRUCT
    nEnergy                 : LINT;   // Energy in kWh
    fEnergyFraction         : LREAL;  // Fraction of energy in kWh
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Description
nEnergy	LINT	Energy in kilowatt hours (kWh) as an integral value.
fEnergyFraction	LREAL	Decimal places of the energy in kilowatt hours (kWh).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

5.5 Global constants

5.5.1 GVL_PMA

The power monitoring library contains the following constants in the PLC.

```
{attribute 'qualified_only'}
VAR_GLOBAL
    eEventTraceLevel         : TcEventSeverity := TcEventSeverity.Info;
END_VAR
VAR_GLOBAL CONSTANT
    cMA_MaxDest              : UDINT := 20;           // maximum destinations for one analysis block
    cMA_MaxID                : UDINT := 600;          // maximum ID which can be used (=maximum numb
er of analysis blocks
    cOption_MarkInterruption : DWORD := 16#0000_0001; // mark first buffer element (of each channel)
 as invalid (NaN) if time series collection has been interrupted.
    cMbRMS_MaxBands          : UDINT := 300;          // maximum number of bands usable in multiband
 rms
    cMaxQuantiles            : UDINT := 40;           // maximum number of quantiles
    cMinArgLog10             : LREAL := 2.3E-308;     // min value to calculate logarithm
END_VAR
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

6 Examples

General

Two examples are available for single or three-phase systems. In the first example only the highly dynamic time-based algorithms are used. In the second example a combination of time-based and frequency-based algorithms is used. This additionally enables a precise evaluation of harmonics to be made. On account of the much higher computing power required for the frequency-based algorithms, these are executed in a separate, slower task.

6.1 Time-based example

The time-based example is suitable for the highly dynamic analysis of electrical systems. Only the time-based algorithms from the Power Monitoring library are used for this.

Overview

One analysis chain is assembled in this example. The individual function block IDs are defined in the structure E_AnalysisIDs. The analysis chain begins either with an input from a bus terminal, for example the EL3783, or with the signal generator. Switching is carried out with the variable nInputSelect. The input signal is transferred to the source function block, which subsequently forwards the signal to the analysis function blocks assigned to it. These include the frequency calculation with the fbFrequency function block, the calculation of the basic values with the fbBasicValues function block and the calculation of power values with the fbPowerValues function block.

Program parameters

The most important parameters for influencing the input signal are shown in the following table.

Variable	Description	Default value
nInputSelect	Selection of the input signal	1 (signal generator)
fSetPhaseDiff	Specified phase shift	5 °
fFreqSetpoint	Specified frequency	50 Hz
fAmplitudeVoltage	Voltage amplitude	325.27 V
fAmplitudeCurrent	Current amplitude	1.414 A
bEnableHarmonics	Generation of harmonics	FALSE

Global constants

The following global constants are defined:

```
VAR_GLOBAL CONSTANT
    cOversamples   : UDINT := 10;
    cSamplerate    : UDINT := 10000;
END_VAR
```

Download

Single-phase sample program	https://infosys.beckhoff.com/content/1033/TF3650_TC3_Power_Monitoring/Resources/zip/5517982603.zip
Three-phase sample program	https://infosys.beckhoff.com/content/1033/TF3650_TC3_Power_Monitoring/Resources/zip/5517987211.zip

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

6.2 Time and frequency-based sample

This sample shows the possibilities of the highly dynamic evaluation of the time-based function blocks in combination with the frequency-based function blocks.

Overview

Two analysis chains are assembled in this sample. The individual function block IDs are defined in the structure E_AnalysisIDs. The analysis chains each begin with an input from a bus terminal, for example the EL3783, or with the signal generator. Switching is carried out with the variable nInputSelect.

The fast analysis chain for the time-based function blocks is executed only in the MAIN program. The input signal is transferred to the source function block fbSource_TimeBased, which subsequently forwards the signal to the analysis function blocks assigned to it. These include the frequency calculation with the fbFrequency function block, the calculation of the basic values with the fbBasicValues function block and the calculation of power values with the fbPowerValues function block. The buffer lengths are each equal to the oversampling factor.

The slower analysis chain for the frequency-based function blocks begins, also in the MAIN program, with the source function block fbSource_FrequencyBased. This collects the incoming data up to the configured buffer length cBufferLength and then sends them to the frequency-based function blocks for the power calculation (fbPowerValues), for the calculation of the spectrum (fbSpectrum) and for the calculation of the harmonics (fbHarmonics) in the slower program MAIN_SLOW.

Program parameters

The most important parameters for influencing the input signal are shown in the following table.

Variable	Description	Default value
nInputSelect	Selection of the input signal	1 (signal generator)
fSetPhaseDiff	Specified phase shift	5 °
fFreqSetpoint	Specified frequency	50 Hz
fAmplitudeVoltage	Voltage amplitude	325.27 V
fAmplitudeCurrent	Current amplitude	1.414 A
bEnableHarmonics	Generation of harmonics	FALSE

Global constants

The following global constants are defined:

```
VAR_GLOBAL CONSTANT
    cOversamples     : UDINT := 10;
    cSamplerate      : UDINT := 10000;
    cFFTLength       : UDINT := 4096;
    cWindowLength    : UDINT := 3200;
    cBufferLength    : UDINT := cWindowLength/2;
    cHarmonicBands   : UDINT := 20;
    cNumQuantiles    : UDINT := 2;
    cFreqMode        : LREAL := 50.0;
END_VAR
```

Download

Single-phase sample program	https://infosys.beckhoff.com/content/1033/TF3650_TC3_Power_Monitoring/Resources/zip/5517984907.zip
Three-phase sample program	https://infosys.beckhoff.com/content/1033/TF3650_TC3_Power_Monitoring/Resources/zip/5517989515.zip

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.22	PC or CX (x86, x64)	Tc3_PowerMonitoring

7 Appendix

7.1 FAQ

In this section frequently asked questions are answered, in order to facilitate your work with the TwinCAT 3 power monitoring library.
If you have further questions, please contact our support (-157).

Which characteristic values can be calculated with the TwinCAT 3 power monitoring library? [▶ 132]

Which characteristic values can be calculated with the TwinCAT 3 power monitoring library?

The following characteristic values can be calculated: RMS, mean, maximum and minimum values of current, voltage, active, reactive, distortion reactive and apparent power, as well as frequencies and frequency spectra, total harmonic distortion, and the harmonic.

7.2 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages:
http://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49(0)5246/963-0
Fax:	+49(0)5246/963-198
e-mail:	info@beckhoff.com

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49(0)5246/963-157
Fax:	+49(0)5246/963-9157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49(0)5246/963-460
Fax:	+49(0)5246/963-479
e-mail:	service@beckhoff.com