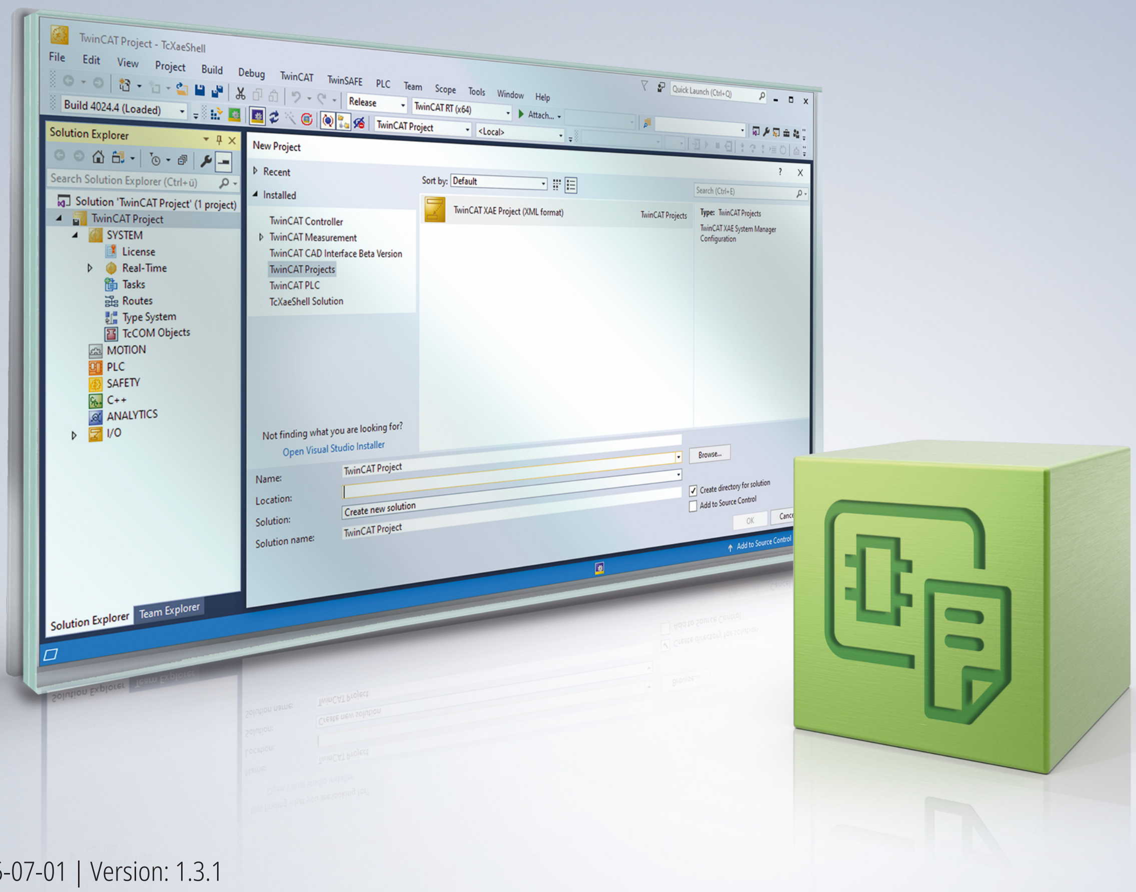
BECKHOFF New Automation Technology

Manual | EN

TE1030

TwinCAT 3 | Documentation Generation


TwinCAT Project - TcXaeShell
File Edit View Project Build Debug TwinCAT TwinSAFE PLC Team Scope Tools Window Help
Solution Explorer
Solution 'TwinCAT Project' (1 project)
TwinCAT Project
SYSTEM
License
Real-Time
Tasks
Routes
Type System
TcCOM Objects
MOTION
PLC
SAFETY
C++
ANALYTICS
I/O
New Project
Recent
Installed
TwinCAT Controller
TwinCAT Measurement
TwinCAT CAD Interface Beta Version
TwinCAT Projects
TwinCAT PLC
TcXaeShell Solution
TwinCAT XAE Project (XML format)
Not finding what you are looking for?
Open Visual Studio Installer
Name: TwinCAT Project
Location:
Solution: Create new solution
Solution name: TwinCAT Project
Solution Explorer
Team Explorer

2025-07-01 | Version: 1.3.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

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

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO® , EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

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

Copyright

Third-party trademarks

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

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

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

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

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

Warning of damage to property or environment

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

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Overview

The machinery directive stipulates that documentation must be supplied with every machine, and the product safety act also requires operating instructions for products sold. Due to these legal requirements, the documentation of machines and products is necessary for every company today. The documentation costs time, must be consistent and should always look the same, even if several people are working on it.

TwinCAT Documentation Generation enables the creation of documentation within a machine application or PLC library. To do this, the program uses specific markups within the source code comments to highlight relevant information and format it for documentation. Reference to other documents containing additional information is also possible.

Since the documentation is generated directly from the programming, it can be ensured that the application program and the documentation are consistent. Moreover, the additional effort for developers is moderate because the documentation generation is based on markups. Developers can contribute content to the documentation in TwinCAT Documentation Generation without being familiar with the typical authoring tools.

3 Installation

System Requirements

Technical data	Description
Operating system	Windows 10
Target platform	Windows
Minimum TwinCAT version	TwinCAT 3.1.4026
TwinCAT licenses	TE1030 TwinCAT 3 Documentation Generation

TwinCAT Package Manager: Installation (TwinCAT 3.1 Build 4026)

Detailed instructions on installing products can be found in the chapter Installing workloads in the TwinCAT 3.1 Build 4026 installation instructions.

Install the following workload to be able to use the product:

- TE1030 | TwinCAT 3 Documentation Generation

Licensing

TwinCAT 3 Documentation Generation is an engineering product. The licensing takes place exclusively on the engineering system. The guide to licensing a full version can be found in the TwinCAT 3 Licensing documentation.

ℹ There is no 7-day trial license for the full version of this product. However, you can test the Documentation Generation quick view without a license.

4 Basic principles

With special comments, the developer writing a program with TwinCAT 3 is able to write explanatory text directly into the TwinCAT project. By using special markups in the comments, the text can be evaluated automatically. In addition, the structure of the software is also used to show inheritance relationships and dependencies. A table of contents for the documentation is generated automatically. No further markups are required for this.

TwinCAT Documentation Generation uses markups to process documentation comments in your code and formats them into documentation whose name and location you specify in the **TE1030-Tc3DocGen** window. As output format you can select HTML, among others. Documentation Generation also automatically generates partial class diagrams to show the dependencies of code elements.

The automatic generation of a documentation is based on the following information within a TwinCAT project:

- Structural information
- Information marked up in the source code comment (markups)
- Libraries used

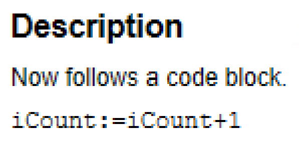
Elements hierarchy

The Description, Summary and Example markups are at the top of the elements hierarchy. These three markups are the prerequisite for using many other markups.

The following example shows how to embed the markup code into the higher-level markup description:

```
(*!<description>Now follows a code block.

<code> iCount:=iCount+1 </code>

</description>*)
```

In the generated documentation it looks like this:



Description

Now follows a code block.

```
iCount:=iCount+1
```

The documentation is automatically divided into individual sections during output, which also reflects this hierarchy:

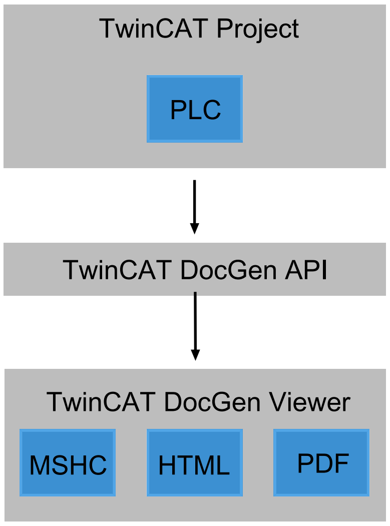
- Overview
    - Is generated automatically and also contains the table of contents.
- Summary
- Description
- Declaration
- Example

Output formats

TwinCAT 3 Documentation Generation allows you to create your documentation in three different output formats:

- HTML
- PDF
- MSHC

The integration in TwinCAT is structured accordingly as follows:


TwinCAT Project

PLC

TwinCAT DocGen API

TwinCAT DocGen Viewer

MSHC

HTML

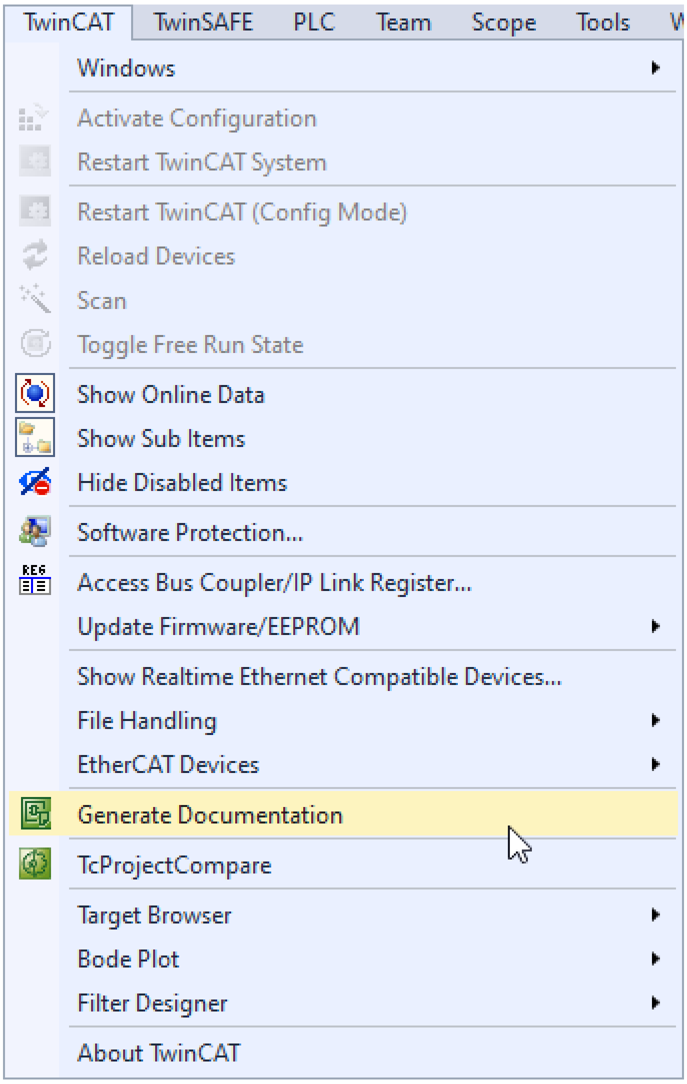
PDF

5 Quick start

The following explains how to open TwinCAT Documentation Generation and which settings you can make when creating and saving a documentation.

Open TwinCAT Documentation Generation

1. Click the **TwinCAT** tab.
2. Click **Generate Documentation**.



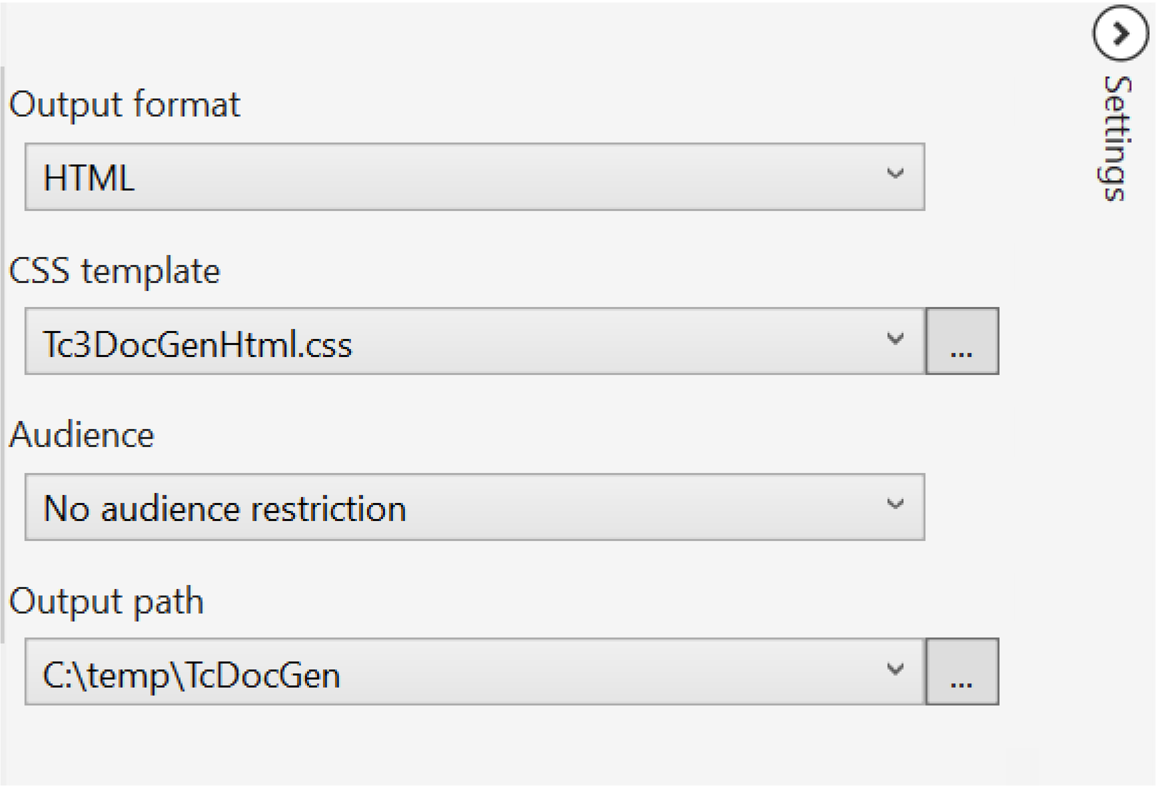
⇨ The **TE1030-Tc3DocGen** window opens.

Create and save documentation

✓ The **TE1030-Tc3DocGen** window is open.

1. Expand the settings area by clicking on ‹ .

⇨ The settings are visible.



Output format
HTML

CSS template
Tc3DocGenHtml.css

Audience
No audience restriction

Output path
C:\temp\TcDocGen

2. First select the output format.
3. Select a layout from the **CSS template** drop-down menu.
4. The **...** button opens the selection window where you can select the CSS file to be used for the formatting of the documentation. (see also Chapter Formatting the documentation [▶ 30])
5. If required, select the desired target group [▶ 22].
6. Select a new destination folder if required.
7. Click on **Generate All**.

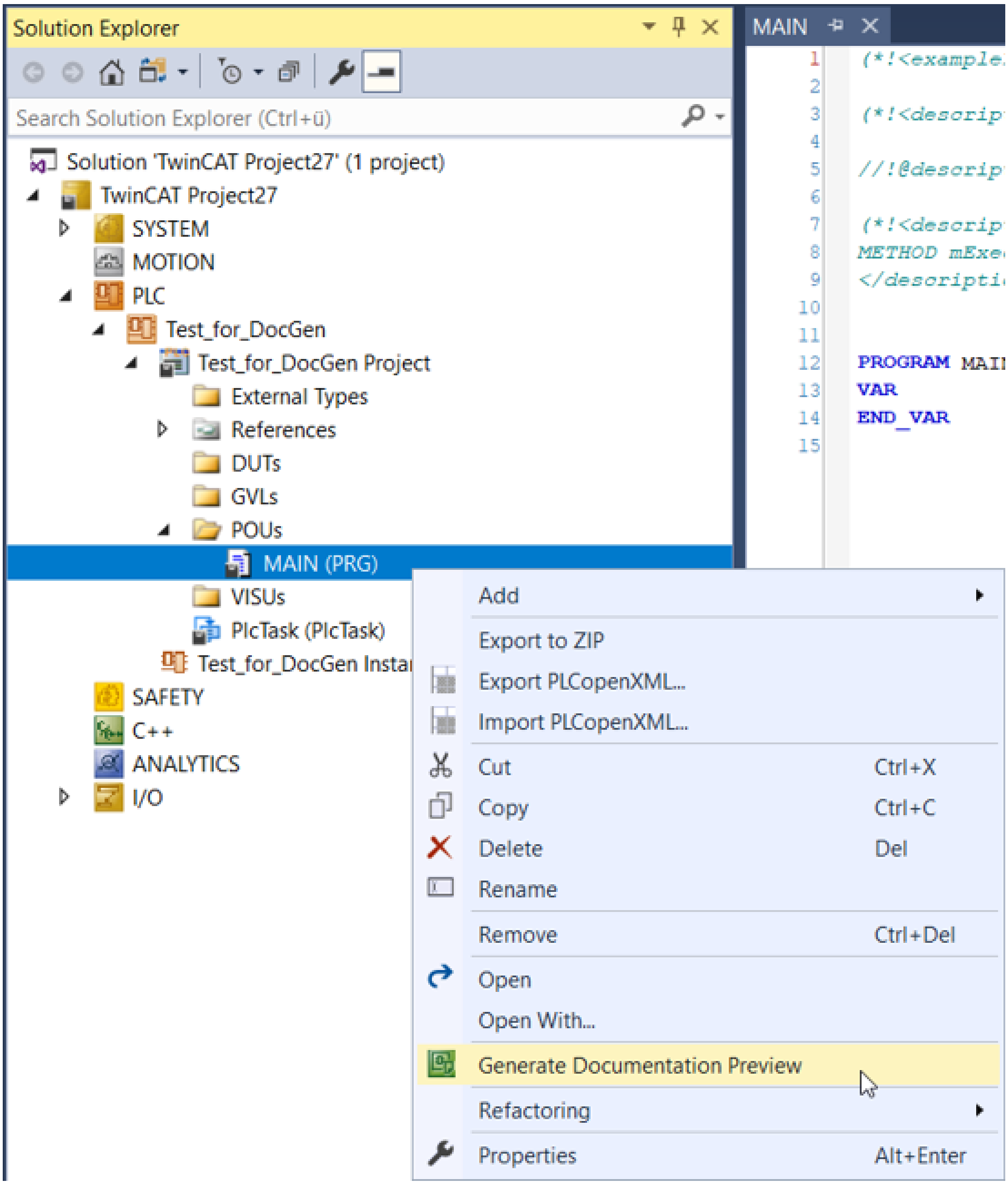
   ⇨ The documentation is stored in the selected folder.
8. You can open the folder where the documentation was saved directly using the **Open recent folder** button.

6 Documentation preview

You can display a preview of the documentation for elements that belong to the PLC.

To create the preview, proceed as follows:

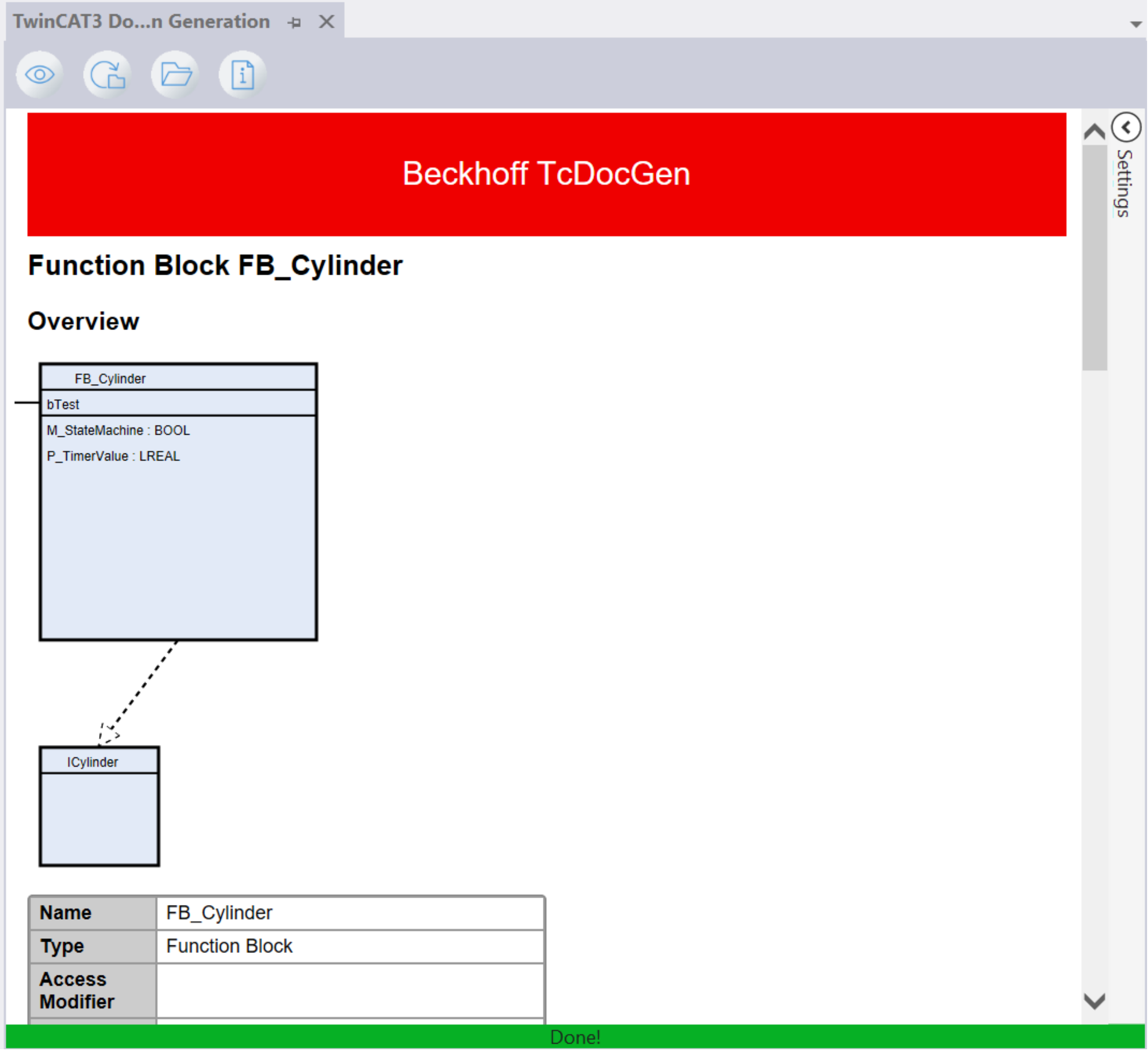
1. Right-click the desired item in the **Solution Explorer**.



```
(*!<example

(*!<descrip

//!@descrip

(*!<descrip
METHOD mExe
</descripti


PROGRAM MAI
VAR
END_VAR

```

2. Select the item **Generate Documentation Preview**.

⇨ The preview of the generated document opens in a separate window in TwinCAT.



Function Block FB_Cylinder

Overview

Name	FB_Cylinder
Type	Function Block
Access Modifier	

Update preview

The preview of the generated document can be updated in the preview window.

1. To do this, click the **Update preview** button.

⇨ The preview will be updated immediately.

7 Markups

The following chapter is about the markups that are used to create the documentation. The markups used for documentation creation are XML-based.

Example

//!<description> This is a description. </description>

Abbreviated forms of the tags

If a documentation element has only the length of one line, the abbreviated form can be used. In this form, the XML tags are written without square brackets and closing elements. Instead of brackets, a leading @ is used to mark the tags.

Example

Normal form:

//!<summary> Summary by using normal form </summary>

Abbreviated form:

//!@summary Summary by using the abbreviated form

7.1 Overview

Below you will find an overview of the markups that are needed for documentation creation.

Normal form	Abbreviated form	Context	Description
<description> </description>	@description		Text section marked as description.
<summary> </summary>	@summary		Text section marked as summary.
<example> </example>	@example		Text section formatted as an example.
<h1> </h1> - <h6> </h6>	-	description, summary, example	Headings up to level 6
<code> </code>	@code	description, summary, example	Section with source code
<literal> </literal>	@literal	description, summary, example	Plain text that is not to be interpreted.
-	$crossreference	description, summary, example	Cross-reference
<see uri=“”> </see>	@see	description, summary, example	Link within a section of text
<seealso uri=“”> </seealso>	@seealso	description, summary, example	Link outside any text section
<param name=“”> </param>	@param		Parameter value
<ul> </ul>	@ul	description, summary, example	Defines an unordered list.
<ol> </ol>	@ol	description, summary, example	Defines a numbered list.
<li> </li>	-	description, summary, example	Element of a list
<table> </table>	-	description, summary, example	Defines a table similar to HTML.
<th> </th>	-	description, summary, example	Creates a table with header.
<tr> </tr>	-	description, summary, example	Creates a table row.
<td> </td>	-	description, summary, example	Creates a table column.
<audience> </audience>	-		Defines a user group for the selected text section.
<preliminary></preliminary>	@preliminary		Marks a section as needing to be changed.
<note type=“”> </note>	-	description, summary, example	Defines safety instructions (Danger, Warning, Note).
<image uri=""></image>	-	description, summary, example	Includes an image.

7.2 Delimiters

To generate the documentation, delimiters must be inserted in the comments. These delimiters work in combination with the markups. For examples of how to use the delimiters, see the examples for each markup.

The following delimiters are allowed:

//!	Single-line documentation comment
(*! …*)	Multi-line documentation comment

7.3 Markup descriptions

The individual markups are described in more detail below and there are examples of the written-out normal form and the abbreviated form in each case.

7.3.1 Description

This markup element is used to mark a text block as a description. Description blocks can be created in various places in the code. In the documentation generated, the individual description blocks are then combined into one text block. So the documentation can be written at the places of the PLC code where it fits and does not have to be at the beginning or end.

The markup description additionally serves as a @mantel element and is accordingly required to be able to use various other elements for the documentation. Other markups can also be used in the abbreviated form of the Description markup.

Example

Normal form:

Single-line comment:

//!<description> Text block that will be marked as description. </description>

Multi-line comment:

(*!<description>This is a description.

<returns> A return value is named here. </returns>

</description>*)

Abbreviated form:

//!@description This could be a description of a code element.

7.3.2 Summary

This markup element can be used to mark a text as a summary of a code element. Within the summary you are allowed to use additional markups. In contrast to the description, however, the summary only exists once per program organization unit. All other places where you use the markup are not output in the documentation. Therefore, write the summary as a coherent text block.

Example

Normal form:

//!<summary> This is the summary of the description in normal form. </summary>

Abbreviated form:

//!@summary This is the summary of the description in the abbreviated form.

7.3.3 Example

The Example markup can be used to mark a text block as an example. Examples can be written in various places in the code. In the generated documentation, the examples are then displayed one below the other under the heading Example. Thus, the examples can be written at the places in the code where they make sense and do not have to be contiguous at the beginning or end.

Additional markups may be used within Examples. There is no abbreviated form for this markup.

Example

Normal form:

(*!<example>This is the description of the following example:

<code>iCount:=iCount+1;</code>

This example shows the increase of the $iCount variable by 1.</example>*)

7.3.4 Header

This markup element can be used to format a heading. You can structure your content with the headings up to the sixth level. To do this, count up the outline number of the markup to "6" accordingly.

The heading markup can be used individually and does not need to be included in any other markup. Besides, you can use it in any place of your documentation.

Example

<h1>This is a first level heading</h1>

<h2>This is a second level heading</h2>

7.3.5 List

These markers can be used to define a list. The following list types are available: unordered list <ul> or ordered list <ol>. Similar to HTML, the markup list elements <li> are used for the elements of both lists. Unordered and ordered lists must be embedded in a Description, Summary or Example.

Example

Normal form:

(*!<description> The following is a list

<ul>

<li> Content of element 1 </li>

<li> List item </li>

<li> List item </li>

</ul>

</description>*)

Abbreviated form:

//!@description @ul <li> list item</li> <li> list item</li> <li> list item</li>

7.3.6 Table

This markup defines a table. No abbreviated form is available for a table. You must embed tables in the Description, Summary or Example markups.

The tags define the appearance of the table. The <tr> tag creates a table row. The <td> tag creates a table column.

Example

Normal form:

(*!<description> <table>

<tr>

<td> Contents of cell 1 </td>

<td> Contents of cell 2 </td>

</tr>

<tr>

<td> Contents of cell 3 </td>

<td> Contents of cell 4 </td>

</tr>

</table> </description>*)

Create table with header row

You also have the option to create the table with header row. To do this, use the <th> markup instead of <td>.

Example

(*!<description> <table>

<tr>

<th> Header cell 1 </th>

<th> Header cell 2 </th>

</tr>

><tr>

<td> Contents of cell 1 </td>

<td> Contents of cell 2 </td>

</tr>

</table> </description>*)

7.3.7 Cross references

To create a reference to another element within the documentation, use the "$" character. If the reference target is not within the same code element, a qualified name, i.e. a full domain name, must be used. TwinCAT Documentation Generation automatically searches for this element and creates a hyperlink in the documentation. You use the cross-reference within the Description, Summary and Example markups.

Example

Normal form:

//!<description> You can find an example in `$Codeblock` </description>

Abbreviated form:

//!@description You can find an example in `$Codeblock`

7.3.8 Links

Within your documentation you can link to external addresses. The target of the link is specified by using a URI (Uniform Resource Identifier). The URI must be enclosed in normal quotes so that the XML library can process the string correctly. You must always insert links within a Description, Summary or Example. You can either insert links in the same line of text or place a line break before the link. To do this, use the "see" and "seealso" markups.

See

Use the <see> markup to create a link within a line of text.

Example

Normal form:

(*!<description> You can find more information on our homepage

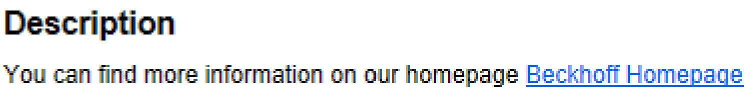
<see uri="http://www.beckhoff.de/TE1030"> Beckhoff Homepage </see>

</description>*)

Abbreviated form:

//!@description You can find more information on our homepage @see uri=http://www.beckhoff.de/TE1030 Beckhoff Homepage

In the documentation it looks like this:



Description

You can find more information on our homepage Beckhoff Homepage

Seealso

Use the <seealso> markup to create a line break before a link, placing the link in a new text section.

Example

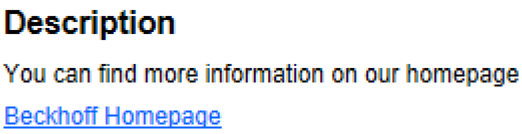
Normal form:

//!<description> You can find more information on our homepage <seealso uri="http://www.beckhoff.de/TE1030"> Beckhoff Homepage </seealso> </description>

Abbreviated form:

//! @description You can find more information on our homepage @seealso http://www.beckhoff.de/TE1030

In the documentation it looks like this:



Description

You can find more information on our homepage

Beckhoff Homepage

7.3.9 Code

The <code> markup can be used to mark a text block as source code. This markup must always be embedded in a Description, Summary or Example.

Example

Normal form:

(*!<description>Now follows a code block.

<code> iCount:=iCount+1 </code>

</description>*)

Abbreviated form:

(*!<description>Now follows a code block.

@code iCount:=iCount+1;

</description>*)

7.3.10 Literal

The <literal> markup can be used to indicate text as plain text that should not be interpreted. This function can be useful to make annotations within the source code of the documentation.

Use Literal within the markups Descriptions, Summaries and Examples.

Example

Normal form:

//!<description> This description contains text that should not be interpreted. As follows. <literal> The <b> bold </b> tag is not interpreted here, but the tags are written to the documentation. </literal> Here follows text again in which the <b> bold </b> marking is interpreted. </description>

Abbreviated form:

//@description This description contains text that should not be interpreted. As follows. @literal The <b> bold </b> tag is not interpreted here, but the tags are written to the documentation.

7.3.11 Preliminary

Allows you to specify whether a section or element is provisional and subject to change. This markup element is only an identifier and cannot contain any other internal information. Preliminary must be included in a Description, Summary or Example.

Example

Normal form:

(*!<description>Now follows a preliminary text section.

<preliminary> Preliminary </preliminary>

</description>*)

Abbreviated form:

//!@description @preliminary Preliminary

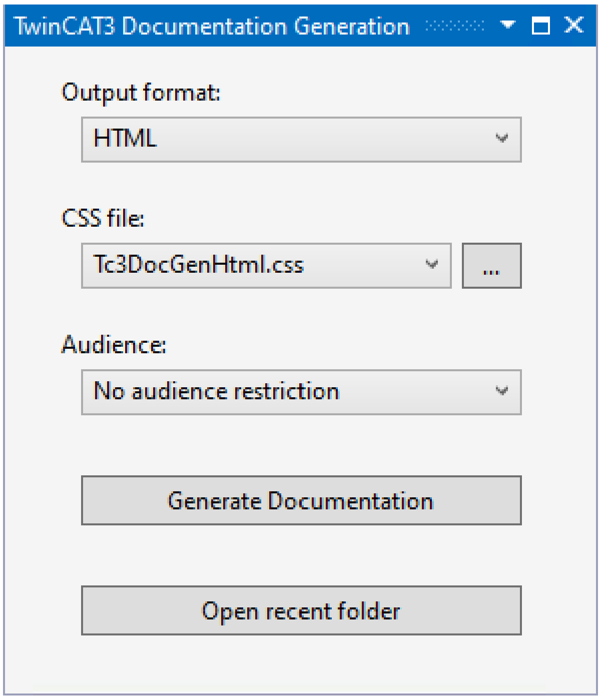
7.3.12 Audience

With this markup you can define that parts of the documentation are generated only for a defined user group. The markup must be included in a Description, Summary or Example.

The desired target group must be defined in two places.

On the one hand, it must be set during the configuration of the documentation.

1. To do this, select either "general" or "internal" from the drop-down menu **Audience**.


TwinCAT3 Documentation Generation

Output format: HTML

CSS file: Tc3DocGenHtml.css

Audience: No audience restriction

On the other hand, the target group must be specified as "audience type" within the markup.

Example

Normal form:

(*!<description>This is a normal part of the documentation.
<audience type="internal">This part is displayed for internal use only.
</audience></ description>*)

If the target group has been set in both places, the documentation will be output accordingly. Currently, the two target groups "general" and "internal" are available. If you set the target group to "internal", the text within the markup will not be output to the documentation.

7.3.13 Param

The markup Param allows to describe a parameter value of e.g. variable declarations. It can be used optionally if there is not enough space behind the parameter definition in the declaration editor, or to have the complete documentation in the same place.

The Param stands alone and does not need to be included in a Description, Summary or Example. The markup Param allows you to create a table in which the variables and their values are listed.

Example

Normal form:

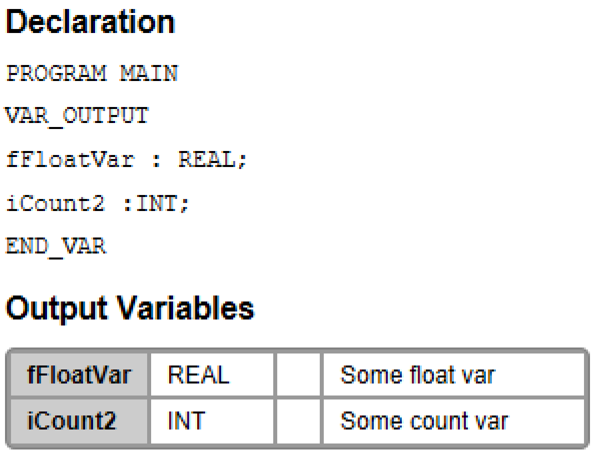
```
//!<param name="fFloatVar">Some float var</param>

//!<param name="iCount2">Some count var</param>

PROGRAM MAIN

VAR_OUTPUT

   fFloatVar : REAL;

   iCount2 :INT;

   END_VAR
```

Abbreviated form:

//!@param fFloatVar Some float var

//!@param iCount2 Some count var

In the documentation it looks like this:



Declaration

```
PROGRAM MAIN
VAR_OUTPUT
fFloatVar : REAL;
iCount2 :INT;
END_VAR
```

Output Variables

fFloatVar	REAL		Some float var
iCount2	INT		Some count var

7.3.14 Note

You use the markup Note to define a note. You must embed the entire markup in either a Description, Summary, or Example. There is no abbreviated form for the markup Note.

A note can be of the Information, Warning or Danger type. You define the type of the note within the markup as "Note Type" with a signal word. The signal word is then highlighted in color according to the danger level.

Possible types that you can define:

<note type="hazard"> - Hazard

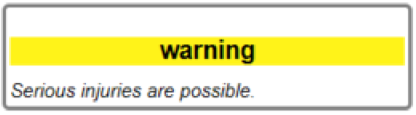
<note type="warning"> - Warning

<note type="information"> - Information

Example

Normal form:

//!<description> <note type="warning">Serious injuries are possible.</note> </description>



warning

Serious injuries are possible.

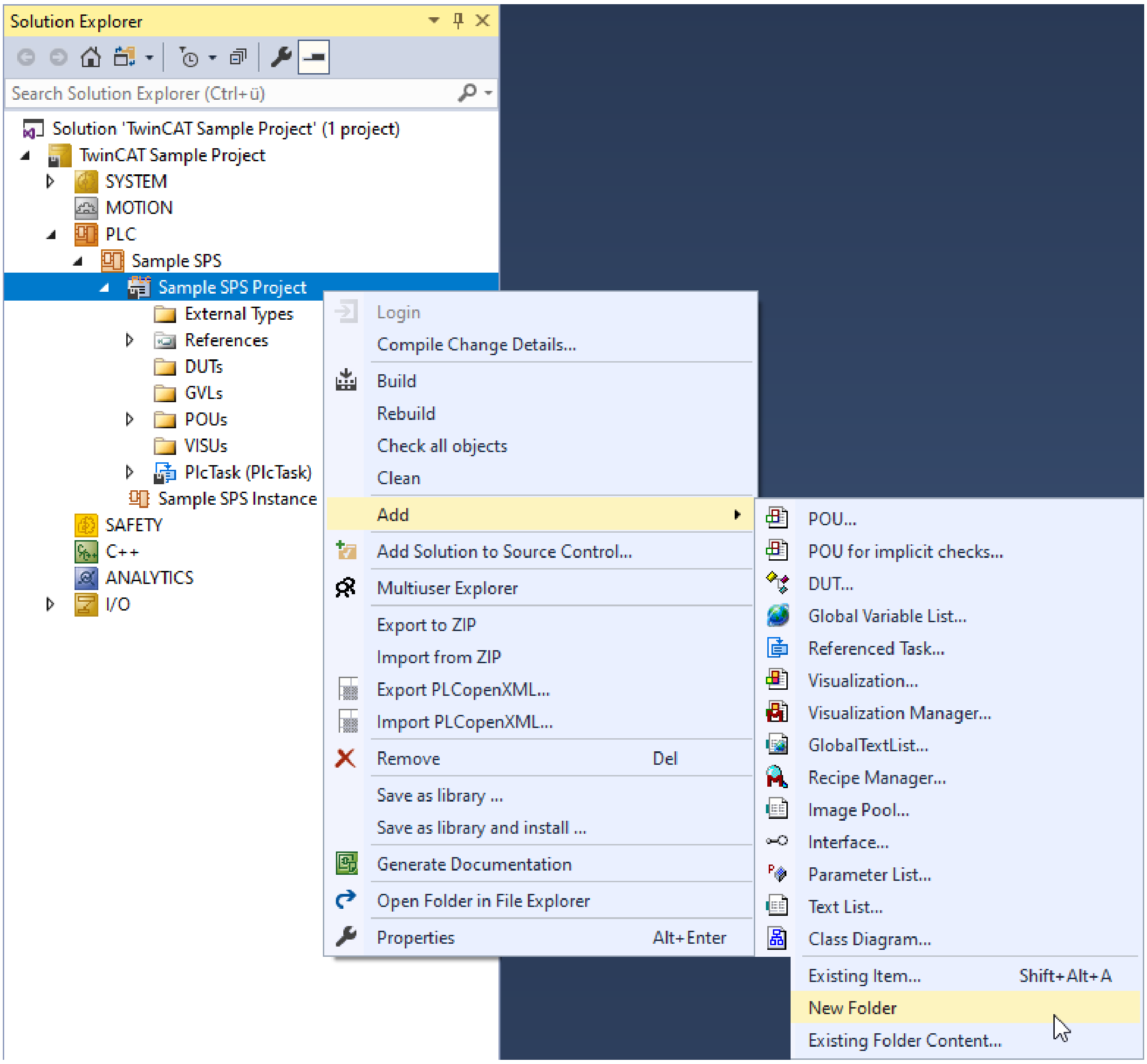
7.3.15 Image

With the help of the markup Image, which has to be inserted into a Description, Summary or Example, you can include images in your documentation.

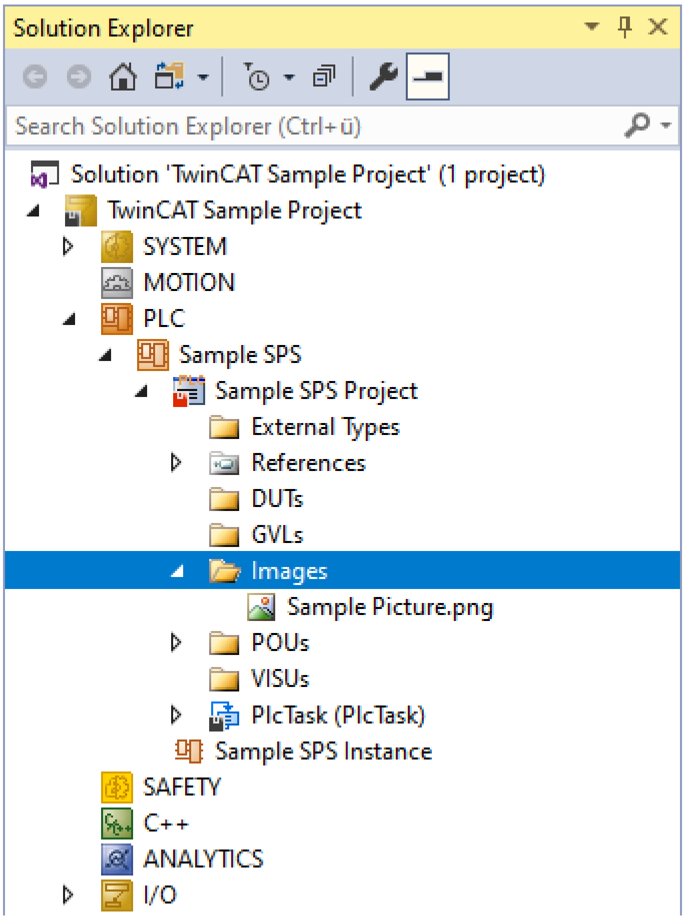
To include images, create a folder called "Images" in the folder of the PLC project. The folder "Images" may have subfolders to structure the images. All references to the images are then made within this folder, i.e. "Subfolder/ImageName".

Place the images in the PLC project folder as follows:

1. Create a new folder below your PLC project.



2. Name the folder "Images".



3. Drag and drop all the images you want to use into the folder.

⇨ You can now use the stored images in your documentation.

Include the images in the documentation as follows:

1. Include the markup Image in either the Description, Summary, or Example markup.
2. Inside the markup, specify the name of the image.

⇨ The image is output to the documentation.

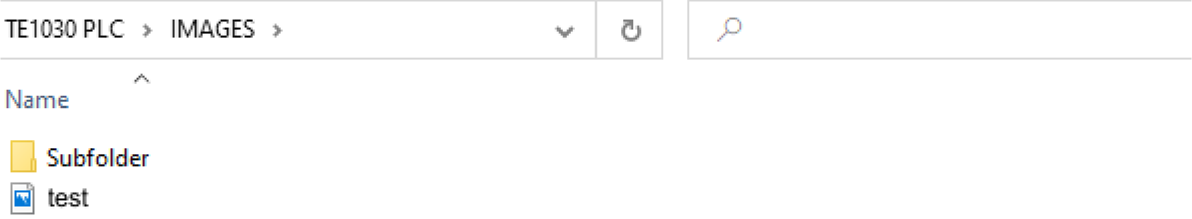
For the image to be found, the image name must match.

- Be sure to write the name of the image correctly in the markup.

Example

(*!<description>This is a description with picture.
<image uri="Sample Picture.png">Replacement text</image></description>*)

Alternatively, you can include the images in the documentation using the absolute storage path.



(*!<description> This is a description with picture.
<image uri="C:/TE1030 PLC/IMAGES/test.jpg">Replacement text</image> </description>*)

If the image cannot be displayed, a replacement text that you create is generated in the documentation instead.

Scale image size

The included images can be scaled using the "width" and "height" markups.

Normal form:

(*!<description>This is a description with picture.

<image uri="test.jpg">image title

width="10"

height="20"

</image></description>*)

7.3.16 Character formats

You can use different character formats for your documentation. These are based on the HTML convention in TwinCAT Documentation Generation.

Bold

You have the option to mark individual expressions in a text in bold. This works by using the <b> markup.

Example:

(*!<description> In the following description, a word is <b>bold</b>. </description>*)

Italic

You have the option to mark individual expressions in a text in italics. This works by using the <i> markup.

Example:

(*!<description> In the following description, a word is written in <i>italics</i>. </description>*)

Code

You have the possibility to mark single variable expressions in a text as code. This works by using the <c> markup.

Example:

(*!<description> The variable <c>iCount</c> is now formatted together with this text block. <code>iCount:=iCount+1;</code> </description>*)

Underlined

You have the option to underline individual expressions in a text. This works by using the <u> markup.

Example:

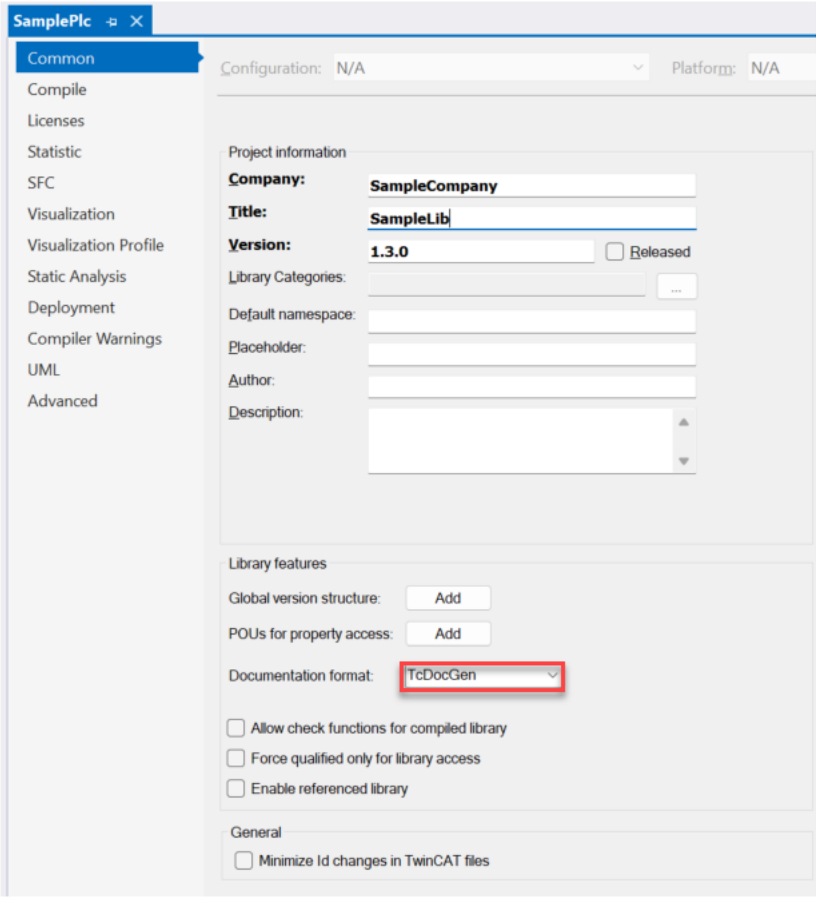
(*!<description> In the following description, a word is <u>underlined</u>. </description>*)

8 Documenting PLC libraries

You can create documentation for PLC libraries as follows:

✓ Document a library as you create it.

1. Open the properties dialog of the project via the context menu of the PLC project node.
2. Enter the name of the company, the title of the library and the version number.
3. Select "TcDocGen" as the **documentation format**.



Configuration: N/A
Platform: N/A

Project information

Company: SampleCompany
Title: SampleLib
Version: 1.3.0 □ Released
Library Categories:
Default namespace:
Placeholder:
Author:
Description:

Library features

Global version structure: Add
POUs for property access: Add
Documentation format: TcDocGen

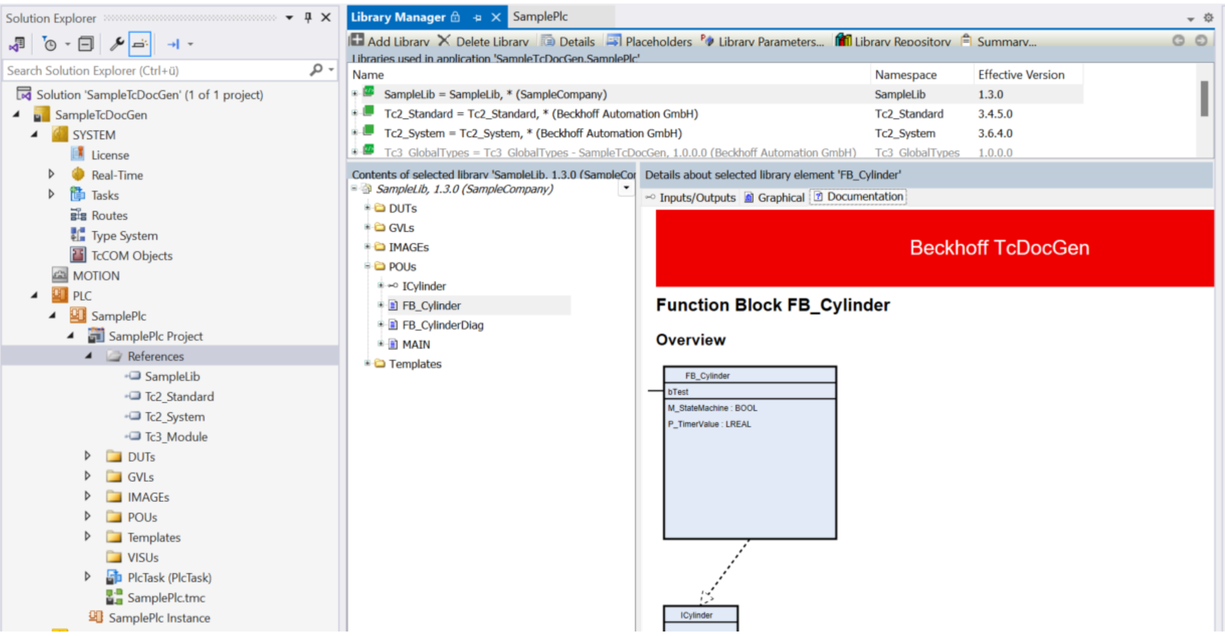
□ Allow check functions for compiled library
□ Force qualified only for library access
□ Enable referenced library

General

□ Minimize Id changes in TwinCAT files

4. Create and install the library.

⇨ The library can now be used as usual. Library users can also view the documentation in the Library Manager.



Libraries used in application 'SampleTcDocGen.SamplePlc'

Name	Namespace	Effective Version
SampleLib = SampleLib, * (SampleCompany)	SampleLib	1.3.0
Tc2_Standard = Tc2_Standard, * (Beckhoff Automation GmbH)	Tc2_Standard	3.4.5.0
Tc2_System = Tc2_System, * (Beckhoff Automation GmbH)	Tc2_System	3.6.4.0
Tc3_GlobalTypes = Tc3_GlobalTypes - SampleTcDocGen, 1.0.0.0 (Beckhoff Automation GmbH)	Tc3_GlobalTypes	1.0.0.0

Contents of selected library 'SampleLib, 1.3.0 (SampleCo

Details about selected library element 'FB_Cylinder'

Function Block FB_Cylinder

Overview

9 Formatting the documentation

The basic output format for the generated documentation is HTML. For formatting HTML files, the World Wide Web Consortium (W3C) has provided CSS (Cascading Style Sheets), a style sheet language that separates the display specifications from the content. When generating the documentation, you can specify how the generated documentation should be formatted by selecting a CSS file (see CSS template [▶ 32] selection box).

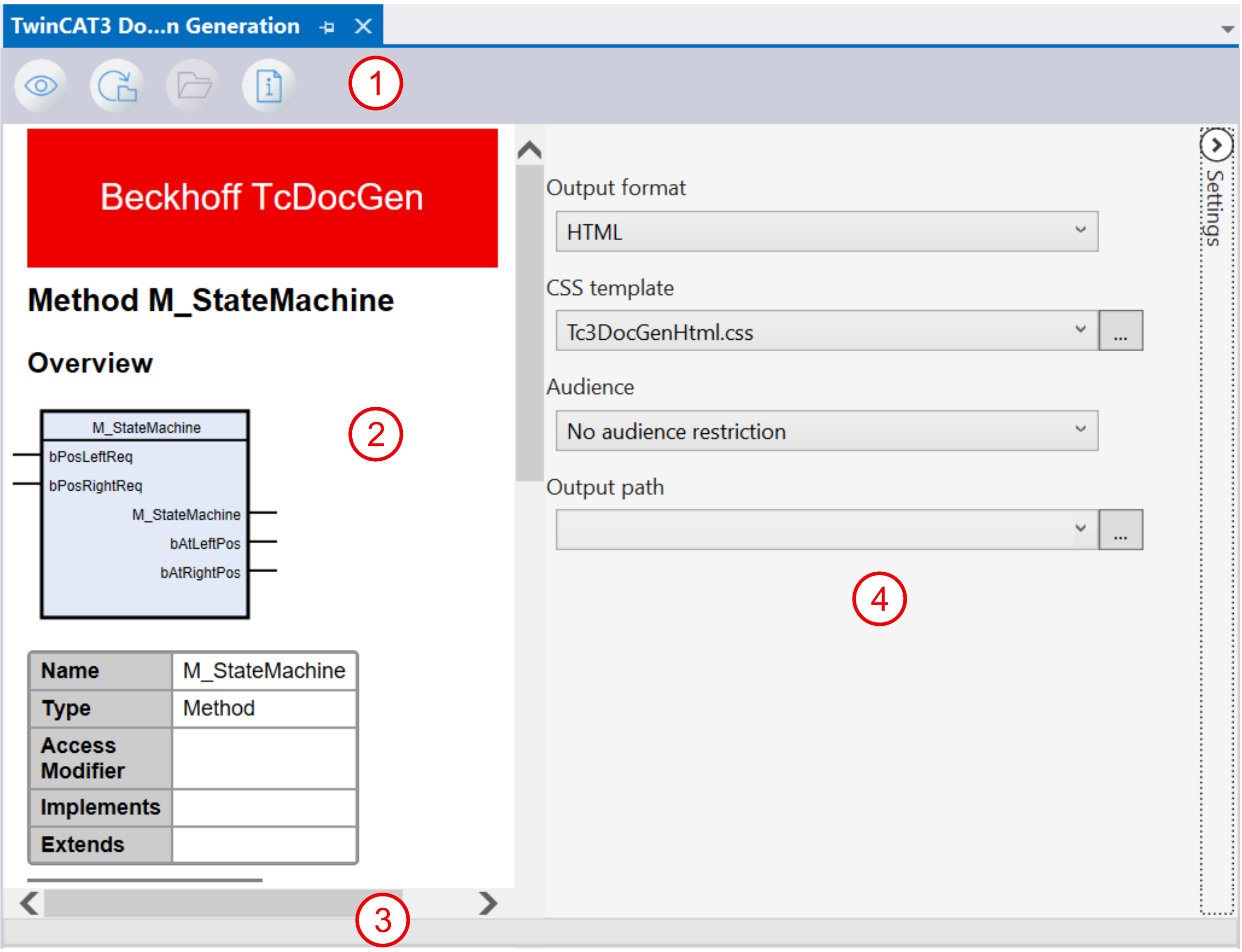
The advantage of this approach is that you can create the documentation in the PLC code without formatting instructions. For formatting, you provide one or more CSS files that adapt the generated documentation accordingly without having to make any changes to the documentation.

After installing the workload *TE1030 | TwinCAT 3 Documentation Generation*, there are two sample templates for CSS files in the folder
C:\Program Files (x86)\Beckhoff\TwinCAT\3.1\Components\TcDocGen\Templates\CSS
. You can expand or modify these according to your needs. The modified or newly created templates must be available again in this folder in order to be used.
There are two ways to make the templates available in this folder:

1. You can manually copy the changed or new templates to the folder.
2. You can use the **...** button behind the CSS templates to open a selection window and select the templates. They will be automatically copied to the folder.

10 Reference user interface

The TwinCAT 3 Documentation Generation user interface consists of the following components:



Output format: HTML

CSS template: Tc3DocGenHtml.css

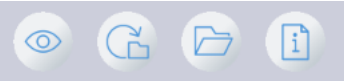
Audience: No audience restriction

Output path:

1	Menu bar
2	Preview window
3	Status bar/ progress bar
4	Settings

10.1 Menu bar

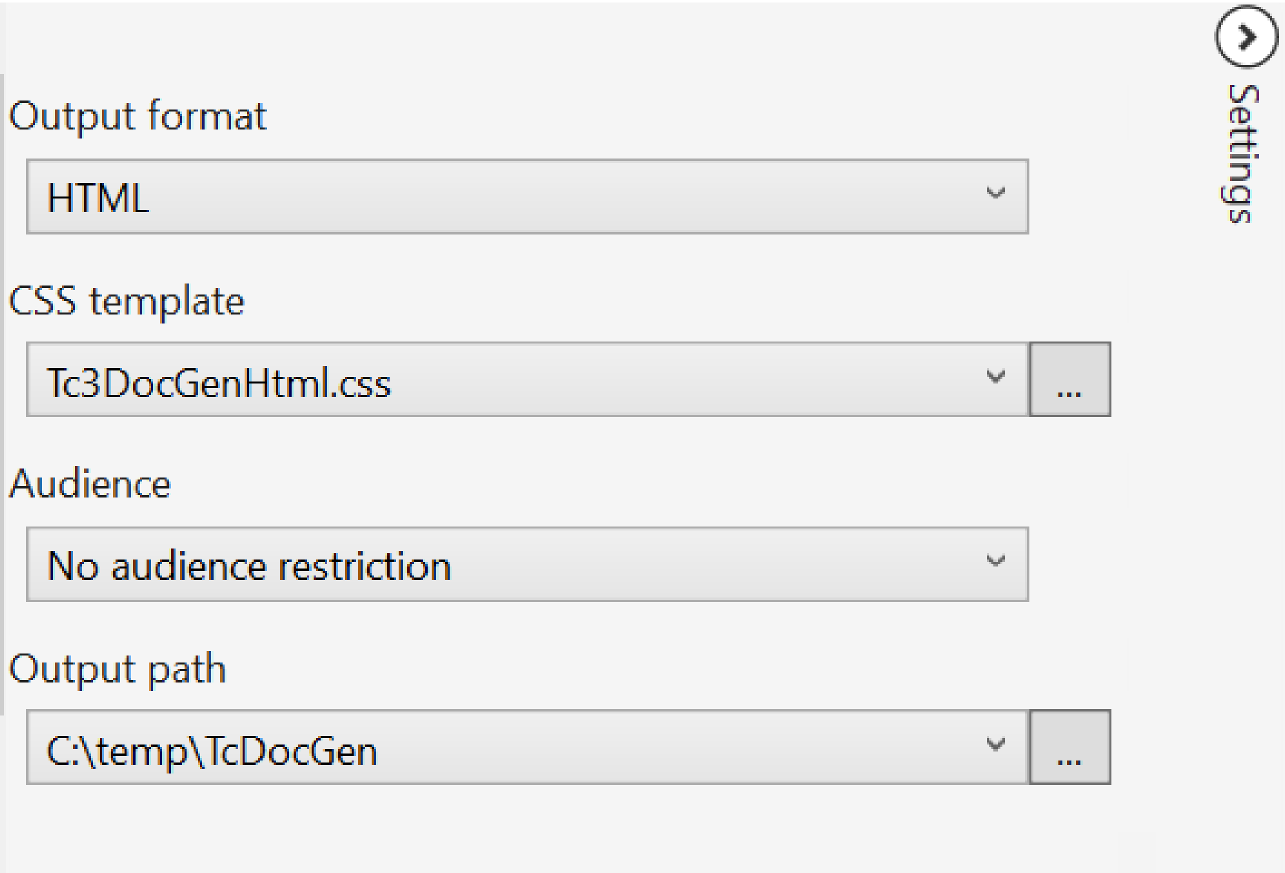
The TwinCAT 3 Documentation Generation menu bar contains the following entries:



👁	Create/update preview.
⟳	Generate complete documentation.
📂	Open the folder of the generated documentation.
ℹ	Info box

10.2 Settings

The following settings can be made and affect the generation of all documentation:



Settings

Output format: HTML

CSS template: Tc3DocGenHtml.css

Audience: No audience restriction

Output path: C:\temp\TcDocGen

Output format:	Selection of the output format for the documentation (HTML, PDF, MSHC)
CSS template:	CSS template to be used for the generated documentation. Button behind selection window to add CSS file
Audience:	User group for which the documentation is to be created.
Output path:	Folder in which the documentation will be generated.

The content of the **Audience** drop-down menu is automatically created based on the audience tags used in the documentation. If you open the documentation for the first time before creating it, it will parse (search) the documentation for the user groups you are using and may cause a slight delay.

11 Example

```
(*! @summary a program with a simple counter
<description>This is a sample program in which I will increase a counter variable. Of course the
description of this program could be longer than one line </description> *)
//! @param iCount2 variable to increase the counter
PROGRAM MAIN
VAR
    btest AT%Q*   : BOOL;   //! A sample variable not needed for this POU
    iCount2       : INT;    // This is my internal comment which will not be visible in the docu
    myPOU         : POU     //! This is a sample of a FB instanciated within this program
END_VAR

(*! <example> This sample shows how to increase the variable <c>iCount2</c>.
The following code can be used to increase $iCount:
<code>iCount2:=iCount2+1; </code></example>*)
//! @summary sample function block for the PS meeting

FUNCTION_BLOCK FB_Cylinder IMPLEMENTS ICylinder
VAR_INPUT
    bTest :BOOL;
END_VAR
VAR

//! @param tTimerValue internal TIME variable that is being accessed via property "P_Timervalue"
(Get/Set)
    tTimerValue      : TIME := T#500MS;   //internal comment
    iState           : INT;               //! state of the state machine
    bAtLeftPosFbk    : BOOL;              //internal comment
    bAtRightPosFbk   : BOOL;              //! software watchdog to reach the end position
    TimerReach       : TON;               //! software watchdog to reach the end position
    TimerStart       : TON;               //! Start timer
END_VAR

//! <description> regular FB that additionally implements an interface "ICylinder" </description>
```

12 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

Trademark statements

More Information:
www.beckhoff.com/te1030

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