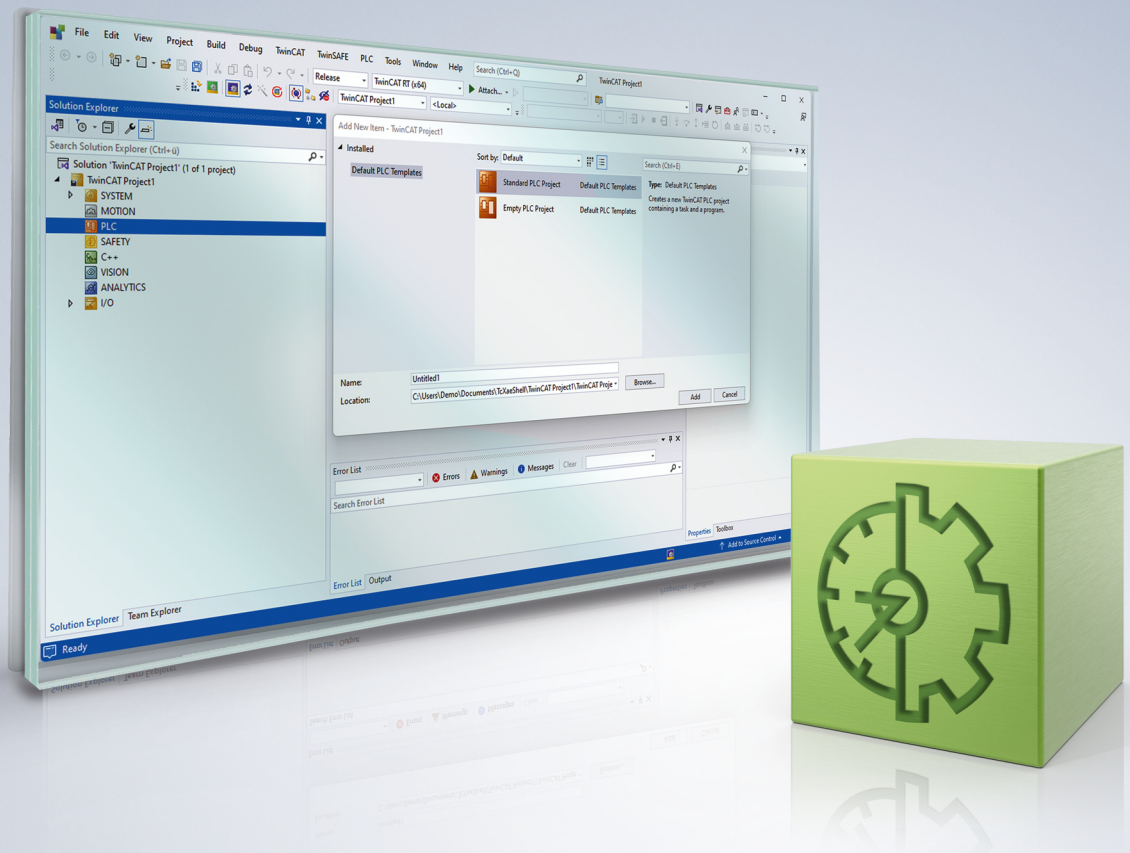
BECKHOFF New Automation Technology

Manual | EN

TE1420

TwinCAT 3 | Target for FMI



2025-12-02 | Version: 1.1.0

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

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

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

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 Functional Mock-up Interface (FMI) is a free standard for exchanging or coupling simulation models created with different simulation tools. This is often the case when the models are provided by different suppliers or domain-specific simulation tools are used to simulate the different aspects of a machine or plant (e.g. thermal simulation, control engineering simulation).

A simulation tool that supports FMI allows a model to be exported or imported as a Functional Mock-up Unit (FMU). This can be done as a model exchange or co-simulation. In the case of the former, the pure simulation model is exported. In the latter case, the associated solvers are also exported, so that the calculations of the model in the "target simulation tool" happen analogously to the initial tool.

When exporting an FMU, some simulation tools offer the choice of encapsulating the behavior of the model in a DLL or / and exporting the behavior of the model as source code. The latter option is mandatory for an FMU to be imported into TwinCAT. Thus, only FMUs whose behavior is available in the FMU as source code can be imported.

3 Installation

System requirements

The same requirements apply for the Target for FMI as for TwinCAT 3 C/C++. The detailed description of the TwinCAT 3 C/C++ requirements can be found in Chapter 4 Installation in the TwinCAT 3 C++ manual.

In the following sections, these requirements are only referred to briefly, not in detail.

On the engineering PC

- Microsoft Visual Studio 2017, 2019, or 2022 Professional, Premium, Ultimate, or Community Edition
  - Select the Visual C++ checkbox when installing Visual Studio.
- TwinCAT 3 XAE

On the runtime PC

- IPC or Embedded CX PC with Microsoft operating system based on Windows NT kernel (Win 10 and corresponding embedded versions)
- TwinCAT 3 XAR
  - TwinCAT 3.1 supports 32-bit and 64-bit operating systems. If the target is a x64 system, the created drivers must be signed. See also "x64: Driver signing" in the TwinCAT 3 C++ manual.

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:

TE1420 | TwinCAT 3 Target for FMI

Setup: Installation (TwinCAT 3.1 Build 4024)

If neither one of the supported Visual Studio versions nor a TwinCAT 3 setup is available, proceed as follows during the installation:

1. Install one of the supported Visual Studio versions.
2. Download the TwinCAT 3 setup from the Beckhoff website in the Download section. See also: Installation up to TwinCAT 3.1 Build 4024
3. Start TwinCAT 3 Setup.
4. Run the setup wizard and follow the instructions to install TwinCAT 3.
   ⇨ TwinCAT 3 is installed.
5. Then start the TE1420-TargetForFMI setup to install the Target for FMI.
6. Follow the instructions in the setup wizard.

⇨ Target for FMI is installed.

If a Visual Studio and/or a TwinCAT installation already exists but the Visual Studio version does not meet the requirements stated above (e.g. Visual Studio Shell or Visual Studio without Visual C++), proceed as follows:

1. Install a suitable Visual Studio version.
2. Execute the TwinCAT 3 setup.

⇨ TwinCAT 3 is integrated into the new or modified Visual Studio version.

Driver signing for targets with x64 operating system

To use an x64-operating system as the runtime PC, the drivers must be signed. Details can be found in the TC3 C++ manual under Driver signing.

4 Licenses

Two licenses are required to use the full functionality of the TE1420 TwinCAT Target for FMI. On the one hand, you need the TE1420 engineering license for creating TwinCAT objects from an FMU and, on the other hand, a runtime license for executing these objects during the TwinCAT runtime.

Engineering license

The license **TE1420 Target for FMI** is required on the engineering system for creating TcCOM and PLC function blocks from FMUs. Licensing is therefore carried out exclusively on the Engineering system.

There is no 7-day trial license available for this product.

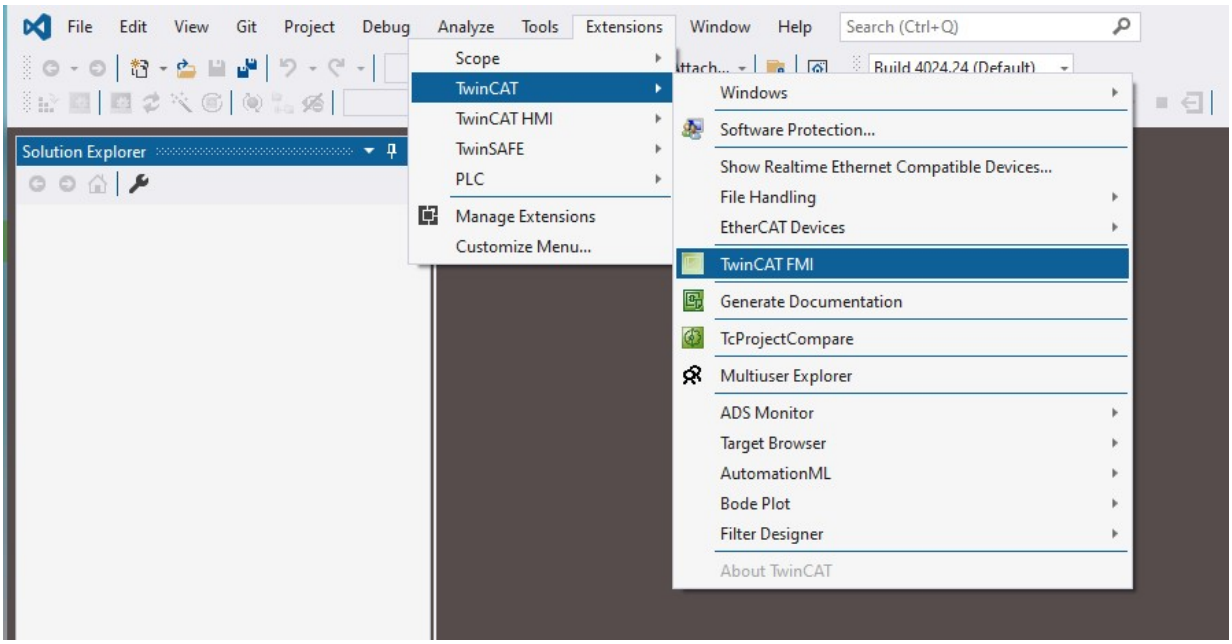
Runtime license

You need the TF1420 license to start a TwinCAT configuration with a TwinCAT object generated from an FMU. Without an activated license, you cannot start the module and therefore also the TwinCAT system. The TF1420 license is an extension of the C++ runtime licenses and therefore requires a TC1300 or a TC1210 with included PLC license as basic requirement.

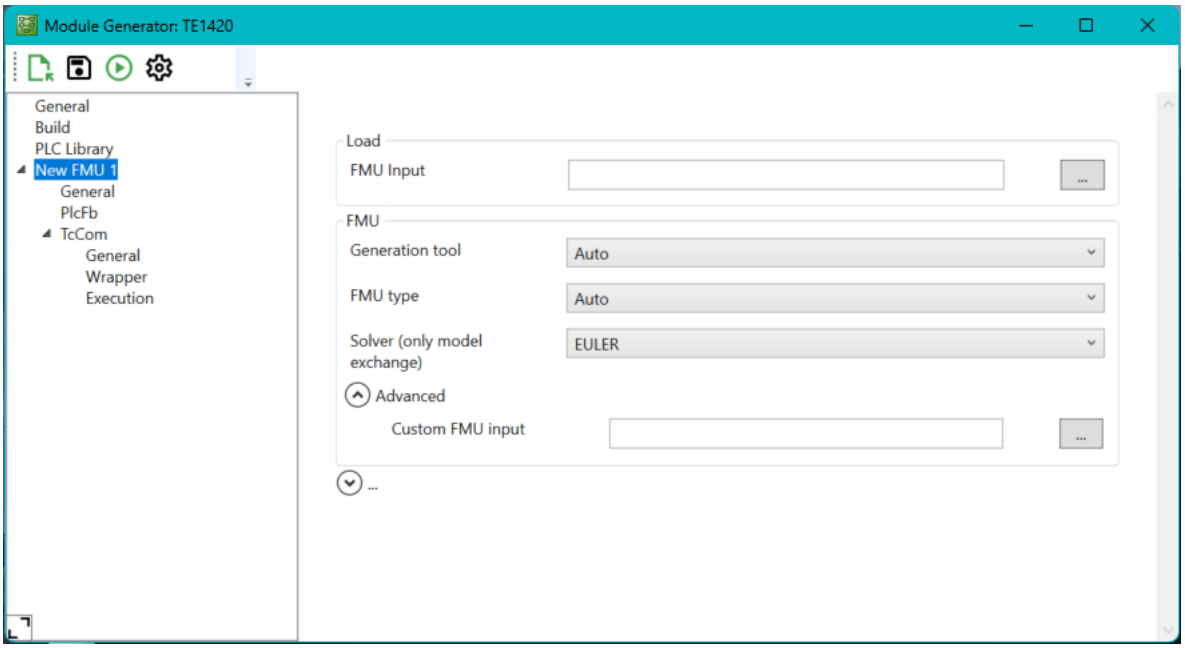
5 Quick start

Creating a TcCOM module from an FMU

1. Open the full version of a Visual Studio with integrated TwinCAT.
2. Select the option **TwinCAT FMI** in the menu **Extensions -> TwinCAT.**



⇨ The **FMI Interface Tool** window opens.



3. Use the button **… under FMU Input** to navigate to the FMU that is to be converted into a TcCOM module. If the view does not open as shown in the previous figure, first select the "New FMU 1" node in the tree view.
4. In the drop-down menu **Generation tool** select the tool from which the FMU was exported.
   Since version 2.7.2 of the TE1420, the generation tool is automatically read from the FMU. FMU type is then set to **Auto**.
5. In the **Solver** drop-down menu, select the solver to be used for the FMU simulation model calculations.
   If the model was exported as a co-simulation, then the solver is already part of the FMU. In this case, use **COSIMULATION** here.
6. Use the button **Start** to start the generation of the TcCOM model.

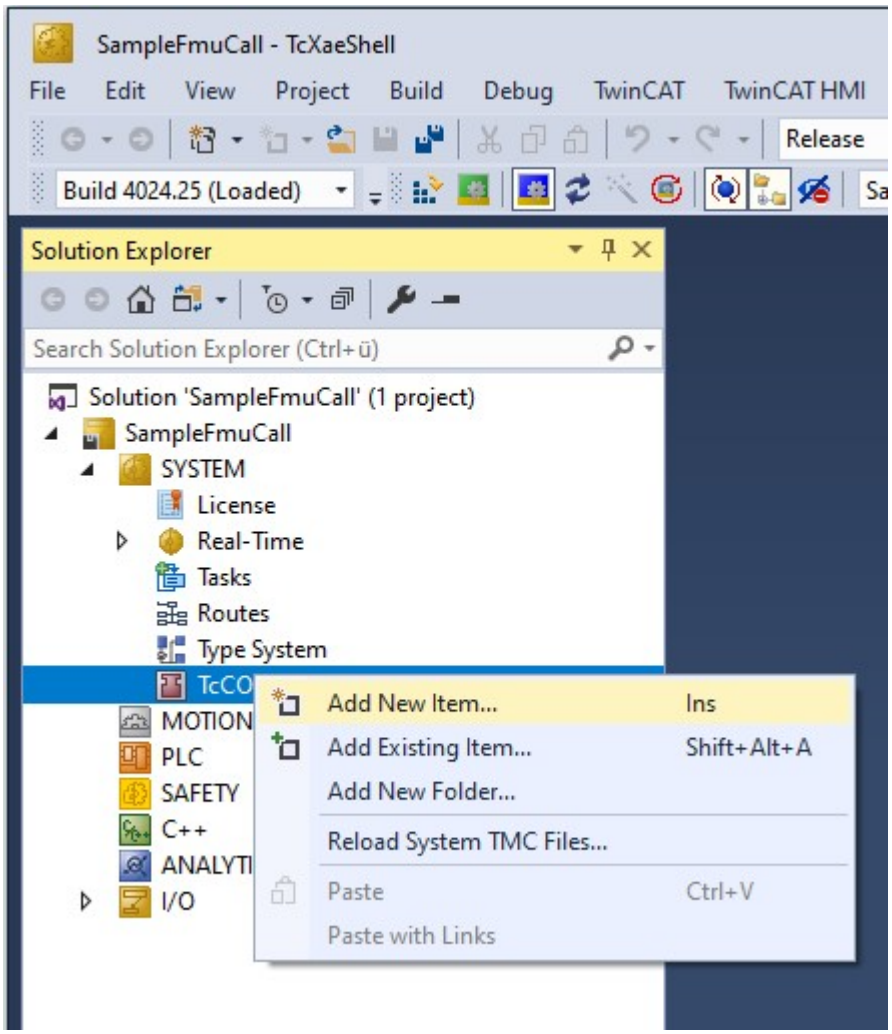
⇨ If the generation is successful, the output window of the TE1420 displays the message "Publish procedure completed successfully..."

Background to the selection of a Generation Tool

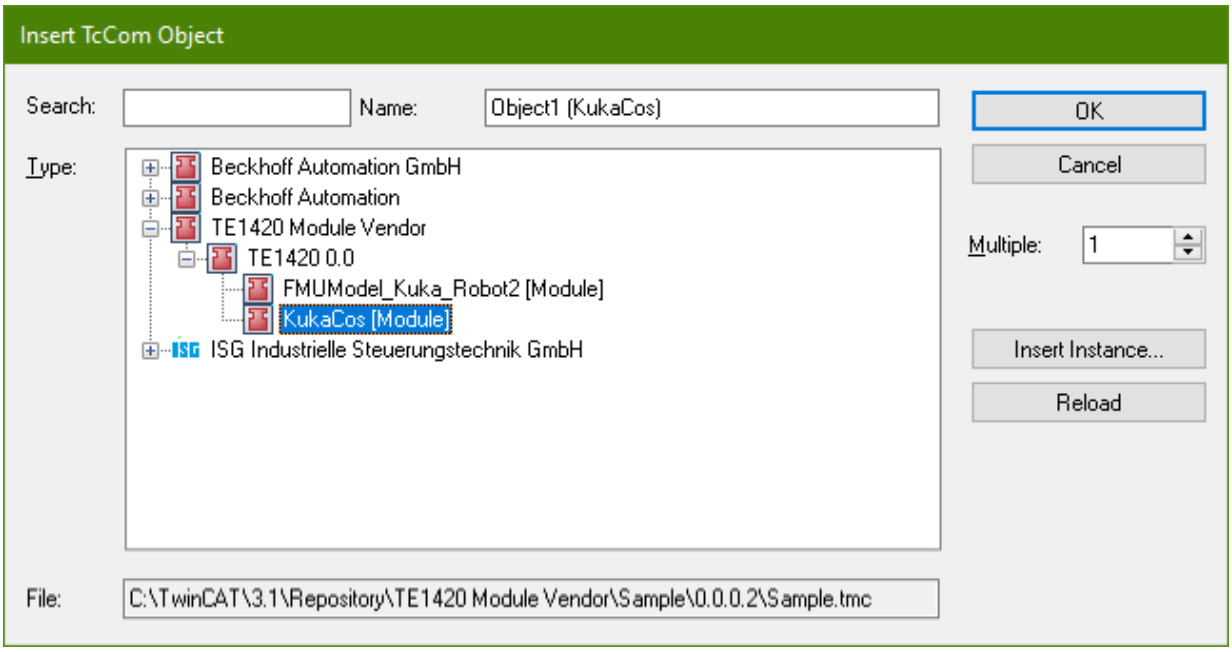
Since the export of the source code of a simulation model is handled differently by the various tools, different settings are required in the Visual Studio C++ project, which is automatically generated and used during conversion, depending on the tool. For tools already supported by TE1420, the configuration of these settings is done automatically with the selection of the Generation tool. Further tools can also be connected by yourself (see chapter Integration of additional tools [▶ 38]).

Integrating the TcCOM module in a TwinCAT project

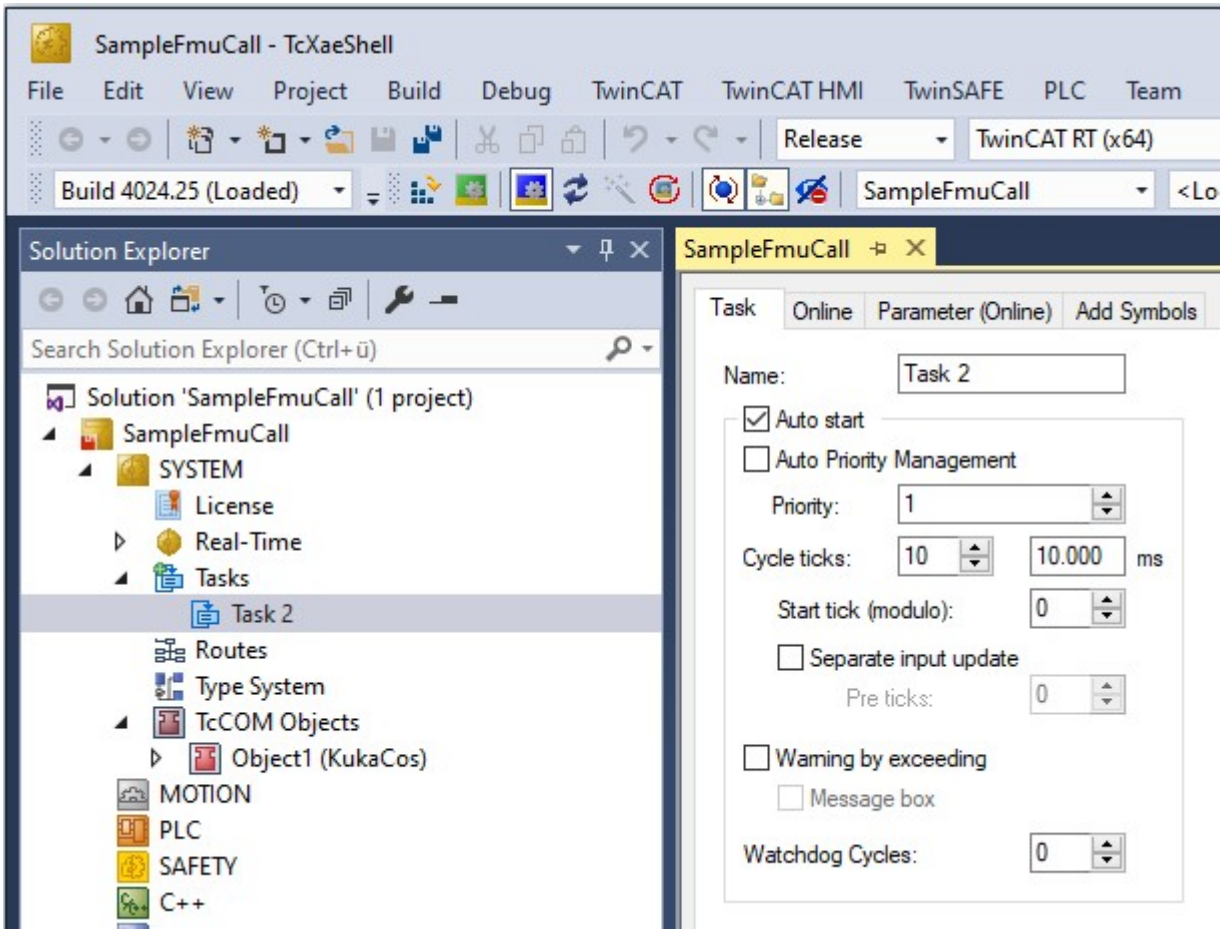
1. Open TwinCAT (TwinCAT XAE or TwinCAT in a Visual Studio environment).
2. Instantiate a new TcCOM object.



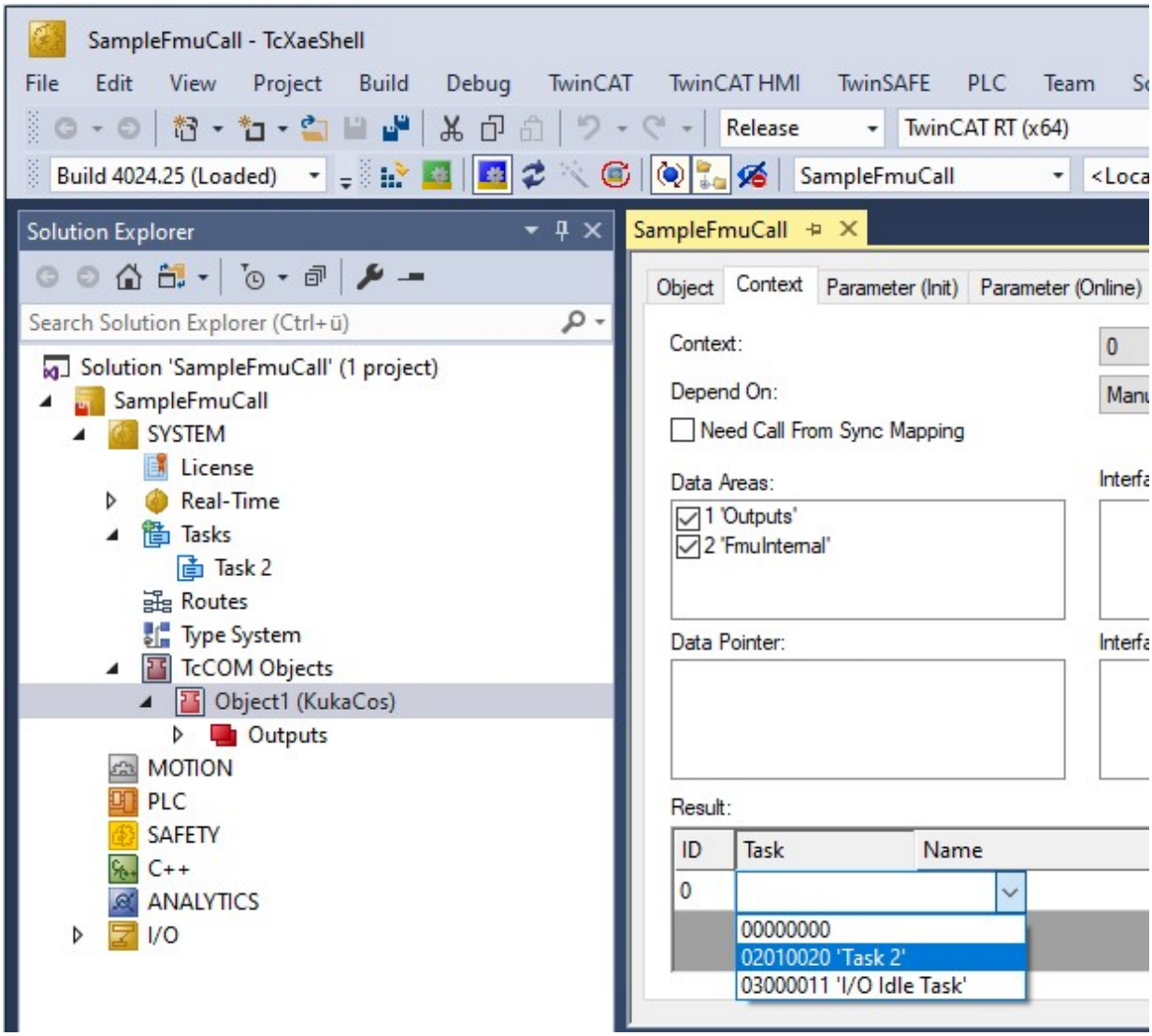
3. Select the desired object.



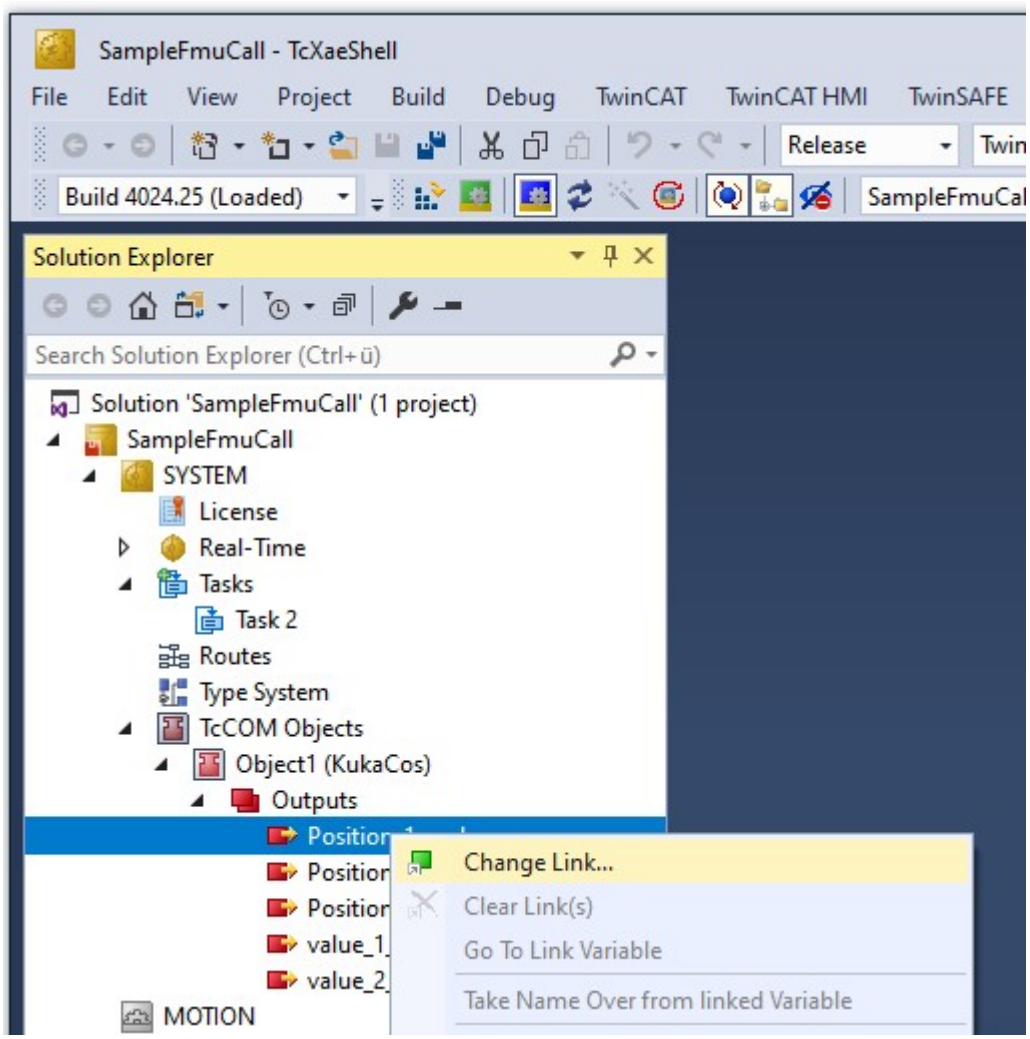
4. Create a cyclic task.



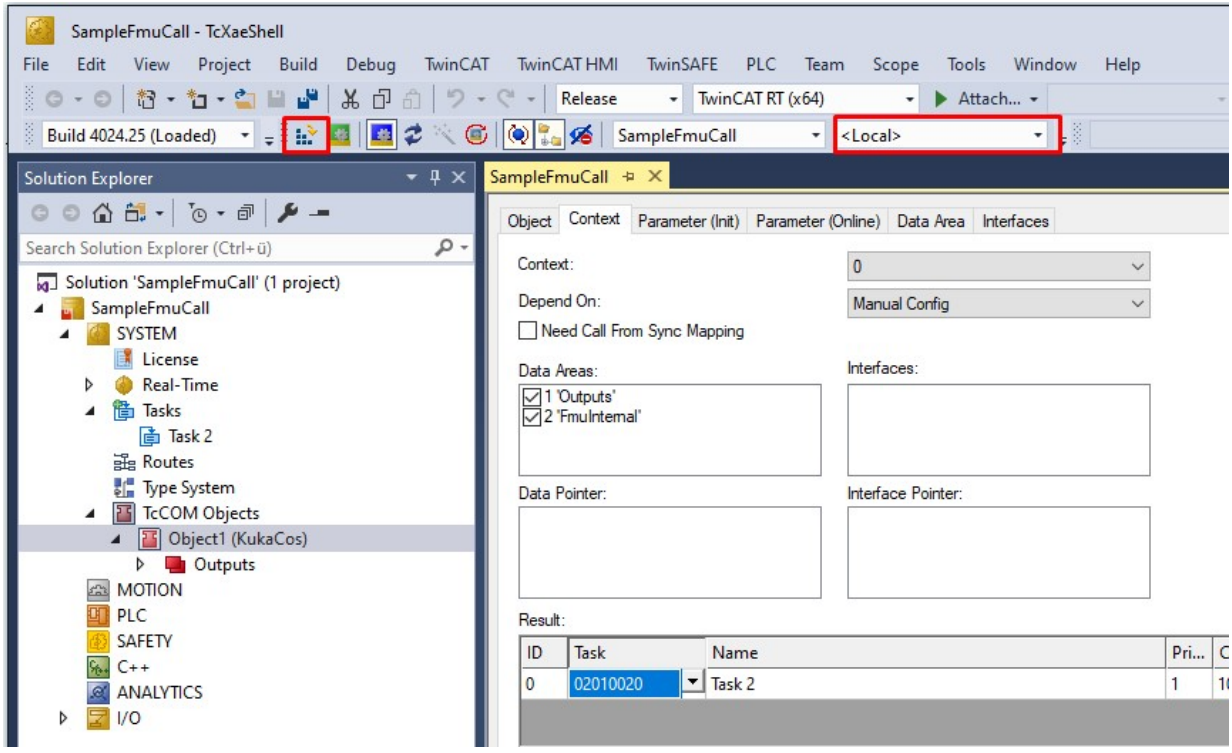
5. Assign the created task to your TcCOM instance.



6. Link the inputs and outputs of the module as required.



7. Select the target system and activate the configuration.



⇨ The TcCOM module has now been integrated into your TwinCAT project.

Use FMI Tool Standalone

In addition to the version integrated in Visual Studio, the FMI Interface Tool can also be used in standalone mode. An installed Visual Studio is also required when using the standalone FMI Interface Tool.

- Start the executable file FMUConfigurationApp.exe in the directory *C:\Program Files (x86)\Beckhoff\TwinCAT\Functions\TE1420-TargetForFmi\GUI* to use the FMI Interface Tool Standalone.
- Then continue with step 3 as described above.

6 Setting up driver signing

Create an OEM certificate level 2

The TwinCAT objects generated using the Target for FMI are based on a tmx driver (TwinCAT Module Executable), just like TwinCAT C++ objects. The drivers must be signed with an OEM certificate level 2, so that they can be loaded on the runtime PC during the TwinCAT runtime.

See the following links for detailed documentation on how to create an OEM certificate for driver signing:

- General documentation on OEM certificates
- Application-related documentation for tmx driver signing

The most important facts in brief:

- You can create your own certificate. To do this, go to Visual Studio at:
  Menu bar > TwinCAT > Software Protection...
- You need an OEM certificate Crypto Version 2 (option: *Sign TwinCAT C++ executables (*.tmx)*).
- All drivers (for 32-bit and for 64-bit operating systems) must be signed.
- Drivers can also be created without signing and signed afterwards.
- For testing purposes in the development phase, a non-countersigned certificate is sufficient.
- Countersigned certificates can be ordered free of charge from Beckhoff (TC0008).

Use of an OEM certificate level 2 for driver signing

There are three possible variants for signing tmx drivers.

1. You can set a default certificate on an engineering PC that is always used for TwinCAT C++ and Target for FMI unless you explicitly specify a different certificate.
2. You can explicitly name a certificate for each build operation.
3. You can build without a certificate and sign afterwards with the TcSignTool.

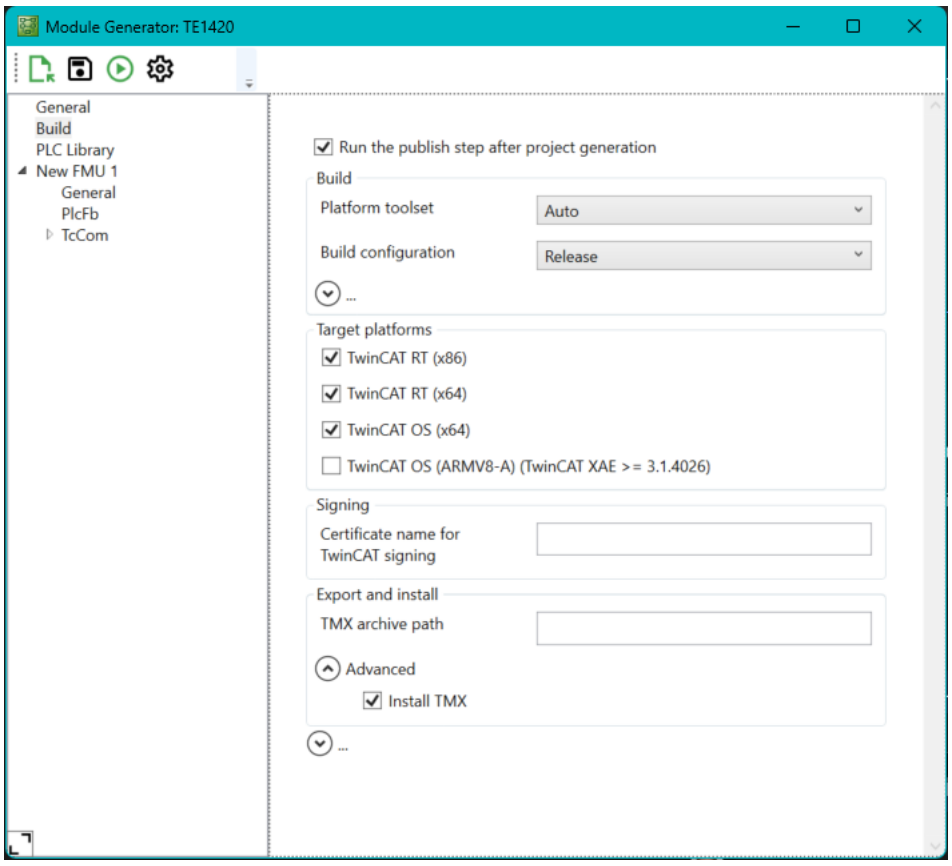
For **Variant 1** use a Windows environment variable. Create a new environment variable at **User > Variables** with:

Variable: *TcSignTwinCatCertName*

Value: Name of the desired certificate
(Available certificates are located at *TwinCAT\3.1\CustomConfig\Certificates*).

For **Variant 2** you do not have to make any further settings in advance. Before each build process, you can define a certificate of your choice for precisely this build process.

TwinCAT FMI > Build > Certificate name for TwinCAT signing



For **Variant 3** you can use the TcSignTool. The TcSignTool is a command line program located in the path *C:\TwinCAT\3.x\sdk\Bin*. With `tcsigntool /?` or `tcsigntool sign /?` you get help how to use the tool concretely.

```
TcSignTool sign /f "C:\TwinCAT\3.1\CustomConfig\Certificates\ MyCertificate.tccert" /p MyPassword
"C:\TwinCAT\3.1\Repository\TE140x Module Vendor\ModulName\0.0.0.1\TwinCAT RT (x64)\MyDriver.tmx"
```

For **variants 1 to 3**, the associated password must be stored in the system in addition to specifying the certificate with the TcSignTool. The password should not be entered in the source code for security reasons. With the TcSignTool you can store passwords belonging to your certificates encrypted in the registry of the Windows operating system.

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

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

The password is deleted with the following parameters:

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

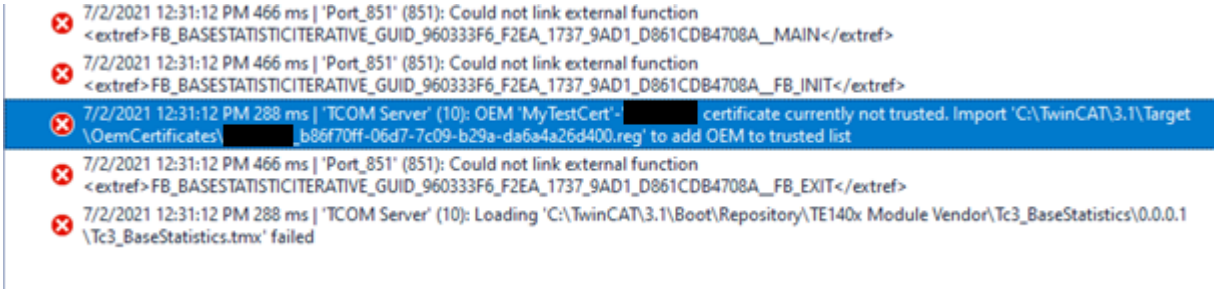
The unencrypted password is stored under:

```
HKEY_CURRENT_USER\SOFTWARE\Beckhoff\TcSignTool\
```

Behavior of the TwinCAT runtime

If a TwinCAT object with signed driver created by means of the Target for FMI is used in a TwinCAT solution and loaded to a target system with **Activate Configuration**, please note the following:

Each TwinCAT runtime (XAR) has its own white list of trusted certificates. If the certificate used for signing is not included in this white list, the driver will not be loaded. A corresponding error message is output in TwinCAT Engineering (XAE).



The error message contains the instruction to execute a registry file, which was automatically created on the target system, on the target system as administrator. This process adds the used certificate to the white list.

Registry file is only dependent on the OEM certificate

The registry file can also be used on other target systems. It only contains information about the OEM certificate used and is not target system dependent.

If you use a non-countersigned OEM certificate for signing, you must also put your target system into test mode. To do this, run the following command as an administrator on the target system:

```
bcdedit /set testsigning yes
```

If you are using a countersigned OEM certificate, this step is not necessary.

7 Parameterization of the module generation

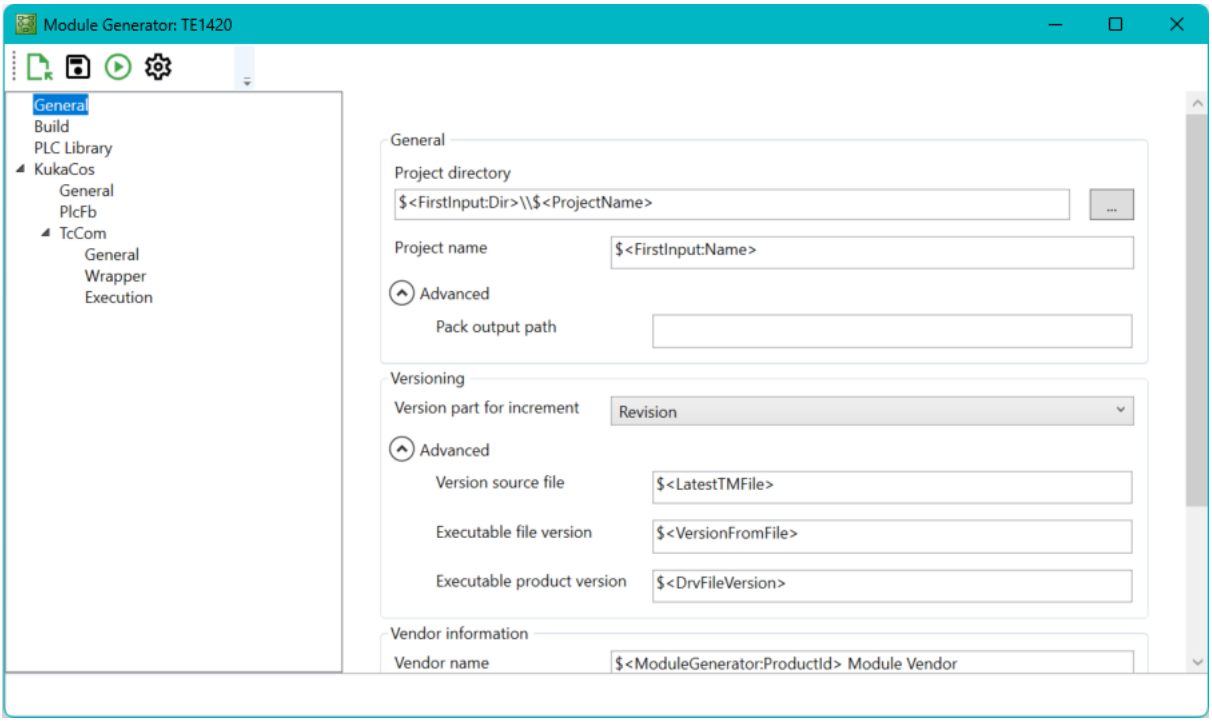
A number of settings are available for importing FMUs, which influence the generation of a TwinCAT runtime object (TcCOM modules / PLC function block). These will be explained in this chapter.

7.1 Creation of versioned drivers

Each runtime object imported from an FMU into TwinCAT 3 automatically contains version information. This makes it possible to transfer several versions of the same model into TwinCAT runtime objects. These can then be instantiated version-selectively in TwinCAT.

Setting the version control

Under the tab **General** you can set for the entries **Version source file** and **Version part for increment** how the version control of the runtime modules should be done.



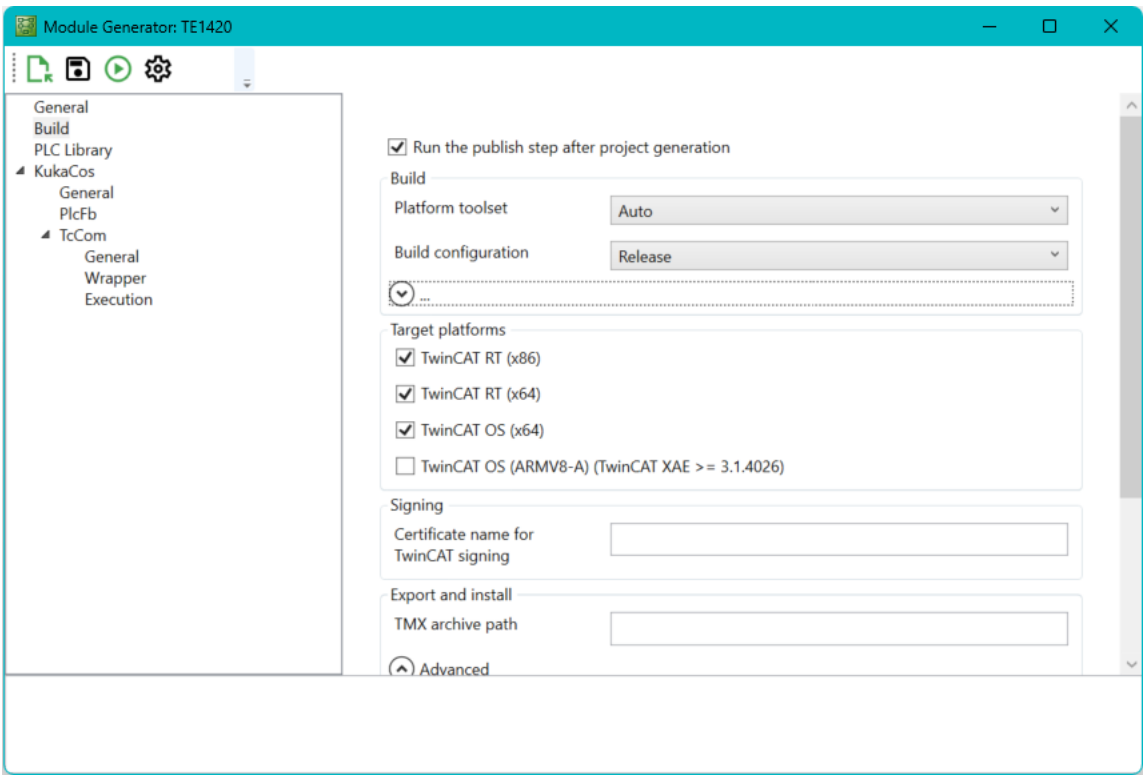
The basic version on which a version update is to be created is specified via **Version source file**. In the standard case $<LatestTMFile> is specified there. This searches for the last available version of the model on the local engineering PC and then uses this as the basis for the version increment. The version number consists of four digits, e.g. 1.0.3.2 or 2.12.123.14. Each digit can be incremented separately according to the scheme: <Major>.<Minor>.<Build>.<Revision>. If *None* is selected, no version update takes place and the last version on the engineering PC is overwritten.
Via **DrvFileVersion** you can specify a fixed version. To do this, enter the target version in the input field.

7.2 Creating TMX archives

In order to use the created TwinCAT objects in a project, they must be available on the engineering system in the corresponding repositories/folders. The corresponding copying to these folders can be done manually, but this is error-prone. The solution to this is the creation of a TMX archive. This is a self-extracting exe archive that copies the files into the appropriate directories so that they can be found by the TwinCAT Engineering environment.

Procedure:

1. Open the tab **Build**.



2. In the **TMX Archive** field, enter the path and name of the TMX archive to be created with the next build.

⇨ In the specified directory you will find a self-extracting archive containing the tmx files.

You can also use wildcards when specifying the path and name:
e.g.: *C:\temp\[Date]-[Time]-[LibName][LibVersion].exe*

This creates, for example, a TMX archive *2022-06-27-150403-Sample0.0.0.7.exe*, which can then be copied to any location on an engineering system and executed.

7.3 Bundling of several models in one TwinCAT driver

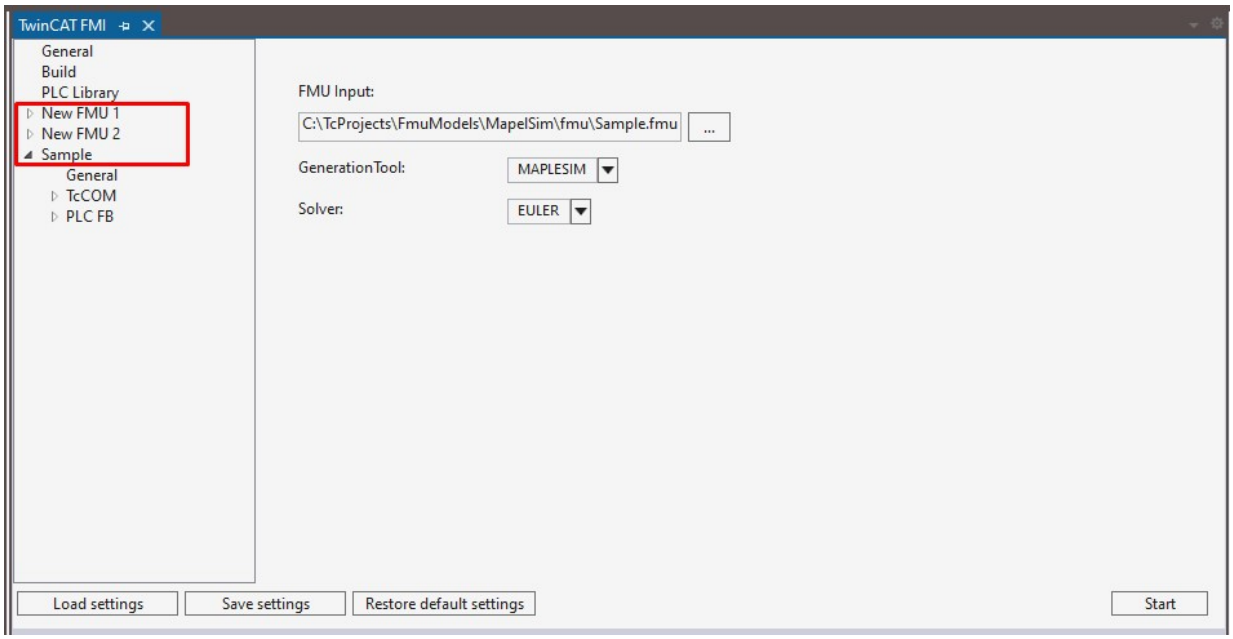
With the help of the TE1420 it is possible to bundle several models in one TMX driver respectively in one PLC library. This function can be used, for example, to make a collection of models (e.g. automation technology components) available in a library.

Proceed as follows:

✓ You have already created an FMI project with a configured FMU. See step 2 in the chapter Quick start [▶ 11].

1. Right-click on a free area in FMI Project Construction.
2. Select the option **Add FMU**.

3. Configure these FMUs as needed and repeat these steps until all FMUs you need are included and configured in the FMI project.



4. Press the button **Start**.

⇨ The desired FMUs are bundled.

When exporting FMUs, some export tools use static variables internally. For this reason, multiple FMUs from the same tool cannot be packed into one driver or library for such tools. For an overview of tools that do not support bundling, see the chapter FAQ [▶ 39].

8 Execution of modules in TwinCAT

The following chapter describes the ways in which modules can be executed in TwinCAT.

8.1 Execution as TcCOM modules

The execution of a TcCOM module created using Target for FMI is done as described in chapter Quickstart [▶ 11].

8.2 Execution within the PLC

With the help of the Target for FMI it is not only possible to create a TcCOM module that can be instantiated directly in the TwinCAT project. You can also create a PLC library that allows you to call up the behavior of a model imported via Target for FMI within the PLC.

Calling the imported model in the PLC works as follows:

1. Create a PLC project.
2. Add the generated library to the PLC project.

8.3 Exception Handling

Floating point exceptions can occur during runtime when processing a TcCOM module in TwinCAT that was generated with the aid of the TE1420. This can happen, for example, when an unexpected value (e.g. initial value) is passed to a function. The handling of such exceptions is described below.

What is a floating point exception?

A floating point exception occurs when an arithmetically not exactly executable operation is instructed in the floating point unit of the CPU. IEEE 754 defines these cases: *inexact, underflow, overflow, divide-by-zero, invalid-operation*. If one of these cases occurs, a status flag is set, which indicates that the arithmetic operation cannot be executed exactly. It is further defined that each arithmetic operation must return a result – one that in the majority of cases leads to the possibility of ignoring the exception.

For example, a division by zero results in +inf or -inf. If a value is divided by inf in the further code, this results in zero, so that no consequential problems are to be expected. However, if inf is multiplied or other arithmetic operations are performed with inf, these are *invalid operations,* whose result is represented as a Not-a-Number (NaN).

How does the TwinCAT Runtime react in case of exceptions?

TwinCAT C++ Debugger not active

The following explanations only apply if the C++ debugger is **not** activated on the TwinCAT runtime system. When the C++ debugger is enabled, exceptions are caught by the debugger and can be handled, see Debugging.

Default behavior

Default setting in TwinCAT is that at "divide-by-zero" and "invalid-operation" the execution of the program is stopped and TwinCAT issues an error message.

Task setting: Floating Point Exceptions

This default setting can be changed on the level of each TwinCAT task. If the checkbox "Floating Point Exception" is unchecked, an exception **does not** lead to a TwinCAT stop and **no** error message is issued. This setting is then valid for all objects that are called by this task. As a consequence, care must be taken in the application that NaN and inf values are handled accordingly in the program code.

Check for NaN and Inf

If, for example, a NaN is passed on via mapping to a TwinCAT object that has activated floating point exceptions, an arithmetic operation with NaN naturally leads to an exception in this object and subsequently to a TwinCAT stop. Therefore, NaN or inf must be checked directly after mapping. In the PLC, corresponding functions are available in the Tc2_Utilities library, e.g. LrealIsNaN.

Try-Catch statement

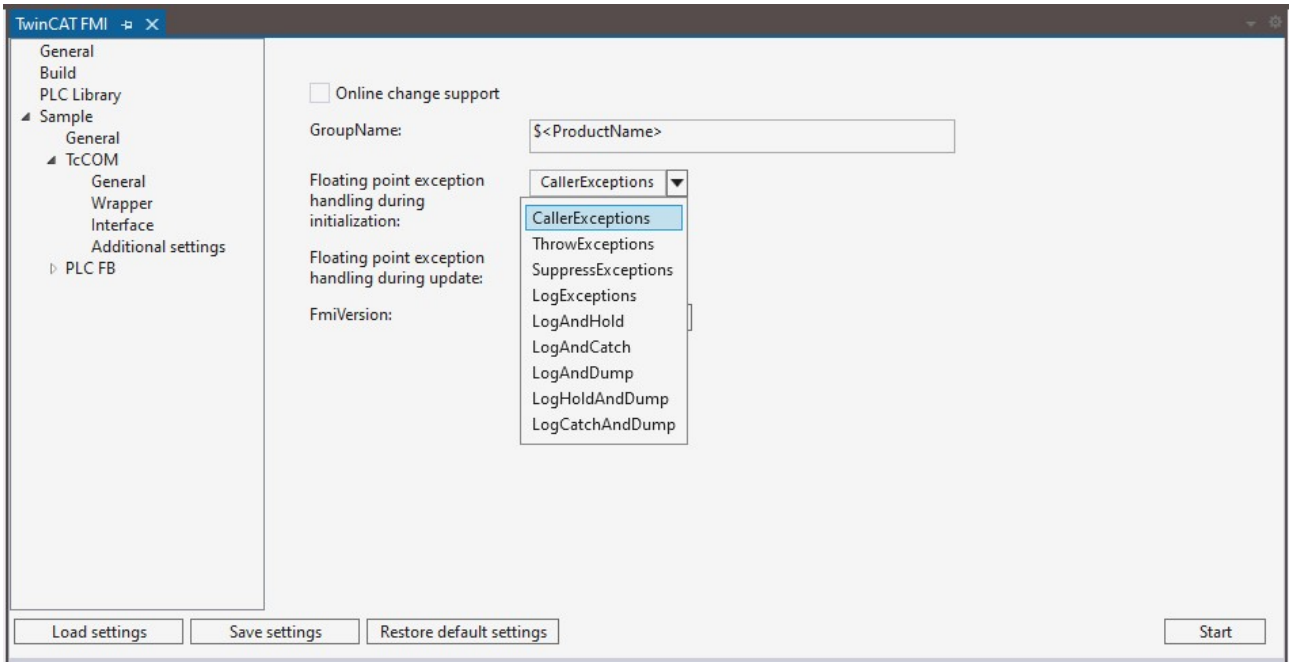
Another way to handle exceptions is to embed them in a try-catch statement. In the PLC the instructions __TRY, __CATCH, __FINALLY, __ENDTRY are available for this purpose. If floating point exceptions are enabled on the calling task and an exception occurs within the Try-Catch, it is caught in the Catch branch and can be handled. Accordingly, no variables are set to inf or NaN in this approach. However, it is also important to note that the code in the Try branch is run through only up to the point of the exception and then a jump is made to the Catch branch. In the application code, it should be noted that internal states in the Try branch may not be consistent.

Dump Files

From TwinCAT 3.1.4024.22 (XAR), dump files can be created at runtime in case of exceptions in the TcCOM object.

Definition of the object behavior in case of occurring exceptions

You can define the behavior of a TcCOM object when exceptions occur under the **TcCOM>General** tab. You must define the behavior separately for the initialization phase of the TcCOM and for the runtime phase (update phase).



A total of 9 different settings are available:

- **CallerExceptions** (default): Exceptions are triggered as configured in the calling task.
- **ThrowExceptions**: Exceptions in the TwinCAT object are triggered in any case, regardless of how the task is configured.
  - An exception causes a TwinCAT error message and a TwinCAT stop.
- **SuppressExceptions**: Exceptions are not triggered, regardless of how the task is configured.
  - An exception does not cause a TwinCAT stop.
  - Outputs or internal states can be NaN or inf.
- **LogExceptions**: Exceptions are triggered, but do not lead to a TwinCAT stop.
  - An exception does not cause a TwinCAT stop.
  - Outputs or internal states can be NaN or inf.

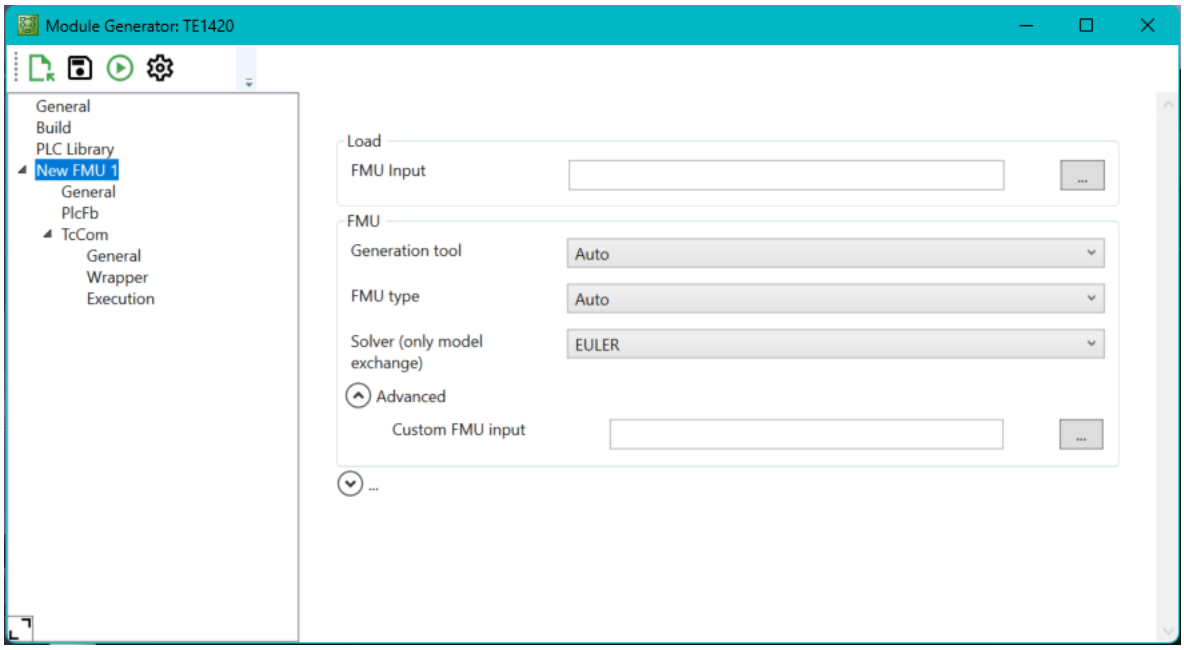
  - The `ExecutionInfo` output is filled with information about an exception in the current cycle. If several exceptions occur in one cycle, only the first exception is displayed at the output. When the TwinCAT object is called again, the information is reset.
- **LogAndHold**: Exceptions are triggered. The execution of the TwinCAT object is stopped.
  - An exception does not cause a TwinCAT stop.
  - Outputs or internal states can be NaN or inf.
  - The `ExecutionInfo` output is filled with information about an exception in the current cycle. If several exceptions occur in one cycle, only the first exception is displayed at the output. When the TwinCAT object is called again, the information is reset.
  - The execution of the TwinCAT object is stopped after an exception occurs. TwinCAT itself remains in run mode.
    Restart execution: ReleaseObjectStop.
- **LogAndCatch**: Exceptions are caught with try-catch in the TwinCAT object. The execution of the TwinCAT object is stopped.
  - An exception does not cause a TwinCAT stop.
  - Outputs or internal states **cannot** contain NaN or inf.
  - The `ExecutionInfo` output is filled with information about an exception in the current cycle.
  - The execution of the code ends at the point of the exception. From there, the program jumps to the catch branch, i.e. internal states can be inconsistent.
  - The execution of the TwinCAT object is stopped after an exception occurs. TwinCAT itself remains in run mode. Restart execution: ReleaseObjectStop.
- **LogAndDump, LogHoldAndDump and LogCatchAndDump**
  - Behavior in case of LogExceptions
  - Additionally a dump file is stored on the runtime system in the TwinCAT folder *Boot*. For more information on dump files, see the chapter Exception Handling in the TE1400 Target for Simulink documentation.

9 Reference user interface

With the aid of the TE1420 Target for FMI, TwinCAT runtime objects (TcCOM modules or PLC function blocks) can be generated from FMUs. These are packaged in a TwinCAT driver (*.tmx) or in a PLC library. From a TwinCAT FMI project always exactly one TMX driver and/or PLC library is generated.

The user interface is therefore divided into a general area, in which settings can be set that apply to all runtime objects in this TwinCAT driver/PLC library, and an object-specific part, which contains the settings for the respective runtime object.

The TE1420 can also be operated completely via the command line. (See chapter Command line interface [▶ 37].) For this reason, the respective command line parameter corresponding to a setting is listed in the last column of the tables for clarity.



Menu bar

Icon	Description
	Loads an FMU or configuration.
	Saves the current configuration.
	Generates the current configuration.
	Sets the configuration level.
	Shows the output window.
	Hides the output window.

Context menu in the project tree

Right-click on an FMU:

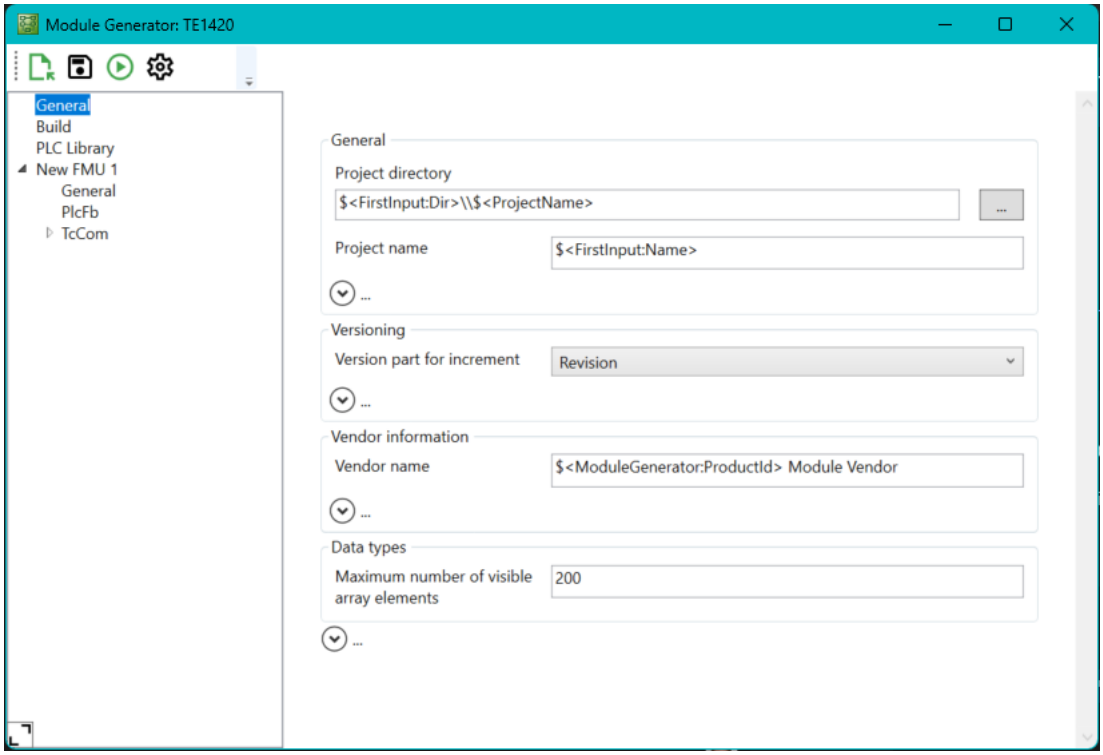
Setting	Meaning
Remove	Removes the FMU from the TwinCAT FMI project.
Restore defaults	Sets the settings of the FMU node to the default values. After that, the FMU can be selected again.

Right-click on a free area in the project tree:

Setting	Meaning
Restore defaults	Restores the default values for all FMUs in the TwinCAT FMI project.
Add new FMU	Adds another FMU to the TwinCAT FMI project.

9.1 'General' tab

In the **General** tab, you can adjust general settings. By default, these are provided with placeholders and are assigned the default values. Changed settings are persisted in an xml file and are retained for new projects. These can be reset using the **Restore defaults** context menu option.



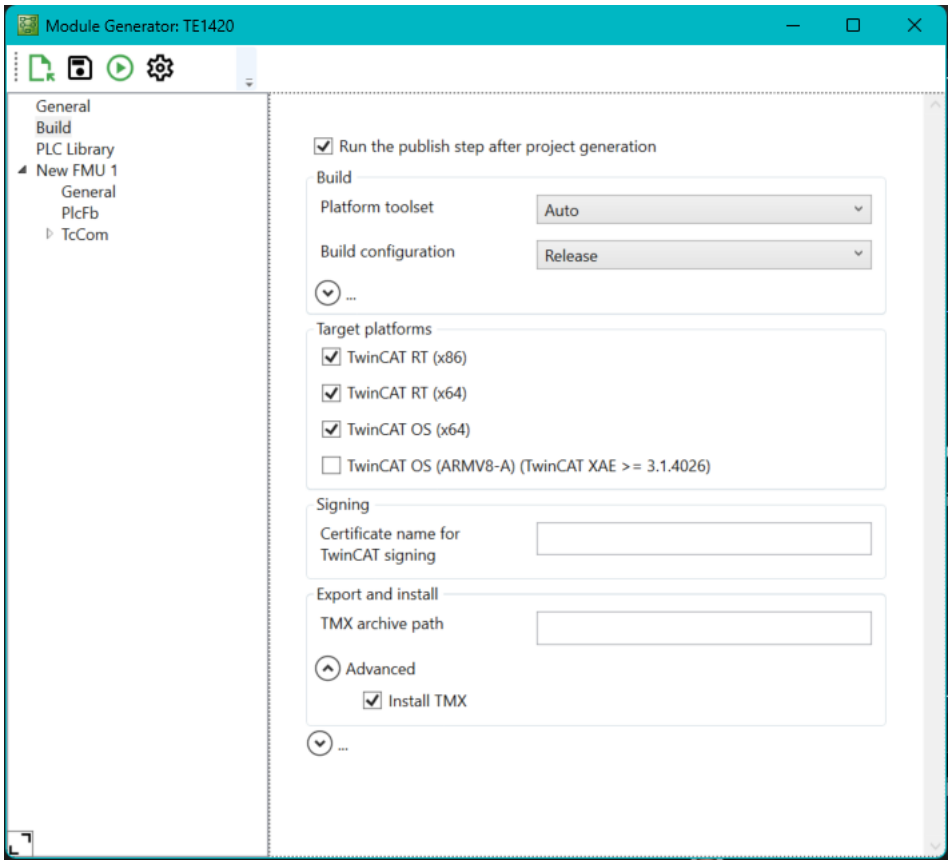
Setting	Meaning	Command line parameters
Project directory	Project directory under which the TcCOM module is to be generated.	--Project.ProjectDir
Project Name	Name of the project	--Project.ProjectName
Version part for increment	Part of the version number to be incremented each time a new one is created.	--Project.IncrementVersion Default: Revision
Vendor Name	Vendor name, the name under which the generated TcCOM module is sorted in the module selection dialog.	--Project.VendorName
Maximum number of visible array elements	Maximum number of visible array elements	--Project.MaxVisibleArrayElements Default: 200

There are also the following extended parameters, which are collapsed by default and do not normally need to be adjusted. In most cases, templates are already stored here, which automatically adjust the parameter values.

Setting	Meaning
Pack output path	Path where the generated C++ project is stored.
Version source file	Version number of the source code file
Executable file version	Version number of the generated module or library
Executable product version	Product version of the generated module or library
Class factory name	Name of the TcCOM Classfactory or driver
Product name	Product name
Copyright notice	Copyright notice
Executable description	Description text of the module
Additional options	Options to overwrite internal code generator parameters
ID of OEM	ID of OEM, is required when using OEM licenses.
IDs of OEM licenses	List of OEM licenses that are queried when the module is started.

9.2 'Build' tab

You can configure the build settings under the **Build** tab.



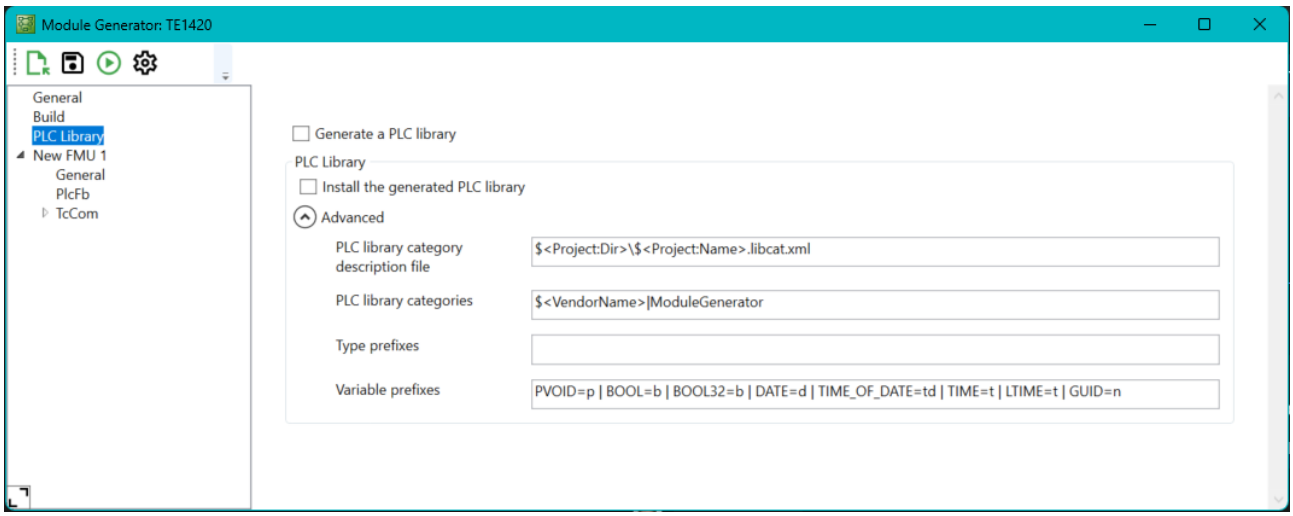
Setting	Meaning	Command line parameters
Run the publish step after project generation	Selection whether only the code should be generated or the tmx driver should also be built for the selected platforms.	--Project.Publish Default: „true“
Platform Toolset	Selection of the Visual Studio toolset used for building.	--Project.PublishPlatformtoolset Default: Auto
Build configuration	Selection of the build configuration (debug/release).	--Project.PublishConfiguration Default: "Release"
TwinCAT RT(x86)	Creating a tmx driver for the TwinCAT RT(x86) platform.	--Project.PublishTcRTx86 Default: "true"
TwinCAT RT(x64)	Creating a tmx driver for the TwinCAT RT(x64) platform.	--Project.PublishTcRTx64 Default: "true"
TwinCAT OS(x64)	Creating a tmx driver for the TwinCAT OS(x64) platform.	--Project.PublishTcOSx64 Default: "true"
TwinCAT OS(ARMV8-A)	Creating a tmx driver for the TwinCAT OS(ARMV8-A) platform.	--Project.PublishTcOSARMv8A Default: "false"
Certificate name for TwinCAT signing	TwinCAT certificate name to be used for signing the generated tmx driver.	--Project.SignTwinCatCertName Default: ""
tmx Archive	Name of a tmx archive that can be used for sharing the tmx files.	--Project.TmxArchive Default: ""

There are also the following extended parameters, which are collapsed by default and do not normally need to be adjusted.

Setting	Meaning
Specify C++ language standard version	Selection of the C++ standard used
Code generation and build verbosity	The level of detail with which code generation and build messages are logged in the output window.
Always rebuild all source files on publish	Select this option to ensure that all source code files are always rebuilt.
Build parallel to publish	When multiple build platforms are selected, the compilation processes are independent of each other and executed sequentially. Activating this option causes all build processes to be executed in parallel on your engineering system. Parallelization is performed by the operating system. To avoid unnecessary waiting times, only create binaries for the platforms you use.
Install TMX	Installs the generated TMX into the repository.
Deployment project	Absolute path to the TwinCAT 3 project (*.tsproj) in which the module is to be automatically updated.
Activate and restart deployment project	Activate and restart the project on the target system stored in the project.

9.3 'PLC Library' tab

You set whether the FMI project should also be created or installed as PLC library under the **PLC Library** tab.



Setting	Meaning	Command line parameters
Generate a PLC library	Setting whether a PLC library should be created.	--Project.GeneratePlcLibrary Default: "false"
Install the generated PLC library	Setting whether the generated PLC library should also be installed.	--Project.InstallPlcLibrary Default: "false"

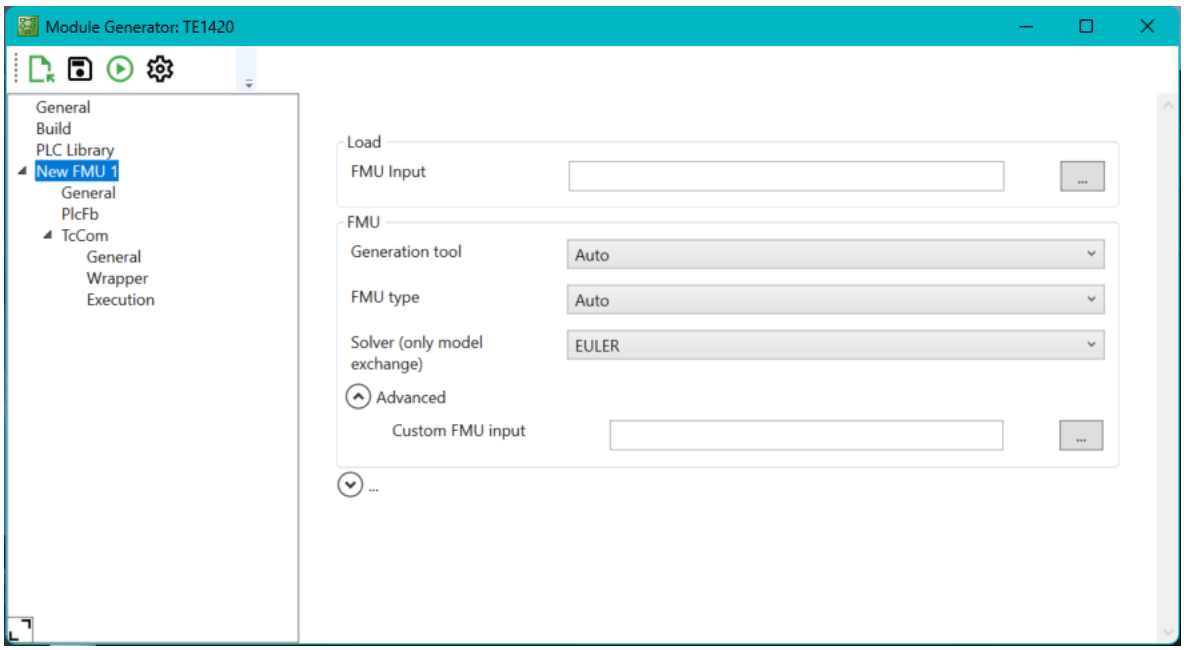
There are also the following extended parameters, which are collapsed by default and do not normally need to be adjusted.

Setting	Meaning
PLC library category description file	Path to the description file of the PLC library categories
PLC library categories	PLC library categories of the library to be generated
Type prefixes	Definition of type prefixes
Variable prefixes	Definition of variable prefixes

9.4 'NewFmuModule1' tab

In an FMI project, one or more FMUs can be combined in a tmx driver or in a PLC library.

To add a new FMU to a project, right-click on a free area in the project tree. In the context menu that opens, you have the option to remove an FMU from the project, restore the default state or add a new FMU. In the default state or when an FMU has been added, the interface looks like the following figure:

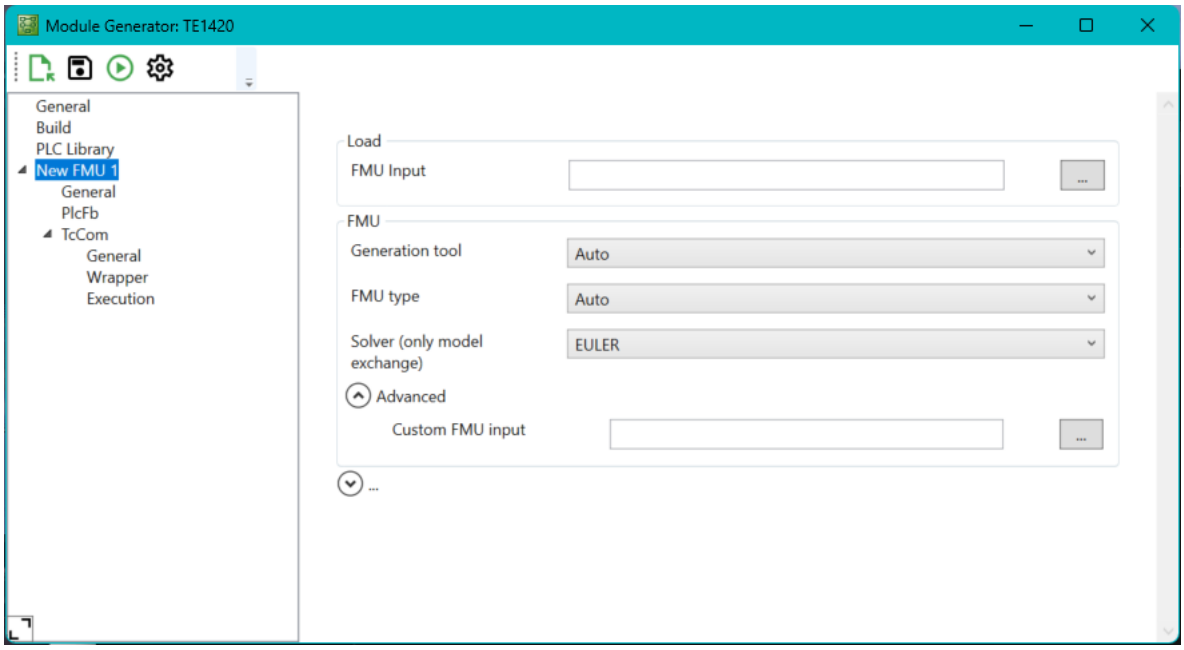


The settings made or the last exported FMI project is persisted, so that when exporting again, e.g. after changes in the model, these do not have to be made again.

9.4.1 'General' tab

You make general settings for the respective FMU under the **General** tab. This primarily involves selecting the FMUs to be imported. The other parameters, such as the Generation tool or the FMU type, are automatically recognized in the FMU using the Modeldescription.xml file. These settings are necessary because the exporting tools do not always export all the required information to an FMU or because of tool-specific peculiarities. The exporting tools supported by the respective version can be recognized by the *Generation tool* drop-down field. If the tool you are using (which generates pure source code FMUs) is not listed here, you can store the information on the build process in a description file as described at Integration of additional tools [▶ 38] and thus integrate the tool into the TE1420.

For Model Exchange FMUs, another necessary setting is the solver to be used for solving the differential equations.



Setting	Meaning	Command line parameters
FMU Input	Path to the FMU to be imported.	InputFile
Generation tool	Selection of the tool that generated the FMU. When selecting **Auto**, the selection is made automatically based on the model description.	--Class.GenerationTool Default: "Auto"
FMU Type:	Selection of FMU type (model exchange or co-simulation). When selecting **Auto**, the selection is made automatically based on the model description.	--Class.FmuVersionType Default: "Auto"
Solver:	Selection of the solver to be used.	--Class.Solver Default: "Euler"

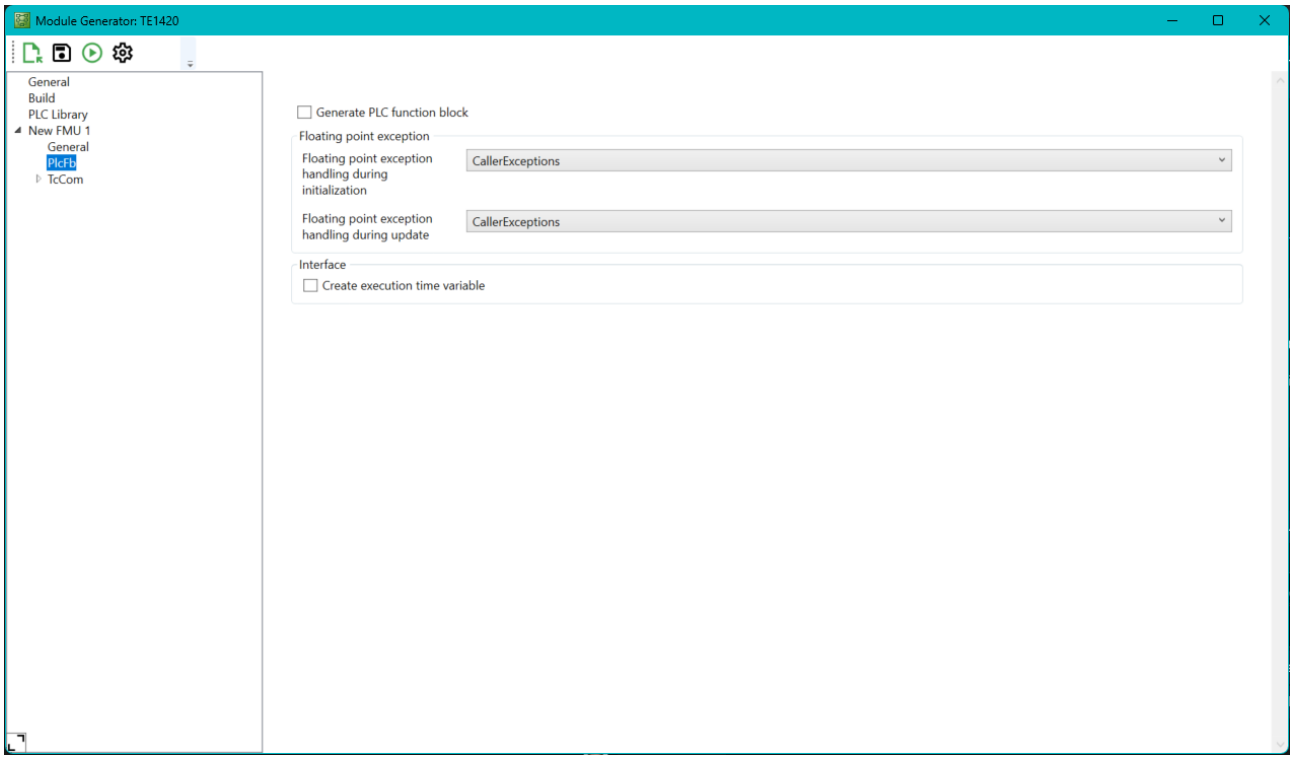
If an FMU is exported via Model Exchange, it contains only the corresponding differential equations describing the model. Not included in the FMU are then the solvers to solve these equations. Solvers from the internal TwinCAT solver library are available in the **Solver** selection box. The solvers can be selected for solving the equations. If an FMU was exported via co-simulation, it already contains the solver. **Co-Simulation** must then be selected in the **Solver** selection box.

There are also the following extended parameters, which are collapsed by default and do not normally need to be adjusted.

Setting	Meaning
Custom FMU Import	Path to the description file for the integration of additional tools (see chapter Integration of additional tools [▶ 38]).
Binary type size for input and output (only FMI 3)	Size of the binary data type
String type size	Size of the string data type

9.4.2 'PLC FB' tab

Under the **PLC FB** tab further settings can be made, which affect the function blocks in the generated PLC library. See 'PLC Library' tab [▶ 29].



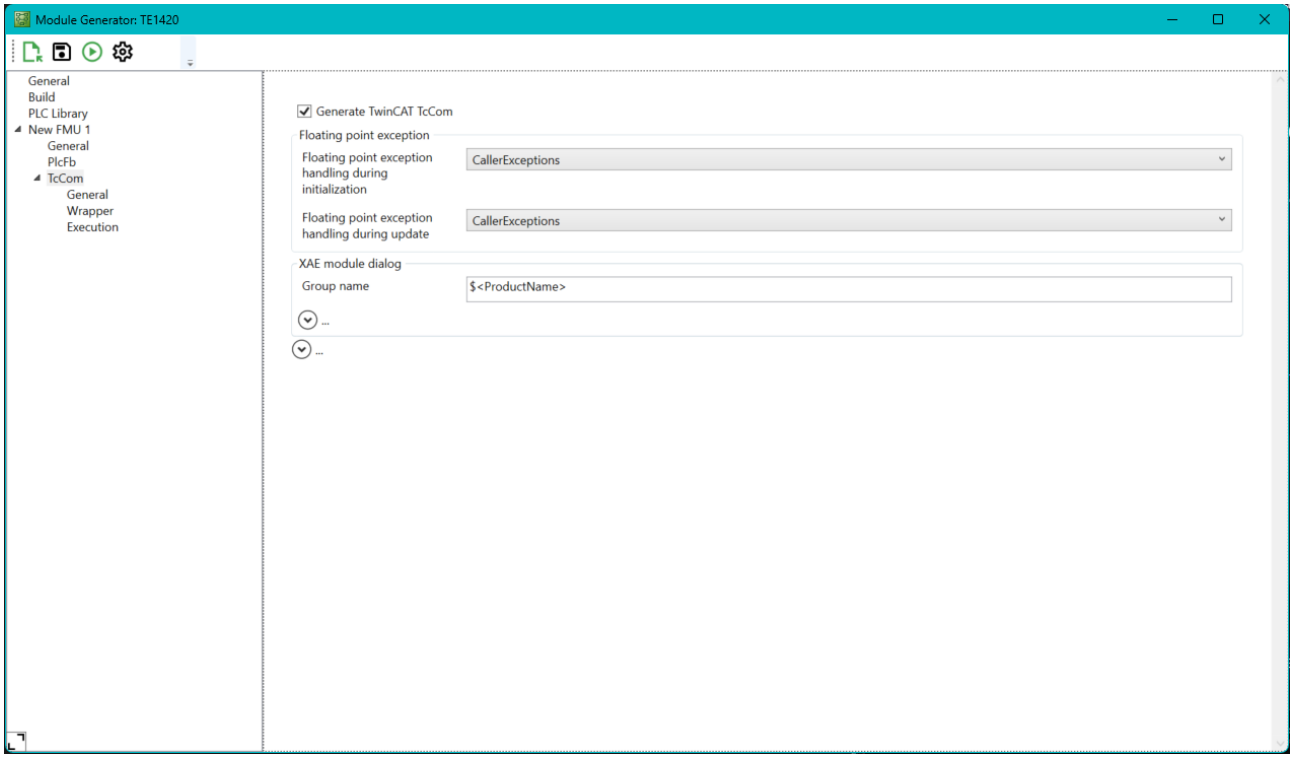
Setting	Meaning	Command line parameters
Generate TwinCAT PLC Function Block	Activates that this FMU (as a function block) becomes part of the PLC library to be generated.	--Class.PlcFb.Generate
Floating point exception handling during initialization	Handling of floating point exceptions during initialization of the TcCOM module.	--Class.PlcFb.InitExceptionHandling
Floating point exception handling during update	Handling of floating point exceptions during the call of the TcCOM module.	--Class.PlcFb.UpdateExceptionHandling
Create execution time variable	Creates a variable that displays the measured execution time.	--Class.PlcFb.MonitorExecutionTime

9.4.3 'TcCOM' tab

Under the **TcCOM** tab you can make further settings that affect the TcCOM module to be generated. The basic settings are selected in such a way that in most cases an executable TcCOM module is generated even without further settings in this area.

9.4.3.1 'General' tab

Under the tab you can change the general settings of the TcCOM module to be generated. Among other things, this affects the group name of the module and the handling of floating point exceptions.



Setting	Meaning	Command line parameter
Generate TwinCAT TcCOM	Activates that a TcCOM module is generated for this FMU.	--Class.TcCom.GenerateTcCom Default: "true"
Floating point exception handling during initialization	Handling of floating point exceptions during initialization of the TcCOM module.	--Class.TcCom.InitExceptionHandling
Floating point exception handling during update	Handling of floating point exceptions during the call of the TcCOM module.	--Class.TcCom.UpdateExceptionHandling
GroupName	Group name under which the generated TcCOM module is sorted in the selection dialog **Insert TcCom Object** in the TwinCAT 3 development environment.	--Class.TcCom.GroupName

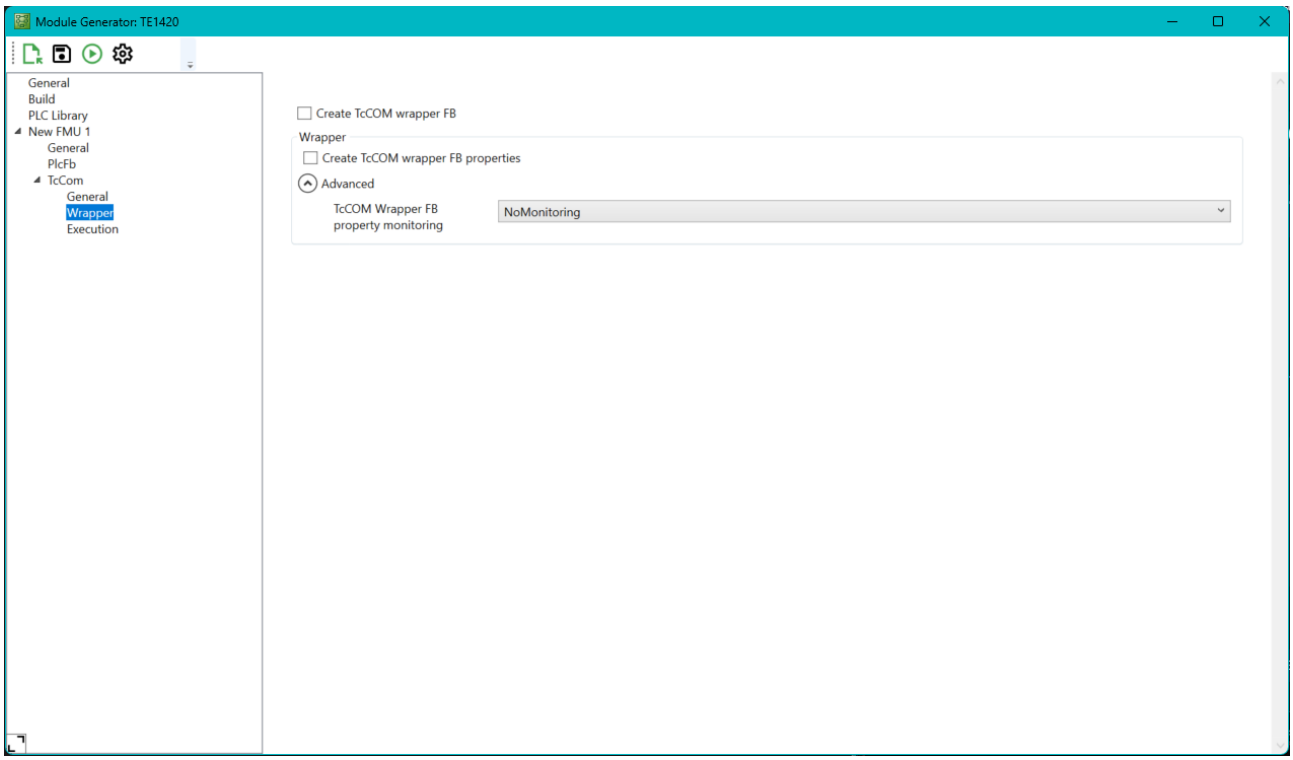
There are also the following extended parameters, which are collapsed by default and do not normally need to be adjusted. In most cases, templates are already stored here, which automatically adjust the parameter values.

Setting	Meaning
Group Display Name	Group name under which the TcCOM module is sorted in the Insert TcCom-Object dialog.
Group icon	Icon of the group under which the TcCOM module is sorted.
Module icon	Icon which is stored for the TcCOM module. Visible in the Insert TcCOM Object dialog
Online Change Support	Switches on the online change support for the generated TcCOM module.
TMC properties	Allows you to store additional properties in the TMC file. Name1=Value1| Name2=Value2|...
Use stop time	Defines whether the internal execution of the code is stopped after the stop time defined in the simulation tool and stored in the model description. If this field is not active, the simulation continues until the TcCOM module or the task is stopped.
Init state transition	Selection of the state in which the initializations in the generated TcCOM module take place. This has an influence on the available stack size
Create online parameters for initial values	Creates online parameters for the initial values.
IDs of OEM licenses	The license IDs for OEM licenses that are to be queried by the generated module on startup can be stored here.

9.4.3.2 'Wrapper' tab

Under this tab you can set whether a wrapper function block should also be generated for a TcCOM module. This is generated in the PlcOpen XML format and can be imported manually into a PLC project.

Unlike the settings in 'PLC Library' tab [▶ 29], which create a PLC library in which the individual models are represented as function blocks, the settings in the tab **TcCOM->Wrapper** create a function block that "wraps" the call of the TcCOM module. In this case, the actual TcCOM module "lives" outside the PLC and is only called or influenced by it.



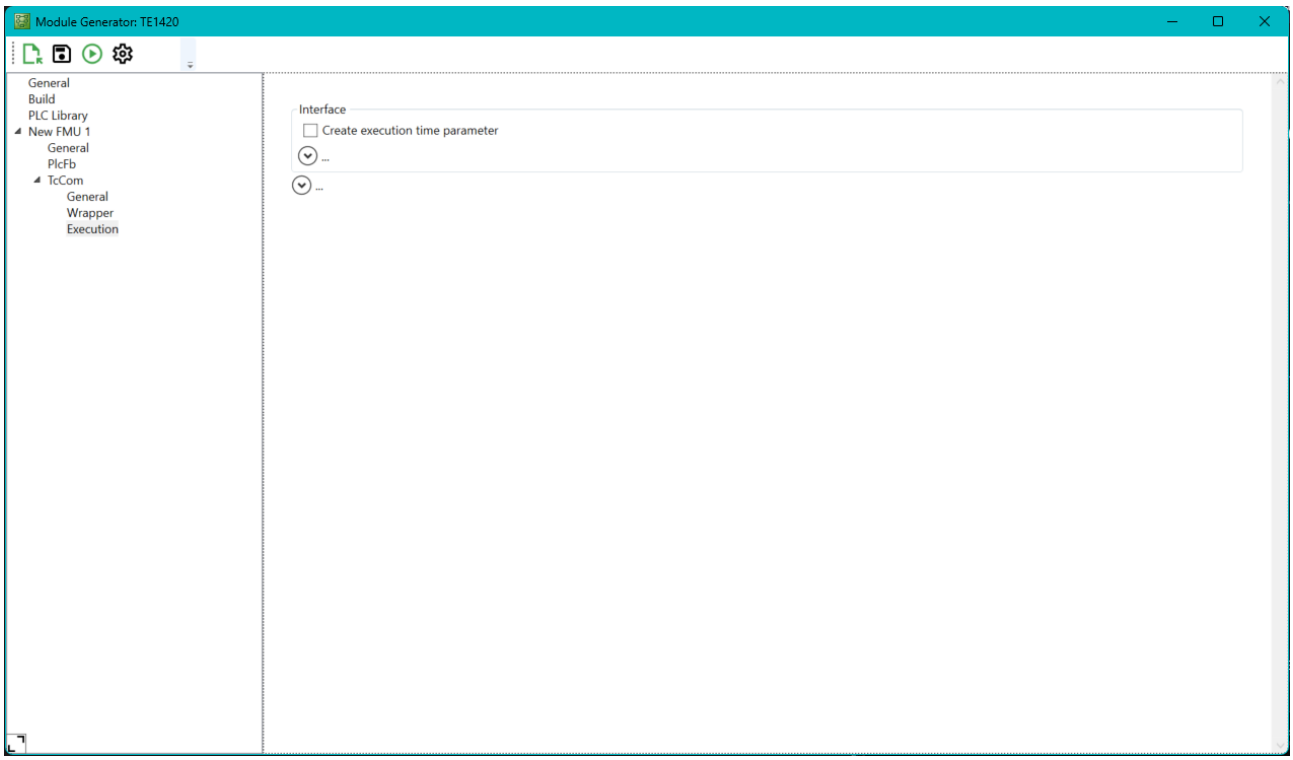
Setting	Meaning	Command line parameters
TcCOM Wrapper FB	Creates a Wrapper FB.	--Class.TcCom.TcComWrapperFb
TcCOM Wrapper FB properties	Creates a Wrapper FB where the variables are accessible as properties.	--Class.TcCom.TcComWrapperFbProperties

There are also the following extended parameters, which are collapsed by default and do not normally need to be adjusted.

Setting	Meaning	Command line parameters
TcCOM Wrapper FB porperty monitoring	Creates a variable to monitor the execution time.	--Class.TcCOM.MonitorExecutionTime

9.4.3.3 'Execution' tab

This tab allows you to set various parameters related to the execution of the generated TcCOM module.



Setting	Meaning
Create execution time parameter	Creates a module parameter in the Parameter (Online) tab, which can be used to read the execution time.

There are also the following extended parameters, which are collapsed by default and do not normally need to be adjusted.

Setting	Meaning
Create ‚ExcutionInfo‘ output	Creates a "ExcutionInfo" variable as output, which contains the execution time as part of the structure, among other things.
Default module caller	Defines how the module is called (from another module or a task).
Verify caller	Defines whether the caller is verified by the calling object.
Default step size adaptation mode	Defines which time is used for the step size of the simulation calculations
Default execution sequence	Defines the execution sequence of the processing (e.g. IO at the start of the task).
Execute model code after startup	Defines whether execution is started directly after startup (analogous to the autostart function of the PLC).

10 Command line interface

The command line interface is available since version 2.7.2.0 of the TE1420 | Target for FMI.

The TE1420 Target for FMI offers a command line interface that can be used to automatically convert FMUs into TcCOM modules. This interface is described below.

The command line interface is used via the TcModuleGenerator.exe application, which is installed by default in the "*C:\Program Files (x86)\Beckhoff\TwinCAT\Functions\TE1420-TargetForFmi\Win32*" folder.

The call via command line is made according to the following scheme:

```
TcModuleGenerator generate [FmuPath1 FmuPath2 …] [options]
```

In addition to a detailed call containing all parameters, it is also possible to pass a settings file in the call. The parameters are stored in this file as XML. The call is then made as follows:

```
TcModuleGenerator.exe generate C:\Users\<UserName>\AppData\Local\Temp\configTE14xx.xml
```

The settings file can be created directly via the UserInterface (see chapter Reference user interface [▶ 26]; Saving the configuration).

The call parameters of the command line interface can be requested with the interface itself using the command

```
TcModuleGenerator.exe --help
```

or

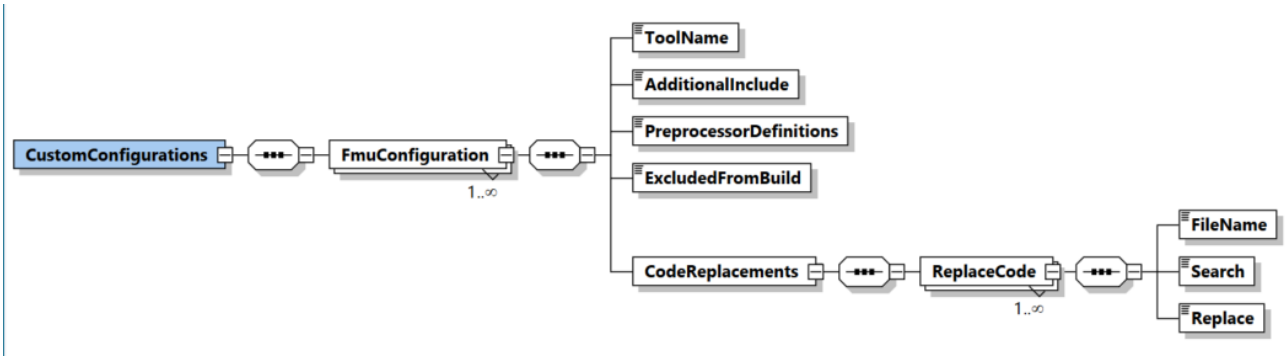
```
TcModuleGenerator.exe generate --help
```

The exact call parameters for the individual available options are specified in the "User interface" documentation (see chapter Reference user interface [▶ 26]).

11 Integration of additional tools

To convert FMUs containing source code into TcCOM modules that can be executed in a TwinCAT 3 runtime environment using TE1420, the FMUs must first be converted into a Visual Studio project and then compiled. Although the FMI standard prescribes the structure of the source code, different settings may be required for FMUs from different source systems in the Visual Studio project. These can be additional Include paths, preprocessor settings or even replacements in the source code. To enable connection to other FMU exporting tools from other tool manufacturers or customers without assistance from Beckhoff, the necessary settings must be stored in an XML file. The storage location of these files is *C:\ProgramData\Beckhoff\TE1420-TargetForFmi*. This folder also contains a template file (CustomFMU.xml). This file can be copied and renamed, and then filled in to connect additional tools. The corresponding .xsd file for validating the file can be found at *...\TE1420-TargetForFmi\SDK\XML\CustomFmu.xsd*".

Xml schema of the extension file:



CustomFMU.xml Template:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<CustomConfigurations
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="CustomFmu.xsd">
<FmuConfiguration>
<ToolName>CustomFmu</ToolName>
<AdditionalInclude>$(ProjectDir);$(ProjectDir)sources;</AdditionalInclude>
<PreprocessorDefinitions></PreprocessorDefinitions>
<ExcludedFromBuild></ExcludedFromBuild>
<CodeReplacements>
<ReplaceCode>
<FileName></FileName>
<Search></Search>
<Replace></Replace>
</ReplaceCode>
</CodeReplacements>
</FmuConfiguration>
</CustomConfigurations>
```

Name	Meaning
CustomConfiguration	Root node of the configuration file
FmuConfiguration	FmuConfiguration for each additional tool
ToolName	Name of the tool as it should be displayed under FMU type (see 'General' tab [▶ 33]).
AdditionalInclude	Additional Include paths separated by commas.
PreprocessorDefinitions	Preprocessor definitions that must be used when building the FMU in Visual Studio, separated by commas.
ExcludedFromBuild	Files that are to be excluded from a build.
CodeReplacements	Top node for all replacements in the source code, if any are required.
ReplaceCode	Entry for each replacement in the source code
FileName	Name of the file in which a replacement must be carried out.
Search	Code string to be searched for.
Replace	Code string to replace the searched string.

12 FAQ

Why can't my FMU be imported?

For the TE1420, the FMUs must be generated with source code. These are then compiled into a TwinCAT runtime module (TcCOM module) using the TE1420 Target for FMI.

Which FMI versions are supported?

Both FMI 2.0 and FMI 3.0 are supported

13 Samples

You can download sample FMUs and associated TwinCAT 3 projects from Beckhoff's GitHub page: TE142x - FMU samples. The samples and their function are also explained there.

14 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

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.

Third-party trademark statements

DSP System Toolbox, Embedded Coder, MATLAB, MATLAB Coder, MATLAB Compiler, MathWorks, Predictive Maintenance Toolbox, Simscape, Simscape™ Multibody™, Simulink, Simulink Coder, Stateflow and ThingSpeak are registered trademarks of The MathWorks, Inc.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

More Information:
www.beckhoff.com/te1420

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