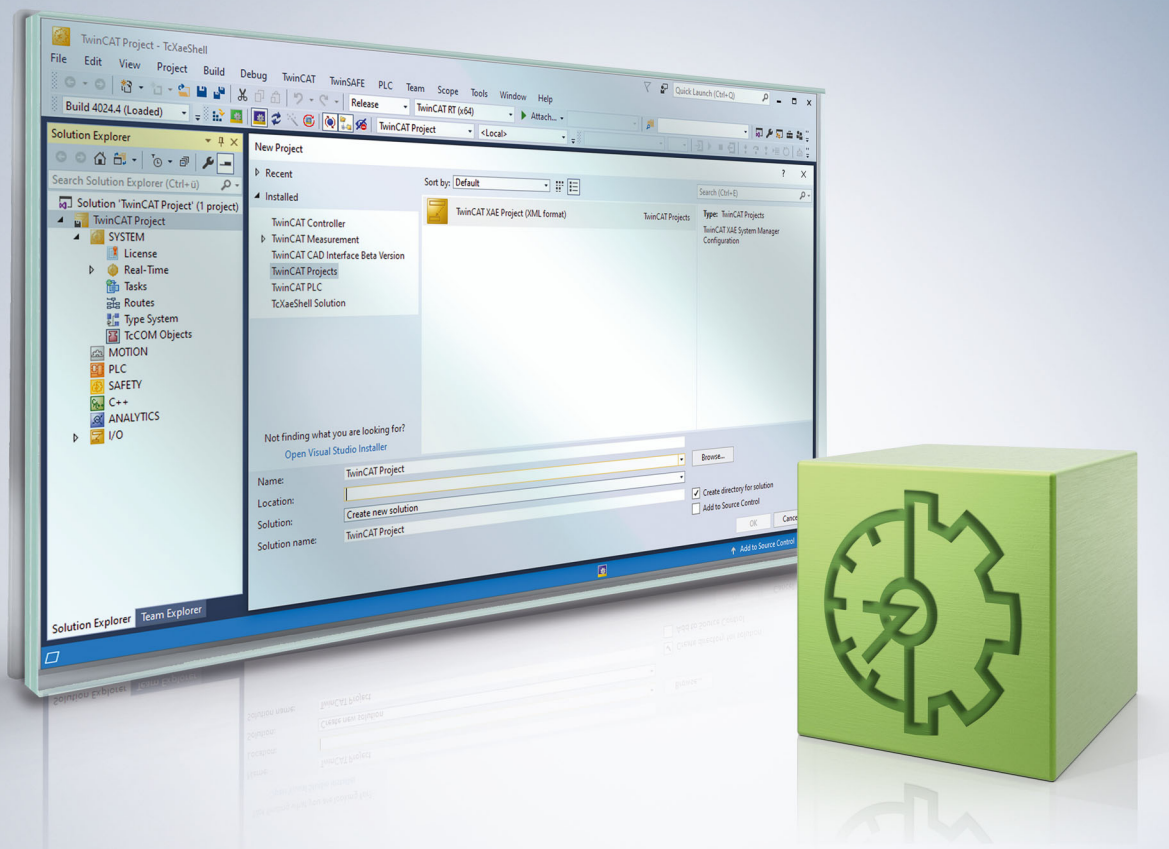
BECKHOFF New Automation Technology

Manual | EN

TE1420

TwinCAT 3 | Target for FMI



2022-07-27 | Version: 1

Table of contents

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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

Trademarks

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

Patent Pending

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

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

Copyright

1.2 Safety instructions

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Description of symbols

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

⚠ DANGER

Serious risk of injury!

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

⚠ WARNING

Risk of injury!

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

⚠ CAUTION

Personal injuries!

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

NOTE

Damage to the environment or devices

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

Tip or pointer

This symbol indicates information that contributes to better understanding.

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 the manual TwinCAT 3 C++ Chapter 4 Requirements.

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

On the engineering PC

- Microsoft Visual Studio 2017 or 2019 Professional, Premium, Ultimate or Community Edition
  - For Visual Studio 2017 or 2019 check the Visual C++ checkbox during installation.
- 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 "x64: driver signing" in the TC3 C++ manual

Setup instructions

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. Start TwinCAT 3 Setup.
3. Then start the TE1420-TargetForFMI setup to install the TE1420.
4. Follow the instructions in the setup wizard.

⇨ The setup is installed.

If a Visual Studio and a TwinCAT installation already exists but the Visual Studio version does not meet the requirements mentioned 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 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, 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 for the engineering system to create 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

The TF1420 license is required to start a TwinCAT configuration with a TwinCAT object generated from an FMU. Without activated license, the module and consequently the TwinCAT system cannot be started. 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 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.

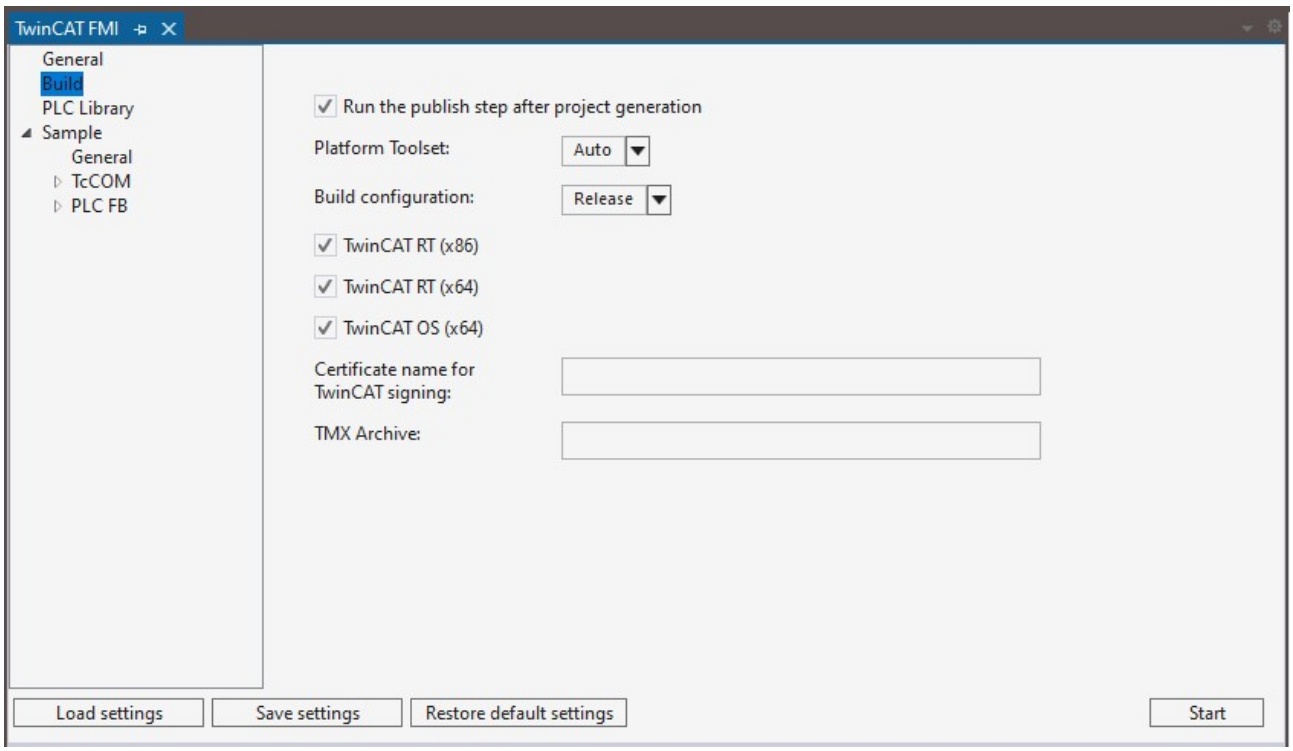
For **Variant 1** use a Windows environment variable. Create a new environment variable at **User > Variables** with:

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:

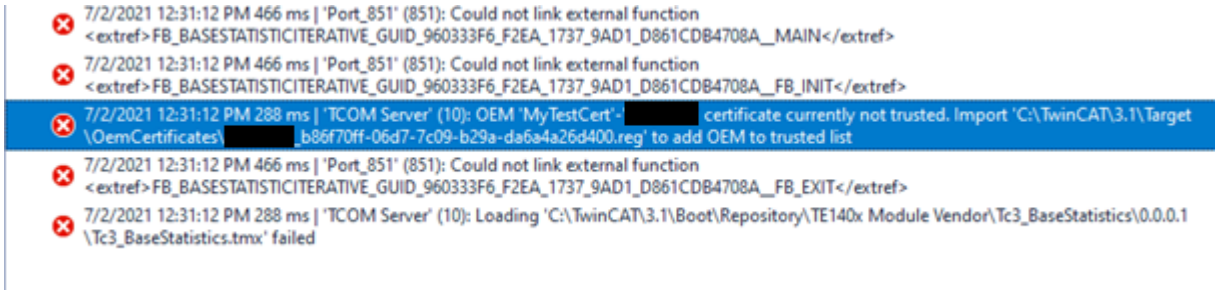
```
tcsigntool grant /f "C:\TwinCAT\3.1\CustomConfig\Certificates\MyCertificate.tccert" /r
```

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:

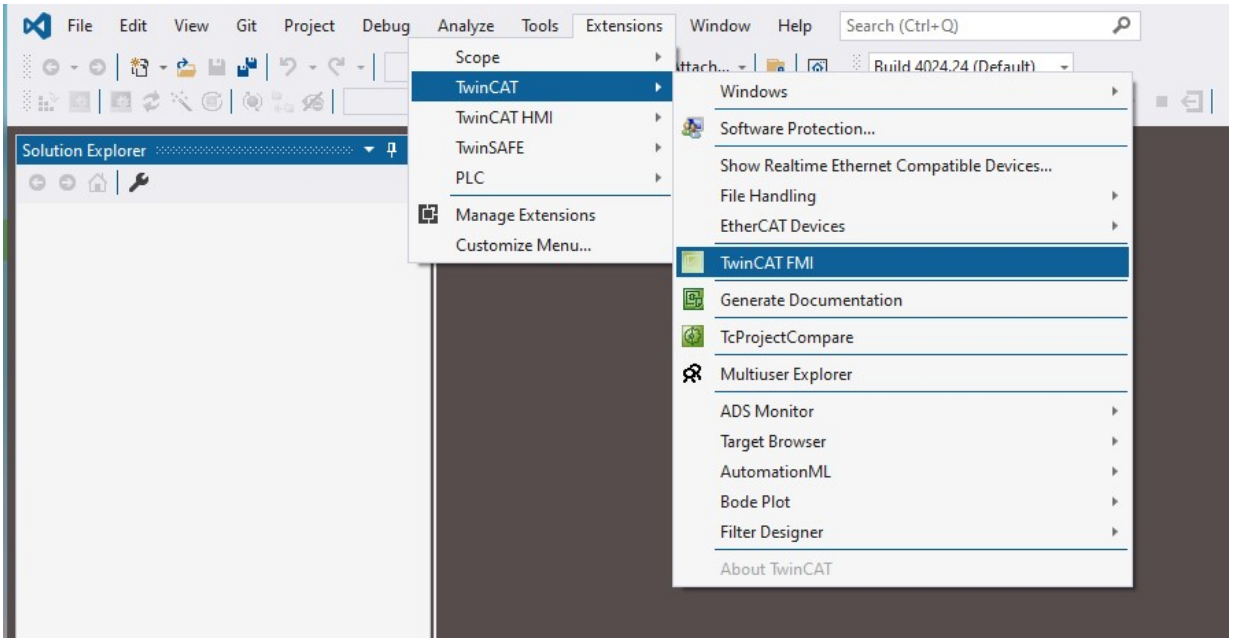
```
bcdedit /set testsigning yes
```

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

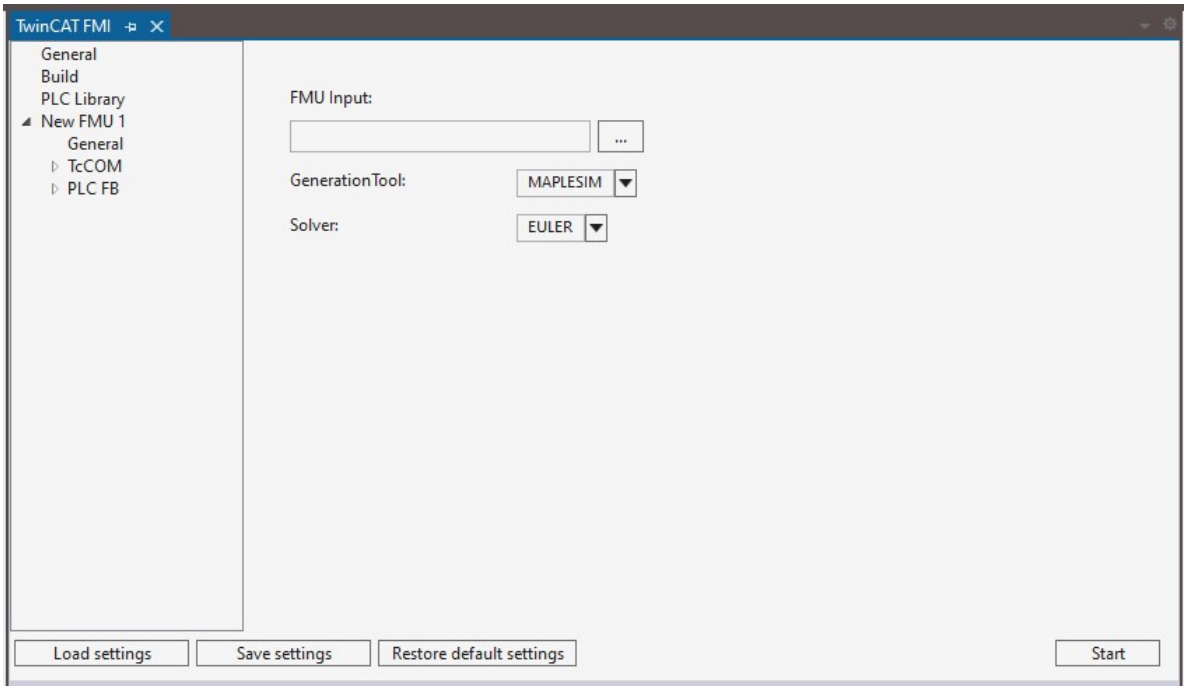
6 Quick start

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. If one or more FMUs are already included in the FMI project, restore the default state shown in the picture by pressing the button **Restore default settings**.
4. Use the button **...** to navigate to the FMU which should be converted to a TcCom module.
5. In the drop-down menu **GenerationTool** select the tool from which the FMU was exported.
6. 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 the selection **COSIMULATION** here.
7. Use the button **Start** to start the generation of the TcCom model.

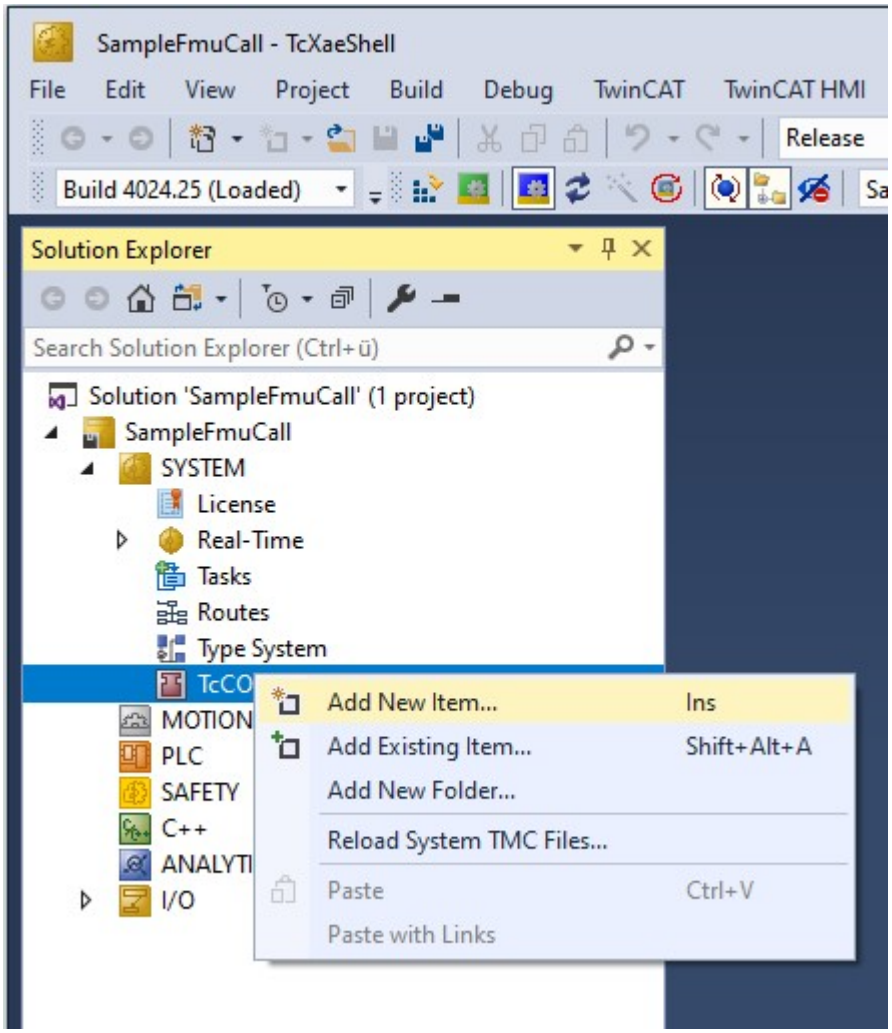
⇨ If the generation is successful, the output window of the TE1420 displays the message "Publish procedure completed successfully..."

Background to the selection of a Generation Tool

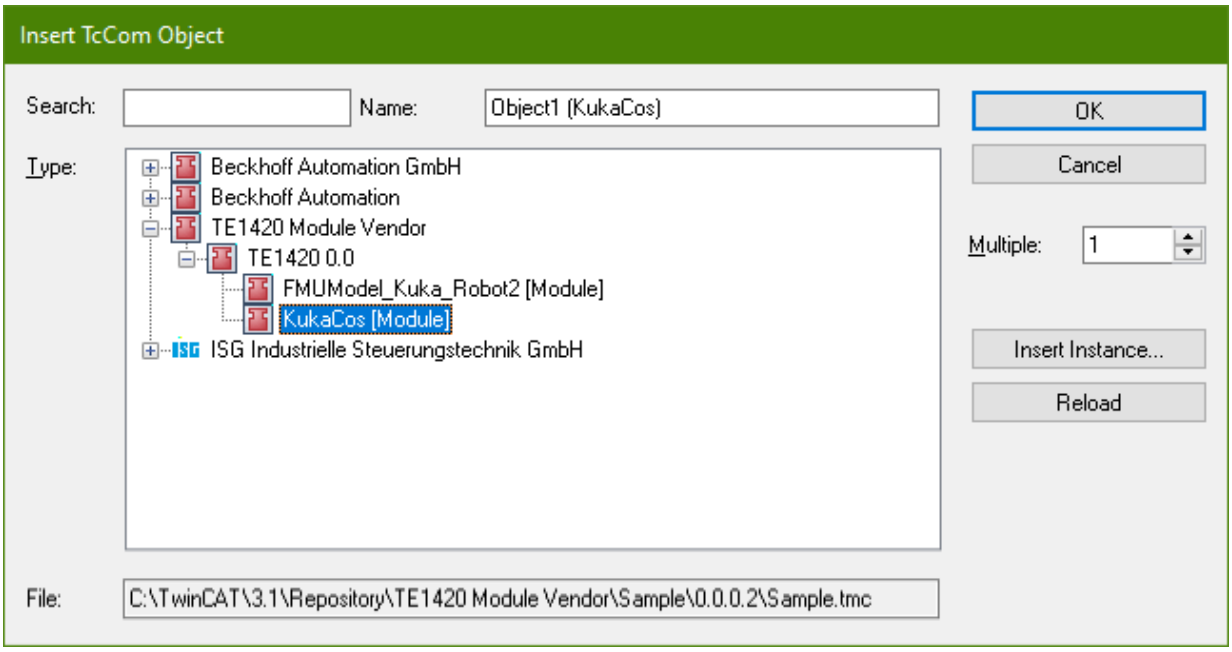
Since exporting the source code of a simulation model is done differently by the different tools, you also set some settings differently depending on the tool in the Visual Studio C++ project that is automatically created and used during the conversion. For the already supported tools, the configuration of these settings is done automatically with the selection of the Generation Tool. Further tools can also be connected by yourself. This is described in more detail in the chapter .

Integrating the TcCom module in a TwinCAT project

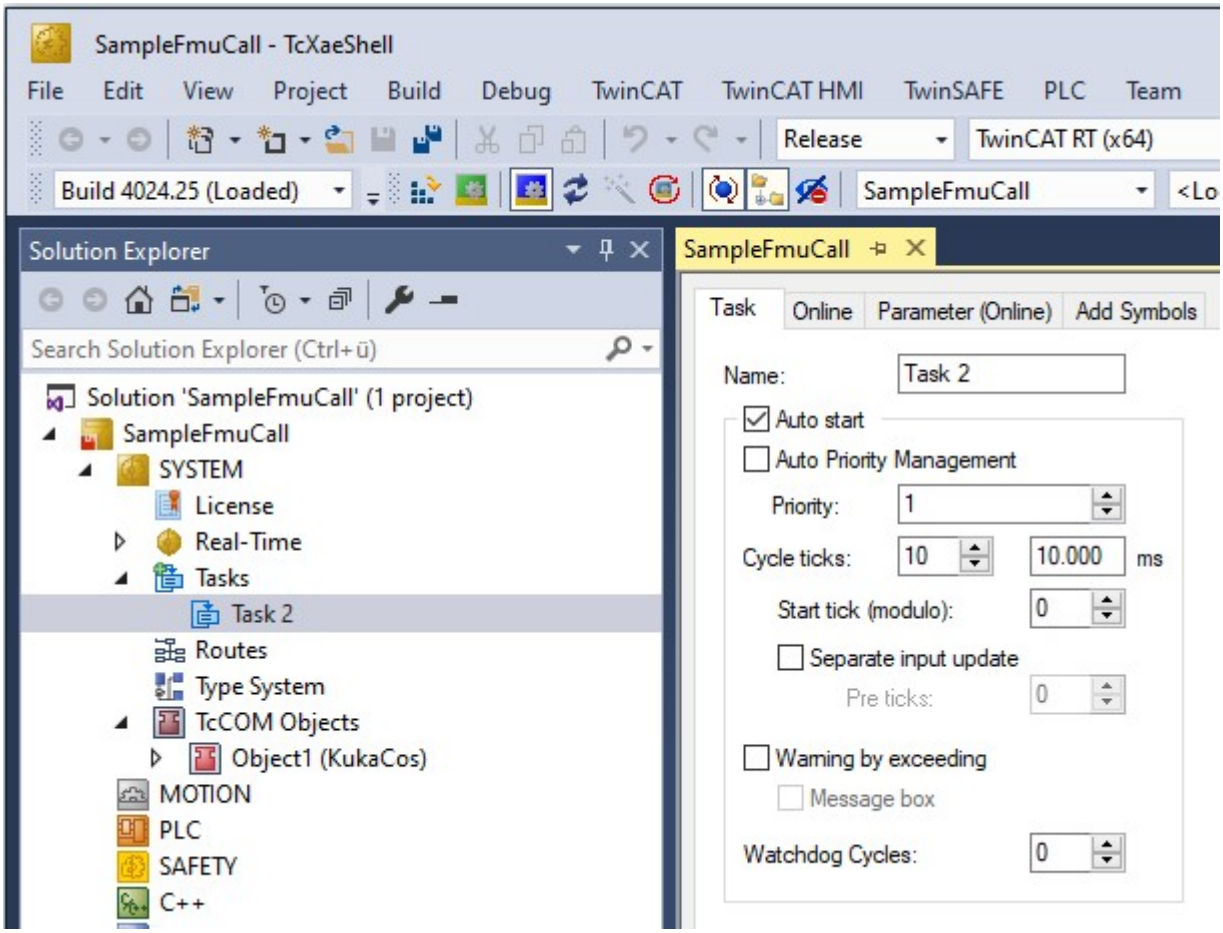
1. Open TwinCAT (TwinCAT XAE or TwinCAT in a Visual Studio environment).
2. Instantiate a new TcCOM object.



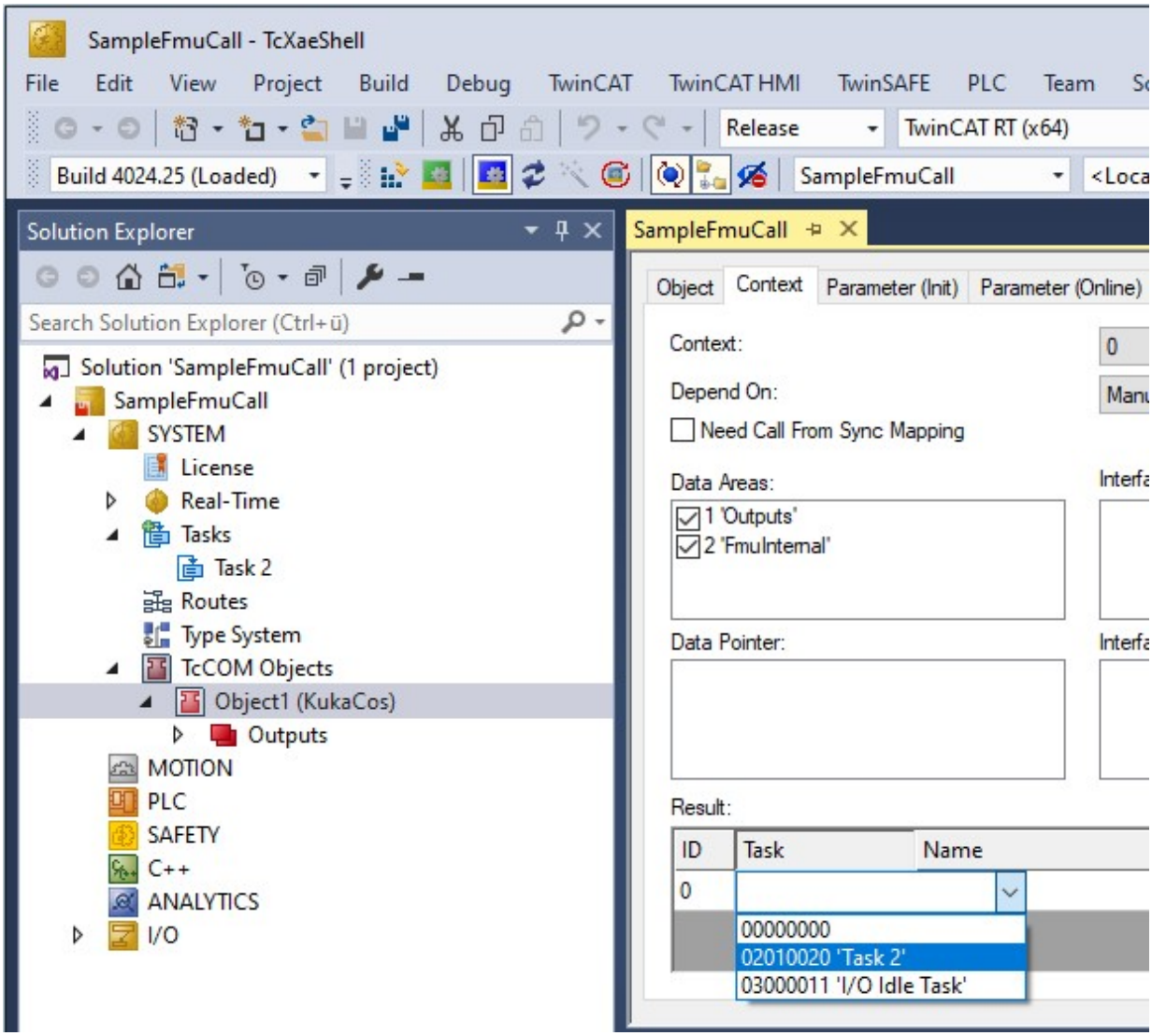
3. Select the desired object.



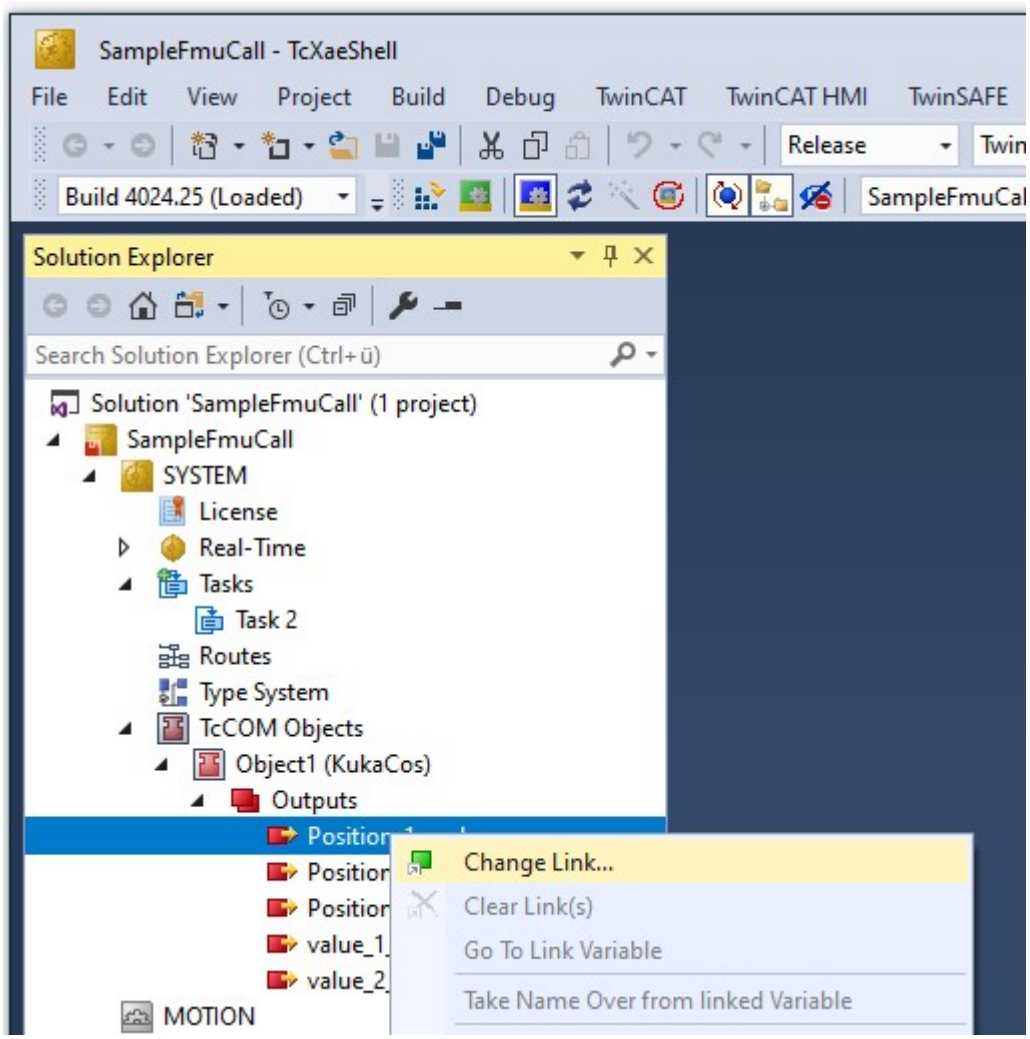
4. Create a cyclic task.



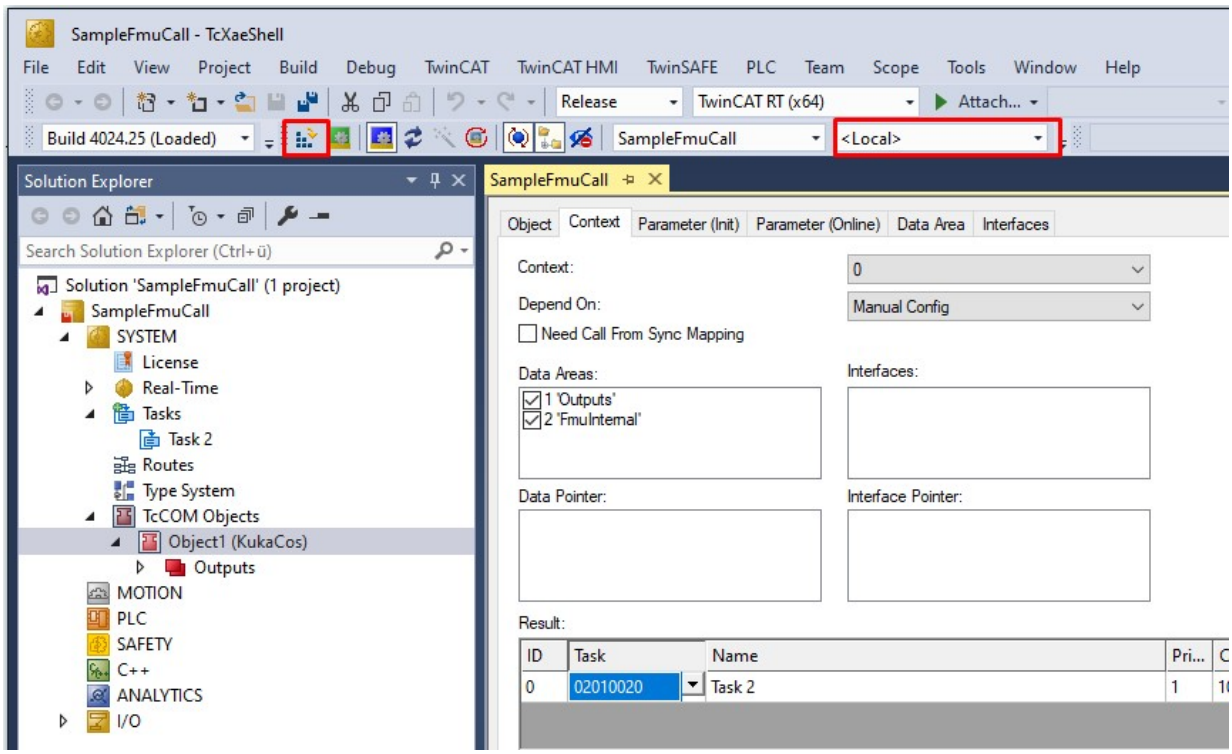
5. Assign the created task to your TcCOM instance.



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.

7 Parameterization of the module generation

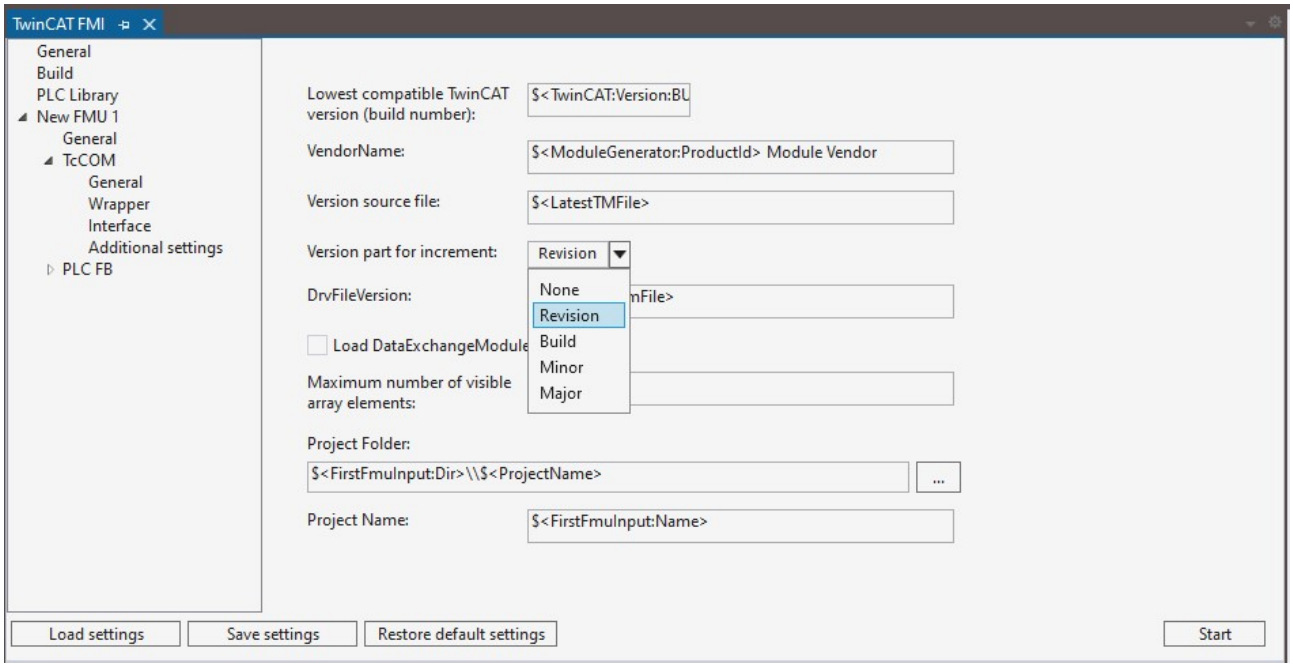
A number of settings are available for importing FMUs, which influence the generation of a TwinCAT runtime object (TcCom modules / PLC function block). These will be explained in this chapter.

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.



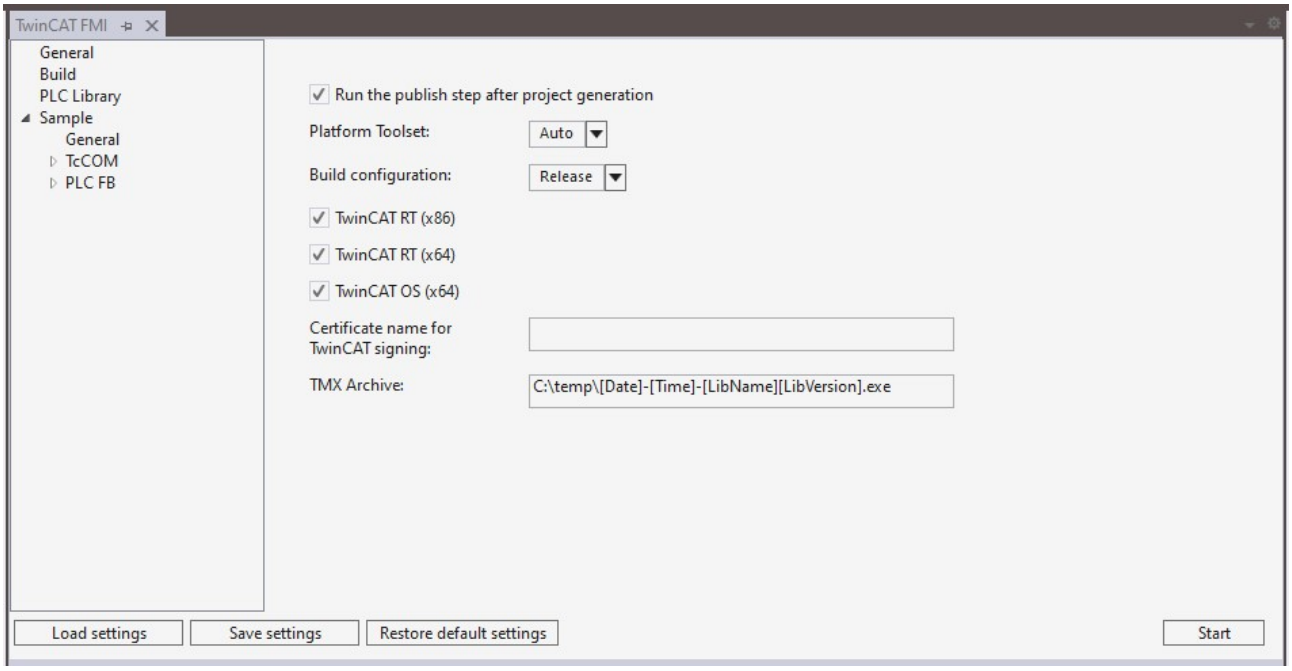
The basic version on which a version update is to be created is specified via **Version source file**. In the standard case $<LatestTMFile> is specified there. This searches for the last available version of the model on the local engineering PC and then uses this as the basis for the version increment. 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

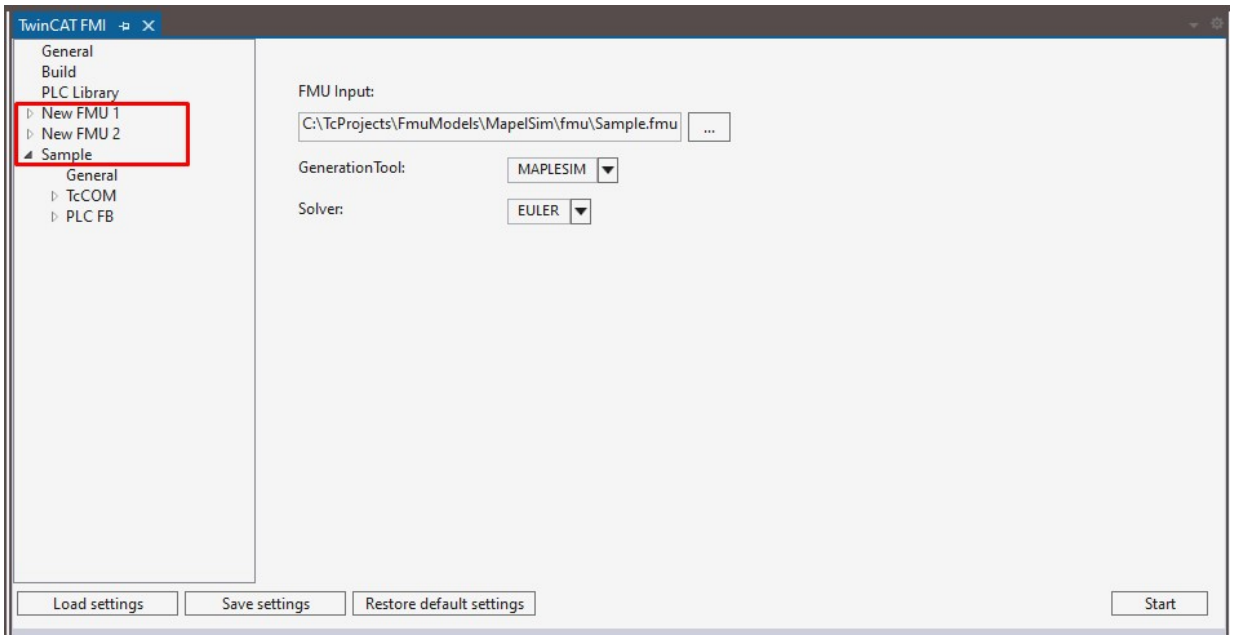
With the help of the TE1420 it is possible to bundle several models in one TMX driver respectively in one PLC library. This function can be used, for example, to make a collection of models (e.g. automation technology components) available in a library.

Proceed as follows:

✓ You have already created an FMI project with a configured FMU. See step 2 in the chapter Quick start [▶ 14].

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.

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 [▶ 14].

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. Furthermore, a PLC library can be created which allows to call the behavior of a model imported via the Target for FMI also 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.
   See also Creating and installing a PLC library.

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.

Standard 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.

Checking 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

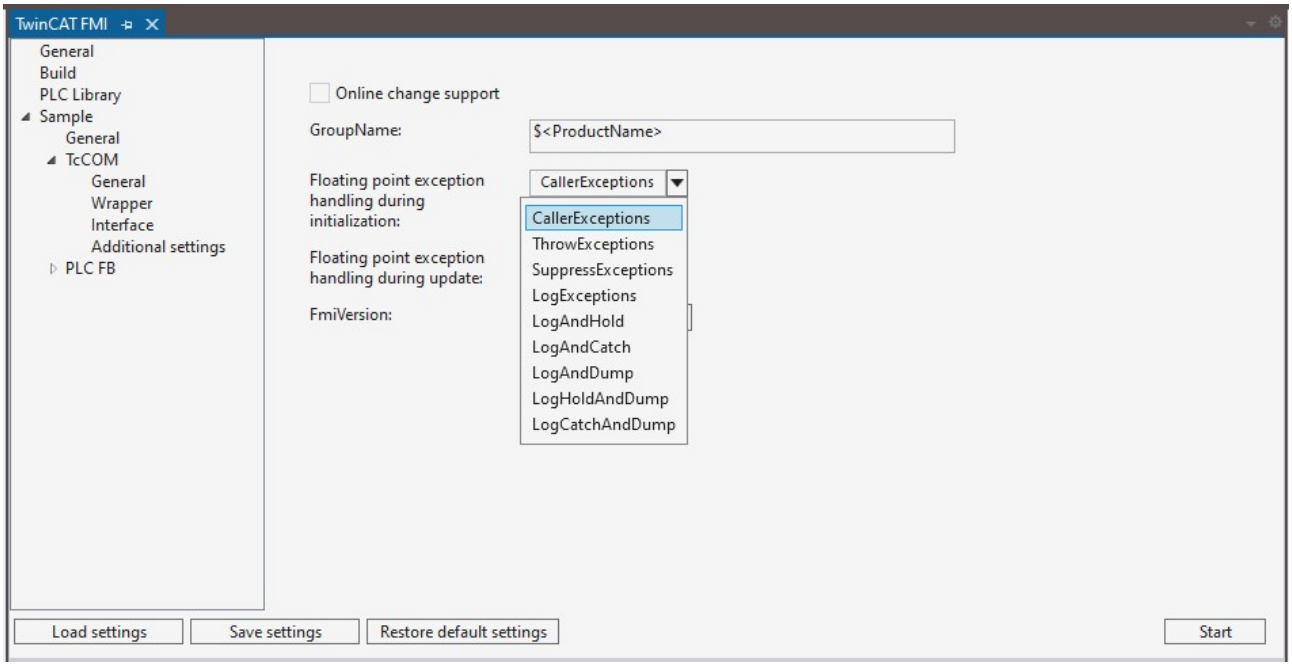
Another way to handle exceptions is to embed them in a try-catch statement. In the PLC the instructions __TRY, __CATCH, __FINALLY, __ENDTRY are available for this purpose. If floating point exceptions are enabled on the calling task and an exception occurs within the Try-Catch, it is caught in the Catch 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

As of 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 in the **Code Generation Settings** in Simulink®. 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 thrown as configured at the calling task.
- **ThrowExceptions**: Exceptions in the TwinCAT object are thrown 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 thrown, 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 thrown, 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 thrown. 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: .
- **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: .
- **LogAndDump, LogHoldAndDump and LogCatchAndDump**
    - Behavior in case of LogExceptions
    - Additionally a dump file is stored on the runtime system in the TwinCAT folder *Boot*. You can see more about dump files here.

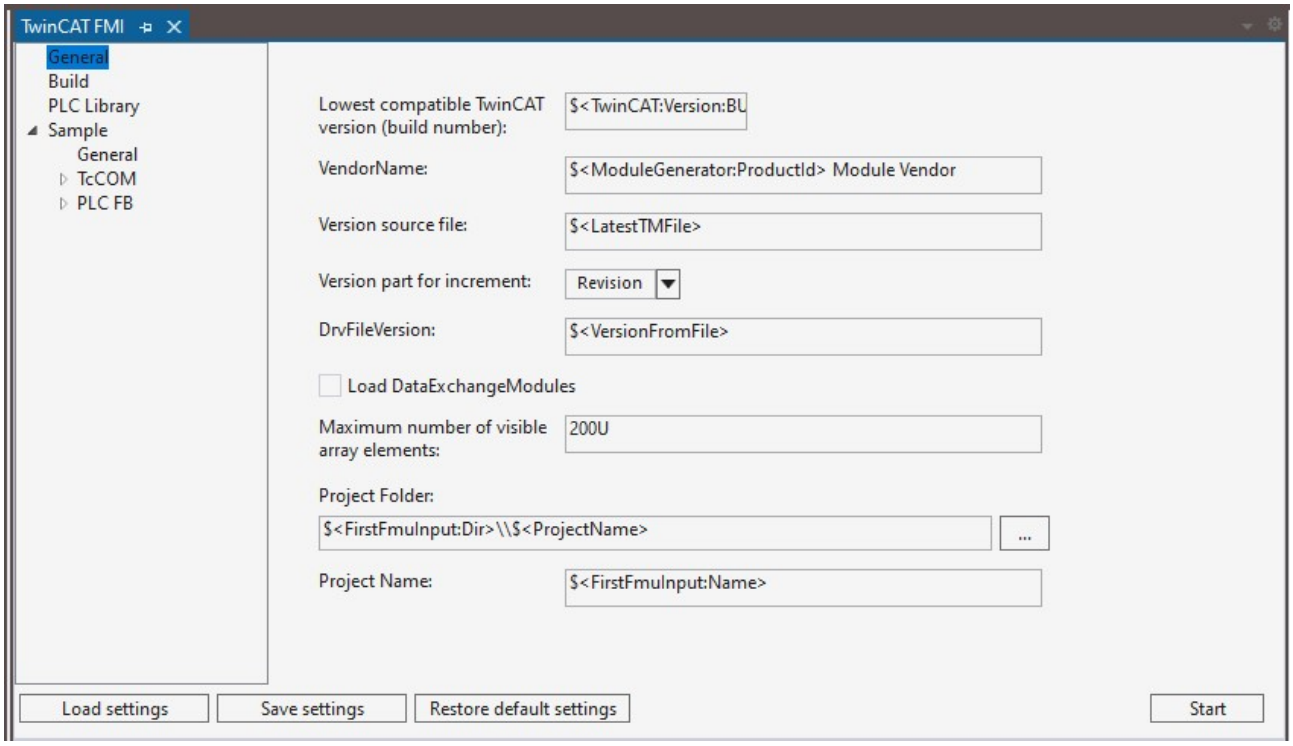
9 Reference user interface

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.

If you press the **Start** button, the conversion to TwinCAT runtime objects starts. At the same time, the current settings are saved to an XML file so that you can continue working on this project the next time you start the tool window. To reset the project press the **Restore default settings** button.

Use the buttons **Load settings** or **Save settings** to load or save FMI projects.



Load settings	Loads an FMI project.
Save settings	Saves the settings of the current FMI project.
Restore default settings	Restores a default project.
Start	Starts the import or conversion of the FMUs into TwinCAT runtime objects.

Context menu in the project tree

Right-click on an FMU:

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:

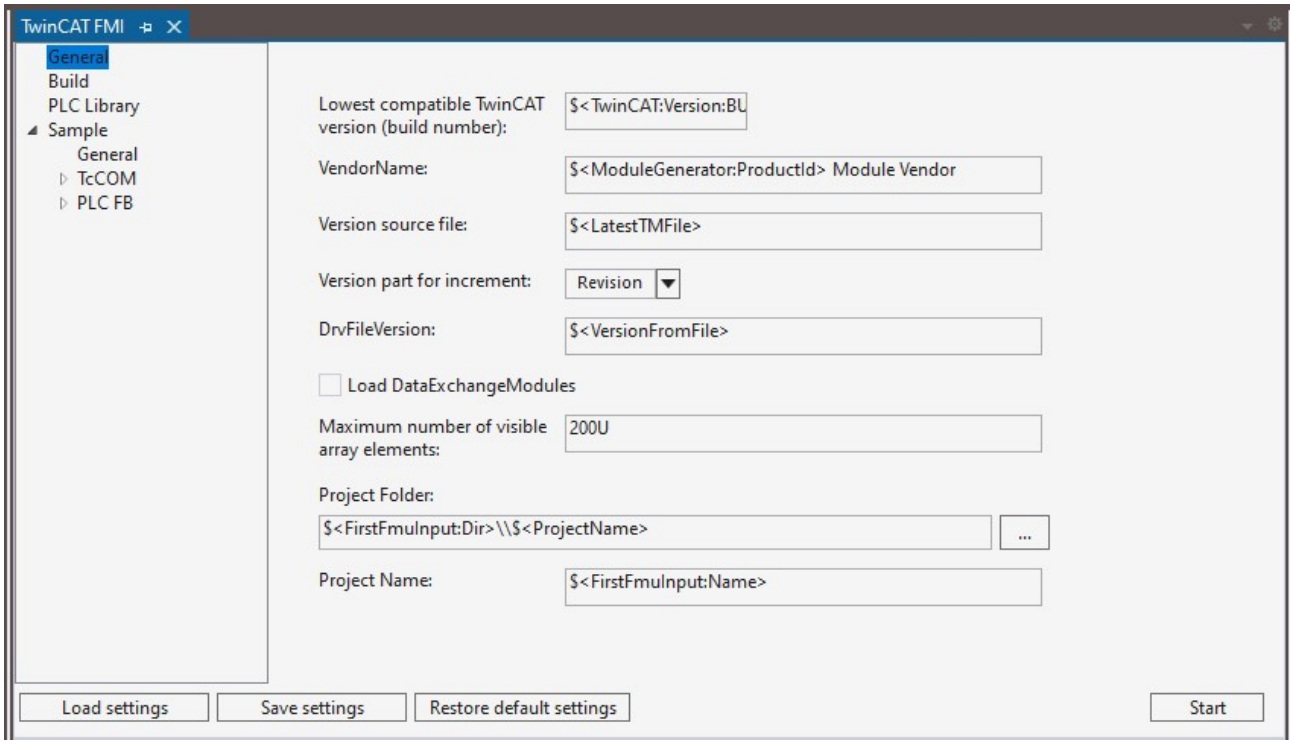
Restore defaults	Restores the default values for all FMUs in the TwinCAT FMI project.
Add new FMU	Adds another FMU to the TwinCAT FMI project.

Also see about this

'General' tab [▶ 26]

9.1 'General' tab

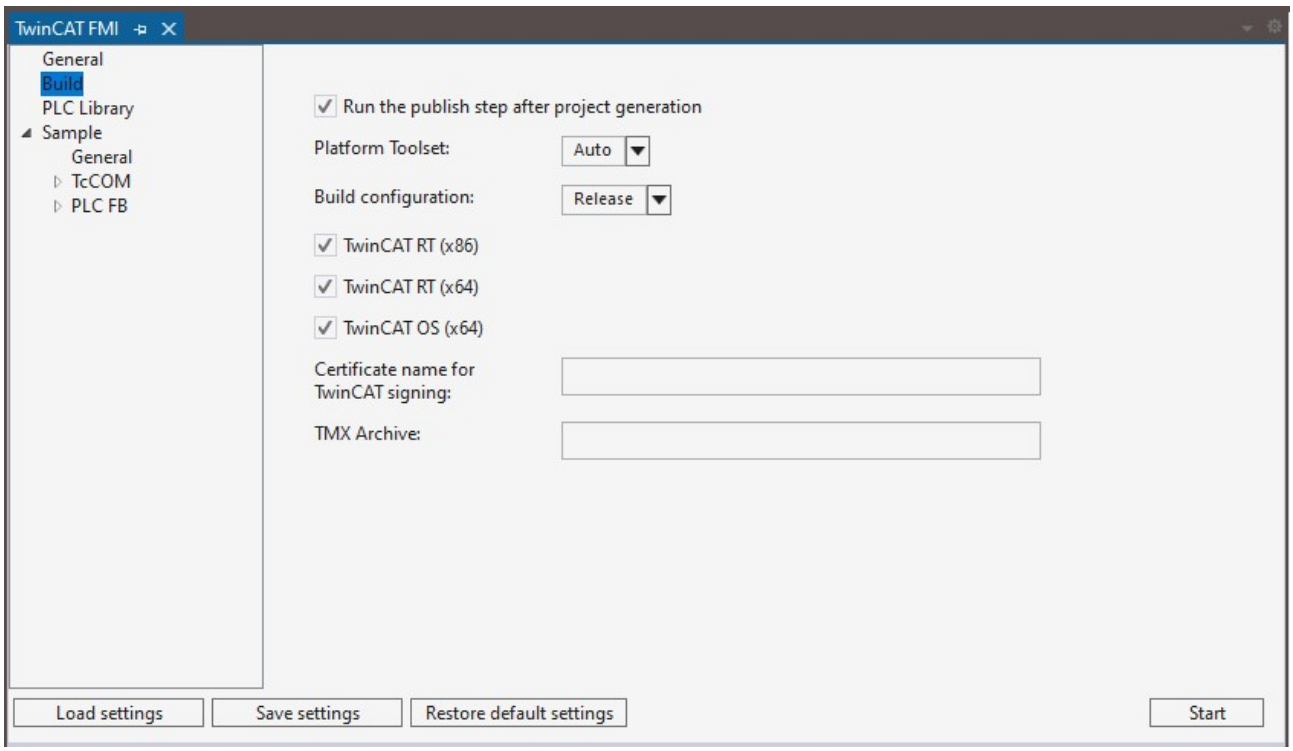
General settings can be made in the **General** tab. By default, these are provided with placeholders and are assigned the default values. Changed settings are persisted in an xml file and are retained for new projects. These can be reset via the **Restore default settings** button.



Lowest compatibel TwinCAT version	Minimum version of the TwinCAT runtime, so that the generated TcCOM module starts.
VendorName	Vendor name, the name under which the generated TcCOM module is sorted in the module selection dialog.
Version source file	File in which the version information is kept.
Version part for increment	Part of the version number to be incremented each time a new one is created.
DrvFileVersion	File version of the generated TcCOM module
Load DataExchangeModules	Loads the data exchange TcCom modules into memory so that they can be accessed from the generated module.
Maximum number of visible array elements	Maximum number of visible array elements
Project Folder	Project folder under which the TcCOM module is to be generated.
Project Name	Name of the project

9.2 'Build' tab

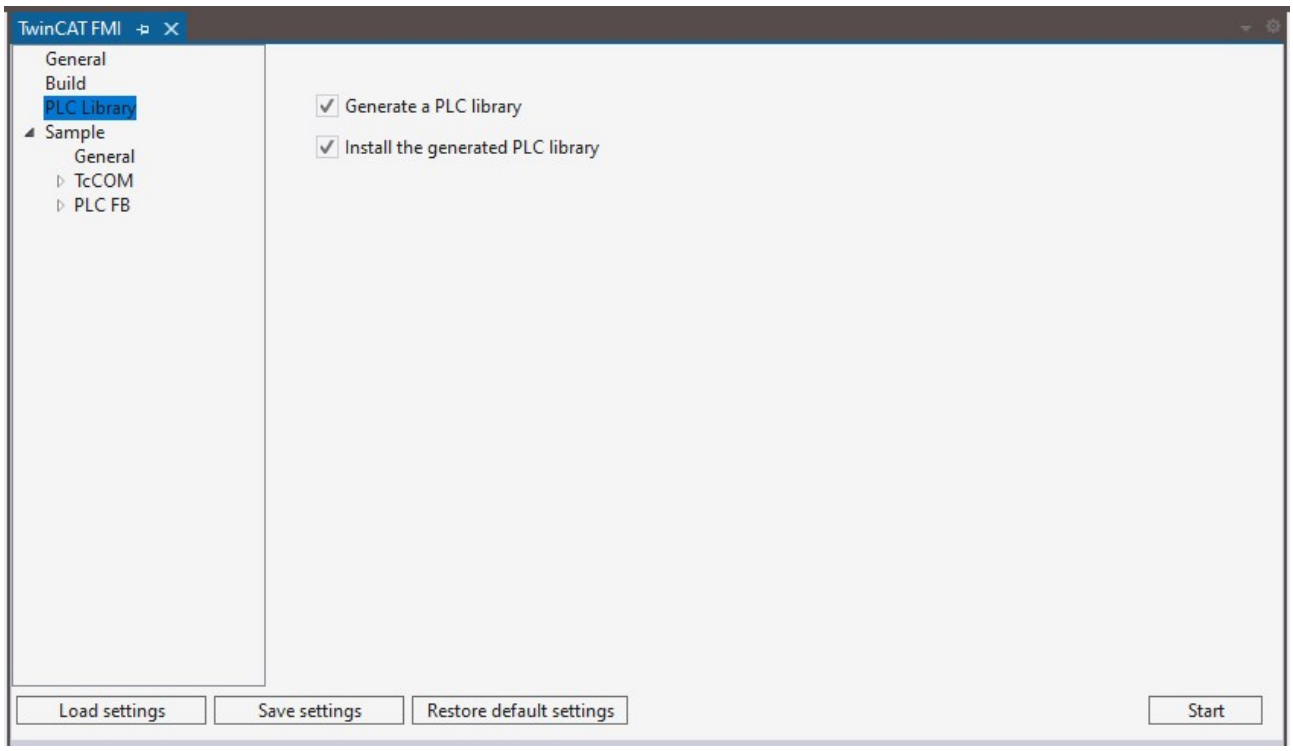
You can configure the build settings under the **Build** tab.



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.
Platform Toolset	Selection of the Visual Studio toolset used for building.
Build configuration	Selection of the build configuration (debug/release).
TwinCAT RT(x86)	Creating a tmx driver for the TwinCAT RT(x86) platform.
TwinCAT RT(x64)	Creating a tmx driver for the TwinCAT RT(x64) platform.
TwinCAT OS(x64)	Creating a tmx driver for the TwinCAT OS(x64) platform.
Certificate name for TwinCAT signing	TwinCAT certificate name to be used for signing the generated tmx driver.
tmx Archive	Name of a tmx archive that can be used for sharing the tmx files.

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.

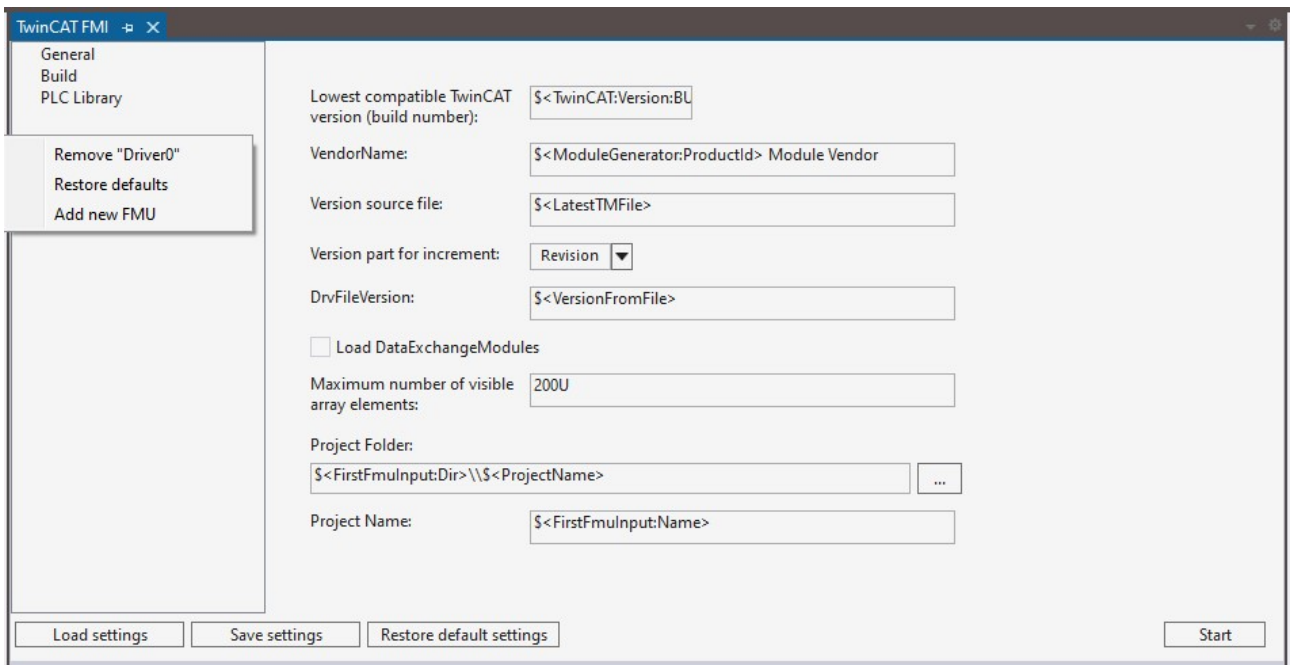


Generate a PLC library	Setting whether a PLC library should be created.
Install the generated PLC library	Setting whether the generated PLC library should also be installed.

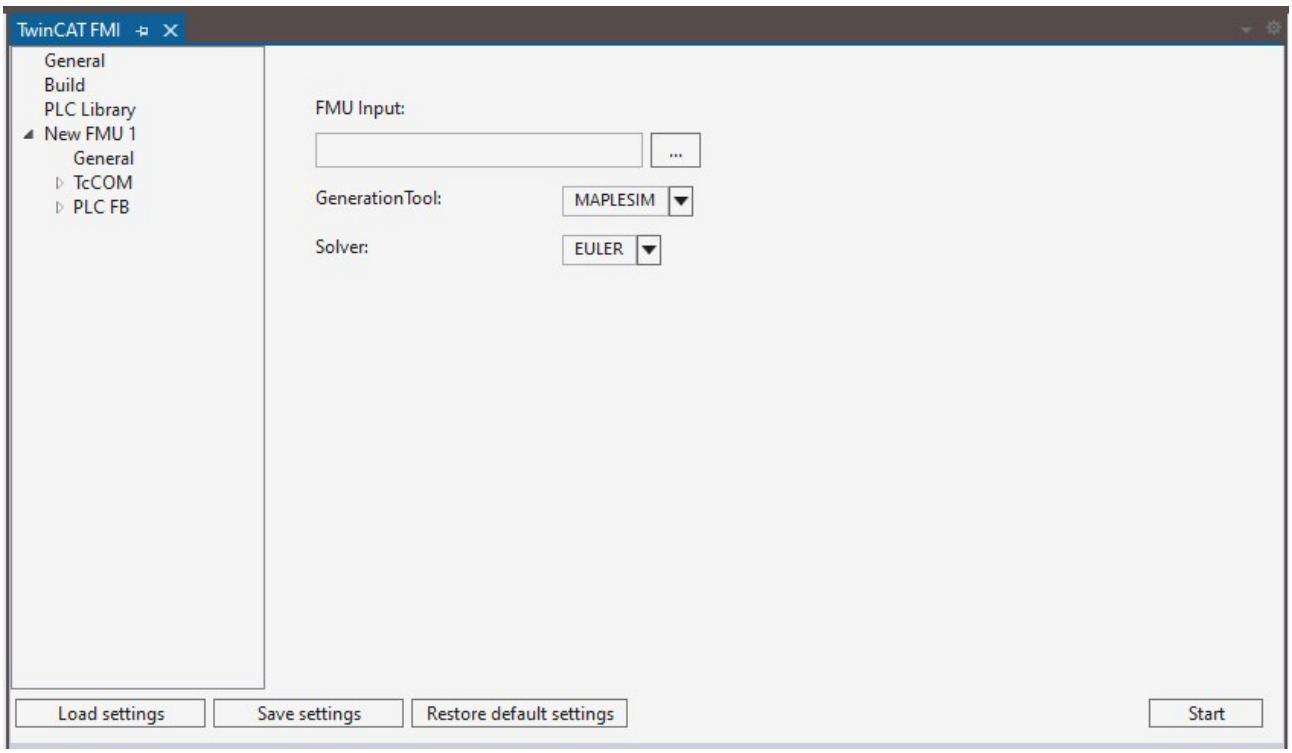
9.4 'NewFmuModule1' tab

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.



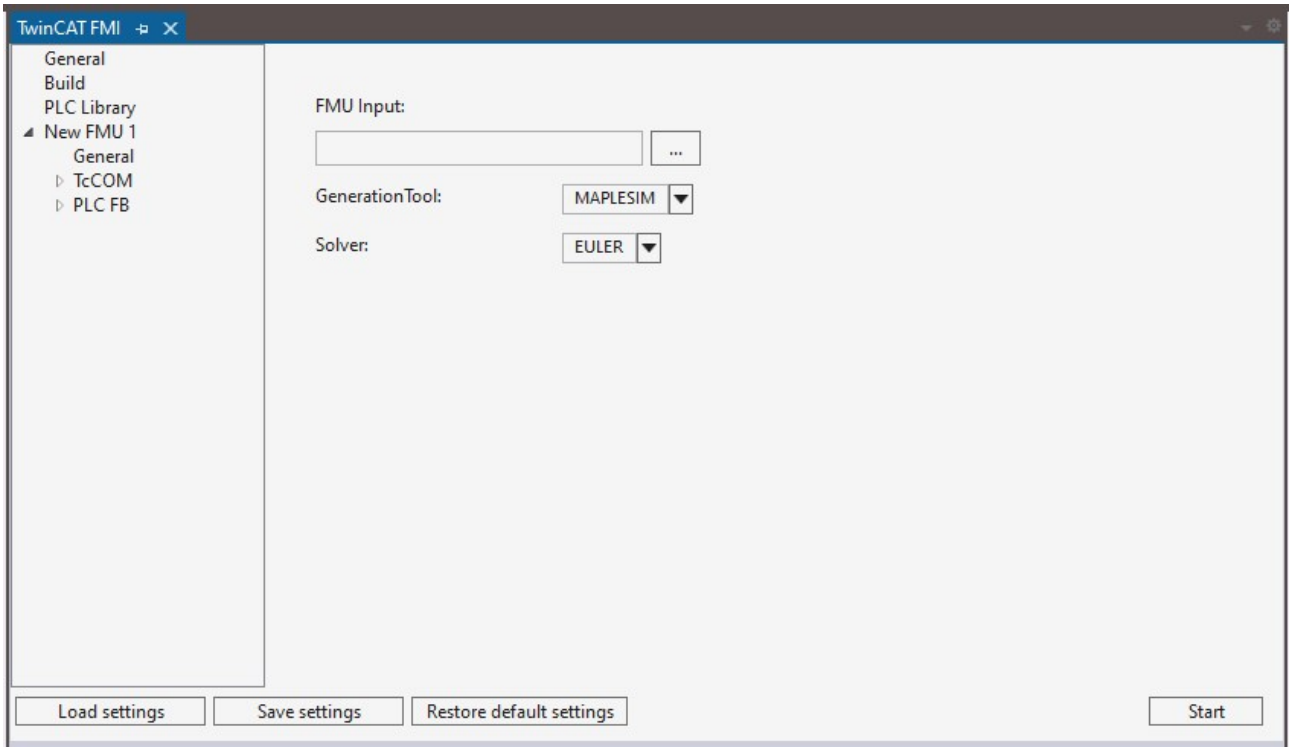
In the default state or when an FMU has been added, the interface looks like the following figure:



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. These are, for example, the selection of the tool that generated the FMU or the selection of the solver to be used for the calculation of the FMU. 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.



FMU Input:	Selection of the FMU to be imported.
Generation Tool:	Selection of the tool that generated the FMU.
Solver:	Selection of the solver to be used.

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 field. The solvers can be selected for solving the equations.

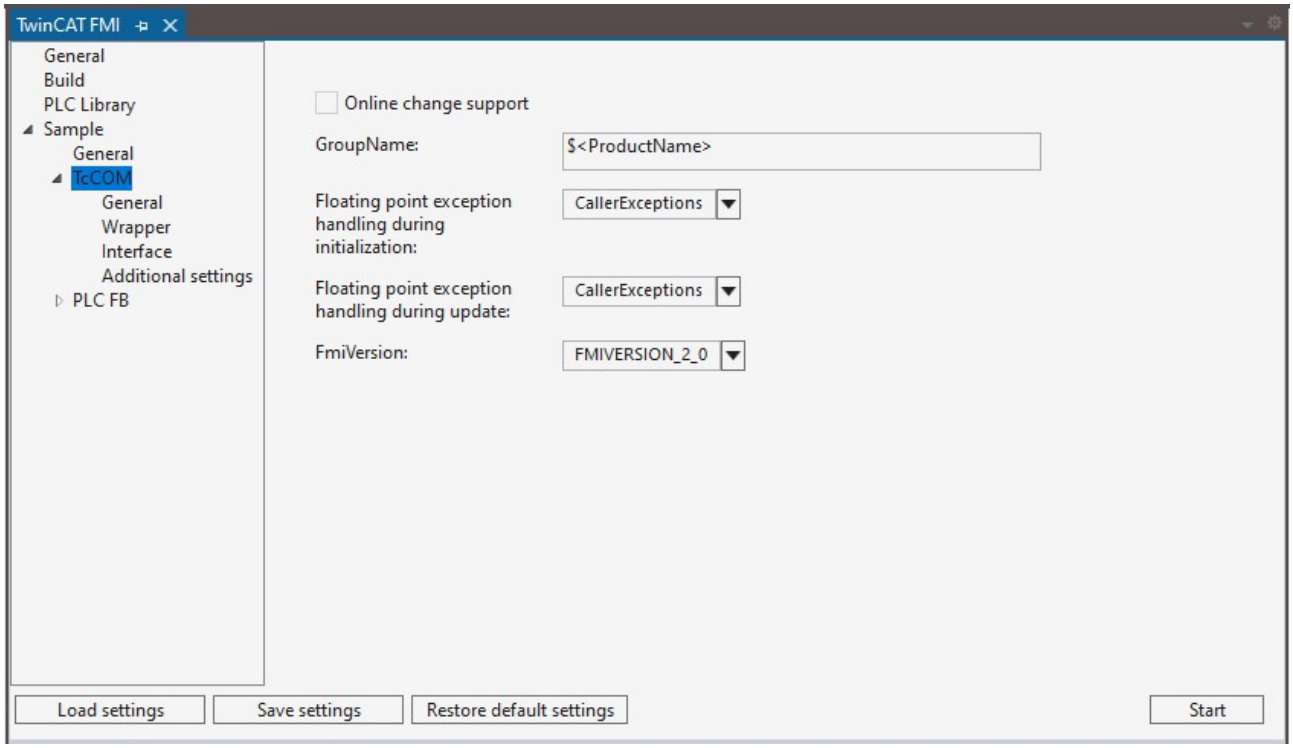
If an FMU was exported via co-simulation, it already contains the solver. Co-simulation must then be selected in the **Solver** selection field.

9.4.2 'TcCom' tab

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

9.4.2.1 'General' tab

Under the tab you can change the general settings of the TcCom module to be generated. This affects not only the group name of the module, but also the online change capability and the handling of floating point exceptions.

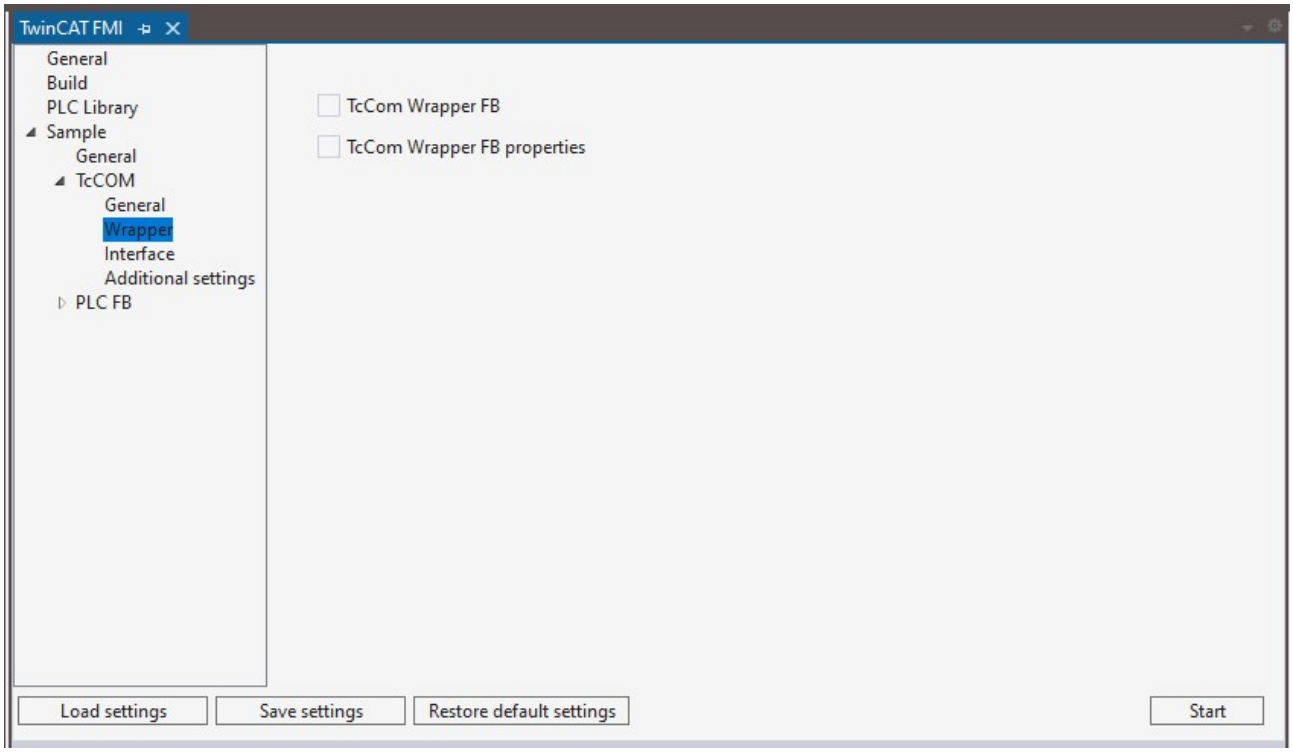


Online change support	Enable/disable online change support.
GoupName	Group name under which the generated TcCom module is sorted in the selection dialog **Insert TcCom Object** in the TwinCAT3 development environment.
Floating point exception handling during initialization	Handling of floating point exceptions during initialization of the TcCom module.
Floating point exception handling during update	Handling of floating point exceptions during the call of the TcCom module.
FmiVersion	Used version of the FMI standard.

9.4.2.2 '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 formation PlcOpen XML and can be imported manually into a PLC project.

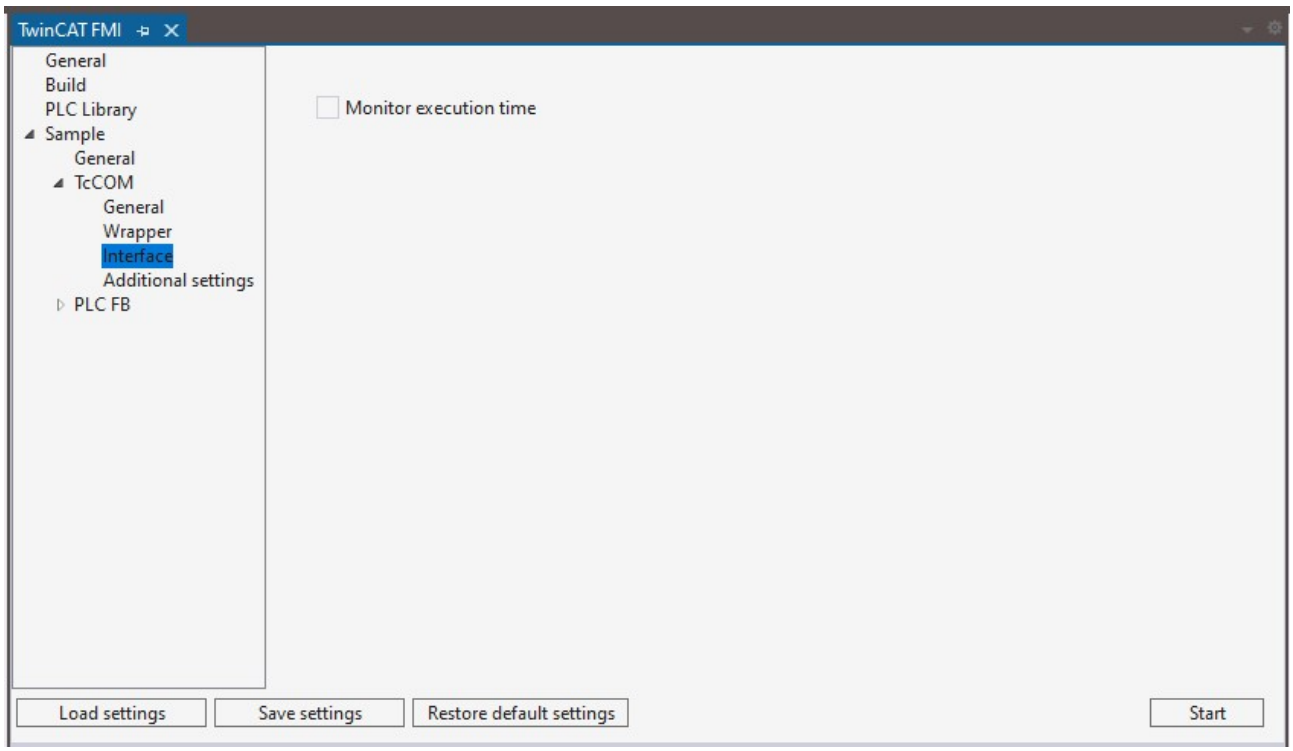
Unlike the settings in 'PLC Library' tab [▶ 27], which create a PLC library in which the individual models are represented as function blocks, the settings in the tab **TcCom->Wrapper** create a function block that "wraps" the call of the TcCom module. In this case, the actual TcCom module "lives" outside the PLC and is only called or influenced by it.



TcCom Wrapper FB	Creates a Wrapper FB.
TcCom Wrapper FB properties	Creates a Wrapper FB where the variables are accessible as properties.

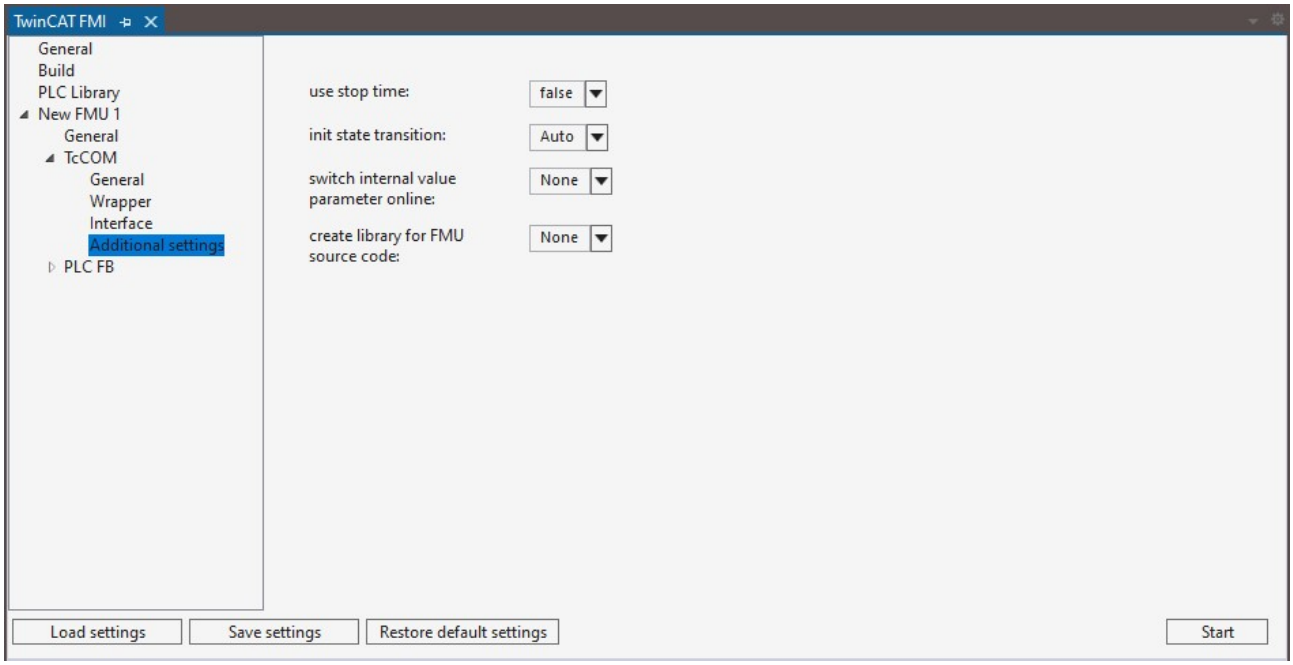
9.4.2.3 'Interface' tab

Under this tab you can set whether a cycle time monitoring is installed in the module itself, which calculates the execution time in each cycle.



9.4.2.4 'Additional settings' tab

Under this tab, additional settings can be made which are not normally required.



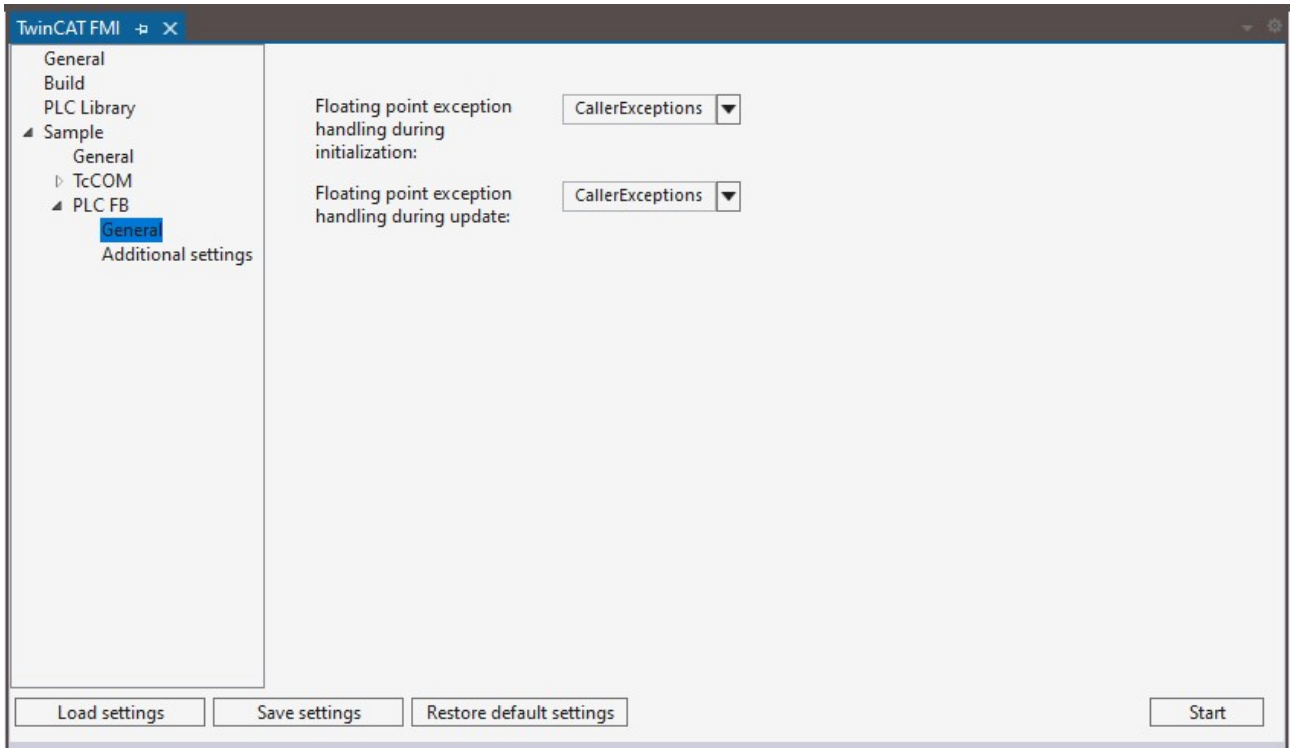
Use stop time	Setting whether the execution of calculations stops after the duration set in the simulation system or not.
Init state transition	Setting at which state transition the initialization of the internal variables takes place.
Switch internal value parameter online	Setting whether internal parameters may be switched online.
Create library for FMU source code	

9.4.3 '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 [▶ 27].

9.4.3.1 'General' tab

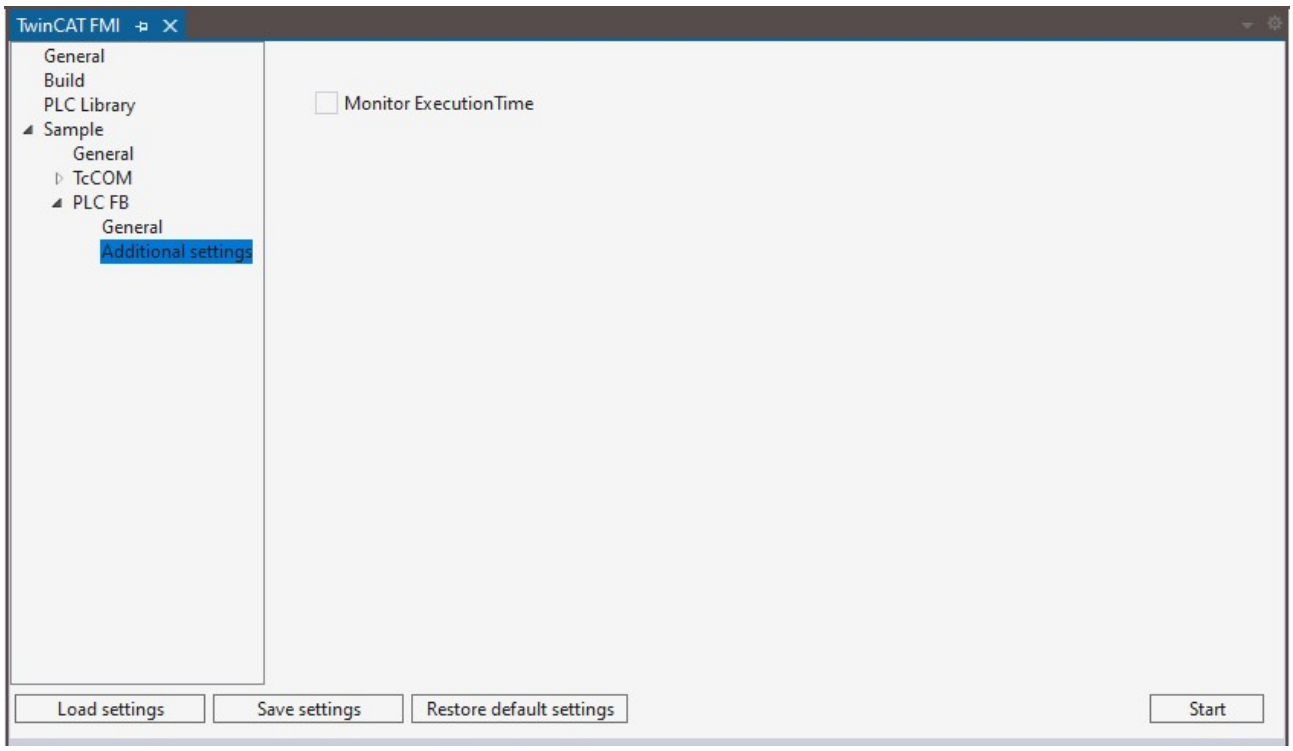
Under this tab you can make general settings for the generated PLC function blocks.



Floating point exception handling during initialization	Handling of floating point exceptions during initialization of the TcCom module.
Floating point exception handling during update	Handling of floating point exceptions during the call of the TcCom module.

9.4.3.2 'Additional settings' tab

Under this tab, additional settings can be made that affect the generated PLC function blocks.



Monitor Execution Time	Enable/disable monitoring of the execution time.

10 Support and Service

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

Beckhoff's branch offices and representatives

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

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages: https://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
Fax:	+49 5246 963 9157
e-mail:	support@beckhoff.com

Beckhoff Service

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

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

Hotline:	+49 5246 963 460
Fax:	+49 5246 963 479
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963 0
Fax:	+49 5246 963 198
e-mail:	info@beckhoff.com
web:	https://www.beckhoff.com

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