

## SIMATIC/SIMOTION ProjectGenerator

Application manual

Preface

Application description

1

System and error messages

2

Tips and assistance

3

Contact

A

## Legal information

### Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 <b>DANGER</b>
---

indicates that death or severe personal injury <b>will</b> result if proper precautions are not taken.
--

 <b>WARNING</b>
--

indicates that death or severe personal injury <b>may</b> result if proper precautions are not taken.
---

 <b>CAUTION</b>
--

indicates that minor personal injury can result if proper precautions are not taken.
--

<b>NOTICE</b>
---------------

indicates that property damage can result if proper precautions are not taken.
--

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

### Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

### Proper use of Siemens products

Note the following:

 <b>WARNING</b>
--

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.
--

### Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

### Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

# Preface

## General information

---

### Note

The standard applications are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. The standard applications do not represent specific customer solutions, but are only intended to provide support for typical tasks. You are responsible for the proper operation of the described products. These standard applications do not relieve you of your responsibility regarding the safe handling when using, installing operating and maintaining the equipment. By using these standard applications, you agree that Siemens cannot be made liable for possible damage beyond the mentioned liability clause. We reserve the right to make changes and revisions to these standard applications at any time without prior notice. In the case of any differences between the suggestions made in these standard applications and other publications from Siemens, such as catalogs, the contents of the other documentation have priority.

---

## Warranty conditions, liability, and support

If the application has been made available free of charge, the following applies:

We do not provide a warranty for any of the information contained in this document.

All other rights and claims against Siemens AG irrespective of legal basis are excluded. In particular claims for damages against Siemens AG in the case of product outage, downtime, loss of profit, either directly, indirectly or consequential damage are excluded.

This does not apply when liability is compulsory by law, e. g. in the case of the Product Liability Act, premeditation, an act of gross negligence by superiors and managerial staff of Siemens AG or in cases of fraudulent concealment of defects.

This limitation of liability also applies to sub-contractors, suppliers, delegates, superiors and managerial staff of Siemens AG.

German law shall apply to this agreement for customers with head offices in Germany; Swiss law for customers with head offices outside Germany. Application of the United Nations Convention on Contracts for the International Sale of Goods as of 11.04.1980 (CISG) is excluded.

If the application has been made available against payment, the appropriate alternative applies for the respective business transaction:

- Alternative 1: (Internal business)

If nothing else has been negotiated, then the "Conditions for the supply and services in Siemens internal business" applies in the version that is valid at the time that the equipment is purchased.

- Alternative 2: (Domestic business of Siemens AG)

If nothing else was negotiated, the "General License Conditions for Software for Automation and Drives for Customers with a Registered Office in Germany" valid at the time of sale are applicable.

- Alternative 3: (Direct export business of Siemens AG)

If nothing else has been negotiated, then the "General License Conditions for Software Products for Automation and Drives for Customers with a Seat or Registered Office outside Germany", valid at the time of sale, are applicable.

It is not permitted to distribute or duplicate these application examples in any form including excerpts thereof without the express consent of Siemens Industry Sector.

## Notice regarding export identification codes

AL: N

ECCN: N

## About this document

### Objective

This document describes the use of the ProjectGenerator for SIMATIC/SIMOTION with which you can quickly and easily create a SIMOTION SCOUT/SIMATIC STEP 7 project and update libraries and modules in an existing project.  
Previous knowledge of STEP 7 and SIMOTION SCOUT is required.

---

#### Note

This document does not claim to contain all details on devices in any version or to take all conceivable operational cases and applications into account.

Should you require further information or encounter specific problems not covered in enough detail for your field of application, please contact your local Siemens office.

---

## Target group

This document is intended for programmers and commissioning engineers.

# Table of contents

	<b>Preface .....</b>	<b>3</b>
<b>1</b>	<b>Application description .....</b>	<b>7</b>
1.1	General information .....	7
1.1.1	Task of the ProjectGenerator .....	7
1.1.2	Scope of delivery .....	10
1.1.3	Requirements .....	14
1.2	Operating the ProjectGenerator .....	15
1.2.1	Preparations .....	15
1.2.2	Starting the ProjectGenerator .....	15
1.2.3	Representation of the configuration sequence .....	17
1.2.4	Creating a sample project .....	19
1.2.4.1	Prerequisites .....	19
1.2.4.2	Creating a new project with SIMATIC devices .....	19
1.2.4.3	Adapting/expanding an existing project with SIMOTION devices .....	27
1.2.4.4	Transfer of SIMOTION IT pages .....	35
1.3	User-specific expansions .....	39
1.3.1	Adding devices and standard modules .....	39
1.3.1.1	Expanding the ProjectGenerator .....	39
1.3.1.2	Adding devices .....	39
1.3.1.3	Adding standard modules .....	41
1.3.2	Adding technology objects (SIMOTION only) .....	42
1.3.3	Creating and adding user-specific modules .....	43
1.4	Architecture of the ProjectGenerator .....	45
1.4.1	General information .....	45
1.4.2	Structure of the ProjectGenerator .....	45
1.4.3	Using the ProjectGenerator in silent mode .....	47
<b>2</b>	<b>System and error messages .....</b>	<b>49</b>
2.1	General information .....	49
2.2	System information .....	49
2.3	Warnings .....	54
2.4	Error messages .....	55
<b>3</b>	<b>Tips and assistance .....</b>	<b>63</b>
3.1	Special characters in XML .....	63
3.2	Comments in the source code .....	64
3.3	Outputting a message box .....	64
3.4	Usable parameters, events, and objects .....	64
3.5	Using text files .....	65
3.6	Using a source several times .....	67

3.7	Reloading a user interface .....	68
3.8	Accessing user interface elements generically.....	69
3.9	Changing the user interface generically .....	69
3.10	Performant interface elements.....	77
<b>A</b>	<b>Contact .....</b>	<b>79</b>
A.1	Contacts .....	79
A.2	Internet addresses .....	80

# Application description

## 1.1 General information

### 1.1.1 Task of the ProjectGenerator

#### **Automatic creation of a SIMATIC/SIMOTION project**

With the ProjectGenerator, you can quickly and easily create a new SIMATIC/SIMOTION project or update an existing project on the basis of templates.

#### **Function collection of standard modules and technological modules**

The templates include preconfigured SIMATIC/SIMOTION devices and modules, which are all supplied with the ProjectGenerator. These can be automatically integrated in a project with the ProjectGenerator.

The function collection can also be expanded with user-specific modules.

#### **Configuration and parameterization via user interface**

The ProjectGenerator provides a user interface for the configuration and parameterization of the modules that allows module-specific adaptations.

The modules can be expanded without the source code of the ProjectGenerator having to be changed.

#### **Standardized process hierarchy in accordance with ISA-88**

The ProjectGenerator uses internationally established industrial standards such as IEC, XML, and ISA-88. By using the standardized hierarchies and interfaces based on international industrial standards, the individual modules of the ProjectGenerator can be integrated for data exchange within a machine as well as between the machines in a line. The modules can also be combined with components from other manufacturers and integrated in existing infrastructures.

The ProjectGenerator is suitable for use in the four lower levels of the hierarchy model in accordance with standard ISA-88:

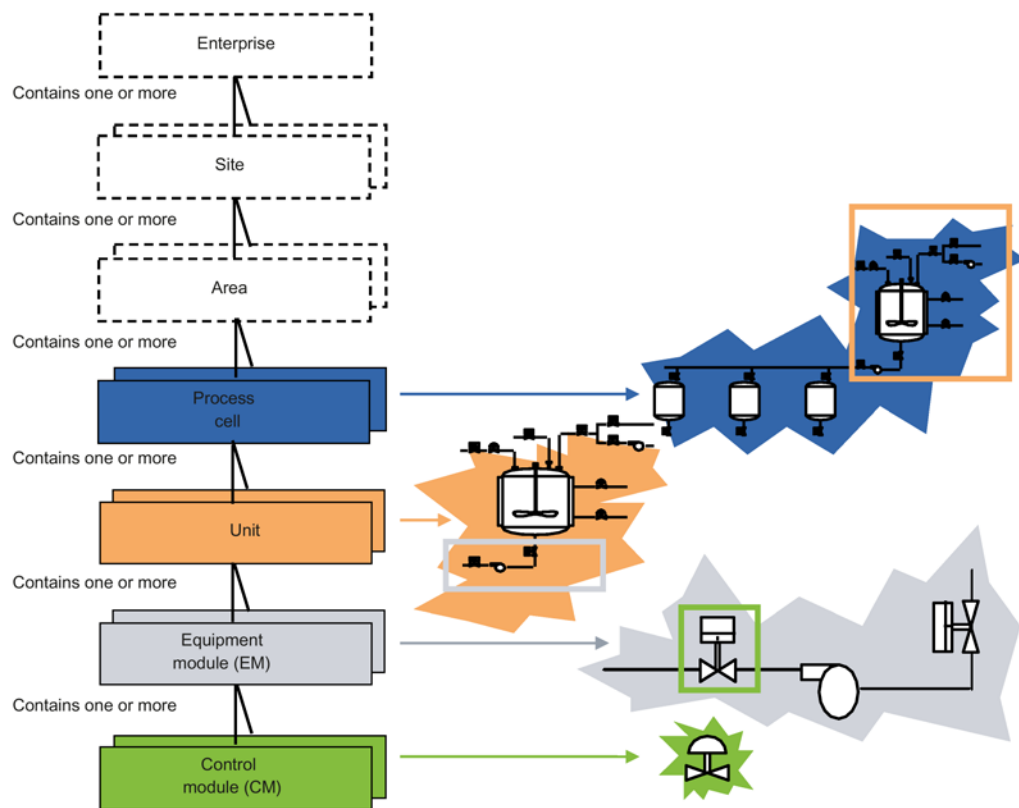


Figure 1-1 Standardized process hierarchy in accordance with ISA-88

The equipment modules (EM) and control modules (CM) contain hardware and software components that only have to be written once and then can be used as often as required in the hierarchies shown in the figure.

According to ISA-88, a unit (machine) can be made up of one equipment module, e.g. the *handling* machine module, or several identical or different equipment modules. An equipment module can in turn contain several equipment modules and/or control modules. Control modules are modules with process interfacing, e.g. an axis or the temperature controller. A control module can in turn contain several control modules.

The ProjectGenerator presently supports one equipment modules level and one control modules level below.

## Overview of the ProjectGenerator functions

You can execute the following functions with the ProjectGenerator:

Table 1- 1 Functions of the ProjectGenerator

Function	ProjectGenerator for SIMATIC	ProjectGenerator for SIMOTION
Create a new project or open and edit an existing project	✓	✓
Create a new device or open and edit of an existing device	✓	✓
Automatic configuration of the modules on the basis of the selected hardware	✓	✓
Copy files/directories in the file system	✓	✓
XML interface for user-specific expansions	✓	✓
<ul style="list-style-type: none"> <li>• Generation of the user interface elements</li> <li>• Generic function calls</li> </ul>	✓ ✓	✓ ✓
Assign programs to the execution system		✓
FTP transport of files to the SIMOTION device (e.g. SIMOTION IT pages)		✓
Import technology objects (TO)		✓
Interconnect technology objects (synchronous operation inter-connection)		✓
Write configuration data and system variables to technology objects		✓
Import libraries:	✓	✓
<ul style="list-style-type: none"> <li>• Change code</li> <li>• Version check and version update</li> <li>• Setting and restoring constants</li> <li>• Creating and restoring user areas</li> </ul>	✓ ✓	✓ ✓ ✓ ✓
Import units		✓
<ul style="list-style-type: none"> <li>• Change code</li> <li>• Version check and version update</li> <li>• Setting and restoring constants</li> <li>• Creating and restoring user areas</li> </ul>		✓ ✓ ✓ ✓
Importing sources	✓	
Importing/adding networks in organization blocks	✓	
Adding variables to the declaration area of an organization block	✓	

### 1.1.2 Scope of delivery

The ProjectGenerator can be downloaded via a link on the **Utilities & Applications** storage medium. **Utilities & Applications** is part of the SIMOTION SCOUT engineering system.

#### Directory structure of the ProjectGenerator

In Version 1.2, the ProjectGenerator is supplied with the following directory structure:

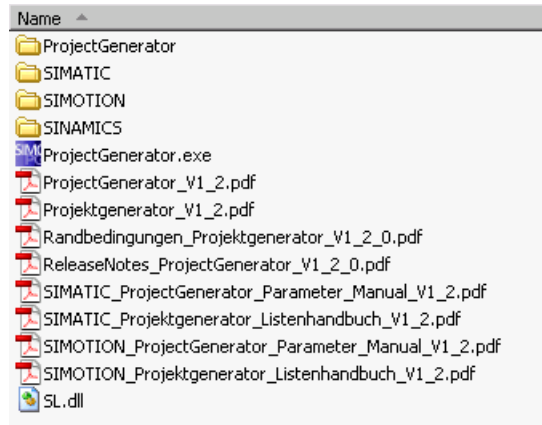


Figure 1-2 Structure of the ProjectGenerator

The top level contains

- The **ProjectGenerator.exe** file with which you start the **ProjectGenerator**.
- The **documentation** for the ProjectGenerator in the form of pdf documents in German and English.

#### ProjectGenerator directory

This contains the files required for operating the ProjectGenerator. The files are system-internal files and should not be changed by the user.

#### SIMATIC, SIMOTION, and SINAMICS directories

The basic data (directories and files) needed in connection with the generation of a SIMATIC or SIMOTION project can be found in the SIMATIC and SIMOTION directories.

Importable SINAMICS devices are in the SINAMICS directory.

The SIMATIC, SIMOTION, and SINAMICS directories contain the following subdirectories:

- **Devices**

The **Devices** directory contains the files of the SIMATIC, SIMOTION, and SINAMICS devices.

It is also possible to add your own preconfigured devices to this directory.

Table 1- 2 Examples of devices

Folder	Subfolder	Devices
SIMATIC	Devices	<b>S7300</b> SIMATIC S7315 SIMATIC S7317 SIMATIC S7319 ...
		<b>S7400</b> SIMATIC S7414 SIMATIC S7416 ...
SIMOTION	Devices	<b>V4.2:</b> SIMOTION C240PN SIMOTION D410PN SIMOTION D425 SIMOTION D435 ...
		<b>V4.3:</b> SIMOTION C240PN SIMOTION D410-2PN SIMOTION D425-2 SIMOTION D435-2 ...
SINAMICS	Devices	<b>1_DriveObjects</b> <b>2_ DriveObjects</b> <b>3_ DriveObjects</b> ... The IO addresses of the drive objects for HW Config can be stored in the additional file to be created, <i>IOConfig.XML</i> .

- **EquipmentModules**

The **EquipmentModules** directory contains version-dependent, preconfigured standard modules of the ProjectGenerator for SIMATIC or SIMOTION.

These are, for example:

Table 1- 3 Examples of SIMATIC and SIMOTION EquipmentModules

Folder	Subfolder	EquipmentModules
SIMATIC	EquipmentModules	<b>Communication_LCom</b> The <i>FBLComMachineCom</i> function block in the <b>LCom</b> library provides the function of a data record-oriented transport protocol, LCom protocol. The TCP transport protocol is used for data transmission via Ethernet.
		<b>OMACV30_LPML</b> The <i>LPMLV30</i> software library provides a user-friendly basis for the configuration of an OMAC-compliant mode manager and a data interface for a SIMATIC/SIMOTION device.
		<b>Weihenstephan</b> The <i>LWeihenstephan</i> library provides the functionality of the Weihenstephan standard for SIMATIC S7 controls.
		...
SIMOTION	EquipmentModules	<b>Communication_LCom</b> The <i>FBLComMachineCom</i> function block in the <b>LCom</b> library provides the function of a data record-oriented transport protocol, LCom protocol. The TCP transport protocol is used for data transmission via Ethernet.
		<b>OMACV30_LPML</b> The <i>LPMLV30</i> software library provides a user-friendly basis for the configuration of an OMAC-compliant mode manager and a data interface for a SIMATIC/SIMOTION device.
		<b>StartupCheck</b> The startup check provides functions for checking I/O modules and technology objects of the axis and external encoder type when the machine starts up. If an error occurs, the user is provided with detailed information about the cause of the fault. When the startup is successful, a signal or a message is output.
		<b>Messagehandling_LMsgHdl</b> The <i>LMsgHdl</i> library provides basic functions for displaying and managing all types of messages of the SIMOTION system (e.g. faults and alarms).
		...

- **Projects (SIMOTION only)**

This contains the original project that is used by the ProjectGenerator when **New project** is selected. It includes an already networked programming device for going online later.

The Ethernet interface is already pre-assigned with the address 169.254.11.99 and only has to be assigned to the network card. This operation is not covered by the ProjectGenerator, because there are too many different network adapters on the market.

If you want to store a customer-specific basic project, you must make sure you do not rename the `\SIMOTION\Projects\Project_Basis` directory, but only replace the project files in this directory.

If a SIMOTION CPU is inserted into the project, it is automatically connected to the standard **Ethernet(1)** subnet if this exists. If it does not exist, the CPU is imported without networking. The IE2/NET (X130) Ethernet interface is always networked as an interface for the SIMOTION D4x5 components. The X200 PN interface is networked for the SIMOTION D410 PN as standard. PN/IE (X127) is used for all SIMOTION D4xx-2 components. Every other CPU also connects to this standard subnet when imported.

If the CPU is to be connected to a different subnet, this can be performed subsequently via NetPro or directly in the database in the ProjectGenerator.

- **Scripts (SIMOTION only)**

This contains the version-dependent configuration and overview scripts that can be imported via the ProjectGenerator.

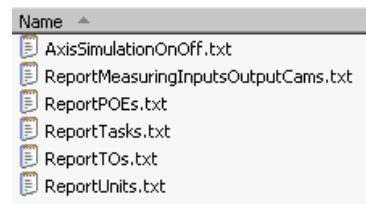


Figure 1-3 Scripts directory

- **TechnologyObjects (SIMOTION only)**

The **TechnologyObjects** directory contains version-dependent XML exports of the technology objects of the axis type, e.g.:

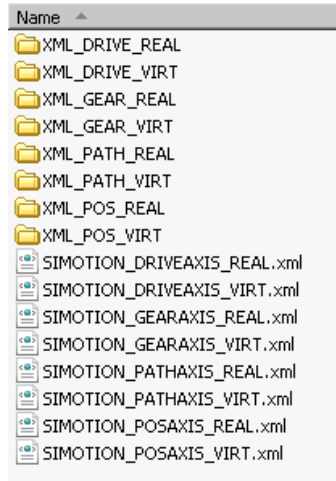


Figure 1-4 TechnologyObjects directory

Irrespective of the technology objects listed here, each module can save and address its own technology objects in the relevant subdirectory.

- **TechnologyPackages (SIMOTION only)**

The **TechnologyPackages** directory contains version-dependent internal system files with the identifiers for the technology packages and devices.

- **TelegramConfig (SINAMICS only)**

The **TelegramConfig** directory contains version-dependent internal system files with the telegram configuration and MLFBs of the SINAMICS Control Units.

### 1.1.3 Requirements

Use of the ProjectGenerator requires an installation of SIMOTION SCOUT V4.2 or higher and STEP 7 V5.5.

If you only want to use the SIMATIC functions, SIMOTION SCOUT does not have to be installed.

## 1.2 Operating the ProjectGenerator

### 1.2.1 Preparations

---

**Note**

Before starting the ProjectGenerator, all STEP 7 or SIMOTION SCOUT applications have to be closed.

---

---

**Note**

To ensure quick and stable running, the ProjectGenerator must be started from a local drive.

---

### 1.2.2 Starting the ProjectGenerator

The ProjectGenerator can be started in **standard mode** and in **silent mode**.

#### Standard mode

Double-click the **ProjectGenerator.exe** file if you want to start the ProjectGenerator in **standard mode**.

The ProjectGenerator interface is loaded and guides you step-by-step from the creation of the project through to the generation.

Further information and the required individual steps can be found in the section Creating a sample project (Page 19).

#### Silent mode

You can execute the ProjectGenerator in **silent mode** without a dialog with the user program. In this way, it is possible to integrate the ProjectGenerator in a separate application.

You activate the **silent mode** by transferring the path to a valid external XML file of the ProjectGenerator to the *ProjectGenerator.exe* file when starting. The commands of the file are then processed directly.

The following is a sample call for a batch file with the extension *.xm/* for starting the ProjectGenerator in **silent mode**:

Table 1- 4 Coding example: ProjectGenerator for SIMOTION

```
C:\SIMOTION\ProjectGenerator.exe C:\SIMOTION\PGEN_DATA_BASE.xml
if errorlevel 1 goto error1
if errorlevel 0 goto no_error
goto End

:error1
Echo Error
goto End

:no_error
Echo No Error
goto End

:End
```

For further information about **silent mode**, see Section Using the ProjectGenerator in silent mode (Page 47).

### 1.2.3 Representation of the configuration sequence

#### Start of the ProjectGenerator

If you have started the ProjectGenerator, the user interface of the ProjectGenerator is loaded after checking the software requirements and confirmation of the disclaimer of liability message. This area **cannot be influenced** by the user (see Fig. *Configuration sequence 1*):

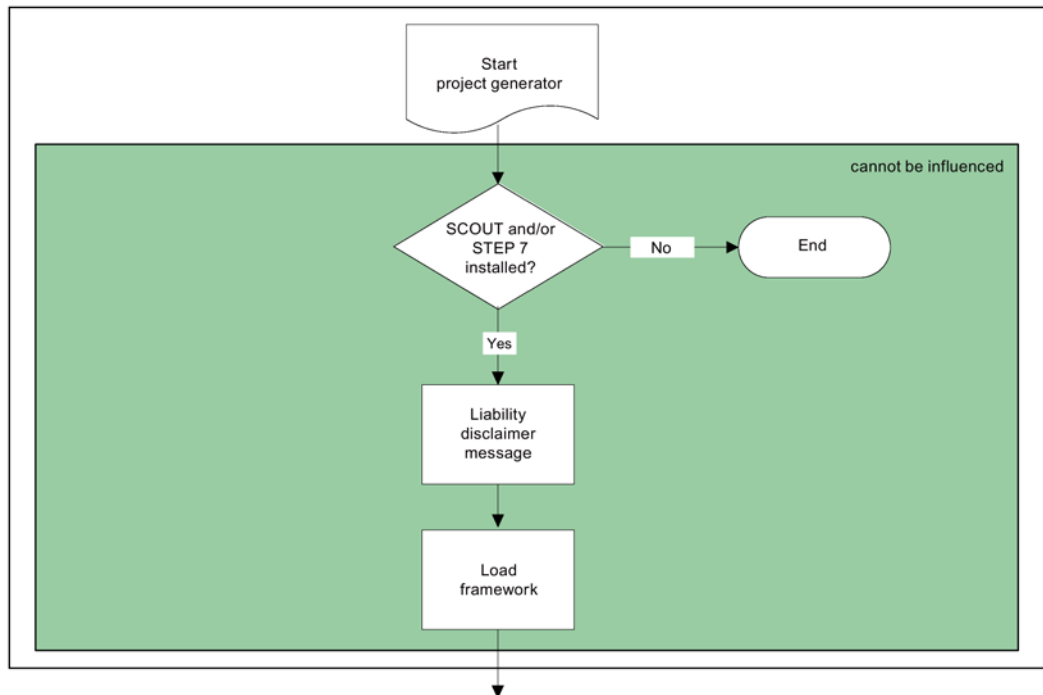


Figure 1-5 Configuration sequence 1

#### Creating a new project or updating an existing project

In the next step you specify whether you want to create a new project or want to load and edit an existing project.

The SIMATIC or SIMOTION device or SIMATIC/SIMOTION devices are then created or selected for the project. Whereby each individual device is configured in succession.

The selection and configuration of the individual modules represents the **area that can be influenced by the user** (area highlighted in the middle of Fig. *Configuration sequence 2*). Standard modules can be integrated or also user-specific modules that can be freely configured (see section Creating and adding user-specific modules (Page 43)).

## Generation of the project and opening of SIMOTION SCOUT

When all devices have been configured, the project is generated. After generation, you can open the project that has just been created or updated. In the case of a SIMOTION project, SCOUT can be opened directly from the ProjectGenerator. The modules which were previously selected and configured are now contained in the project.

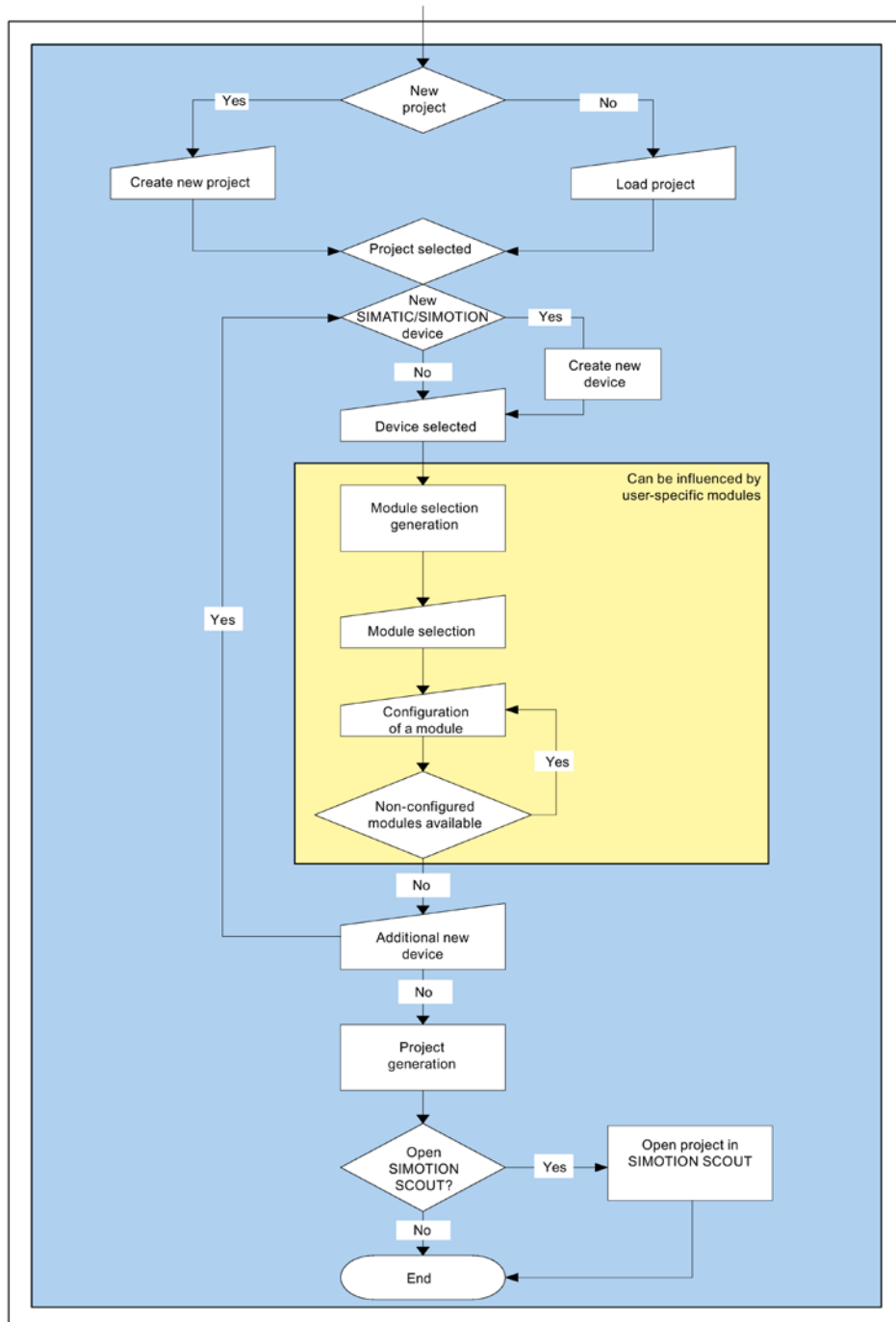


Figure 1-6 Configuration sequence 2

## 1.2.4 Creating a sample project

### 1.2.4.1 Prerequisites

#### Requirements

- SIMOTION SCOUT and STEP 7 are closed
- The functions and properties of the standard modules are known. The documentation of the individual modules is supplied in the files of the ProjectGenerator.

### 1.2.4.2 Creating a new project with SIMATIC devices

In this sample project, a new SIMATIC device is created with the ProjectGenerator using the standard modules *Ethernet Communication LCom* and *Weihenstephan Standard*.

### How to create a new SIMATIC device with standard modules:

1. Start the ProjectGenerator, agree to the liability disclaimer, and select the option **Create a new project**.

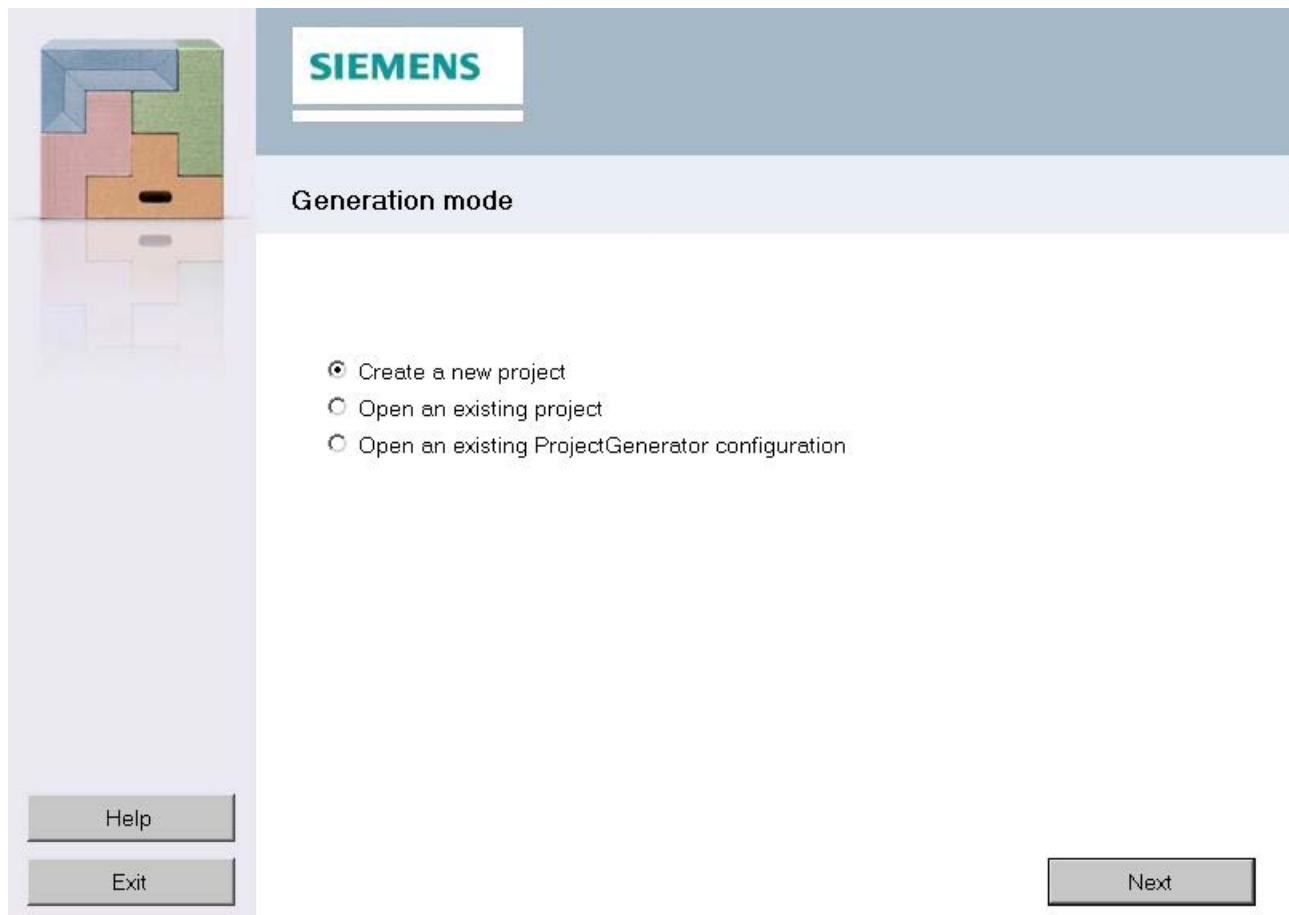
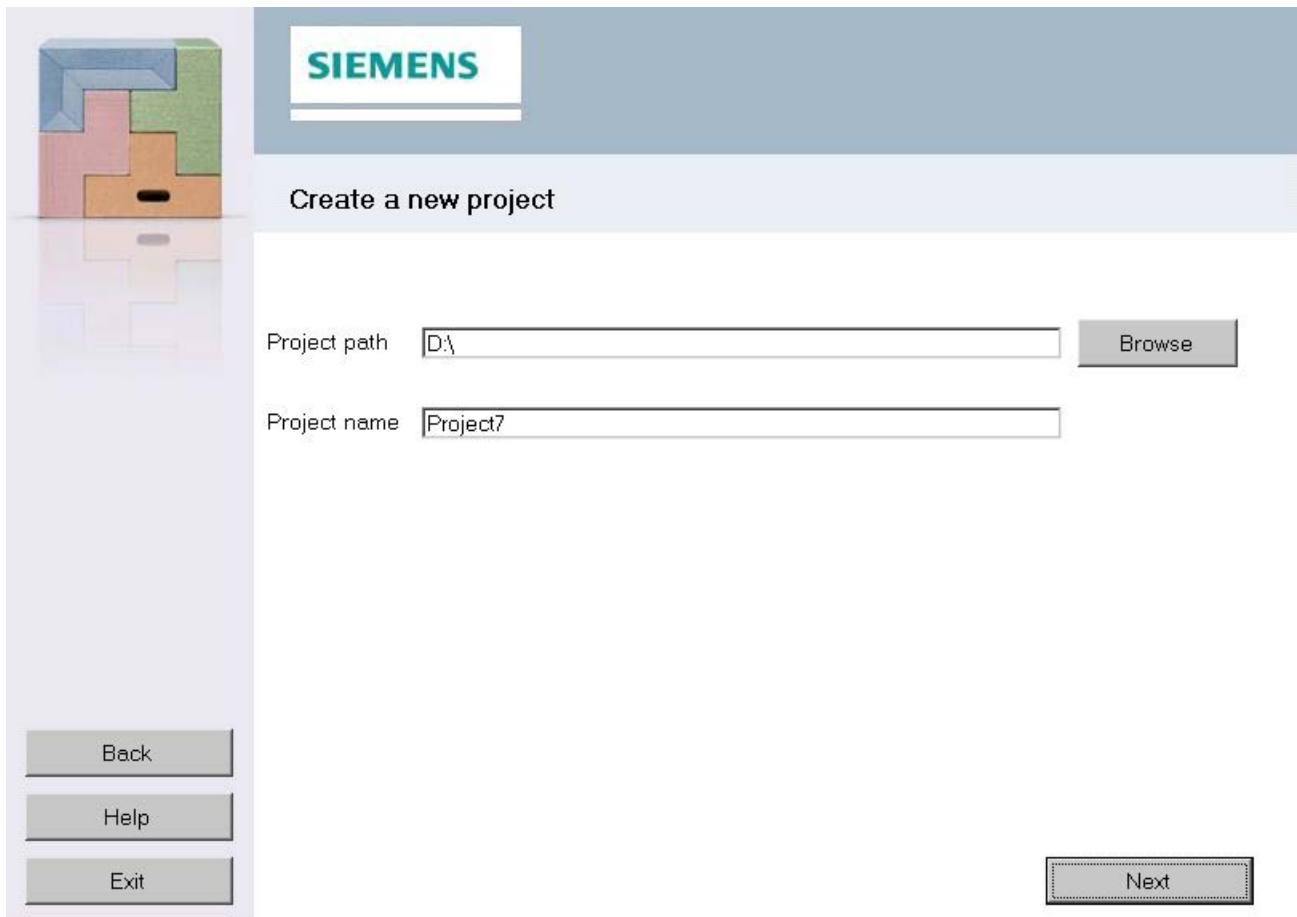


Figure 1-7 Creating or selecting a project

2. If you have selected **Create a new project**, enter a project name and the storage location of the project (the path can also be selected via **Browse**) and click **Next**.



The screenshot shows the 'Create a new project' window of the Siemens ProjectGenerator. The window has a blue header with the 'SIEMENS' logo. Below the header, the title 'Create a new project' is displayed. The main area contains two input fields: 'Project path' with the text 'D:\' and a 'Browse' button to its right; and 'Project name' with the text 'Project7'. On the left side of the window, there is a vertical sidebar with a colorful geometric logo and three buttons: 'Back', 'Help', and 'Exit'. At the bottom right of the window, there is a 'Next' button with a dotted border.

Figure 1-8 Project name and storage path for the project

The *Device selection* window is opened.

3. In the *Device selection* window, under **Select device category**, select the device or devices to be integrated in the project:

**SIEMENS**

### Device selection

Add the required devices to the project. A device from the list has to be selected in order to continue with the next step.

Select device category

☐ SIMOTION ☒ SIMATIC

Device name  
S7317

Device family  
S7300

Device type  
SIMATIC\_CPU315

Add device Delete device

Devices in project

Device name	Device type	Version
S7315	SIMATIC_CPU315	S7300
S7317	SIMATIC_CPU315	S7300

Back Help Exit

Next

Figure 1-9 Creating or selecting a device

Depending on the software installation, you can select either only SIMATIC devices or mixed SIMOTION and SIMATIC devices, one after the other.

Input the device name, device family and device type and then click **Add device**. The new device is added to the **Devices in project** list.

If you want to create a further device, repeat the procedure.

4. In the **Devices in project** list, select the device you want to configure and click **Next**.  
The *Equipment module selection* window opens.

5. In the *Equipment module selection* window, select the standard module or modules that you want to integrate in the selected device and click **Next**.

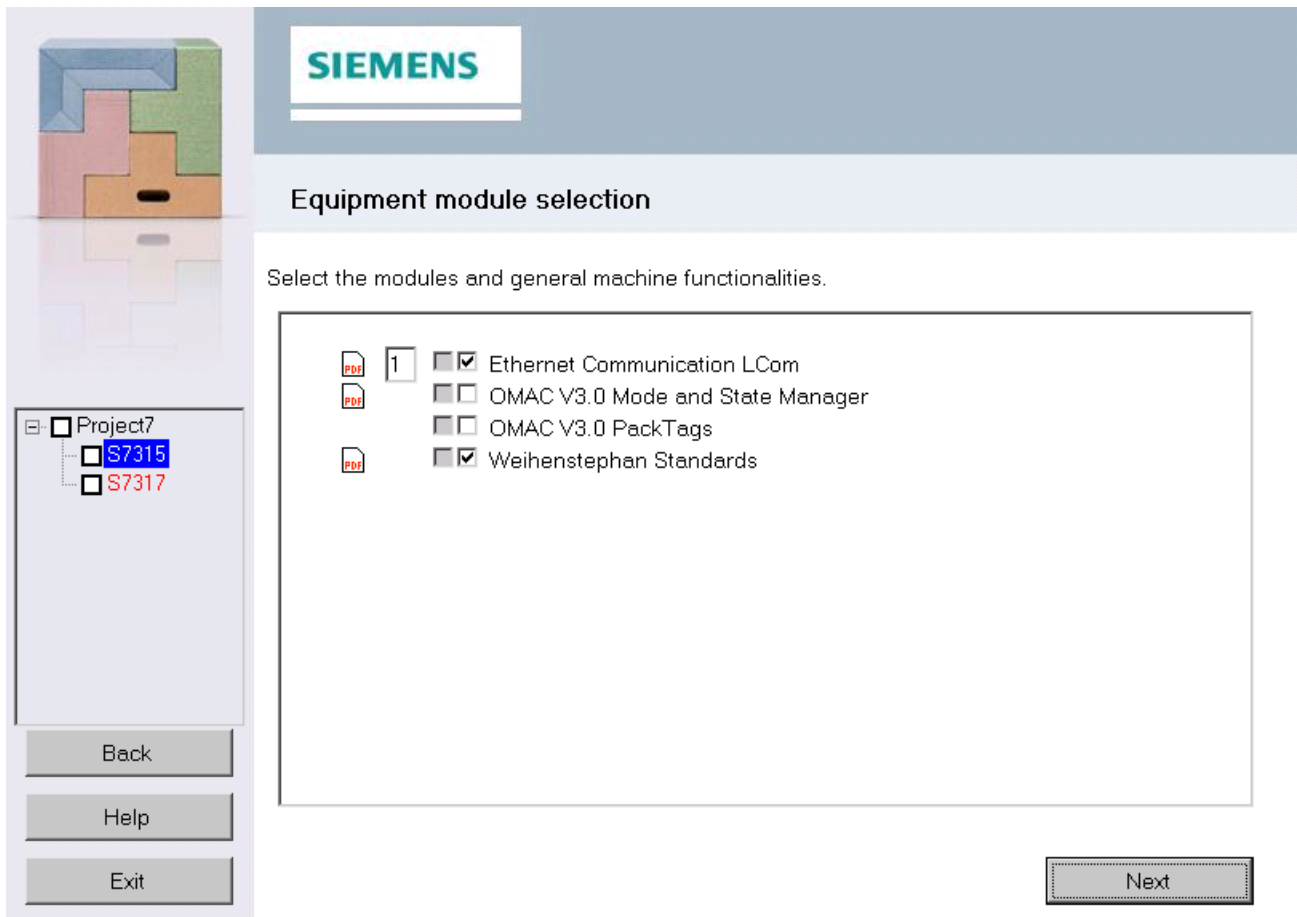


Figure 1-10 Example: Selecting SIMATIC standard modules

Click the **PDF** button to open the documentation for the respective standard module.

For some standard modules, you can enter the number of modules on the left.

For this example, we have selected the *Ethernet Communication LCom* and *Weihestephan Standards* modules, which are configured one after the other in the following screen forms.

6. In the *Ethernet Communication LCom - Function Block Call* window, configure the function block call with the required data blocks and communication parameters and click on **Next**.

Module 1 of 2

### Ethernet Communication LCom - Function Block Call

Fill in the required data and this tool will add a function block call in OB1 to the project.

**Add the FB call to the project**

New instance DB name	DBFBComMachineCom	DB number	105
Parameter DB name	DBLComParameter	DB number	500
Send DB name	DBLComSend	DB number	501
Receive DB name	DBLComReceive	DB number	502

**Communication parameters**

☒ SIMATIC is TCP client (active partner)

IP address (partner): 169 | 254 | 11 | 23

Connection ID: 105

Local port: 3456

Remote port: 3456

Sender cycle time (ms): 500

**Data blocks in device**

Symbolic name	DB number	Data type

Next

Figure 1-11 Configuration of the standard module *Ethernet Communication LCom*

Here, make the following entries:

- **Add FB call to the project**

Under **Add FB call to the project**, you save the names and numbers for the data blocks (DB) required for calling the function block: **New instance DB name**, **Parameter DB name**, **Send DB name**, **Receive DB name**.

- **Communication parameters**

Here, you specify the transfer parameters of the function block call:

If the SIMATIC device is a TCP client, and thus the active connection partner, activate the **SIMATIC is TCP Client** checkbox. In this case, enter the **IP address** and the **Remote port** number.

You can also change the **Local Port** and **Cycle time**.

- **Data blocks in device**

Data blocks already available for the device are shown in the **Data blocks in device** table.

7. Configure the next standard module *Weihenstephan Standards* and click on **Next**.

Module 2 of 2

## Weihenstephan Standards

Fill in the required data and this tool will generate the DBs and add a function block call in OB1 to the project.

☒ Generate Weihenstephan DBs only

List DB number

Search DB number

Data DB number

Path to the PDA-Config file

Data blocks in device

Symbolic name	DB number	Data type
DBFBLComMachineCom	DB 105	FB 105
DBLComParameter	DB 500	DB 500
DBLComReceive	DB 502	DB 502
DBLComSend	DB 501	DB 501

Figure 1-12 Configuration of the Weihenstephan Standards

You can make the following entries here:

- **List DB Number/Search DB Number/Data DB Number**

Enter the desired numbers of the DBs to create a suitable interface for the Weihenstephan standards functionality.

- **Browse**

The **Browse** button can be used to set or change the path of the PDA-Config File.

When you are finished with the configuration and click on **Next**, the *Generate the project* window opens.

8. Configure all the other devices following the above example by clicking on the **Configure devices** button.

Fully configured devices are shown in green with a checkmark on the outer left beneath the project name, while devices that have not yet been configured are red, and devices being worked on are orange.

9. If you do not want to configure another device (**Configure devices**) and want to complete the project, you have the following options.

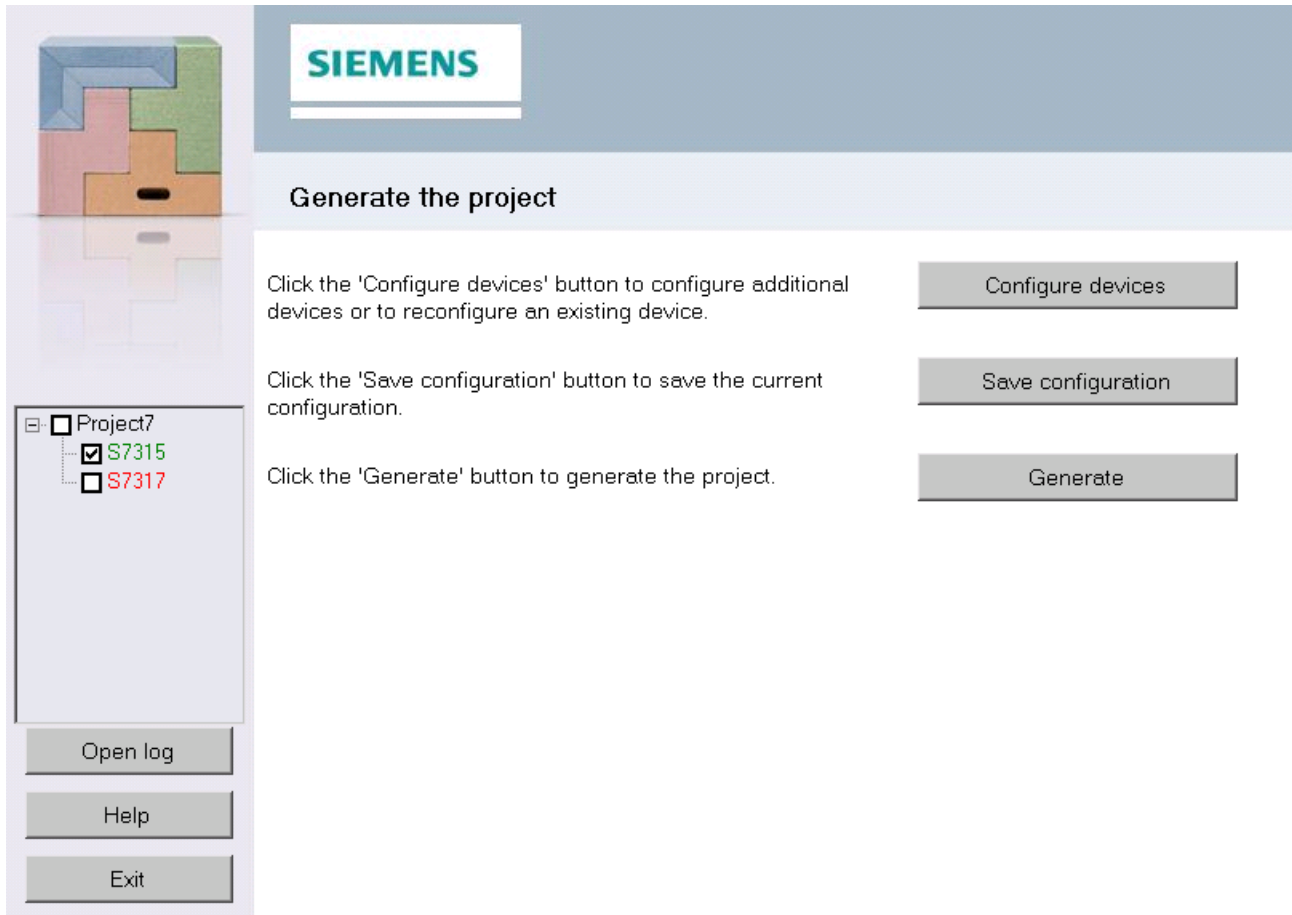


Figure 1-13 Generate the project

- If you would like to save the configuration of the devices, but generate the project at a later time, click on **Save configuration** and enter the storage path in Explorer.
- If you do not wish to configure any other devices and would like to generate the project, click **Generate**.

The project is generated. The duration depends on the type of configuration and is shown using a progress bar.

When the project has been completely generated, the message **Generation finished** appears in the window.

10. Click the **Exit** button to close the ProjectGenerator.

### 1.2.4.3 Adapting/expanding an existing project with SIMOTION devices

#### Sample project

In this example, an existing SIMOTION project is opened with the ProjectGenerator and a new device is created using the standard modules *Message Handling* and *Startup Check*.

#### How to open an existing SIMOTION project and to expand or change it:

1. Start the ProjectGenerator, agree to the liability disclaimer, and select the option **Open an existing project**.

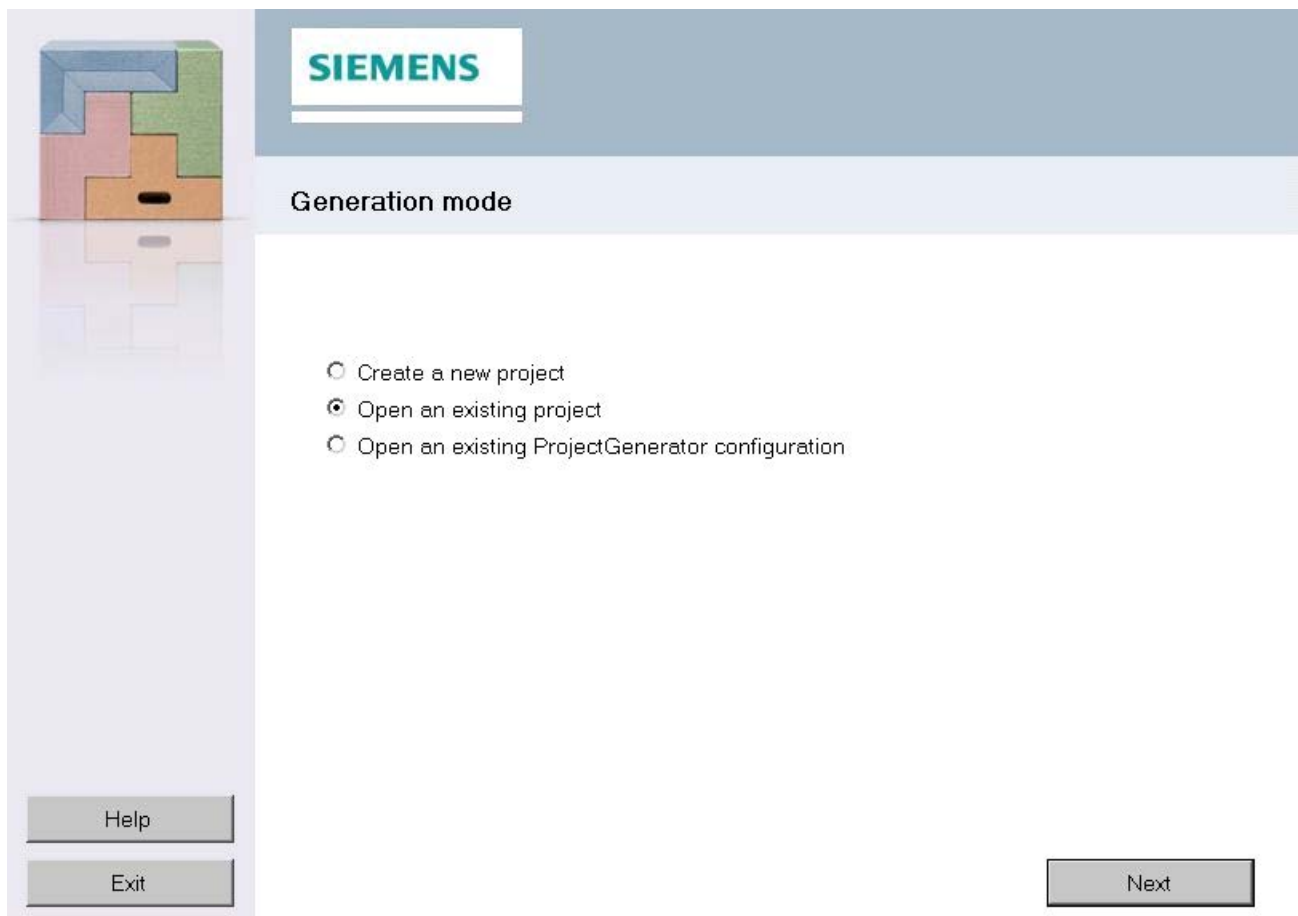


Figure 1-14 Creating or selecting a project

The *Open an existing project* window is opened.

2. Select the project that you would like to open and click on **Next**.

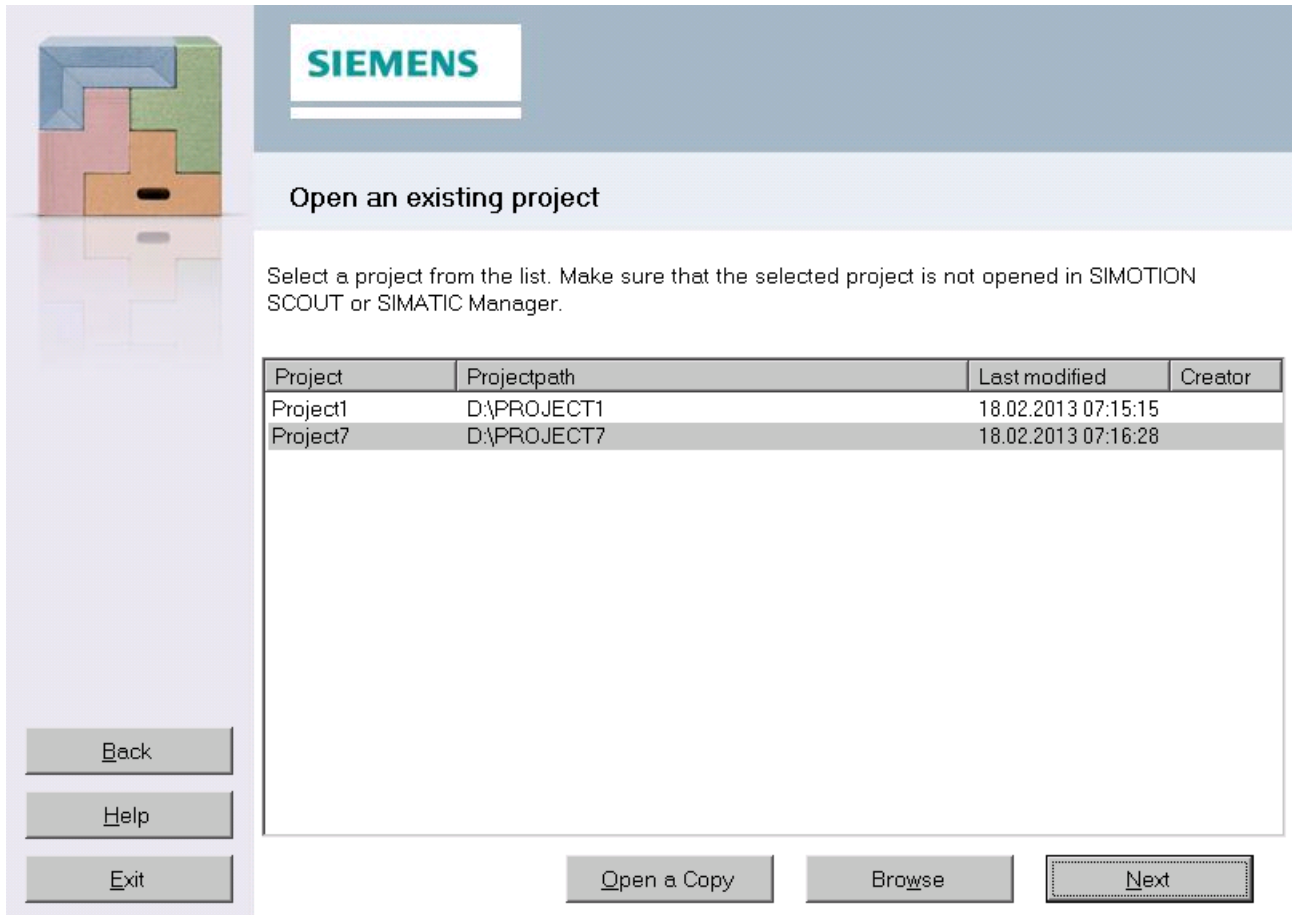


Figure 1-15 Selecting a project

If the desired project is not shown, click on **Browse** and select the path to the desired project with Windows Explorer.

The selected project is accepted into the table view. Select the project in the table and click the **Next** button.

The *Device selection* window is opened with the devices created for the project.

3. In the *Device selection* window, under **Devices in project**, select the SIMOTION device that you wish to edit, or create a new device.

**SIEMENS**

### Device selection

Add the required devices to the project. A device from the list has to be selected in order to continue with the next step.

Select device category  
☒ SIMOTION   ☐ SIMATIC

Device name

Device version

Device type

Device name	Device type	Version
D435	SIMOTION_D435	V4_2
D445_2	SIMOTION_D445-2DP_PN	V4_3

Figure 1-16 Creating or selecting a SIMOTION device

You have the following options:

- You can create additional devices for the project in the left-hand part of the window. To do this, input the version or type and the type name of the device and then click **Add device**. The new device is added to the **Devices in project** list.
  - You can delete a newly created device by selecting the device in the **Devices in project** list and then clicking on **Delete device**. Please note that only newly created devices can be deleted. Devices which already existed when the project was opened cannot be deleted.
  - If you only want to change an existing device, select it in the **Devices in project** window.
4. In the **Devices** list, select the device you want to configure next and click **Next**.

The *Equipment module selection* window opens.

5. In the *Equipment module selection* window, select the standard module or modules that you want to integrate in the selected device and click **Next**.

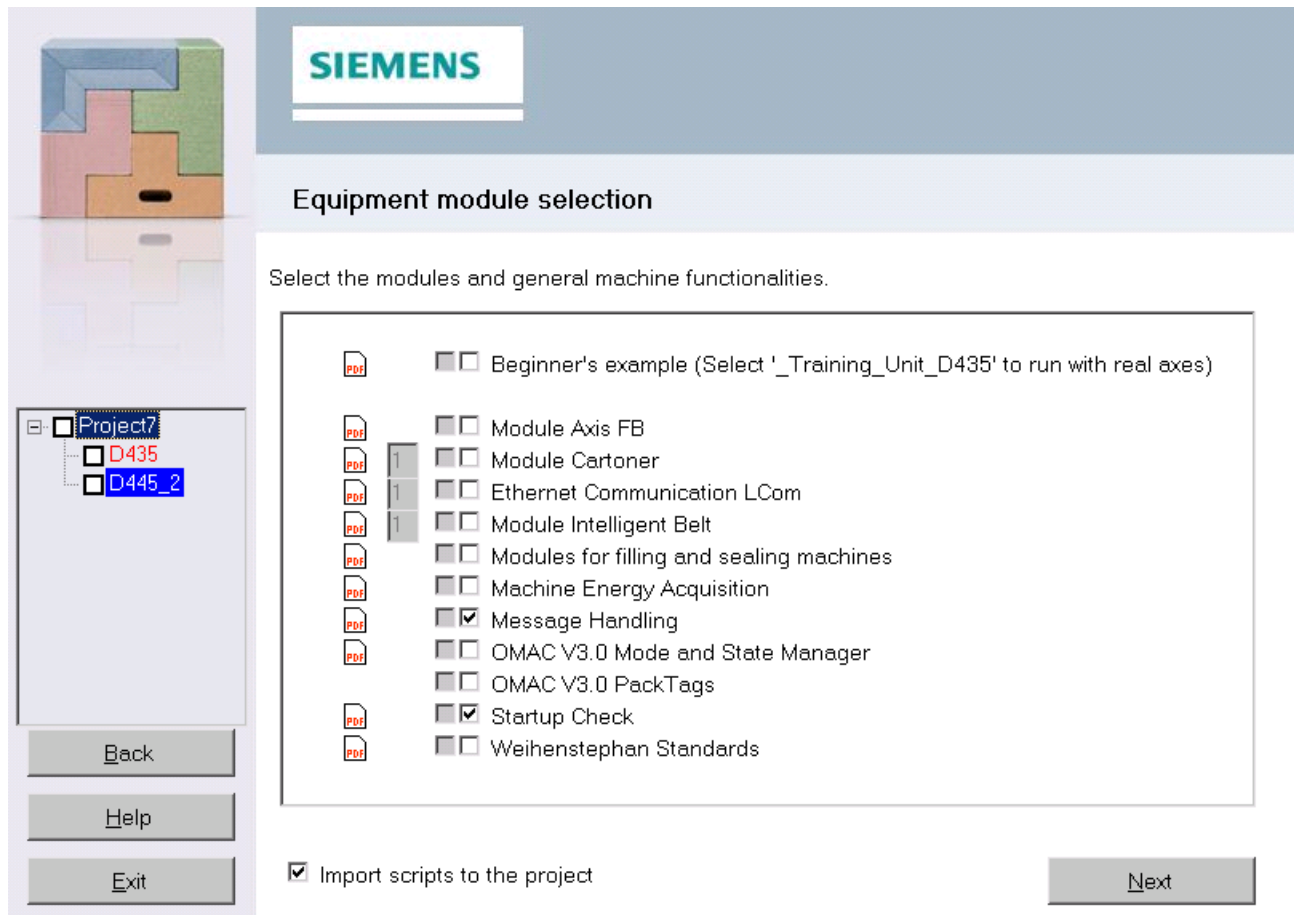


Figure 1-17 Selecting standard modules

Click the **PDF** button to open the documentation for the respective standard module.

For some standard modules, you can enter the number of modules on the left. The left vertical row of selection boxes shows the modules already included in the project.

The following applies only to SIMOTION devices: The **Import scripts to the project** checkbox is activated by default. In this way, all scripts that are in the relative path SIMOTION\Scripts are imported into the project level of the selected project.

For this example, we have selected the *Message Handling* and *Startup Check* modules, which are configured in the following screen forms.

6. In the *Message Handling - Configuration* window, configure the selected standard module **Message Handling** and click **Next**.

SIEMENS

Module 1 of 2

### Message Handling - Configuration

Configure the following settings and click 'Next' to add the Message Handling to the selected device.

Size of active messages list

Size of message archive

☒ DRIVE OBJECT diagnostics

☒ Time synchronization SIMOTION - SINAMICS

☒ Message archive in STRING format

☒ English

☐ Deutsch

☒ Create SIMOTION IT web page

☒ Transfer SIMOTION IT web page to storage medium via FTP

IP address

Project7

- D435
- D445\_2

Help

Exit

Next

Figure 1-18 Configuration of the *Message Handling* standard module

You can make the following entries here:

- **Size of active messages list** with the number of entries
- **Size of message archive** with the number of entries
- **DRIVE OBJECT diagnostics:** Monitoring of the SINAMICS drive objects
- **Time synchronization SIMOTION – SINAMICS:** Synchronization of the time
- **Message archive in STRING format:**  
Select whether you want to use the buffers in the STRING format in the default language **German** or **English**.

If you activate the **Create SIMOTION IT web page** checkbox together with **transfer SIMOTION IT web page to storage medium via FTP**, the SIMOTION IT page is saved to the storage medium of the SIMOTION device. It can then be called up with a standard WEB browser (SIMOTION IT). To do this, you must enter the IP address of the SIMOTION device. The ProjectGenerator then goes online and transfers the files for SIMOTION IT to the storage medium of the SIMOTION device. Use the **Test** button to test whether a connection to the SIMOTION device is possible.

If the **Create SIMOTION IT web page** checkbox is not activated, the data is saved on the PC in the project directory.

7. Configure the selected standard module *Startup Check* and click **Next**.

Module 2 of 2

### Startup Check - Configuration

Configure the following settings and click 'Next' to add the Startup Check to the selected device.

- ☒ Create SIMOTION IT web page
- ☒ Transfer SIMOTION IT web page to storage medium via FTP
  - IP address:

Project7

- ☐ D435
- ☐ D445\_2

Figure 1-19 Configuration of the standard module *Startup Check*

See step 6 for the meaning of **transfer SIMOTION IT web page to storage medium via FTP**.

The *Generate the project* window opens.

8. If you do not want to configure another device using **Configure devices** and want to complete the project, you have the following options:

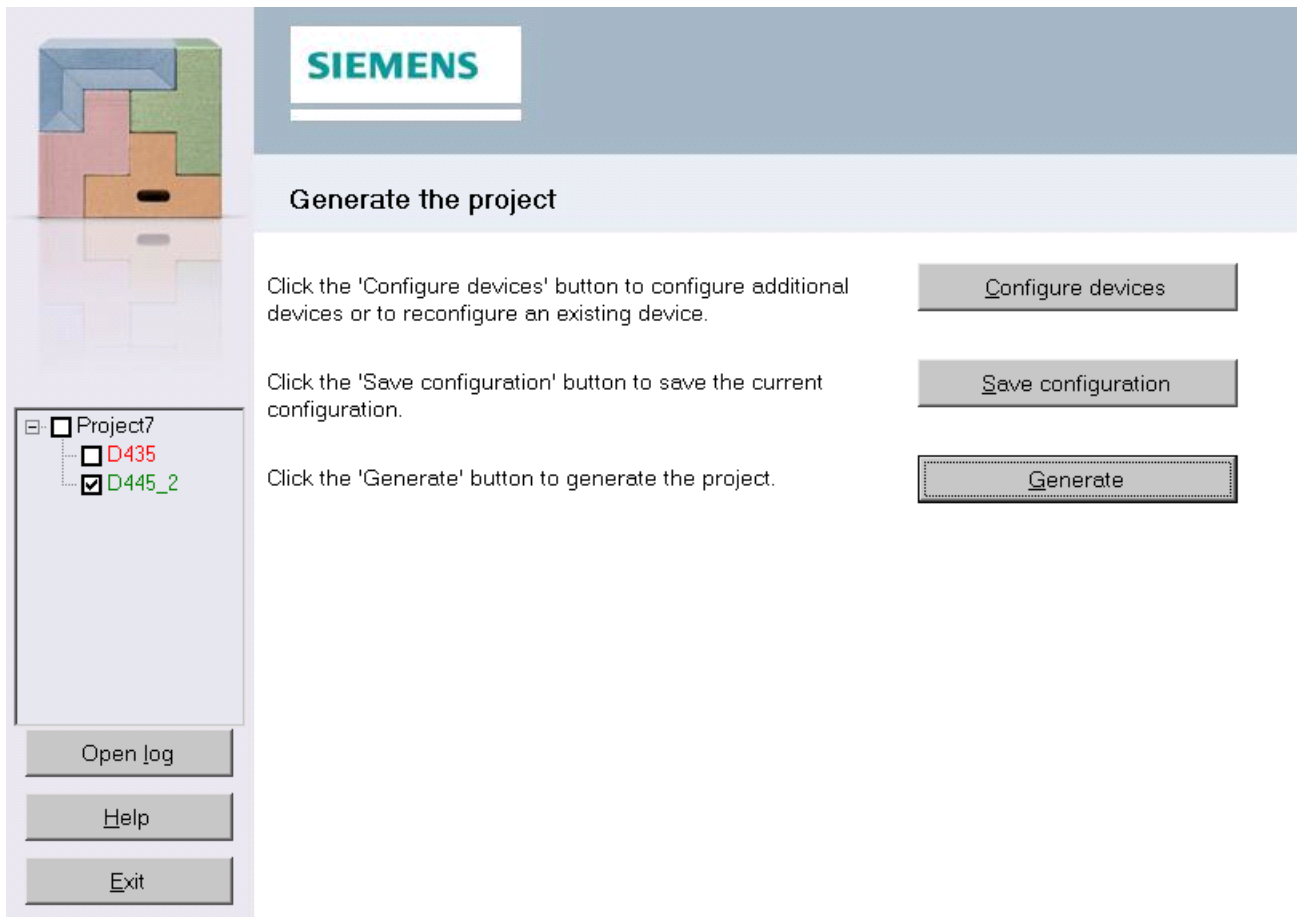


Figure 1-20 Generate the project

- If you would like to save the configuration of the devices, but generate the project at a later time, click on **Save configuration** and enter the storage path in Explorer.
- If you do not wish to configure any other devices and would like to generate the project, click **Generate**.

The project is generated. The duration depends on the type of configuration and is shown using a progress bar.

---

#### Note

Before the project is processed, ProjectGenerator saves the project as an archive in the *Temp* directory of the user, e.g., in: C:\Documents and Settings\User\Local Settings\Temp\PGEN\_SAVED\_EXPORTS\Archive

---

9. When the project has been completely generated, the message **Generation finished** appears in the window and a query whether you want to open SIMOTION SCOUT.

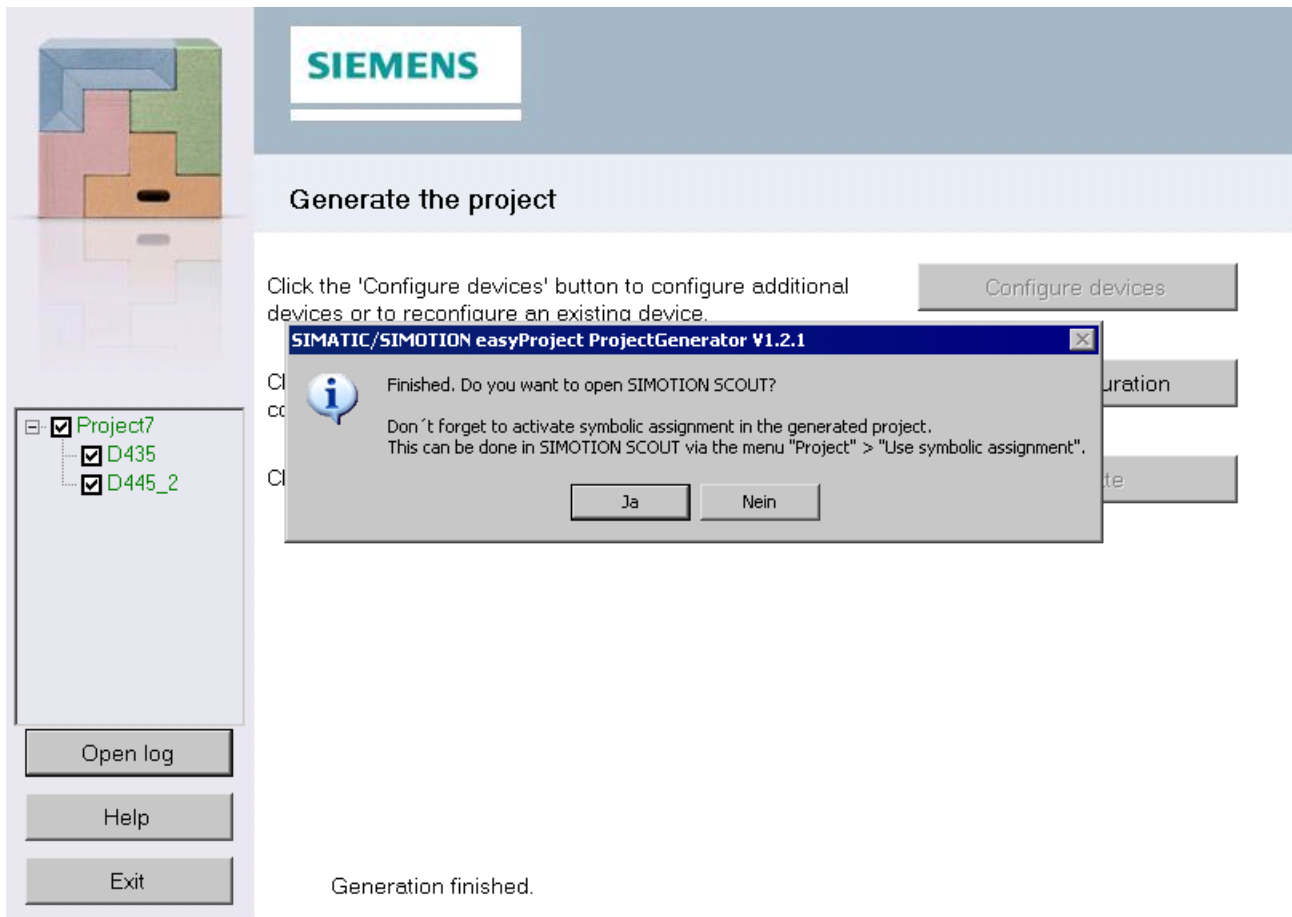


Figure 1-21 Project generation completed

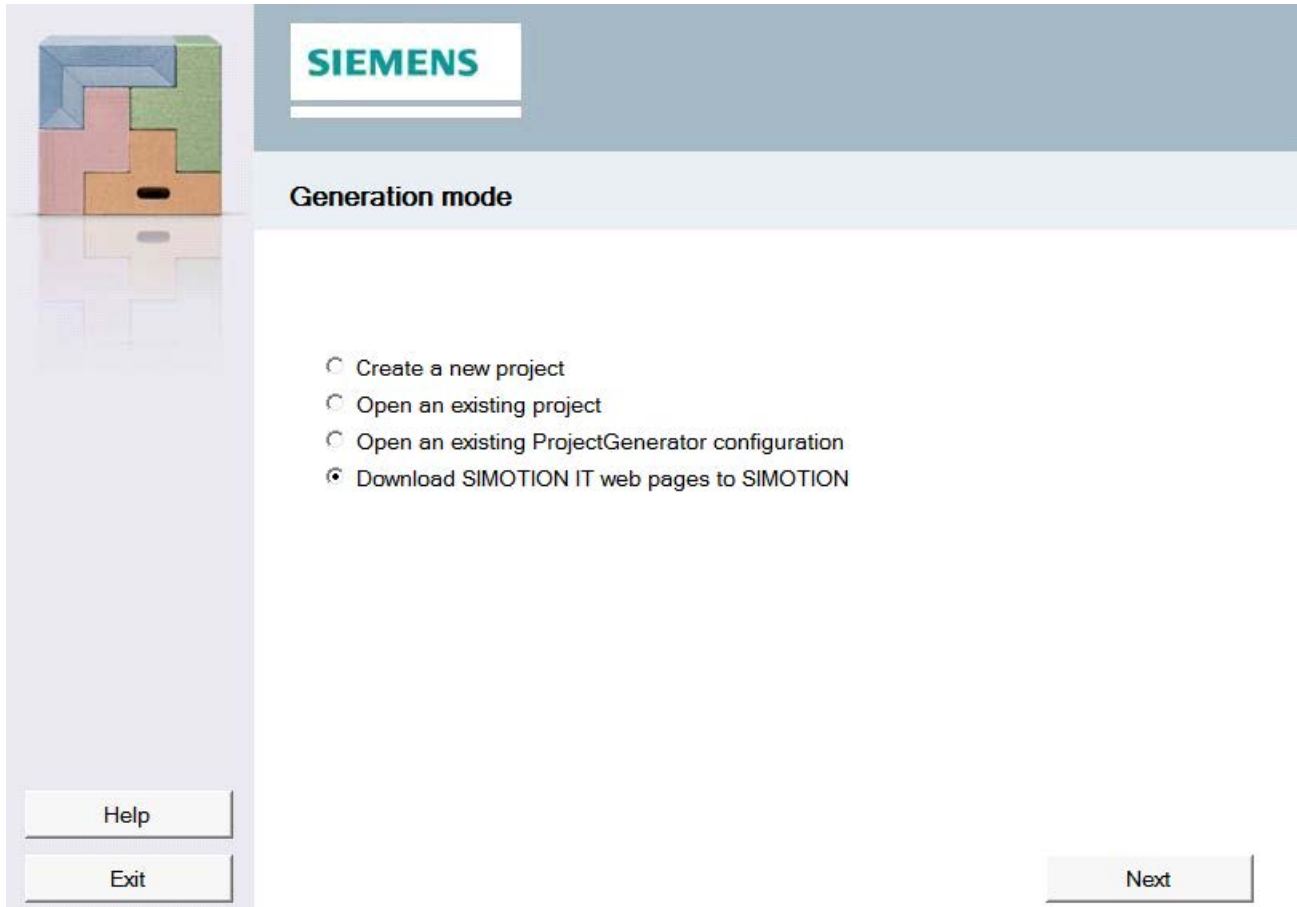
- Click **Yes** if you want to open the project in SIMOTION SCOUT.  
The ProjectGenerator is closed and the project is opened in SIMOTION SCOUT. The configured modules are integrated and ready to use. You can add user-specific functions in the relevant module. Details on this are provided in the documentation for the individual modules.
- If you click on **No**, you can save the data you have input during this ProjectGenerator session by clicking **Save configuration** and leave the ProjectGenerator by clicking **Exit**.

#### 1.2.4.4 Transfer of SIMOTION IT pages

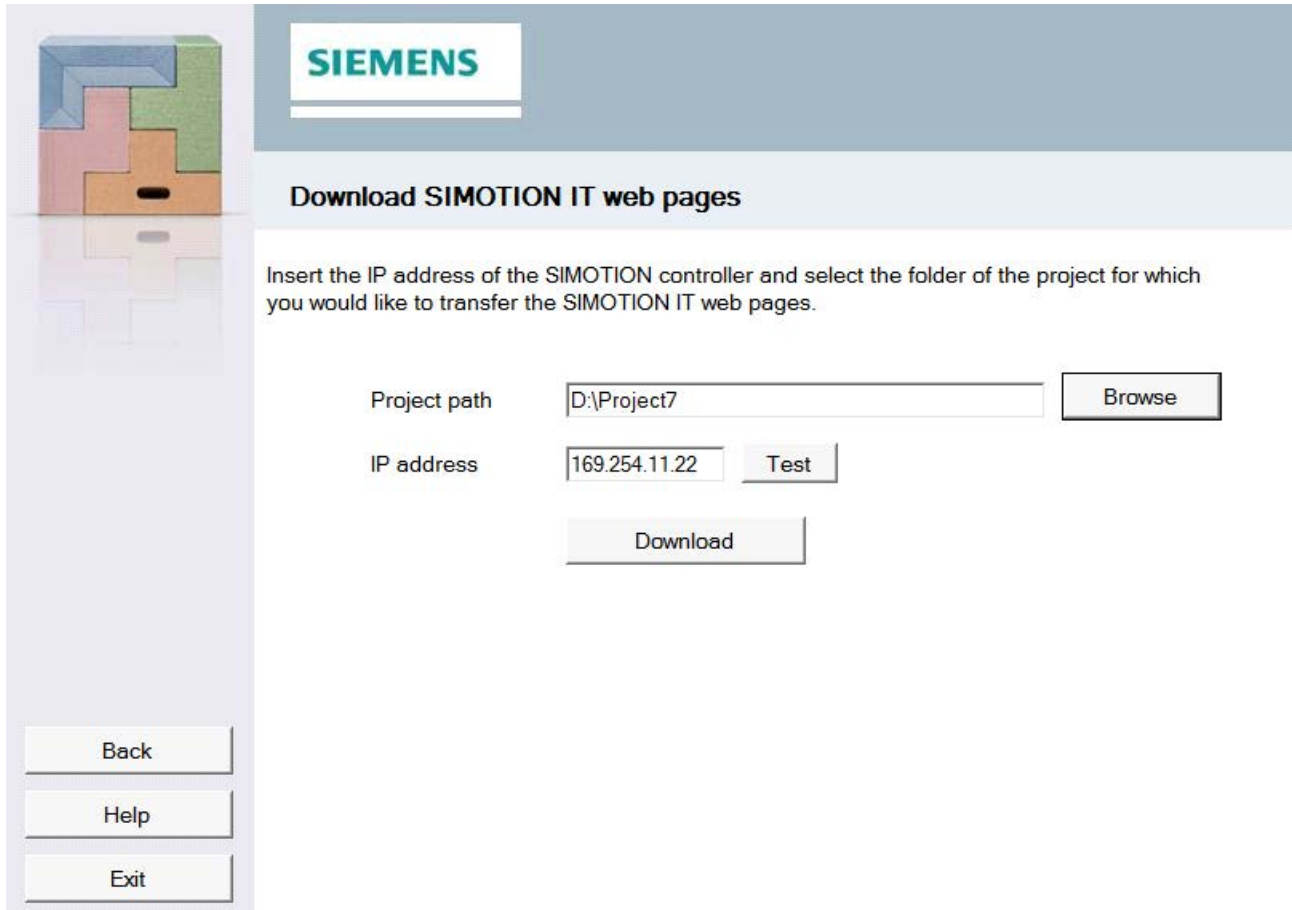
As of Version 1.3.0 of the ProjectGenerator, in addition to the previous possibility of downloading SIMOTION IT pages via FTP transfer to a controller during the generation, it is now possible to download only the SIMOTION IT pages of an existing project to a controller, without having to regenerate the project

This functionality is described in the following.

1. Start the ProjectGenerator, agree to the liability disclaimer, and select the option **Download SIMOTION IT web pages to SIMOTION**.



2. When you have selected **Download SIMOTION IT web pages to SIMOTION**, you can select the path to an existing project for which SIMOTION IT pages have also been generated. Make sure that only the directory that contains the project is selected. The IP address of the SIMOTION controller must also be specified. For this reason, the option to test whether a connection can be established to the controller is also offered.



The screenshot shows a software window titled "Download SIMOTION IT web pages" with the Siemens logo in the top left. The window contains a text box for "Project path" with the value "D:\Project7" and a "Browse" button. Below it, the "IP address" is set to "169.254.11.22" with a "Test" button. A "Download" button is at the bottom. On the left side of the window, there is a vertical sidebar with three buttons: "Back", "Help", and "Exit".

**Note:** The IP address of the network adapter on your computer and the IP address of the SIMOTION controller must be in the same subnet.

3. After specifying the project path and, if required, a successful test of the connection, the transfer can be started. To do this, click the **Download** button. All files under PGEN\_Data\_Files\CardFiles are then downloaded to the controller.

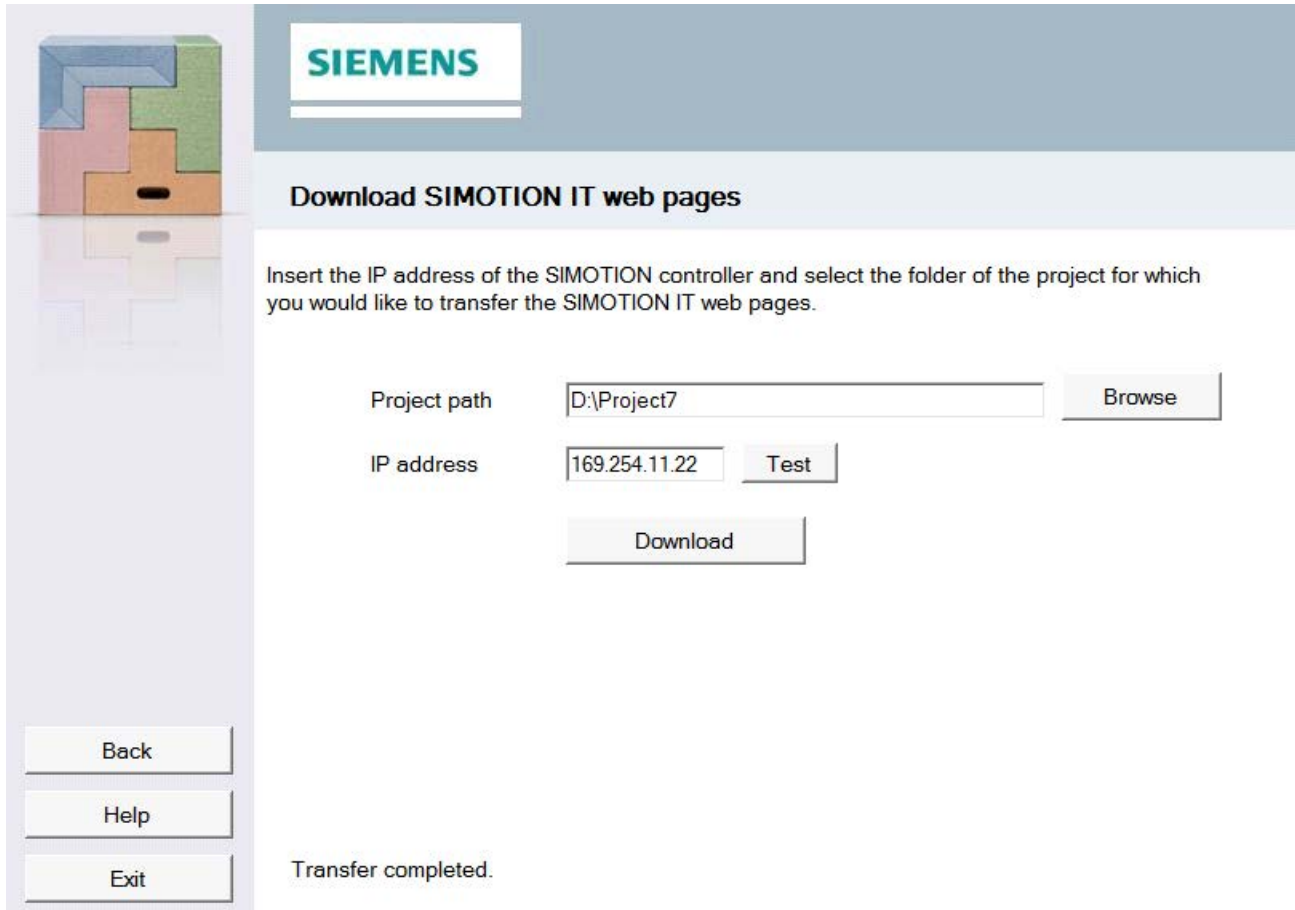
**Note:** As of Version 4.4 of SIMOTION, a user and password must be specified when downloading the SIMOTION IT pages through the ProjectGenerator. This information must be available in the SIMOTION user administration so that the transfer can function.



The screenshot shows the Siemens ProjectGenerator interface. On the left is a vertical sidebar with a colorful geometric logo at the top and three buttons labeled 'Back', 'Help', and 'Exit' at the bottom. The main area has a blue header with the 'SIEMENS' logo. Below the header, the title 'Download SIMOTION IT web pages' is displayed. A text instruction reads: 'Insert the IP address of the SIMOTION controller and select the folder of the project for which you would like to transfer the SIMOTION IT web pages.' The form contains two input fields: 'Project path' with the value 'D:\Project7' and a 'Browse' button to its right; and 'IP address' with the value '169.254.11.22' and a 'Test' button to its right. Below these fields is a large 'Download' button. At the bottom left of the main area, the text 'Transferring ...' is visible.

- During the transfer, **Transferring** in the bottom left of the user interface indicates that the download is still in progress. When the download has been completed, **Transfer completed** is displayed.

When the transfer is complete, you can start another transfer by returning to the generation mode via the **Back** button, or exit the ProjectGenerator.



The screenshot shows the Siemens SIMOTION IT web pages download interface. On the left is a vertical sidebar with a 3D block logo and three buttons: 'Back', 'Help', and 'Exit'. The main area has a blue header with the 'SIEMENS' logo. Below the header, the title 'Download SIMOTION IT web pages' is displayed. A text instruction reads: 'Insert the IP address of the SIMOTION controller and select the folder of the project for which you would like to transfer the SIMOTION IT web pages.' The form contains two input fields: 'Project path' with the value 'D:\Project7' and a 'Browse' button, and 'IP address' with the value '169.254.11.22' and a 'Test' button. A 'Download' button is positioned below the IP address field. At the bottom left of the main area, the status 'Transfer completed.' is shown.

SIEMENS

Download SIMOTION IT web pages

Insert the IP address of the SIMOTION controller and select the folder of the project for which you would like to transfer the SIMOTION IT web pages.

Project path

IP address

Back

Help

Exit

Transfer completed.

## 1.3 User-specific expansions

### 1.3.1 Adding devices and standard modules

#### 1.3.1.1 Expanding the ProjectGenerator

You can expand the ProjectGenerator with devices and modules without having to change the source code of the ProjectGenerator.

For the expansion of the ProjectGenerator with modules, you can use either the standard modules supplied with the ProjectGenerator (see Section Adding standard modules (Page 41)) or also use user-specific modules.

The expansion of the ProjectGenerator is performed in the SIMATIC/SIMOTION subdirectory of the **ProjectGenerator** (see Section Scope of delivery (Page 10)).

#### 1.3.1.2 Adding devices

**Proceed as follows to add a new device:**

You can add your own preconfigured devices to the **SIMATIC/Devices** or **SIMOTION/Devices** directory so that these are also available for selection in future in the ProjectGenerator interface.

### Adding a SIMATIC or SIMOTION device

1. Create a directory with the device name under the relevant version.
  - The following applies for SIMOTION devices: The name of the folder (e.g. SIMOTION\_D435, see figure below) must be identical to the name of the exported XML file of the **device** (e.g. SIMOTION\_D435.xml) from SIMOTION SCOUT.
  - The following applies for all SIMATIC devices: The name of the folder must be identical to the name of the exported CFG file of the **device** from STEP 7.

---

#### Note

We recommend that the device name and folder name begin with SIMATIC\_ or SIMOTION\_.

---

2. Save the exported CFG or XML file of the **device** to the newly created directory.
3. Save the CFG or XML export of the **station** (e.g. XML\_SIMOTION\_D435(station)) to the newly created directory.

At the next start, the ProjectGenerator detects the new device and offers it for import.

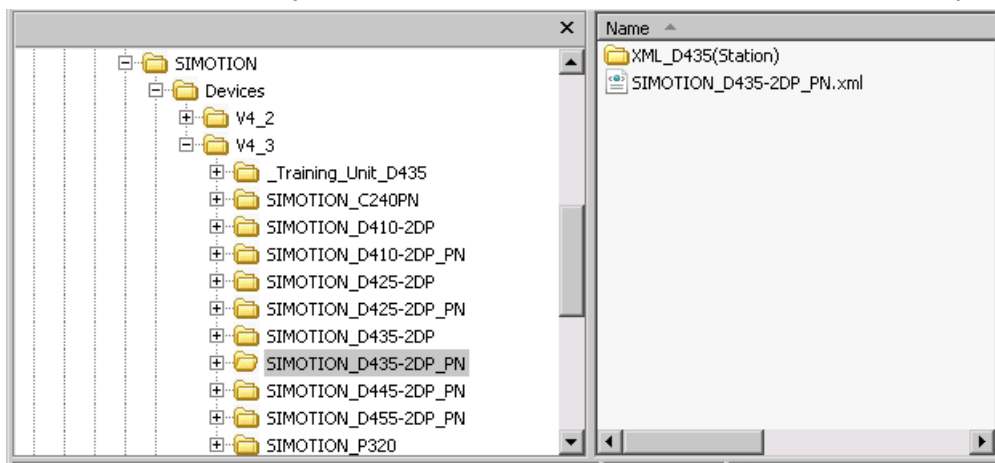


Figure 1-22 Example: Adding a device - SIMOTION D435

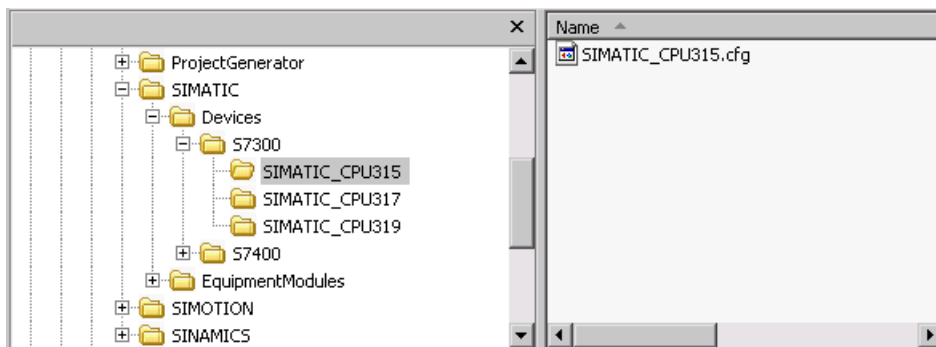


Figure 1-23 Example: Adding a device - SIMATIC devices

### 1.3.1.3 Adding standard modules

Proceed as follows to add a standard module:

1. Create a folder for the new standard module in the **SIMATIC/EquipmentModules** or **SIMOTION/EquipmentModules** directory of the ProjectGenerator.
2. Create a text file (.txt) in this directory, assign it an appropriate name and change the format from .txt to .xml.
3. Open the XML file and define the data structure and description for the new standard module.

Use the XML files of the pre-configured module supplied with the ProjectGenerator as an example.

4. The name of the directory must be identical to the name of the XML file (see figure).

Use the other standard modules with regard to the contents of the directory as an example.

At the next start, the ProjectGenerator detects the new module and offers it for import.

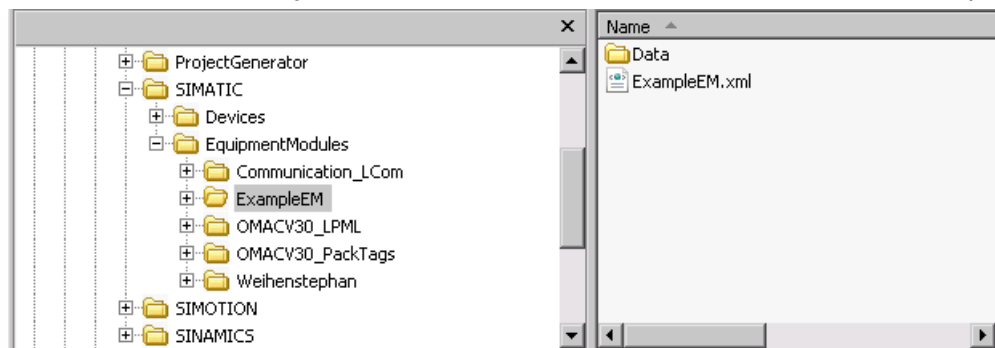


Figure 1-24 Example: Adding a SIMATIC module

### 1.3.2 Adding technology objects (SIMOTION only)

Proceed as follows to add technology objects:

To add a further technology object, store the XML export of the new technology object under the corresponding version directory in the ProjectGenerator directory **SIMOTION/TechnologyObjects**.

At the next start, the ProjectGenerator detects the new technology object and offers it for import.

Each standard module can supply and address its own technology objects. To do this, create technology objects adapted to your requirements below your module and import these instead of the technology objects in the default directory.

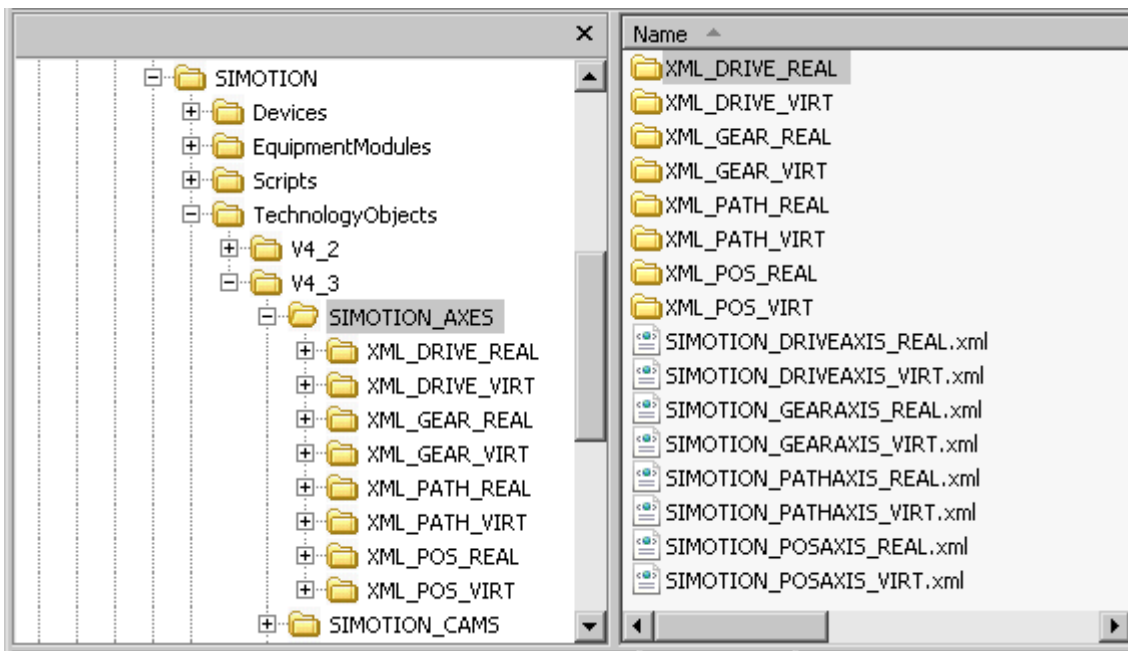


Figure 1-25 Example: Adding SIMOTION technology objects

#### Note

The **output cam**, **measuring input** and **following object** technology objects can only be added in combination with an axis.

### 1.3.3 Creating and adding user-specific modules

You can also insert user-specific modules in a project via the ProjectGenerator without having to deal with the necessary programming.

You can use all the commands with which the existing standard modules are inserted. You can find a concise description of the individual commands in the *SIMATIC\_ProjectGenerator\_List\_Manual* and *SIMOTION\_ProjectGenerator\_List\_Manual* Manuals.

#### Structure of the user-specific modules

---

##### Note

For the creation of user-specific modules, we recommend that you use the supplied standard modules as an example with regard to structure and content. They can be used as a copy template or as an orientation help for your own expansions.

---

The first element in a module is a *CommandList* tag. The name of the library is specified in the *Name* attribute, in order to be able to identify whether there is already a module of this type the next time the ProjectGenerator is run through.

The *DisplayText* attribute describes the text that is displayed when the module is selected.

If a module is not capable of multi-instances, the number can be limited via the *MaxNumberOfModules* attribute.

The following statements apply only to the ProjectGenerator for SIMOTION:

- If a module has no or no unique library, a fixed name of a unit is referenced by setting the attribute *Mode="UnitOnly"*.

#### Example of a CommandList tag

Table 1- 5 Example of a CommandList tag

```
<CommandList Name="pStartupCheck"
              DisplayText="Use Startup Check"
              MaxNumberOfModules="1"
              Mode="UnitOnly"
. . .
>/CommandList>
```

The *CommandList* tag is followed by the *Command* tags as *Child* tags. All *Command* tags have a unique identification number (ID) within the module and the name of the function that is to be called. The ID is always a positive number. The entry level of the module always begins with ID 1, otherwise the ID can be assigned arbitrarily.

If after completion of a command a further command is to be connected directly, this can be performed with the attribute *NCID*, *NextCommandID*. The ID of the next command is entered in the *NCID* attribute. The command name is entered in the *Name* attribute.

## Examples of Command tags

Table 1- 6 Example of Command tag 1

```
<Command ID="1" Name="ImportUnit" NCID="2">
    . . .
</Command>
```

You must use a special logic for some information that is available in the project and in the ProjectGenerator data management, in order to be able to pass on this information to a control, for example.

Example: A text box should display the active device name. The programmer of the XML description files does not know the name. Functions are available that return the information. These functions have the prefix `__Call__` so that the ProjectGenerator detects that a system function is being used. The ProjectGenerator subsequently replaces the system function call with the return information. If, for example, the *Text* tag of a text box is assigned the value `__call_GetSimotionDeviceName`, this function call is replaced by the active device name, e.g. *D435*.

Transfer parameters must be transferred to some of these functions. This is performed using the syntax of *Visual Basic .NET*. The quotation marks are an exception here. These can be omitted in most cases because there is an automatic conversion to the correct format. However, the ' character can also be used instead of quotation marks.

Table 1- 7 Example of Command tag 2

```
<Control Action="add"
    Type="TextBox"
    Name="TB_DeviceName"
    Text="__call_GetSimotionDeviceName"
    ReadOnly="true"
    Location="175, 300"
    Autosize="true"
    ToolTip="Actual device name">
</Control>
```

## 1.4 Architecture of the ProjectGenerator

### 1.4.1 General information

The architecture of the ProjectGenerator is described in this section. This information is especially for users who want to expand the ProjectGenerator with their own user-specific modules.

### 1.4.2 Structure of the ProjectGenerator

The ProjectGenerator consists of the following components:

#### System core consisting of SL object and XML interpreter

- **SL object**  
The **SL object** is in the ProjectGenerator. The SL object contains all the functions for reading and writing the files and the functions that are required for creating and editing a project.
- **XML interpreter**  
The **XML interpreter** accesses a standard module or a user-specific module for the configuration of a module. The XML interpreter generates the user interface of the ProjectGenerator from the commands of the XML configuration files. After completing the configuration, the ProjectGenerator checks whether it still has other modules that have to be configured and whether the *Framework.xml* is processed further.  
Read access to an open project is performed with the XML interpreter.

#### Framework

The ProjectGenerator accesses the **Framework** for the project selection, selection of the device, and the creation of the overview of the existing standard modules.

#### CFG or XML configuration files

When creating a project, the following configuration files are accessed:

- Framework accesses the system-internal configuration files.
- The CFG or XML configuration files of the standard modules and the user-specific modules are accessed for the configuration of the individual modules.

#### Basic data

The basic data that is stored outside the ProjectGenerator in the **SIMATIC** or **SIMOTION** directory is accessed to generate a project (see Section Scope of delivery (Page 10)). The exact path definition for the basic data is performed with the commands of the XML configuration files.

## STEP 7 project

Read or write access to the project is by means of the STEP 7 command interface or the SIMOTION SCOUT scripting interface.

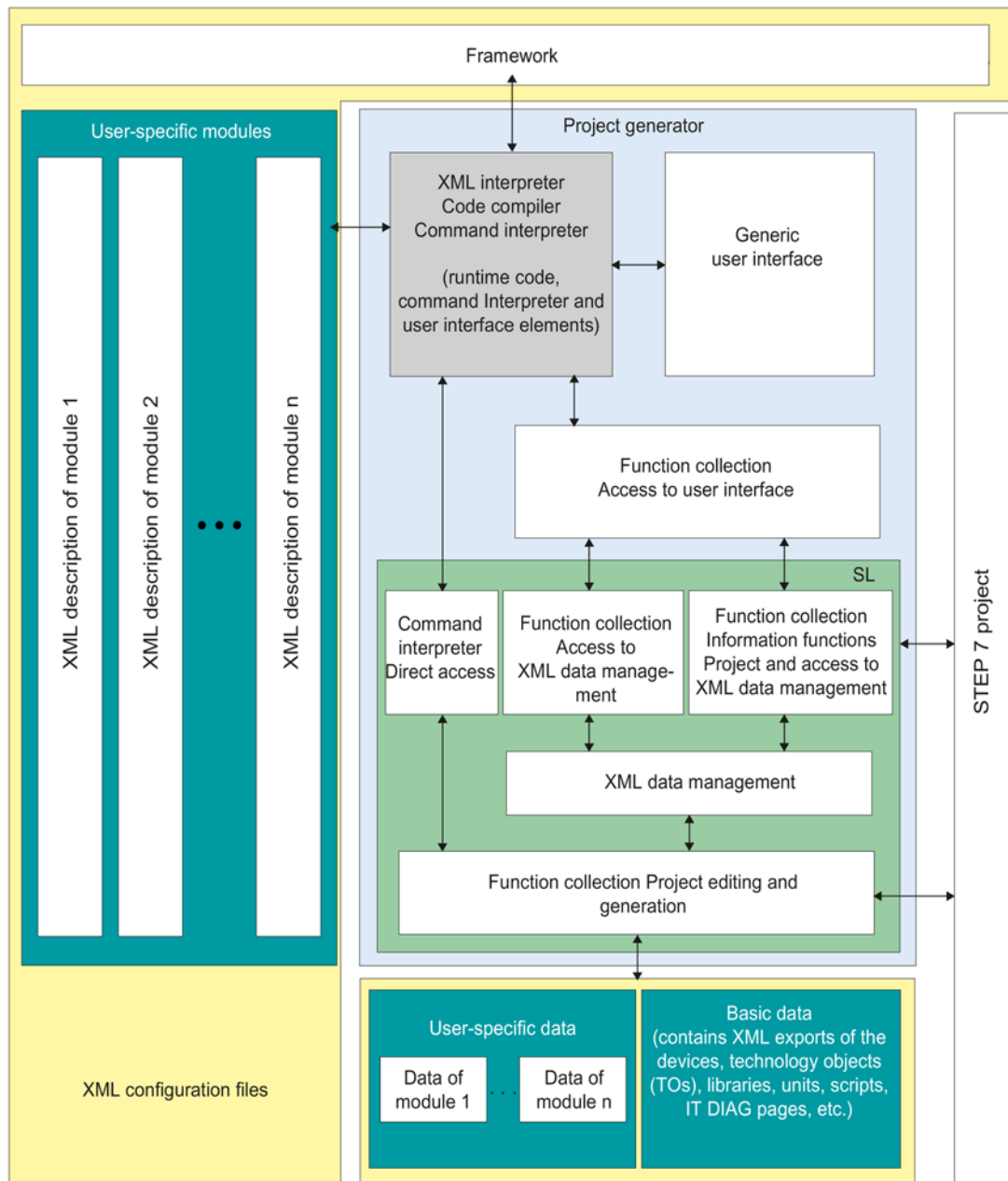


Figure 1-26 System architecture of the ProjectGenerator

### 1.4.3 Using the ProjectGenerator in silent mode

#### ProjectGenerator

The ProjectGenerator can be embedded in user-specific tools via the **silent mode**. This is performed without the user interface of the ProjectGenerator.

In the event of an error, no error messages are output. Errors can be viewed in the log file of the ProjectGenerator.

#### External XML file

The required information and commands are transferred to the ProjectGenerator in **silent mode** via an external XML file with the start call. The commands contained in the file are processed directly and no messages are displayed. If errors have occurred, they can be viewed in the log file.

#### Creation of an external XML file

During the creation and generation of a project with the ProjectGenerator, all entries and configurations are logged and stored in the *Temp* directory of the user (C:\Documents and Settings\User name\Local Settings\Temp) under the file name *PGEN\_DATA\_BASE.xml*.

You can generate an XML file tailored to your requirements by running through the ProjectGenerator with the required settings and configurations and then transferring the generated XML file *PGEN\_DATA\_BASE.xml* to the ProjectGenerator at the start call.

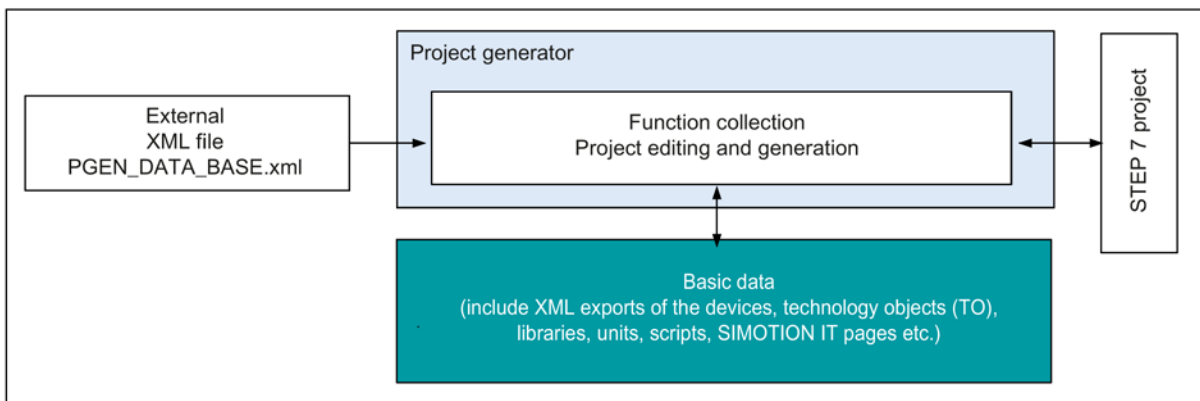


Figure 1-27 System architecture in silent mode

#### Basic data

The basic data in the **SIMATIC** or **SIMOTION** directory is accessed for the generation of the project.

#### STEP 7 project

Read or write access to the project is by means of the STEP 7 command interface or the SIMOTION SCOUT scripting interface.



## System and error messages

### 2.1 General information

The following messages are output on the user interface of the ProjectGenerator or in the system log file.

After running through the ProjectGenerator, the system log file can be found in the Windows *temp* directory under the file name *PGenLogfile.xml*.

### 2.2 System information

The character combinations {0} and {1} are filled with the corresponding name or the type.

#### I\_CMD\_001 to I\_CMD\_052

Table 2- 1 I\_CMD\_001 to I\_CMD\_052

Message	Meaning
I_CMD_001	Value of variable is: {0}
I_CMD_002	Project was not found.
I_CMD_003	Parameter value {0}
I_CMD_004	Project '{0}' ({1}) opened.
I_CMD_005	I/O '{0}' on device '{1}' found.
I_CMD_006	Slave '{0}' removed.
I_CMD_007	I/O '{0}' deactivated.
I_CMD_008	I/O '{0}' removed.
I_CMD_009	Project renamed. New name: '{0}'
I_CMD_010	Library renamed. New name: '{0}'
I_CMD_011	I/O variable '{0}' removed.
I_CMD_012	I/O container '{0}' exported to '{1}'.
I_CMD_013	Library '{0}' removed.
I_CMD_014	Library '{0}' exported to '{1}'.
I_CMD_015	I/O variables imported from '{0}'.
I_CMD_016	Library '{0}' imported from '{1}'.
I_CMD_017	TO '{0}' imported from '{1}'.
I_CMD_018	TO renamed. New name: '{0}'
I_CMD_019	TO '{0}' removed.
I_CMD_020	TO '{0}' exported to '{1}'.
I_CMD_021	DO renamed. New name: '{0}'
I_CMD_022	DO '{0}' removed.

Message	Meaning
I_CMD_023	DO '{0}' exported to '{1}'.
I_CMD_024	Unit renamed. New name: '{0}'
I_CMD_025	Unit '{0}' exported to '{1}'.
I_CMD_026	Unit '{0}' removed.
I_CMD_027	Program '{0}.{1}' added to task '{2}'.
I_CMD_028	Program '{0}.{1}' removed from task '{2}'.
I_CMD_029	Script '{0}' exported to '{1}'.
I_CMD_030	Script '{0}' removed.
I_CMD_031	DP slave '{0}' added.
I_CMD_032	Device '{0}' downloaded.
I_CMD_033	Project '{0}' closed.
I_CMD_034	Project '{0}' copied to '{1}'.
I_CMD_035	Project renamed. New name: '{0}'
I_CMD_036	Slot index '{0}' removed.
I_CMD_037	Separator from slot index '{0}' removed.
I_CMD_038	New slot '{0}' added for slave '{1}'.
I_CMD_039	Device '{0}' downloaded to folder '{1}'
I_CMD_040	I/O variables exported to '{0}'.
I_CMD_041	Slot '{0}' copied to '{1}'.
I_CMD_042	Watch table renamed. New name: '{1}'
I_CMD_043	Unit object '{0}' imported from '{1}'.
I_CMD_044	Old library {0} removed.
I_CMD_045	Slot '{0}' copied.
I_CMD_046	Slot '{0}' moved to '{1}'.
I_CMD_047	STEP 7 project closed.
I_CMD_048	Global variable '{0}' added.
I_CMD_049	Last separator removed from slot '{0}'.
I_CMD_050	STEP 7 HW Config opened.
I_CMD_051	STEP 7 HW Config closed.
I_CMD_052	Device renamed. New name: '{0}'

## I\_SYS\_SL\_001 to I\_SYS\_SL005

Table 2- 2 I\_SYS\_SL\_001 to I\_SYS\_SL005

Message	Meaning
I_SYS_SL_001	Simulation active. Function not started.
I_SYS_SL_002	Parameter not optional at parameter column ({0}).
I_SYS_SL_003	Parameter converted from '{0}' to '{1}'.
I_SYS_SL_004	Call function {0}.
I_SYS_SL_005	Log file stored in {0}.

## PG\_I\_001 to PG\_I\_090

Table 2- 3 PG\_I\_001 to PG\_I\_090

Message	Meaning
PG_I_001	Generation started.
PG_I_002	Opening project.
PG_I_003	Generating project.
PG_I_004	Opening device.
PG_I_005	Importing device.
PG_I_006	Setting device config data.
PG_I_007	Importing TOs.
PG_I_008	Setting master-slave connection.
PG_I_009	Setting config data of TO.
PG_I_010	Setting system data of TO.
PG_I_011	Importing the units.
PG_I_012	Deleting unit defines.
PG_I_013	Setting unit defines.
PG_I_014	Importing watch tables.
PG_I_015	Importing libraries.
PG_I_016	Importing unit to library.
PG_I_017	Deleting library unit defines.
PG_I_018	Setting library unit defines.
PG_I_019	Starting compilation.
PG_I_020	Closing project.
PG_I_021	Downloading to file system.
PG_I_022	Transferring files to file system.
PG_I_023	Transferring files to server.
PG_I_024	Generation finished.
PG_I_025	Finished, but errors occurred at compile time. Do you want to open SIMOTION SCOUT?
PG_I_026	Finished. Do you want to open SIMOTION SCOUT?
PG_I_027	Show the workbench.
PG_I_028	Setting property '{1}' of task '{0}' to value '{2}'.
PG_I_029	Importing project scripts.
PG_I_030	Offline unit '{0}' is newer. Do you want to replace it?
PG_I_031	Unit '{0}' already exists. Do you want to replace it?
PG_I_032	New configuration of unit '{0}' can be different. Please check the correct order of the references.
PG_I_033	Renaming task.
PG_I_034	Setting properties of the tasks.
PG_I_035	Setting name of task '{0}' to value '{1}'.
PG_I_036	Rearranging programs in the execution system.
PG_I_037	Program '{1}' of unit '{0}' in task '{2}' is rearranged to position '{3}'.
PG_I_038	Value '{0}' was outside the limits '{1}'. It will be changed to min/max value.

Message	Meaning
PG_I_039	Text length was greater than '{0}'.
PG_I_040	Do you really want to close this tool?
PG_I_041	Ping was successful.
PG_I_042	Wrong input. The default value will override your input.
PG_I_043	SIMOTION SCOUT is not installed.
PG_I_044	SIMATIC STEP 7 is not installed.
PG_I_045	File not found. This tool will be closed.
PG_I_046	Registry key not found.
PG_I_047	Unit '{0}' already exists. Do you want to replace it?
PG_I_048	Offline library '{0}' is newer. Do you want to replace it?
PG_I_049	Object with the name '{0}' already exists. Choose another name.
PG_I_050	Object cannot be deleted.
PG_I_051	Ping failed.
PG_I_052	Wrong IP address.
PG_I_053	Section between the labels '{0}' and '{1}' not found. (Regex syntax)
PG_I_054	The ProjectGenerator cannot be started from a network drive.
PG_I_055	<p><b>Important information</b></p> <p>The software and the appropriate documentation provided on this medium are made available at no charge. Customers are granted the non-exclusive and non-transferable right to use the software at no charge. This includes the right to modify the software, to copy the software either unchanged, or changed, as well as to link to customer's own software.</p> <p>The software was not subject to the standard system test that Siemens AG normally applies to software. Any liability of Siemens - irrespective of the legal grounds -, in particular due to errors in the software or the appropriate documentation, or damages arising from advice/consultation, is excluded unless mandatory by law, e.g. in cases of willful misconduct, gross negligence, personal injury or death, failure to meet guaranteed characteristics, fraudulent concealment of a defect or in case of breach of fundamental contractual obligations. The above stipulations shall not change the burden of proof to the detriment of the customer.</p> <p>These terms shall be governed by German law without recourse to its conflict of law provisions. The place of jurisdiction shall be Erlangen.</p>
PG_I_056	The ProjectGenerator cannot start modules from a network drive.
PG_I_057	The symbol with name '{0}' already exists with an different address. Do you want to replace the existing symbol?
PG_I_058	The symbol address '{0}' is already in use with a different object. Do you want to replace the existing symbol address?
PG_I_059	Importing HW Config.
PG_I_060	Importing SIMATIC libraries.
PG_I_061	Generating symbol table.
PG_I_062	Importing SIMATIC sources.
PG_I_063	Importing code to Organization Blocks.
PG_I_064	SIMATIC DB '{0}' already exists. Do you want to replace it?
PG_I_065	Importing Weihenstephan Communication SIMATIC sources.
PG_I_066	DB '{0}' already exists. Do you want to replace it?
PG_I_067	Source '{0}' compiled with errors. Do you want to open SIMATIC log file?
PG_I_068	Source '{0}' compiled with errors. Details can be found in SIMATIC log file.

Message	Meaning
PG_I_069	Configured IP-address for next module is '{0}'
PG_I_070	Importing SINAMICS CU320-2.
PG_I_071	File '{0}' is not a valid STEP7 project.
PG_I_072	The SINAMICS station with the name '{0}' already exists on the interface '{1}'. Choose a different name.
PG_I_073	The maximal numbers of DP-Slaves exceeded on this interface.
PG_I_074	The PROFIBUS address is already used. Choose a different one.
PG_I_075	The IP address is already used. Choose a different one.
PG_I_076	Importing I/O tables.
PG_I_077	Connection of TO '{{0}}' to DO '{{1}}' on device '{{2}}' established.
PG_I_078	Generating OPC export files (*.sti).
PG_I_079	Setting SINAMICS parameters & BICOs.
PG_I_080	Importing DOs to the SINAMICS devices.
PG_I_081	Creating HW Config slots for SINAMICS DOs.
PG_I_082	Creating IOs in the address lists.
PG_I_083	Creating global variables.
PG_I_084	Importing AlarmS messages.
PG_I_085	Creating AlarmS messages
PG_I_086	Executing scripts in project.
PG_I_087	Setting preprocessor defines.
PG_I_088	Importing global variable tables.
PG_I_089	Setting unit compile options.
PG_I_090	Deleting unit compile options.

## 2.3 Warnings

The character combination {0} is filled with the corresponding name or type.

### W\_CMD\_001 to W\_CMD\_012

Table 2- 4 W\_CMD\_001 to W\_CMD\_012

Message	Meaning
W_CMD_001	Copying of an opened project not possible. Project '{0}' will be closed. Afterwards reference is automatically generated.
W_CMD_002	No project open.
W_CMD_003	Project must be offline.
W_CMD_004	Project name already exists. Cannot rename.
W_CMD_005	Slot address '{0}' not found.
W_CMD_006	DP subsystem '{0}' not found.
W_CMD_007	New name of imported TO is already used. Renaming is not possible.
W_CMD_008	Reopening of STEP 7 HW Config could cause access problem.
W_CMD_011	Project '{0}' not closed.
W_CMD_012	Task '{0}' not activated.

### W\_SYS\_SL\_001 to W\_SYS\_SL\_009

Table 2- 5 W\_SYS\_SL\_001 to W\_SYS\_SL\_009

Message	Meaning
W_SYS_SL_001	No optional language packet selected or selected message {0} not found.
W_SYS_SL_002	No optional language packet selected.
W_SYS_SL_003	Function will also be executed in simulation mode.
W_SYS_SL_004	Project '{0}' still open. Will be closed automatically.
W_SYS_SL_005	Warning: {0}
W_SYS_SL_007	Directory '{0}' exists.
W_SYS_SL_008	Directory '{0}' exists and will be removed.
W_SYS_SL_009	Log file extension not correct.

## 2.4 Error messages

The character combinations {0}, {1} and {2} are filled with the corresponding name or the type.

### E\_CMD\_002 to E\_CMD\_047

Table 2- 6 E\_CMD\_002 to E\_CMD\_047

Message	Meaning
E_CMD_002	Import from file ({0}) caused an error.
E_CMD_003	Too many project files in folder '{0}'.
E_CMD_004	Project file (.s7p) not found in folder '{0}'.
E_CMD_005	Project not open.
E_CMD_006	No write access to parameter {0}.
E_CMD_007	Project must be online.
E_CMD_008	Enum value '{0}' not defined.
E_CMD_009	Statement '{0}' not defined.
E_CMD_010	Drive object type error: Imported DO type is '{0}'.
E_CMD_011	Optional parameter value '{0}' not defined.
E_CMD_012	Could not go to {0}.
E_CMD_013	Enum value '{0}' not defined.
E_CMD_014	State of device not changeable.
E_CMD_015	Unexpected error in STEP 7 project.
E_CMD_016	Unexpected error with I/O address for I/O '{0}'.
E_CMD_017	No slave on the bus '{0}'.
E_CMD_018	I/O variable '{0}' not correct.
E_CMD_019	DO '{0}' not found.
E_CMD_020	Import of unit not successful.
E_CMD_021	Device '{0}' not found.
E_CMD_022	Object '{0}' not found.
E_CMD_023	Unit '{0}' not found.
E_CMD_024	Project could not be copied. Destination folder already exists.
E_CMD_025	Project could not be closed.
E_CMD_026	Symbol '{0}' not found.
E_CMD_027	Timeout. Expected parameter value not reached in time.
E_CMD_028	Slot address not writeable.
E_CMD_029	CU '{0}' not found.
E_CMD_030	Project could not be opened.
E_CMD_031	Ethernet port '{0}' not found.
E_CMD_032	TO '{0}' not found.
E_CMD_033	Library '{0}' not found.
E_CMD_034	Renaming of TO not possible, name '{0}' already exists.
E_CMD_035	Target slot not empty. Copying not possible.

## 2.4 Error messages

Message	Meaning
E_CMD_036	Technological alarm '{0}' not found.
E_CMD_037	Parameter '{0}' not found.
E_CMD_038	Symbol '{0}' not found.
E_CMD_039	CU '{0}' not found.
E_CMD_040	NO VALUE.
E_CMD_041	Script '{0}' not found.
E_CMD_042	Watch symbol '{0}' not found.
E_CMD_043	STEP 7 HW Config not open.
E_CMD_044	S7 station not found.
E_CMD_045	S7 subsystem not found.
E_CMD_046	Slot with address '{0}' not found.
E_CMD_047	Parameter must be greater than zero.

## E\_CMD\_500 to E\_CMD\_599

Table 2- 7 E\_CMD\_500 to E\_CMD\_599

Message	Meaning
E_CMD_500	Add control to form. Type: '{1}'; name: '{0}'.
E_CMD_501	Start generation.
E_CMD_502	Project already open.
E_CMD_503	New project with name: '{0}' in folder: '{1}' generated.
E_CMD_504	New device with name: '{0}' and version: '{1}' and type: '{2}' is generated.
E_CMD_505	The device is already active.
E_CMD_506	The device '{0}' is already open.
E_CMD_507	Open device '{0}' command is added in XML database.
E_CMD_508	Setting configuration data '{0}' at TO '{2}' to value '{1}'.
E_CMD_509	Connection between master '{0}' and following object '{1}' established.
E_CMD_510	Setting system data '{0}' at TO '{2}' to value '{1}'.
E_CMD_511	Section between '{0}' and '{1}' restored.
E_CMD_512	Constant '{0}' set to value '{1}'.
E_CMD_513	Task information added.
E_CMD_514	Label '{0}' replaced with '{1}'.
E_CMD_515	Information about '{2}' added to section between '{0}' and '{1}'.
E_CMD_516	Adding program '{0}' to execution system.
E_CMD_517	The program '{0}' already exists in the execution system.
E_CMD_518	Preprocessor instruction in unit '{0}' set to '{1}'.
E_CMD_519	Preprocessor instruction in unit '{0}' to library '{2}' set to '{1}'.
E_CMD_520	Importing watch table '{0}' from folder '{1}'.
E_CMD_521	Adding template code '{2}' between the labels '{0}' and '{1}'.
E_CMD_522	Technology package of library '{0}' set to '{1}'.
E_CMD_523	Constants for unit '{0}' saved. Number of VAR_GLOBAL CONSTANT sections found is '{1}'.

Message	Meaning
E_CMD_524	Constants in unit '{0}' restored.
E_CMD_525	Unit '{1}' imported to library '{0}'.
E_CMD_526	Project is being compiled.
E_CMD_527	Ping to IP address '{0}' failed. The transfer will not be executed.
E_CMD_528	Transfer from local folder '{1}' to FTP server '{0}' successfully finished.
E_CMD_529	'{1}' ({0}) read.
E_CMD_530	Value incremented to '{0}'.
E_CMD_531	Generation successfully finished.
E_CMD_532	Entry '{1}' deleted in object '{0}'.
E_CMD_533	Object '{0}' (type '{1}') deleted in XML database.
E_CMD_534	Entry '{1}' created in object '{0}'
E_CMD_535	Writing constant '{0}' in unit '{2}' to value '{1}' in XML database.
E_CMD_536	Writing constant '{0}' in library '{2}' and library unit '{3}' to value '{1}' in XML database.
E_CMD_537	Writing master '{0}' <-> slave '{1}' connection in XML database.
E_CMD_538	Writing new TO '{0}' of type '{1}' in XML database.
E_CMD_539	Setting TO '{0}' config data '{1}' to value '{2}'.
E_CMD_540	Setting TO '{0}' system data '{1}' to value '{2}'.
E_CMD_541	Writing label '{0}' in unit '{2}' to value '{1}' in XML database.
E_CMD_542	Writing constant '{0}' in library '{2}' and library unit '{3}' to value '{1}' in XML database.
E_CMD_543	Writing task information for unit '{0}' in XML database.
E_CMD_544	Writing task information for library '{1}' and unit '{0}' in XML database.
E_CMD_545	Writing device and slave information for unit '{0}' in XML database.
E_CMD_546	Creating backup data of unit '{0}' in XML database.
E_CMD_547	Creating backup data of library '{1}' and unit '{0}' in XML database.
E_CMD_548	Assigning program '{0}' to task '{1}' in XML database.
E_CMD_549	Open browser.
E_CMD_550	Selected path: '{0}'
E_CMD_551	Reading next equipment module.
E_CMD_552	Creating backup of constant in unit '{1}' in library '{0}' in XML database.
E_CMD_553	Setting define '{0}' in unit '{1}' in XML database.
E_CMD_554	Setting define '{0}' in unit '{1}' in library '{2}' in XML database.
E_CMD_555	Setting TP '{1}' in library '{0}' in XML database.
E_CMD_556	Setting FTP transfer from local path '{0}' to IP address '{1}' in XML database.
E_CMD_557	Setting import of unit '{0}' in library '{1}' in XML database.
E_CMD_558	Setting import of watch table '{0}' from path '{1}' in XML database.
E_CMD_559	Configuration data '{1}' for device '{0}' is set to value '{2}'.
E_CMD_560	Configuration data '{1}' with value '{2}' is set for device '{0}' in XML database.
E_CMD_561	Deleting preprocessor define '{1}' in unit '{0}'.
E_CMD_562	Deleting preprocessor define '{1}' in unit '{0}' in library '{2}'.
E_CMD_563	Restoring defines in unit '{1}' in library '{0}'.
E_CMD_564	Deleting define '{0}' in unit '{1}' is set in XML database.
E_CMD_565	Deleting define '{0}' in unit '{1}' in library '{2}' is set in XML database.

## 2.4 Error messages

Message	Meaning
E_CMD_566	Restoring defines in unit '{0}'.
E_CMD_567	Restoring defines in unit '{0}' in library '{1}'.
E_CMD_568	Creating backup of constants in unit '{0}' in XML database.
E_CMD_569	Transfer command for the files in '{0}' to '{1}' is set to XML database.
E_CMD_570	Instruction 'USEPACKAGE' activated in unit '{0}'.
E_CMD_571	File '{0}' not found.
E_CMD_572	No code found to set a constant.
E_CMD_573	No device active.
E_CMD_574	Directory '{0}' not found.
E_CMD_575	No device GUID found.
E_CMD_576	No TP GUID found.
E_CMD_577	GUID of device '{0}' not found.
E_CMD_578	GUID of TP '{0}' not found.
E_CMD_579	Text of the sender was empty.
E_CMD_580	There was no pattern given.
E_CMD_581	Create directory on the server '{0}' not possible.
E_CMD_582	Create file on the server '{0}' not possible.
E_CMD_583	Unit not found.
E_CMD_584	Wrong file extension.
E_CMD_585	Wrong input parameter.
E_CMD_586	CU not found.
E_CMD_587	Object with name '{0}' already exists. Choose another name.
E_CMD_588	Set auto define TP in unit '{0}' in XML database.
E_CMD_589	Set auto-define TP in unit '{0}' in library '{1}' in XML database.
E_CMD_590	'{0}' is not a valid IP address.
E_CMD_591	Path does not exist.
E_CMD_592	Set auto define TP in unit '{0}' in XML database.
E_CMD_593	Set auto-define TP in unit '{0}' in library '{1}' in XML database.
E_CMD_594	Property '{0}' is not available in the task '{1}'.
E_CMD_595	Value '{1}' not allowed for property '{0}'.
E_CMD_596	Property '{0}' not found.
E_CMD_597	Could not set the property '{1}' of task '{0}' to value '{2}'.
E_CMD_598	Property '{0}' not found.
E_CMD_599	Task '{0}' not found.

## E\_CMD\_600 to E\_CMD\_664

Table 2- 8 E\_CMD\_600 to E\_CMD\_664

Message	Meaning
E_CMD_600	The object with the name '{0}' already exists. Choose another name.
E_CMD_601	The object with the name '{0}' already exists. Do you want to replace it?

Message	Meaning
E_CMD_602	The object with the name '{0}' can't be replaced, because it's a real TO.
E_CMD_603	The following object with the index '{1}' of the technology object '{0}' was renamed to '{2}'.
E_CMD_604	The following object with the index '{1}' of the technology object '{0}' was renamed to '{2}'. Index was not found.
E_CMD_605	The following object of the technology object '{0}' cannot be renamed due to wrong index '{1}'.
E_CMD_606	Technology object '{0}' has no following objects to rename.
E_CMD_607	Writing rename following object '{0}' from the object '{1}' in XML database.
E_CMD_608	Data type error in line '{0}'
E_CMD_609	Initial value wrong in line '{0}'
E_CMD_610	The number of the identifiers STRUCT and END_STRUCT is not the same.
E_CMD_611	The length of the cell, row = '{0}' column= '{1}' is wrong.
E_CMD_612	Project '{1}' in path '{0}' already exists. Choose another path or name.
E_CMD_613	Unit with the name '{0}' already exists. Choose another name.
E_CMD_614	IP address of device '{0}' changed to address '{1}' on module '{2}'
E_CMD_615	Writing label '{0}' in SIMATIC source '{2}' to value '{1}' in XML database.
E_CMD_616	Setting symbolic block name '{0}' of block number '{1}' in XML database.
E_CMD_617	Adding networks in path '{1}' to Organization Block '{0}' at position '{2}'.
E_CMD_618	Writing constant '{0}' in source '{2}' to value '{1}' in XML database.
E_CMD_619	Importing IL (AWL) source '{0}' from folder '{1}'.
E_CMD_620	Port '{0}' not found. Port renamed to '{1}'.
E_CMD_621	Setting TO '{0}' to DO '{1}' on device '{2}' in XML database.
E_CMD_622	Removing TO - DO connection on TO '{0}' from the XML database.
E_CMD_623	Creating SINAMICS device '{0}' of type '{1}' with address '{2}'.
E_CMD_624	The device with the name '{0}' already exists. Proceeding with existing device.
E_CMD_625	Importing DO '{0}' and changing the power module to order no '{1}'.
E_CMD_626	Importing DO '{0}' and connecting it to the power unit of DO '{1}'.
E_CMD_627	Importing DO '{0}'.
E_CMD_628	Creating HW Config slots for device '{0}'.
E_CMD_629	Set BiCo connection from SrcName '{0}', SrcNumber '{1}' and SrcIndex '{2}' to target Name '{3}', Number '{4}' and Index '{5}'.
E_CMD_630	Setting parameter '{1}' on DO '{0}' to value '{2}'.
E_CMD_631	Writing new DO '{0}' to device '{1}' in XML database.
E_CMD_632	SINAMICS device '{1}' not found. Can't insert new DO '{0}'.
E_CMD_633	Writing new SINAMICS device '{0}' at CPU Interface '{1}' in XML database.
E_CMD_634	The SINAMICS device '{0}' at CPU Interface '{1}' already exists in XML database.
E_CMD_635	The I/O variable '{0}' already exists in the project.
E_CMD_636	Writing new I/O variable '{0}' with address '{1}' in XML database.
E_CMD_637	The global variable '{0}' already exists in the project.
E_CMD_638	Writing new global variable '{0}' with data type '{1}' in XML database.
E_CMD_639	The AlarmS message '{0}' already exists in the project.
E_CMD_640	Writing new AlarmS message '{0}' in XML database.
E_CMD_641	Writing AlarmS messages in path '{0}' in XML database.

## 2.4 Error messages

Message	Meaning
E_CMD_642	Can't convert MessageClass number '{0}' into an Integer. Proceeding with value 0.
E_CMD_643	Can't convert AlarmID number '{0}' into an Integer. Proceeding with value -1.
E_CMD_644	The AlarmS message '{0}' will be removed.
E_CMD_645	Creating AlarmS message '{0}'.
E_CMD_646	Creating global variable '{0}'.
E_CMD_647	Creating I/O variable '{0}'.
E_CMD_648	The DO '{0}' has no free X2 slot.
E_CMD_649	The X2 slot is already free at the DO '{0}'.
E_CMD_650	Deleting X2 slot at the DO '{0}'.
E_CMD_651	DO '{0}' on device '{1}' not found.
E_CMD_652	Setting parameter '{0}' to value '{1}'.
E_CMD_653	Connecting BiCo '{0}' to '{1}'.
E_CMD_654	Subsystem '{0}' found. Proceeding with this subsystem.
E_CMD_655	'{0}' subsystem '{1}' created.
E_CMD_656	Writing '{0}' device with name '{1}' in XML database.
E_CMD_657	Overwriting '{0}' device with name '{1}' in XML database.
E_CMD_658	The device '{0}' is a CPU and can't be used for a SINAMICS import.
E_CMD_659	Writing subsystem '{0}' property '{1}' with value '{3}' in XML database.
E_CMD_660	Subsystem '{0}' not found.
E_CMD_661	Setting property '{1}' to value '{2}' at subsystem '{0}'.
E_CMD_662	Sinamics device '{1}' at subsystem '{0}' not found.
E_CMD_663	Setting property '{1}' to value '{2}' in device '{3}' at subsystem '{0}'.
E_CMD_664	No CPU in the imported station found. Please check the .cfg file.

## E\_SYS\_SL\_001 to E\_SYS\_SL\_509

Table 2- 9 E\_SYS\_SL\_001 to E\_SYS\_SL\_509

Message	Meaning
E_SYS_SL_001	Exception: {0}
E_SYS_SL_002	Message '{0}' not found in external language file.
E_SYS_SL_006	Parameter '{0}' does not exist for function '{1}'.
E_SYS_SL_007	Parameter column '{0}' not defined.
E_SYS_SL_008	Function attribute: Critical error.
E_SYS_SL_009	Command '{0}' not found in standard library.
E_SYS_SL_010	{0} parameter conversion error: Parameter '{1}' (exception: {2})
E_SYS_SL_011	Parameter conversion error: '{0}' expected.
E_SYS_SL_012	No parameter in parameter column {0}.
E_SYS_SL_013	Internal error: XML node not found.
E_SYS_SL_014	Message '{0}' not found.
E_SYS_SL_015	Resource '{0}' not found.
E_SYS_SL_016	Standard library function could not be started.

Message	Meaning
E_SYS_SL_017	XML file save error. Log file could not be written.
E_SYS_SL_018	Task name '{0}' not defined.
E_SYS_SL_019	Element '{0}' of the enum list '{1}' is not defined.
E_SYS_SL_020	Directory '{0}' not found.
E_SYS_SL_021	STEP 7 project file (*.s7p) not found.
E_SYS_SL_500	Conversion error "Router active".
E_SYS_SL_501	IP address of router is not valid.
E_SYS_SL_502	IP address is not valid.
E_SYS_SL_503	SIMATIC station type not found or not implemented.
E_SYS_SL_504	Symbolic-Name '{0}' of block not found.
E_SYS_SL_505	Program container was not found.
E_SYS_SL_506	Searching for program container '{0}'. No project found.
E_SYS_SL_507	Variable '{0}' already exists in source.
E_SYS_SL_508	Variable '{0}' with type '{1}' generated.
E_SYS_SL_509	Selected Weihenstephan PDACONF.XML is incorrect ('{0}').



## Tips and assistance

### 3.1 Special characters in XML

As some characters have a certain meaning within XML, these have to be re-written with entities. This also applies to the contents of comments, for example.

#### How do I depict special characters in XML?

Use the associated entities instead of special characters in XML.

#### Special characters and rewriting

Special characters	Entity
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	Chr(39)

---

#### Note

The XML file must be coded in UTF-8 format if you wish to use umlauts and other special characters in it.

---

## 3.2 Comments in the source code

In the Visual Basic code, comments cannot be inserted with single quotation marks (') for compatibility reasons.

### How do I insert comments in the source code?

To insert your source code comments, use the REM instructions.

### Example

Table 3- 1 Code example

```
REM Comment
```

## 3.3 Outputting a message box

The 'and' character (&) cannot be used directly as a concatenation operator. Program either as a special character or use the plus sign (+).

### How do I output a message box?

The Visual Basic *Show* function of the message box class can be used to output a message box from the user code.

### Code example

```
MessageBox.Show('Message ' + tmpText1)
```

## 3.4 Usable parameters, events, and objects

Due to the multitude of possible parameters and events, all points cannot be described in this documentation.

### Which parameters, events, and objects are possible?

Further information on the parameters can be found on the Internet in the Microsoft.NET Framework Class Library (<http://msdn.microsoft.com/en-us/library/ff361664.aspx>).

## 3.5 Using text files

### How do I work with text files?

Time and again it may be necessary to read or write files alien to ProjectGenerator; e.g., as an interface to other programs.

### Example

The following example shows how a file is written and then subsequently reread in a button's event code.

Table 3- 2 Code example

```
<Control Action="add"
    Type="Button"
    Name="BT_WriteRead "
    Text="Write & Read "
    Location="650, 531"
    Size="130, 30"
    Enabled="true"
    ToolTip="Write & read a file">
<Events>
    <VisibleChanged code="@BT_Next@.Focus()" />
    <Click code="
        Dim tmpWindowsPath As String
        Dim axNameArray(4) As String
        REM get windows temp path
        tmpWindowsPath = IO.Path.GetTempPath()

        REM create temp path for ProjectGenerator. projgen always use tmpProjectGenerator
        IO.Directory.CreateDirectory(tmpWindowsPath + 'tmpProjectGenerator\')

        REM copy text to array
        For i As Integer = 0 To Microsoft.VisualBasic.UBound(axNameArray)
            axNameArray(i) = 'Line' + i.ToString()
        Next

        REM Save some information for next steps
        IO.File.WriteAllLines(tmpWindowsPath + 'tmpProjectGenerator\' + 'axNameArray.txt',
        axNameArray, System.Text.Encoding.Default)

        REM read and fill automatically
        Dim tmpProjGenPath As String
        Dim readAxNamesArray() As String

        REM get windows temp path
        tmpProjGenPath = IO.Path.GetTempPath() + 'tmpProjectGenerator\'
```

### *3.5 Using text files*

```
    If IO.File.Exists(tmpProjGenPath + 'axNameArray.txt') Then REM file exists?
        REM read file to the array
        readAxNamesArray = IO.File.ReadAllLines(tmpProjGenPath + 'axNameArray.txt', Sys-
tem.Text.Encoding.Default)

        REM show all read lines
        For Each Line As String In readAxNamesArray REM for all lines
            Microsoft.VisualBasic.Interaction.MsgBox(Line, , 'Show read lines')
        Next

    Else REM file does not exist
        Microsoft.VisualBasic.Interaction.MsgBox('File not found. Path = ' + tmpPro-
jGenPath + 'axNameArray.txt',64, 'File not found')
    End If"/>
</Events>
</Control>
```

## 3.6 Using a source several times

### How do I use a source several times in the execution system?

In order to be able to use a source several times in the execution system, all program names and all global variables located in the interface must have a unique name. Here, the name of the source is offered. This must be used in each of the necessary variables and program names.

To have this task taken care of by ProjectGenerator, create labels (e.g., <UnitName>) in the export of the unit.

### Example

Table 3- 3 Code example (extract from the module LCom)

```
//SIEMENS AG
//(c)Copyright 2008 All Rights Reserved
//-----
//file name:      pCom.st
//library:        uses LCom
//version:        SIMOTION / SCOUT 4.1.1.6
//restrictions:
//functionality:  communicate with other controllers with tcp
//-----
//change log table:
//version  date      expert in charge      changes applied
//01.01.00  05.2011  TM                    multi instance corrected for scripting
//=====
INTERFACE
//----- Import -----
  USELIB LCom;
//----- Device Global Type Definitions -----
  TYPE
    //types for example data
    //-----
    s<sgUnitName>UserDataSendType : STRUCT; //type for example send data
    r32Spare1 : REAL;
    r32Spare2 : REAL;
    b32Spare1 : DWORD;
    b32Spare2 : DWORD;
    b8Spare1  : BYTE;
    b8Spare2  : BYTE;
  END_STRUCT;
  ...
END_TYPE
//----- Device Global Variables -----
  VAR_GLOBAL
    gbo<sgUnitName>Enable          : BOOL := TRUE; //connect if cpu running
```

### 3.7 Reloading a user interface

```

    gbo<sgUnitName>Communicate      : BOOL := TRUE; //communicate if cpu running
    gul6<sgUnitName>SendDataLength : UINT := 500; //byte length of send data
    gs<sgUnitName>UserDataSend     : s<sgUnitName>UserDataSendType;
...
    END_VAR
//----- Export -----
    PROGRAM <sgProgramName>;
//-----
END_INTERFACE

IMPLEMENTATION
//-----
    PROGRAM <sgProgramName>
...
END_IMPLEMENTATION

```

## 3.7 Reloading a user interface

### How do I reload the user interface?

When creating user modules it is very useful if the user interface is reloaded without having to exit the module. The following construct can be used for this purpose. This allows changes in the XML code to be visible directly just by pressing a button on the user interface.

### Example

In the example, the "Reload" button is added to the user interface. If the button is pressed, the user interface closes down and then opens up again automatically.

Table 3- 4 Code example

```

<Command ID="1" Name="ChangeForm" >
  <Control Action="add"
    Type="Button"
    Name="BT_Reload"
    Text="Reload"
    Location="450, 531"
    Size="130, 30"
    Enabled="true"
    Visible="true"
    ToolTip="Configuration of the Axis FB">
    <Events>
      <Click code="MyApp.NextCommand(100)" />
    </Events>
  </Command>
<Command ID="100" Name="DestroyForm" NCID="1" />

```

## 3.8 Accessing user interface elements generically

### How can I generically access user interface elements?

With very complex pages, it is necessary to be able to access the user interface elements generically. You can find all the elements of the current user interface in the `"MyApp.Controls('Panel').Controls"` collection.

### Example

The example hides all user interface elements that begin with the name "CB\_".

Table 3- 5 Code example

```
For Each tmpControl As Object In MyApp.Controls('Panel').Controls
    If tmpControl.Name.ToUpper.StartsWith('CB_') Then
        tmpControl.Visible = False
    End If
Next
```

## 3.9 Changing the user interface generically

### How do I change the user interface generically with user XML?

With complex modules it may be necessary to change the user interface generically to get to the target of an executable module more quickly.

### Example

The following example shows how the contents of the user interface can be adapted generically using an XML file (`fcInitSyncDataList.xml`). The starting point is the screen in Figure 3.1. The screen contains twenty invisible groups of controls (`visible = false`). Four of these groups are shown in the figure for the purposes of illustration. A group consists of the following controls:

- A label for describing the parameter (`LBL_FC1Para1_X`). This label is always displayed and contains the name of the parameter.
- A text box (`TB_FC1ParaX`), a combo box (`CmB_FC1ParaX`), and a checkbox (`CB_FC1ParaX`). Which of these three controls is to be displayed is specified in the XML file.
- A label for a possible unit (`LBL_FC1ParaX`). In the figure, these are the labels with the text "Unit". This label is only displayed when the second control is a text box.

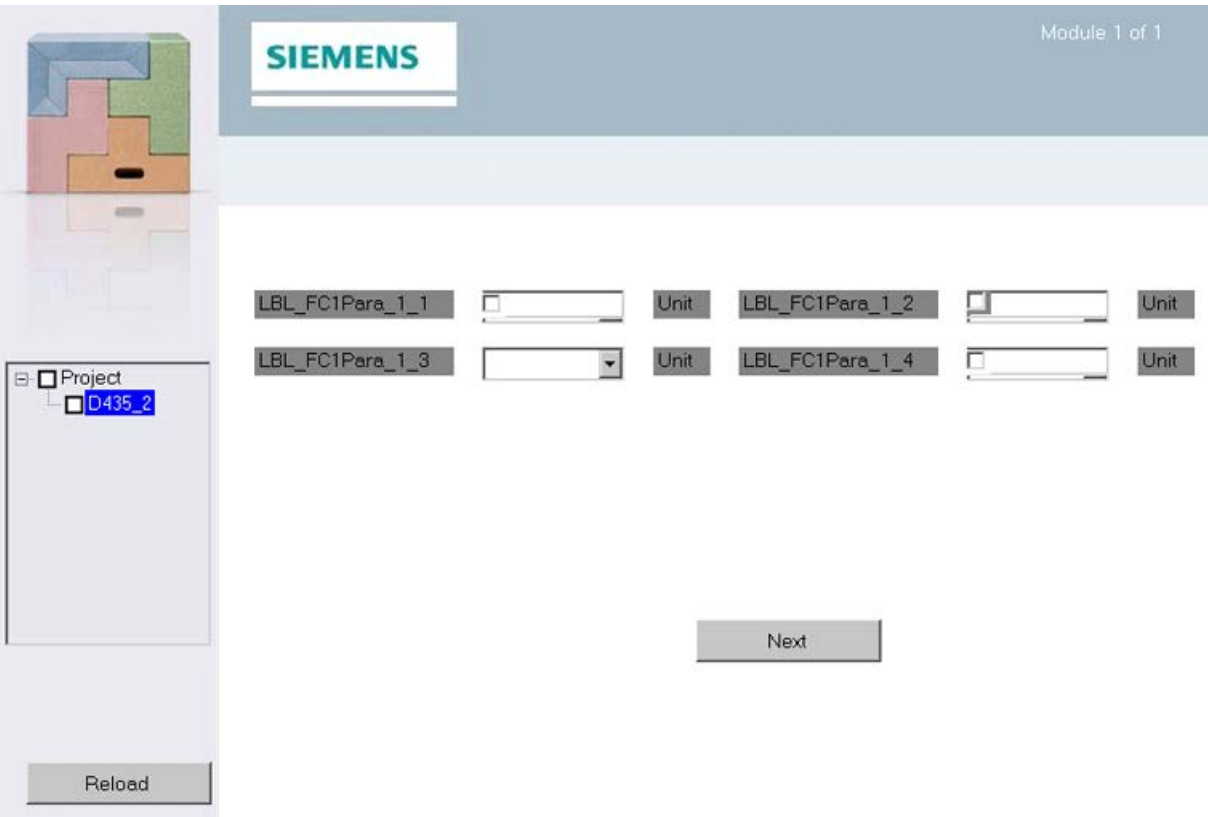


Figure 3-1 Example: User interface with all elements

The following table shows a section of code from the XML file of the equipment module. In this section, the module is defined and the user interface is relabeled with the controls from group 1.

Table 3- 6 Creating a user interface code example

```
<CommandList Name="DocExample"
    DisplayText="DocExample"
    MaxNumberOfModules="1"
    ModuleInfoFile=" "
    ToolTip=" ">
<Command ID="1" Name="DestroyForm" NCID="2"/>
<Command ID="2" Name="ResizeForm" NCID="3" X="860" Y="600"/>
<Command ID="3" Name="ChangeForm" NCID=" ">
    <Control Action="add"
        Type="Label"
        Name="LBL_FC1Para_1_1"
        Text="LBL_FC1Para_1_1"
        Visible ="True"
        Location="180, 200"
        AutoSize="False"
        Size="140,20"
        BackColor="__call_SetColor(Gray)">
    </Control>
    <Control Action="add"
        Type="CheckBox"
        Name="CB_FC1Para1"
        Text=" "
        Location="340,202"
        Size="17, 17"
        AutoSize="False"
        Visible ="True"
    </Control>
    <Control Action="add"
        Type="TextBox"
        Name="TB_FC1Para1"
        Text=" "
        Location="340, 200"
        Size="100, 20"
        Autosize="false"
        Visible ="True">
```

### 3.9 Changing the user interface generically

In the XML file (fclnitSyncData-List.xml), each control group is configured by an XML node (item). Because there are only twenty control groups, the number of items in the XML file must not exceed this number. Certain parameters must be specified depending on the control selected. The following section of code shows the structure of the XML file expected from the source code. To skip certain control groups and thus have influence over the display, a dummy control "none" is defined.

Table 3- 7 Structure of fclnitSyncDataList.xml code example

```
<GenericControls>
  <!-- TextBox selection -->
  <Item Control="TextBox"
    ParameterName ="Label text (LBL_FC1Paral_X)"
    Parameter="Label tooltip (LBL_FC1Paral_X)"
    InfoTooltip="TextBox tooltip"
    DefaultValue="TextBox text"
    RotaryUnit="Label unit text"/>

  <!-- CheckBox selection -->
  <Item Control="CheckBox"
    ParameterName ="Label text (LBL_FC1Paral_X)"
    Parameter="Label tooltip (LBL_FC1Paral_X)"
    InfoTooltip="CheckBox tooltip"
    DefaultValue="Checked setting: True or False"/>

  <!-- ComboBox selection -->
  <Item Control="ComboBox"
    ParameterName ="Label text (LBL_FC1Paral_X)"
    Parameter="Label tooltip (LBL_FC1Paral_X)"
    InfoTooltip="ComboBox tooltip"
    DefaultValue="Text of item to be selected initially"
    MaxItems="Number of items, provide at least this number of items"
    Item1="Item1 text"
    Item2="Item2 text"
    <!-- add more ComboBox items if needed --> />

  <!--skip control-->
  <Item Control="None"/>

</GenericControls>
```

The following table gives a concrete example with three controls. The second control group is not displayed.

Table 3- 8 fclnitSyncDataList.xml code example

```
<GenericControls>

  <!-- CheckBox to configure the new axis as a virtual axis -->
  <Item Control="CheckBox"
    ParameterName ="Add virtual axis "
    Parameter=""
    InfoTooltip="Check to add a virtual axis"
    DefaultValue="False"/>

  <!-- skip second group -->
  <Item Control="None"/>

  <!-- TextBox to enter the name of the new axis -->
  <Item Control="TextBox"

    ParameterName ="Axis name "
    Parameter=""
    InfoTooltip="Name of the new axis"
    DefaultValue="NewAxis"
    RotaryUnit="" />

  <!-- Combobox to select axis type -->
  <Item Control="ComboBox"
    ParameterName ="Type of axis "
    Parameter=""
    InfoTooltip="Select the type of the new axis"
    DefaultValue="PosAxis"
    MaxItems="4"
    Item1="DriveAxis"
    Item2="PosAxis"
    Item3="GearAxis"
    Item4="PathAxis" />

</GenericControls>
```

### 3.9 Changing the user interface generically

In the source code in the following table, the configuration data for the user interface is imported from the XML file, the selected controls are displayed, and are initialized with the imported data.

Table 3- 9 fclInitSyncDataList.xml code example

```

REM =====
REM Read fclInitSyncDataList and insert Labels, Textboxes, CheckBoxes and ComboBoxes
REM =====

REM Read the xml file and save the items in the Arraylist fclInitSyncDataList
Dim fclInitSyncDataList As System.Collections.ArrayList =
My-
App.myIsl.readxmlfile('SIMOTION\EquipmentModules\V4_3\DocExample\Data\Lists\fclInitSyncData
List.xml')

REM Check the number of the xml items. Exit the module (Command0) if any er-
ror
If fclInitSyncDataList.count = 0 Then          REM No items in XML file or file is miss-
ing
    Microsoft.VisualBasic.Interaction.MsgBox('List with parameters
    SIMOTION\EquipmentModules\V4_3\UserModuleName\Data\Lists\fclInitSyncDataList.XML
    could not be found or is empty.' + Microsoft.VisualBasic.vbCrLf + 'Please check
    fclInitSyncDataList.xml.',64, 'fclInitSyncDataList is missing or corrupt')
    MyApp.NextCommand(0)
Else If fclInitSyncDataList.count > 20 Then      REM More than 20 items in XML file
    Microsoft.VisualBasic.Interaction.MsgBox('Parameter list contains more parameters
    than allowed.' + Microsoft.VisualBasic.vbCrLf + 'Please check
    fclInitSyncDataList.xml',64,'fclInitSyncDataList: too many parameters')
    MyApp.NextCommand(0)
Else
    REM File and count ok

REM Loop through the items in the XML file
For i As Integer = 0 To fclInitSyncDataList.count - 1
    REM If the item is None continue with the next item
    If fclInitSyncDataList(i)('CONTROL') = 'None' Then
        Continue For
    End If

    REM Show the label LBL_FC1Para_1_X and set the text as in the
    REM attribute ParameterName in the XML file
    With MyApp.Controls('Panel').Controls('LBL_FC1Para_1_' + Cstr(i+1))
        .Visible = True
        .Text = fclInitSyncDataList(i)('PARAMETERNAME')
    End With

    REM Set the tooltip text of the label LBL_FC1Para_1_X as in the
    REM attribute 'PARAMETER' in the XML file
    MyApp.ToolTip.SetToolTip(MyApp.Controls('Panel').Controls('LBL_FC1Para_1_' +
    Cstr(i+1)), fclInitSyncDataList(i)('PARAMETER'))

```

```

REM =====Textbox selected =====
If fcInitSyncDataList(i)('CONTROL') = 'TextBox' Then

    REM Show the textbox 'TB_FC1ParaX' and get the text from the variable
    REM with same name. If the variable does not exist use the attribute
    REM DefaultValue in the XML File
    With MyApp.Controls('Panel').Controls('TB_FC1Para' + Cstr(i+1))
    .Visible = True
    .Text = MyApp.myISL.getValueOfTemporaryVariable('TB_FC1Para' +
        Cstr(i+1), 'Local', fcInitSyncDataList(i)('DEFAULTVALUE'))
    End With

    REM Set the tooltip of the textbox as in the attribute InfoTooltip
    MyApp.Tooltip.SetToolTip(MyApp.Controls('Panel').Controls('TB_FC1Para' +
        Cstr(i+1)), fcInitSyncDataList(i)('INFOTOOLTIP'))

    REM Set the text of the unitlabel as in the attribute RotaryUnit
    MyApp.Controls('Panel').Controls('LBL_FC1Para' + Cstr(i+1)).Text =
    fcInitSyncDataList(i)('ROTARYUNIT')

    REM Make the unitlabel visible
    MyApp.Controls('Panel').Controls('LBL_FC1Para' + Cstr(i+1)).Visible = True

REM ===== Checkbox selected =====
Else If fcInitSyncDataList(i)('CONTROL') = 'CheckBox' Then

    REM Show the checkbox 'CB_FC1ParaX' and set the tooltip
    MyApp.Controls('Panel').Controls('CB_FC1Para' + Cstr(i+1)).Visible = True
    MyApp.Tooltip.SetToolTip(MyApp.Controls('Panel').Controls('CB_FC1Para' +
        Cstr(i+1)), fcInitSyncDataList(i)('INFOTOOLTIP'))

    REM Get the checked property from the variable with same name.
    REM If the variable does not exist
    REM get the value from the attribute DefaultValue in the XML
    REM file and save it in a variable with
    REM the same name as the checkbox
    If MyApp.myISL.getValueOfTemporaryVariable('CB_FC1Para' +
        Cstr(i+1), 'Local', 'noVariableFound') = 'noVariableFound' Then
        CType(MyApp.Controls('Panel').Controls('CB_FC1Para' +
            Cstr(i+1)), System.Windows.Forms.CheckBox).Checked =
            fcInitSyncDataList(i)('DEFAULTVALUE')
        MyApp.myISL.setTemporaryVariable('CB_FC1Para' +
            Cstr(i+1), 'Local', fcInitSyncDataList(i)('DEFAULTVALUE'))
    Else REM Variable exists, load saved value
        CType(MyApp.Controls('Panel').Controls('CB_FC1Para' +
            Cstr(i+1)), System.Windows.Forms.CheckBox).Checked =
            MyApp.myISL.getValueOfTemporaryVariable('CB_FC1Para' +
            Cstr(i+1), 'Local', fcInitSyncDataList(i)('DEFAULTVALUE'))
    End If

```

### 3.9 Changing the user interface generically

```

REM ===== Combobox selected =====
Else If fcInitSyncDataList(i)('CONTROL') = 'ComboBox' Then
    REM Show the combobox 'CmB_FC1ParaX' and set the tooltip
    MyApp.Controls('Panel').Controls('CmB_FC1Para' + Cstr(i+1)).Visible = True
    MyApp.Tooltip.SetToolTip(MyApp.Controls('Panel').Controls('CmB_FC1Para' +
    Cstr(i+1)),
    fcInitSyncDataList(i)('INFOTOOLTIP'))

    REM Add items to the combobox
    For j As Integer = 1 To fcInitSyncDataList(i)('MAXITEMS')
        CType(MyApp.Controls('Panel').Controls('CmB_FC1Para' +
Cstr(i+1)),System.Windows.Forms.ComboBox).Items.Add(fcInitSyncDataList(i)
        ('ITEM' + Cstr(j)))
    Next

    REM Get the item to be selected from a variable or from the XML file
    If MyApp.myISL.GetValueOfTemporaryVariable('CmB_FC1Para' +
    Cstr(i+1),'Local','noVariableFound') = 'noVariableFound'
    Then
        MyApp.Controls('Panel').Controls('CmB_FC1Para' + Cstr(i+1)).Text =
        fcInitSyncDataList(i)('DEFAULTVALUE')
        MyApp.myISL.setTemporaryVariable('CmB_FC1Para' +
        Cstr(i+1),'Local',fcInitSyncDataList(i)('DEFAULTVALUE'))
    Else
        MyApp.Controls('Panel').Controls('CmB_FC1Para' + Cstr(i+1)).Text =
        MyApp.myISL.GetValueOfTemporaryVariable('CmB_FC1Para' +
        Cstr(i+1),'Local',fcInitSyncDataList(i)('DEFAULTVALUE'))
    End If
End If
Next
End If

```

## 3.10 Performant interface elements

### How can I improve the performance of the user interface?

To improve the performance of the user interface you can consider the following tips.

#### **Use of transparency on the user interface**

If you set the background color of the user interface to transparent, the performance is reduced.

Use a standard color instead.

#### **Use of Autosize**

If you have the size of the controls of the user interface of *.NET* defined automatically, the performance is reduced.

Instead, provide a fixed size via the *SIZE*element.

#### **Use of DestroyForm and ChangeForm**

Use these functions sparingly. Often, better results can be achieved if the whole user interface is constructed in one step and then adapted from the source code by showing and hiding controls.



## Contact

### A.1      **Contacts**

Siemens AG

Digital Factory

Factory Automation

Production Machines

DF FA PMA APC

Frauenauracher Strasse 80

D-91056 Erlangen, Germany

Fax: +49 9131 98 1297

<mailto:tech.team.motioncontrol@siemens.com>

## A.2 Internet addresses

Additional information on various topics is provided on the following Internet pages.

### See also

SIMOTION ([www.siemens.com/simotion](http://www.siemens.com/simotion))

SINAMICS ([www.siemens.com/sinamics](http://www.siemens.com/sinamics))

Motion Control / Application Center ([www.siemens.com/motioncontrol/apc](http://www.siemens.com/motioncontrol/apc))

Packaging ([www.siemens.com/packaging](http://www.siemens.com/packaging))