Excel code generator for TIA Portal Openness

TIA Portal Openness / Excel code generator

https://support.industry.siemens.com/cs/ww/en/view/109770550

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.
By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.
The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

Table of contents

1 Introduction

1.1 Overview

1.1.1 Basic knowledge required

In order to use the code generator, general knowledge in the field of design in the TIA Portal is required.

1.1.2 Validity

This document is valid for the following products/product versions:

- Excel code generator V3.1

1.1.3 Requirement

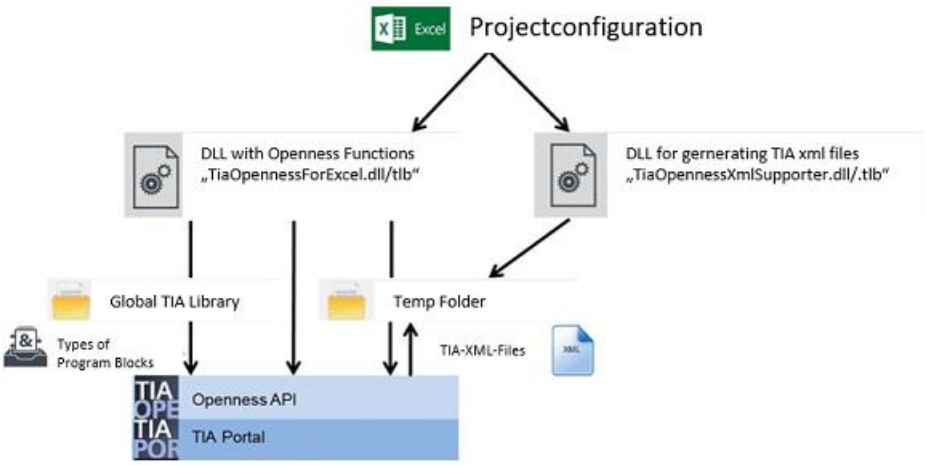
- Microsoft Excel 2007 32-/64-bit
- STEP 7 V15 (TIA Portal), STEP 7 V15.1 (TIA Portal) or STEP 7 V16 (TIA Portal)

1.2 Principle of operation

The TIA Portal Openness DLLs are based on the .NET framework version 4.6.2 and are not COM-visible, which means that they cannot be used directly in COM applications such as Microsoft Excel.

The Excel Code Generator closes this gap. The Excel file "*Datenbase_automatic_Generation.xlsm*" references two DLLs and thus enables the generation of a PLC program in the TIA Portal from Excel. On the basis of the Excel data entered, the DLL "*TiaOpennessXmlSupporter*" creates the TIA XML files that are necessary to generate program blocks, PLC data types, global data blocks and tag tables in the TIA Portal. The second DLL "*TiaOpennessForExcel*" comprises selected functions of the TIA Portal Openness DLLs and makes them available for use in Excel. The "*TiaOpennessForExcel*" DLL is responsible for accessing the TIA Portal, the project, and the global TIA corporate library, as well as importing the TIA XML files to generate the PLC program.

Abbildung 1-1 Principle of operation of the Excel code generator



Projects can be generated automatically using the code generator STEP 7 V15, STEP 7 V15.1 or STEP 7 V16. A large number of similar projects can be generated. This is useful, for example, for series machines with different configurations.

With the code generator, the following can be generated in a STEP 7 V15, STEP 7 V15.1 or STEP 7 V16 project:

- PLC data types
- Global data blocks
- PLC tag tables
- Program blocks with KOP networks
    - Cyclic organization blocks (OBs)
    - Functions (FCs)
    - Function blocks (FBs)
        - Single-instance data blocks
        - Multi-instances
- Import of library blocks (KOP, FUP, GRAPH, SCL and AWL)

The project structure is created in the code generator and the logic is imported from a library. For this a standardized global library with all necessary program blocks, which contain the complete program logic, must be available. The library blocks are interconnected within the code generator and placed at the appropriate position in the call hierarchy.

1.3 Components used

The following hardware and software components were used to create this application example:

Table 1-1

Component	Quantity	Article number	Note
TIA Portal V15, TIA Portal V15.1 or TIA Portal V16 incl. TIA Portal Openness	1	6ES7822-1..05-..	

This application example consists of the following components:

Table 1-2

Component	File name	Note
Code generator	Codegenerator_Excel2TIA-singleSolution.zip	These files are not described in detail in the application example. They are intended for your own use of the program containing source code.
Code generator	Codegenerator_Excel2TIA_V31.zip	The components contained in this folder are covered in this application example and described in more detail in the course of this section.
Subfolder	Application example	Contains the complete application example with the components: Global library, empty TIA Portal project and Excel file for configuration
Subfolder	Codegenerator_Setup	Contains the installation files

2 Engineering

2.1 Installation

1. Extract the file "Codegenerator_Excel2TIA_V31.zip" required for the application example into any empty directory.
2. Navigate to the folder "Codegenerator_Excel2TIA_V31.zip > Codegenerator_Setup".
3. Run the " Codegenerator_Excel2TIA_v1.1.exe" file (if necessary as administrator).
4. Ensure the successful installation, if necessary, by means of existing dialogs. You can also check this by ensuring that a folder "Codegenerator_DLLs_and_Data" has been created under the path "C:\Program Files".

Note | An alternative guide to installing older versions can be found in section 3.2

2.2 Configuration and parameter assignment via Excel

2.2.1 General information

Open the Excel file "Database_automatic_Generation.xlsm" and activate the table data sheet "BaseSettings". If a security warning appears, you must click on "Activate contents" to ensure that the code generator is working correctly. To better understand the structure of the code generator, this file already offers an example project with the name.

The syntax must match the syntax rules in the TIA Portal. Any syntax errors that occur are output in a log file after generation. This log file is created in the temporary folder "Temp\Log" in the current path of the Excel file.

2.2.2 Procedures

To get to know the workflows with the code generator, it is a good idea to go through the following steps one after another. The data used are selected as examples. The focus, however, is on the general procedure.

With the code generator, a basic framework of a STEP 7 project can be created from tag tables, PLC data types, global data blocks and program blocks. After the first run of the code generator, the engineering is continued in the STEP 7 project, since the code generator does not represent the full range of functions of the TIA Portal.

Variant 1: Generate new project

Here a new project with the name "ExampleProject" and a S7-1518 controller is created. The elements configured in the code generator are then inserted into the empty project. A new project is created with each generation.

Variant 2: Generating parts of an already created project

Starting point: A project has already been created in the TIA Portal and an S7-1500 controller has been created. This allows you to configure the hardware and connection before generating the PLC program.

The code generator connects to the already created and opened project in the TIA Portal by means of the project name. An S7-1500 controller must be configured in the project. The template project contains static elements, such as the hardware and connection configuration. Dynamic elements are generated using the code generator.

2.2.3 BaseSettings

The "BaseSettings" spreadsheet forms the start overview of the code generator and contains general information necessary for the generation.

In the text field "Global library name" (Figure 4.3-1) you must enter the file name of the library without file extension, e.g. "Global_Lib". This corresponds to the designation 2 in **Fehler! Verweisquelle konnte nicht gefunden werden.** without t he extension (e.g. .al15).

The absolute path of the library file must be entered in the text field "Global library path (Figure 2-1). As shown in the figure, this path is composed of two parts (see **Fehler! Verweisquelle konnte nicht gefunden werden.**). To do this, a backslash "\" and the name of the global library with file extension (e.g. .al15) are appended to the path of the folder of the global
library (1).

To define your TIA Portal Version (V15, V15.1 or V16), select the entry "TIA Portal Version" accordingly.

The other settings can remain at the default values.

Figure 2-1: "BaseSettings" spreadsheet

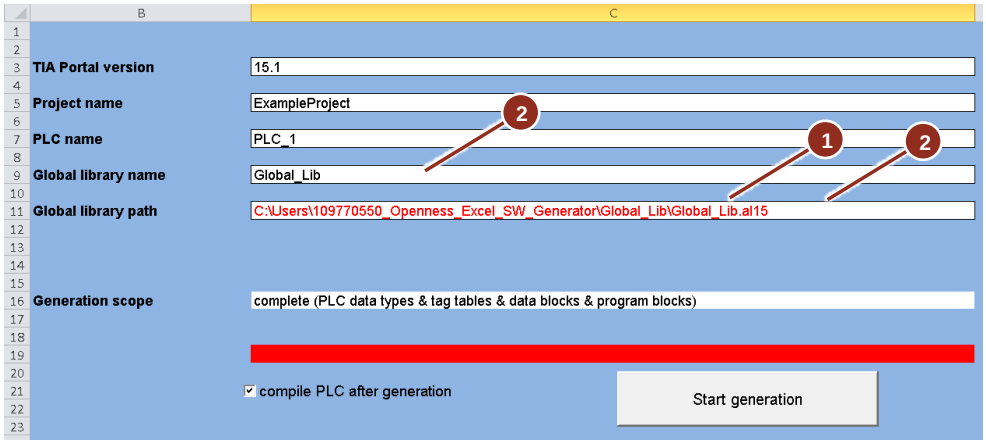


Figure 2-2 Composition oft he path tot he global library

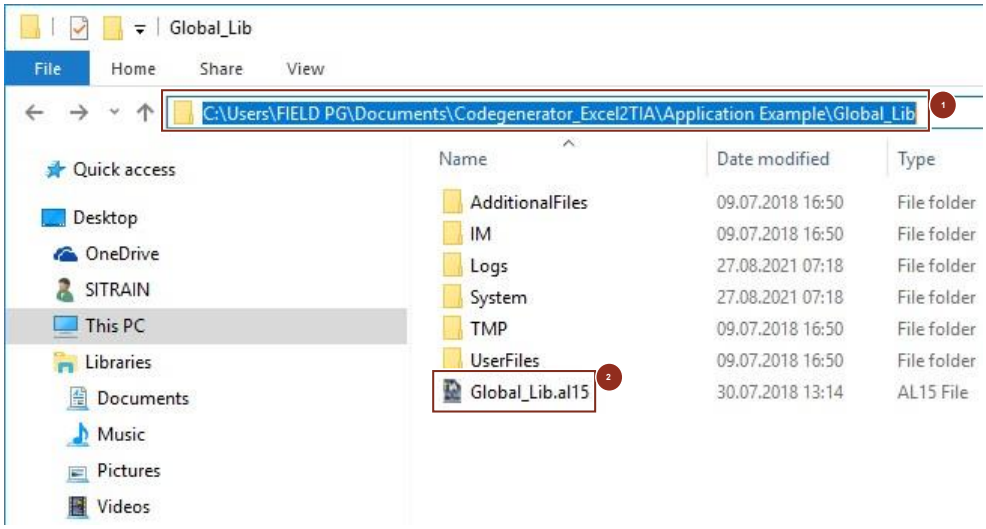


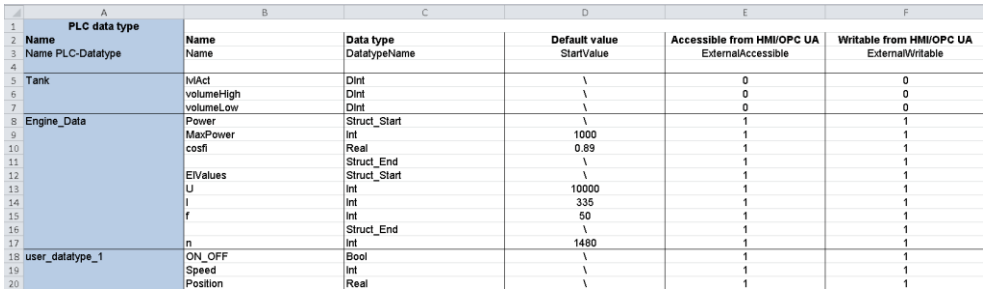
Table 2-1

Input box	Special feature
Name of project	Name of project for variant 2
Name of the PLC.	Name of S7-1500 PLC in the TIA Portal
Name of the global library	Name of library file without the file extension (e.g. .al15)
Global library path	Absolute path of the library file

2.2.4 PLC data types

User-defined data types that are not contained in the library can be created in the "PLC_DataTypes" spreadsheet. Data types that exist in the library and are assigned to a library block that is imported do not have to be created in the code generator.

Figure 2-3: PLC data types



	A	B	C	D	E	F
1	**PLC data type**					
2	**Name**	**Name**	**Data type**	**Default value**	**Accessible from HMI/OPC UA**	**Writable from HMI/OPC UA**
3	Name PLC-Datatype	Name	DatatypeName	StartValue	ExternalAccessible	ExternalWritable
4						
5	Tank	lvlAct	DInt	\	0	0
6		volumeHigh	DInt	\	0	0
7		volumeLow	DInt	\	0	0
8	Engine_Data	Power	Struct_Start	\	1	1
9		MaxPower	Int	1000	1	1
10		cosfi	Real	0.89	1	1
11			Struct_End	\	1	1
12		ElValues	Struct_Start	\	1	1
13		U	Int	10000	1	1
14		I	Int	335	1	1
15		f	Int	50	1	1
16			Struct_End	\	1	1
17		n	Int	1480	1	1
18	user_datatype_1	ON_OFF	Bool	\	1	1
19		Speed	Int	\	1	1
20		Position	Real	\	1	1

Notes on inputting PLC data types

Table 2-2

Input box	Special feature
Name of PLC data type	A new data type is created by an entry in this column and contains all elements up to the next entry in this column.
Name	Name of the tag in the PLC data type
DatatypeName	- All TIA elementary data types - enclose user-defined PLC data types with quotation marks e.g. "Data type" - Arrays can be created analogous to TIA syntax - Struct: start with "Struct_Start", end with "Struct_End". Array syntax: ARRAY[0..n] of data type
StartValue	"\" indicates default value
ExternalAccessible	- Use 1 / 0 for True / False If 0, then a 0 must also be entered for "ExternalWriteable" and "ExternalVisible".
ExternalWriteable	Use 1 / 0 for True / False
ExternalVisible	Use 1 / 0 for True / False
Setpoint	Use 1 / 0 for True / False

Example 1: PLC data types

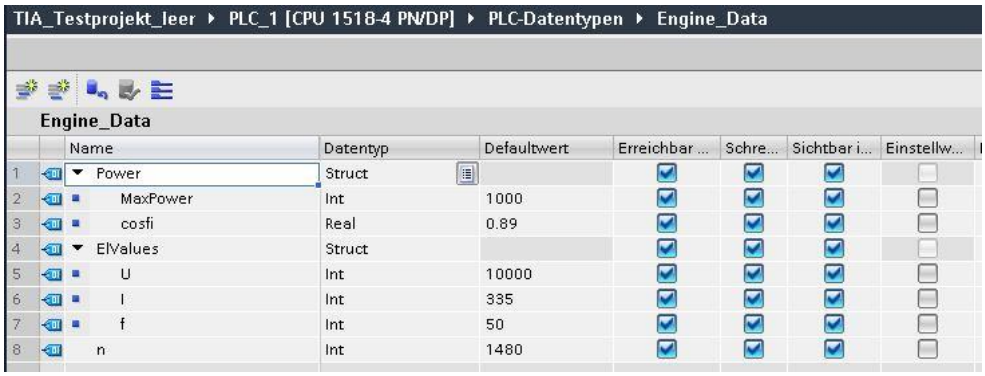
Figure 4.4-2 shows a section of the table sheet PLC data types of the code generator and is intended as an example of the input. A PLC data type with the name "*Engine_Data"* was created here. This PLC data type consists of two structures "*Power*" and "*ElValues*", as well as an integer tag "*n*".

Figure 2-4: Creating PLC data types

	A	B	C	D
1	**PLC data type**			
2	**Name**	**Name**	**Data type**	**Default value**
3	Name PLC-Datatype	Name	DatatypeName	StartValue
4				
5	Tank	lvlAct	DInt	\
6		volumeHigh	DInt	\
7		volumeLow	DInt	\
8	Engine_Data	Power	Struct_Start	\
9		MaxPower	Int	1000
10		cosfi	Real	0.89
11			Struct_End	\
12		ElValues	Struct_Start	\
13		U	Int	10000
14		I	Int	335
15		f	Int	50
16			Struct_End	\
17		n	Int	1480

After generation, this PLC data type corresponds to Figure 4.4-3 in the TIA Portal.

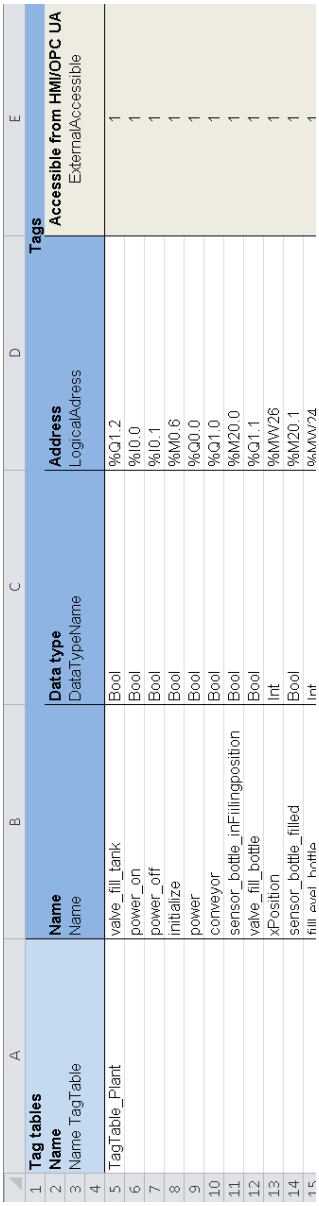
Figure 2-5: Generation of data types in the TIA Portal



2.2.5 Tag tables

The required inputs/outputs and flags can be generated in the spreadsheet "TagTables". If inputs/outputs have to be connected, variant 2 can be used, since the address ranges of the input/output modules are already known in this procedure.

Figure 2-6: Tag tables



	A	B	C	D	E
1	**Tag tables**				**Tags**
2	**Name**	**Name**	**Data type**	**Address**	**Accessible from HMI/OPC UA**
3	Name TagTable	Name	DataTypeName	LogicalAdress	ExternalAccessible
4					
5	TagTable_Plant	valve_fill_tank	Bool	%Q1.2	1
6		power_on	Bool	%I0.0	1
7		power_off	Bool	%I0.1	1
8		initialize	Bool	%M0.6	1
9		power	Bool	%Q0.0	1
10		conveyor	Bool	%Q1.0	1
11		sensor_bottle_inFiilingposition	Bool	%M20.0	1
12		valve_fill_bottle	Bool	%Q1.1	1
13		xPosition	Int	%MW26	1
14		sensor_bottle_filled	Bool	%M20.1	1
15		fill_level_bottle	Int	%MW24	1

Notes on inputting tag tables

Table 2-3

Input box	Special feature
Name TagTable	A new tag table is created by an entry in this column and contains all elements up to the next entry in this column.
Name	Tag name
DataTypeName	enclose user-defined data types with quotation marks e.g. "data type"
LogicalAdress	Address in TIA Portal notation e.g. %Q76.0
ExternalAccessible	- Use 1 / 0 for True / False

Input box	Special feature
	If 0, then a 0 must also be entered for "ExternalWriteable" and "ExternalVisible".
ExternalWriteable	Use 1 / 0 for True / False
ExternalVisible	Use 1 / 0 for True / False

Example 2: Tag tables

Figure 4.5-2 shows a section of the table sheet Tag tables of the code generator and is intended as an example of the input. Here two tag tables were created, one with the name "*TagTable_Plant*" and one with the name "*TagTable_Interconnection*".

In the figure 5 tags of the tag table "*TagTable_Plant*" are shown, which all have the data type Bool and a logical address (e.g. %Q1.2 for an output, %I0.1 for an input and %M0.6 for a flag).

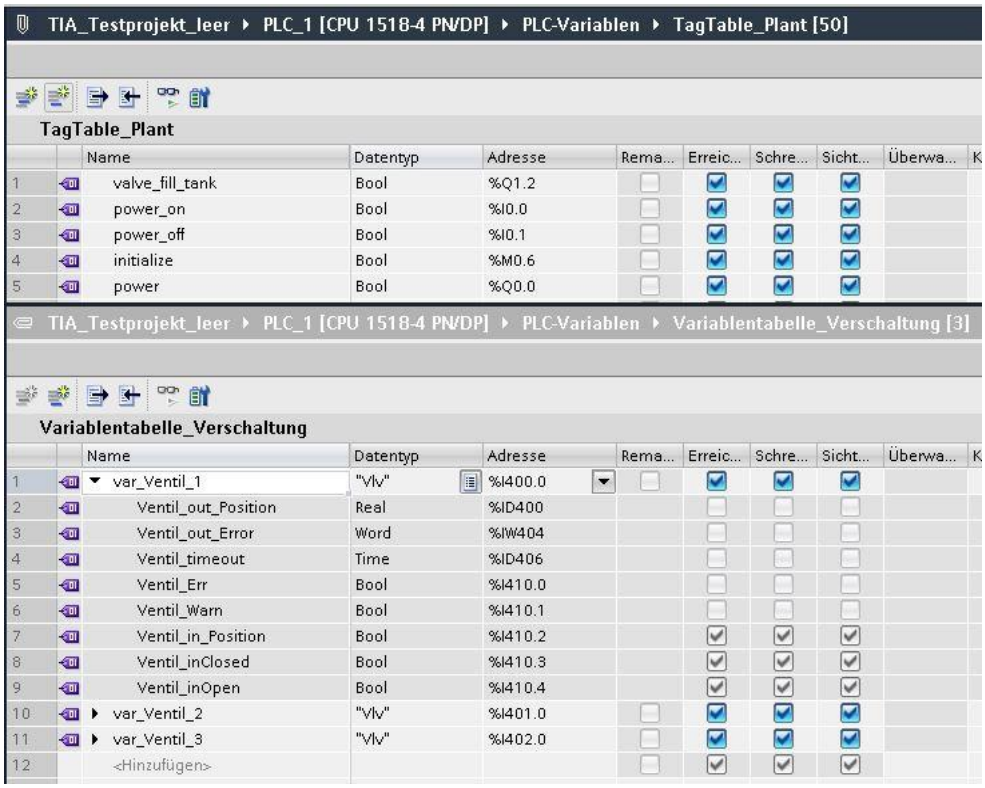
The second tag table consists of three tags which all have the PLC data type "Vlv".

Figure 2-7: Creating tag tables

	A	B	C	D
1	**Tag tables**			
2	**Name**	**Name**	**Data type**	**Address**
3	Name TagTable	Name	DataTypeName	LogicalAdress
4				
5	TagTable_Plant	valve_fill_tank	Bool	%Q1.2
6		power_on	Bool	%I0.0
7		power_off	Bool	%I0.1
8		initialize	Bool	%M0.6
9		power	Bool	%Q0.0
10		conveyor	Bool	%Q1.0
11		sensor_bottle_inFiilingposition	Bool	%M20.0
12		valve_fill_bottle	Bool	%Q1.1
13		xPosition	Int	%MW26
14		sensor_bottle_filled	Bool	%M20.1
15		fill_evel_bottle	Int	%MW24

After generation, these two tag tables "*TagTable_Plant*" and "*TagTable_Interconnection*" correspond to the TIA Portal section shown in Figure 4.5-3.

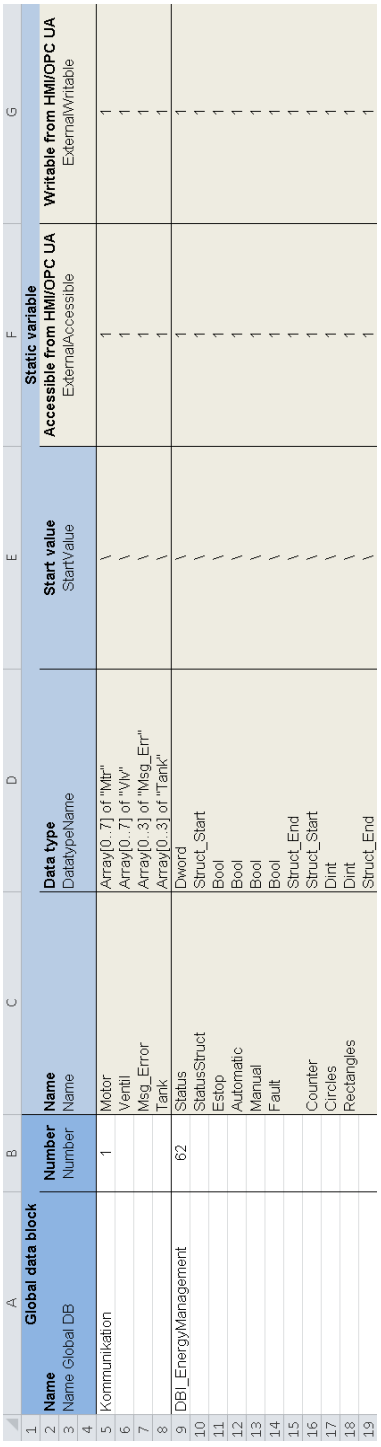
Figure 2-8: Generation of tag tables in the TIA Portal



2.2.6 Data blocks

Global data blocks and their internal structure can be configured in the spreadsheet "DataBlocks". Single-instance data blocks do not have to be configured, as they are automatically generated by their naming in the "ProgramBlocks" spreadsheet.

Figure 2-9: Data blocks



	A	B	C	D	E	F	G
1	**Global data block**					**Static variable**	
2	**Name**	**Number**	**Name**	**Data type**	**Start value**	**Accessible from HMI/OPC UA**	**Writable from HMI/OPC UA**
3	Name Global DB	Number	Name	DatatypeName	StartValue	ExternalAccessible	ExternalWritable
4							
5	Kommunikation	1	Motor	Array[0..7] of "Mtr"	\	1	1
6			Ventil	Array[0..7] of "Vlv"	\	1	1
7			Msg_Error	Array[0..3] of "Msg_Err"	\	1	1
8			Tank	Array[0..3] of "Tank"	\	1	1
9	DB|_EnergyManagement	62	Status	Dword	\	1	1
10			StatusStruct	Struct_Start	\	1	1
11			Estop	Bool	\	1	1
12			Automatic	Bool	\	1	1
13			Manual	Bool	\	1	1
14			Fault	Bool	\	1	1
15				Struct_End	\	1	1
16			Counter	Struct_Start	\	1	1
17			Circles	Dint	\	1	1
18			Rectangles	Dint	\	1	1
19				Struct_End	\	1	1

Notes on inputting global data blocks

Table 2-4

Input box	Special feature
Name of global DB	A new global data block is created by an entry in this column and contains all elements up to the next entry in this column.
Number	- Optional unique number of the data block
Name	Tag name
DatatypeName	- enclose user-defined data types with quotation marks e.g. "data type" - Struct: start with "Struct_Start", end with "Struct_End". Array syntax: ARRAY[0..n] of data type
StartValue	- "\" indicates default value user-defined data types cannot receive a start value
ExternalAccessible	- Use 1 / 0 for True / False If 0, then a 0 must also be entered for "ExternalWriteable" and "ExternalVisible".
ExternalWriteable	Use 1 / 0 for True / False
ExternalVisible	Use 1 / 0 for True / False
Setpoint	Use 1 / 0 for True / False

Example 3: Global data blocks

Figure 4.6-2 shows a section of the table sheet Data blocks of the code generator and is intended as an example of the input. Three global data blocks with the names "*Communication*", "*DBI_EnergyManagement*" and "*DB_Interconnection*" were created here.

The global data block "Communication" is created with the number "DB1" and contains four static global tags which each have an array form as data type (e.g. Array[0..3] of "Tank").

The global data block "DBI_EnergyManagement" is created with the number "DB62" and contains, besides a global Dword tag, two Structs.

The global data block DB_Interconnection is created with the number "DB30" and contains 5 global static tags which each have a PLC data type (e.g. "Mtr") as data type.

Figure 2-10: Creating data blocks

	A	B	C	D
1	**Global data block**			
2	**Name**	**Number**	**Name**	**Data type**
3	Name Global DB	Number	Name	DatatypeName
4				
5	Kommunikation	1	Motor	Array[0..7] of "Mtr"
6			Ventil	Array[0..7] of "Vlv"
7			Msg_Error	Array[0..3] of "Msg_Err"
8			Tank	Array[0..3] of "Tank"
9	DBI_EnergyManagement	62	Status	Dword
10			StatusStruct	Struct_Start
11			Estop	Bool
12			Automatic	Bool
13			Manual	Bool
14			Fault	Bool
15				Struct_End
16			Counter	Struct_Start
17			Circles	Dint
18			Rectangles	Dint
19				Struct_End
20	DB_Verschaltung	30	Motor_1	"Mtr"
21			Motor_2	"Mtr"
22			Ventil_1	"Vlv"
23			Ventil_2	"Vlv"
24			Msg_Error	"Msg_Err"

After generation, these three global data blocks "*Communication*", "*DBI_EnergyManagement*" and "*DB_Interconnection*" correspond to the TIA Portal sections shown in Figures 4.6-3 and 4.6-4.

Figure 2-11: Generation of data blocks in the TIA Portal 1

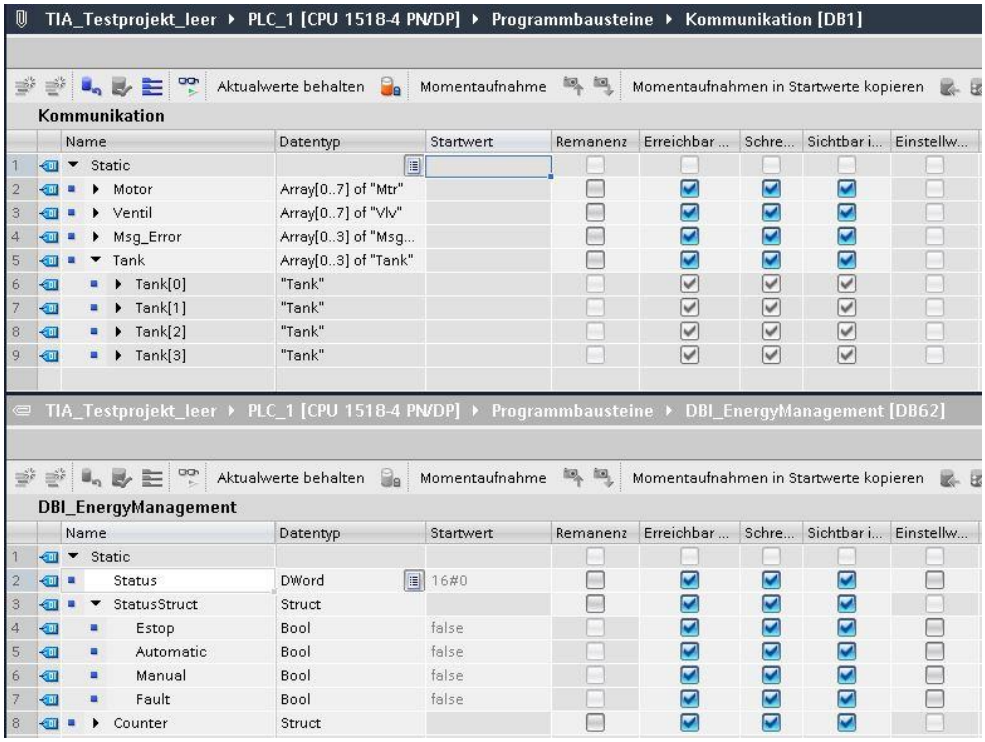
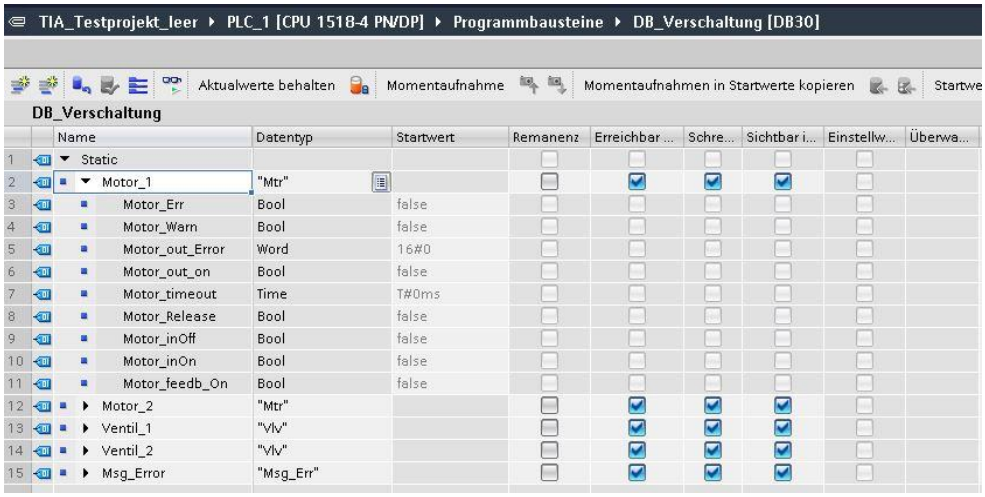


Figure 2-12: Generation of data blocks in the TIA Portal 2



2.2.7 Program blocks

The call structure of the blocks is created in the "ProgramBlocks" spreadsheet. The table starts on the left side with the top level (1) and the subordinate levels (2, 3 and 4) are on the right side.

In a program block (cyclic OB, FC or FB) one FC or FB can be called per LD network. Ideally, the call depth ends with the call of a library block (in which a logic is programmed and standardized) on the 2nd, 3rd or 4th level.

The column "Number of the FB or FC is optional" does not have to be filled. In this case, the numbers are assigned automatically by the TIA portal.

The called FC or FB can be a type of a global library, then the column "*Libray Connection*" must be filled with the Boolean value TRUE, so that the block is imported from the global library into the TIA Portal project. If the called block is not present in the global library ("Libray Connection" = FALSE), then the block is generated via the code generator in the form of an XML file and then imported into the TIA Portal.

In the case of a function block (FB), the name of the individual data block or the multi-instance, indicated by a hash "#" in front of the name, must be entered in the "*Instance name*" column.

Figure 2-13: Call structure of the program blocks

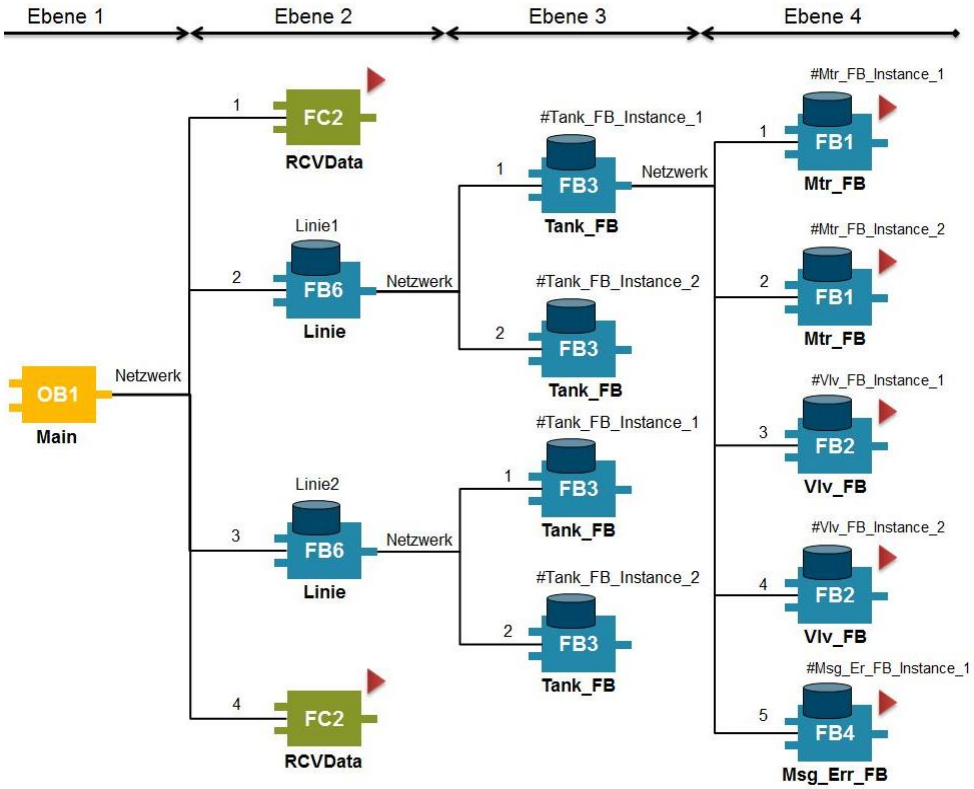
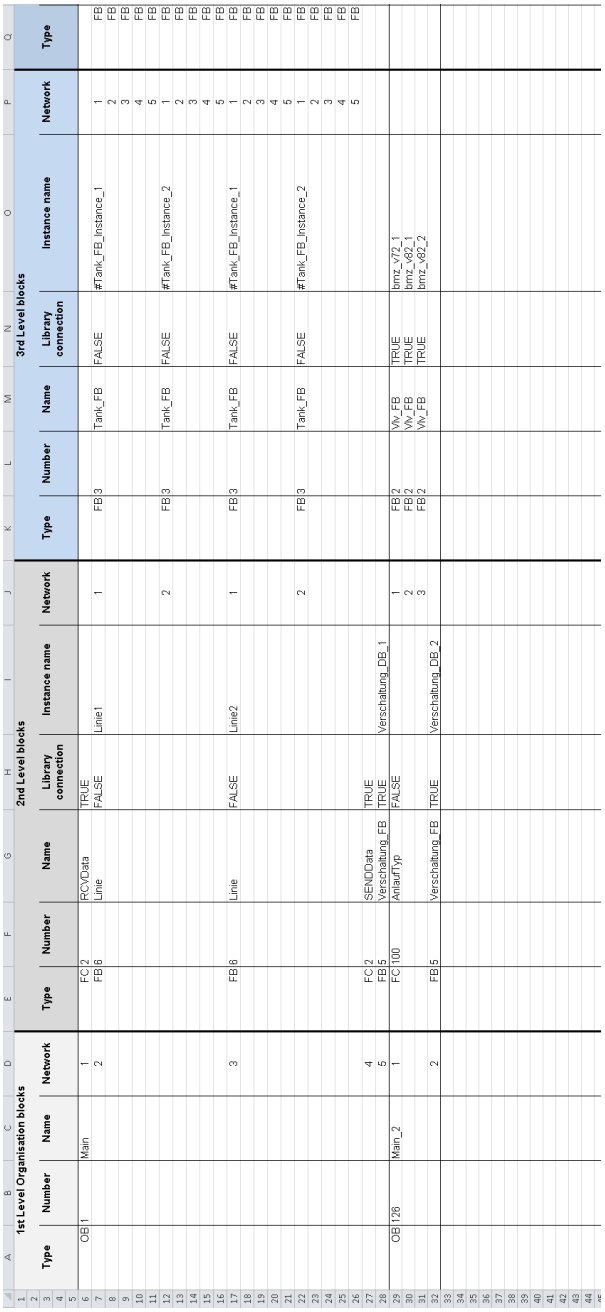
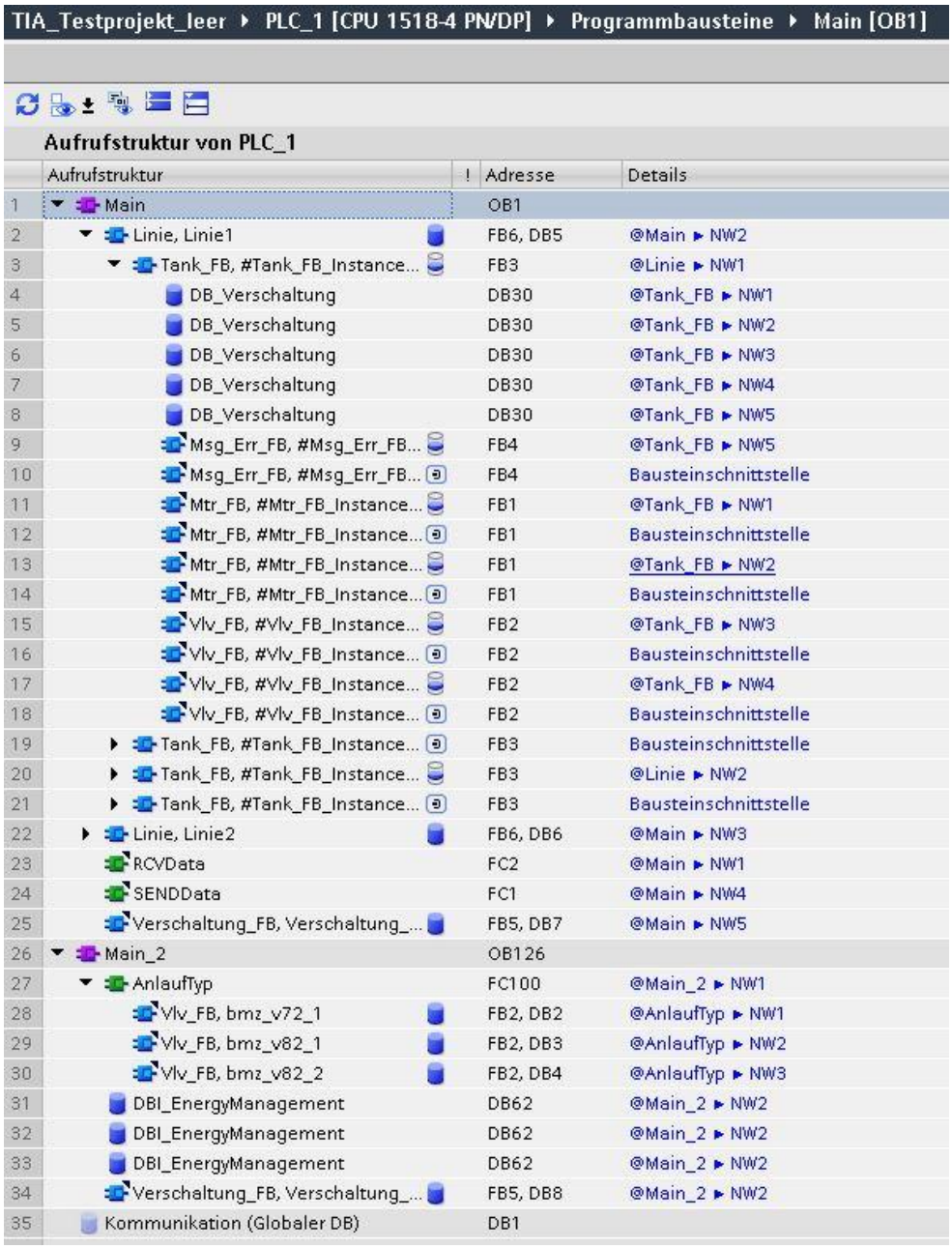


Figure 2-14: Program blocks



1st Level Organisation blocks				2nd Level blocks						3rd Level blocks				
Type	Number	Name	Network	Type	Number	Name	Library connection	Instance name	Network	Type	Number	Name	Library connection	Instance name	Network	Type
OB	1	Main	1	FC	2	RCVData	TRUE		1	FB	3	Tank_FB	FALSE	#Tank_FB_Instance_1	1	FB
			2	FB	6	Linie	FALSE	Linie1							2	FB
															3	FB
															4	FB
															5	FB
									2	FB	3	Tank_FB	FALSE	#Tank_FB_Instance_2	1	FB
															2	FB
															3	FB
															4	FB
															5	FB
			3	FB	8	Linie	FALSE	Linie2	1	FB	3	Tank_FB	FALSE	#Tank_FB_Instance_1	1	FB
															2	FB
															3	FB
															4	FB
															5	FB
									2	FB	3	Tank_FB	FALSE	#Tank_FB_Instance_2	1	FB
															2	FB
															3	FB
															4	FB
															5	FB
			4	FC	2	SENDData	TRUE									
			5	FB	5	Verschaltung_FB	TRUE	Verschaltung_DB_1	1	FB	2	Vlv_FB	TRUE	bmz_v72_1		
OB	126	Main_2	1	FC	100	AnlaufTyp	FALSE		2	FB	2	Vlv_FB	TRUE	bmz_v82_1		
									3	FB	2	Vlv_FB	TRUE	bmz_v82_2		
			2	FB	5	Verschaltung_FB	TRUE	Verschaltung_DB_2								

Figure 2-15: Generation of program blocks in the TIA Portal



Notes on inputting organization blocks 1st level

Table 2-5

Input box	Special feature
Type	must be "OB".
Number	Number of the cyclic OB
Name	Name of the cyclic OB
Network	specifies the number of the LD network in which a block is to be called.

Notes on inputting blocks of the 2nd to 4th level

Table 2-6

Input box	Special feature
Type	"FC", "FB" permissible
Number	Optional
Name	- Libray Connection = FALSE: Name of the FC or FB to be generated Libray Connection = TRUE: Name must correspond to the spelling of the name of the library block; this block is loaded or imported via the name from the library
Libray Connection	- TRUE / FALSE pay attention to upper case or select via drop down menu
Instance name	- only enter the name of the corresponding instance for FBs - for multi-instance: "#" before instance name for single-instance data blocks: only single-instance names
Network	- only fill out if Libray Connection = FALSE defines in which KOP network of this block a block is called.

Example 4: Calling a FB with the single instance DB in the cyclic (1st→ 2nd level)

As shown in the following figure, in the 1st network the OB1 "Main" calls the FC2 "RCVData", which is a library type and is imported from the global library.

In the 2nd network of the OB the FB6 "Line" is called together with the single instance DB "Line1". Since the Libray Connection = FALSE, an XML file is generated for this block and then imported into the TIA portal. The structure of the FB6 "Line" is described in example 5.

Figure 2-16: Connecting program blocks 1

	A	B	C	D	E	F	G	H	I	J
1	1st Level Organisation blocks				2nd Level blocks					
2										
3	Type	Number	Name	Network	Type	Number	Name	Library connection	Instance name	Network
4										
5										
6	OB	1	Main	1	FC	2	RCVData	TRUE		
7				2	FB	6	Linie	FALSE	Linie1	1
8										
9										
10										
11										
12										2
13										
14										
15										
16										
17				3	FB	6	Linie	FALSE	Linie2	1
18										

Example 5: Calling two FBs with one multi-instance each in one FB (2nd→3rd level)

As shown in the following figure, in the 1st network the FB6 "Line" calls the FB3 "Tank_FB" together with the multi-instance (marked by #) "Tank_FB_Instance_1".

In the 2nd network of the FB "Line" the FB3 "Tank_FB" is also called with the second multi-instance "Tank_FB_Instance_2".

Since the Libray Connection of the FB "Tank_FB" = FALSE, an XML file is generated for this block and then imported into the TIA portal. The structure of the FB3 "Tank" is described in example 6.

Figure 2-17: Connecting program blocks 2

E	F	G	H	I	J	K	L	M	N	O	P
2nd Level blocks						3rd Level blocks					
Type	Number	Name	Library connection	Instance name	Network	Type	Number	Name	Library connection	Instance name	Network
FC	2	RCVData	TRUE								
FB	6	Linie	FALSE	Linie1	1	FB	3	Tank_FB	FALSE	#Tank_FB_Instance_1	1
											2
											3
											4
											5
					2	FB	3	Tank_FB	FALSE	#Tank_FB_Instance_2	1
											2
											3
											4
											5
FB	6	Linie	FALSE	Linie2	1	FB	3	Tank_FB	FALSE	#Tank_FB_Instance_1	1
											2
											3

Example 6: Calling five library blocks (FBs) with one multi-instance each in one FB (3rd → 4th level)

As shown in the following figure, in the 1st network the FB3 "Tank_FB" calls the library bound FB1 "Mtr_FB" together with the multi-instance "Mtr_FB_Instance_1".

In the 2nd network of the FB "Line" the library-linked FB1 "Mtr_FB" is also called with the second multi-instance "Mtr_FB_Instance_2".

In the 3rd and 4th network of the FB "Tank_FB" the library-linked FB2 "Vlv_FB" is called once with the corresponding multi-instance "Vlv_FB_Instance_1" or "Vlv_FB_Instance_2".

In the 5th network of the FB "Tank_FB" the library-linked FB4 "Msg_Err_FB" is called together with the multi-instance "Msg_Err_FB_Instance_1".

The library-linked block types "Mtr_FB", "Vlv_FB" and "Msg_Err_FB" together with their library-linked PLC data types "Mtr", "Vlv" and "Msg_Err" are imported from the global library into the TIA Portal project.

Figure 2-18: Connecting program blocks 3

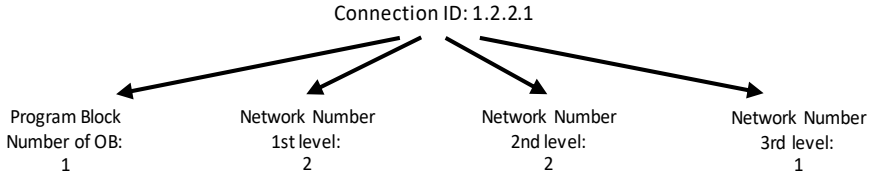
K	L	M	N	O	P	Q	R	S	T	U
3rd Level blocks						4th Level blocks				
Type	Number	Name	Library connection	Instance name	Network	Type	Number	Name	Library connection	Instance name
FB	3	Tank_FB	FALSE	#Tank_FB_Instance_1	1	FB	1	Mtr_FB	TRUE	#Mtr_FB_Instance_1
					2	FB	1	Mtr_FB	TRUE	#Mtr_FB_Instance_2
					3	FB	2	Vlv_FB	TRUE	#Vlv_FB_Instance_1
					4	FB	2	Vlv_FB	TRUE	#Vlv_FB_Instance_2
					5	FB	4	Msg_Err_FB	TRUE	#Msg_Err_FB_Instance_1
FB	3	Tank_FB	FALSE	#Tank_FB_Instance_2	1	FB	1	Mtr_FB	TRUE	#Mtr_FB_Instance_1
					2	FB	1	Mtr_FB	TRUE	#Mtr_FB_Instance_2
					3	FB	2	Vlv_FB	TRUE	#Vlv_FB_Instance_1

2.2.8 Interconnections

In this spreadsheet, the interfaces of the previously created program blocks can be linked with data. The individual program block is addressed using the position at which it is generated. This addressing takes place by means of the interconnection ID. It refers to the network numbers of the networks via the levels into which the block was generated . Thus, the interconnection ID links the information to the call of a block and the connection of its interfaces.

For example, the **interconnection ID 1.2.2.1** means that the OB "*Main*" has the block number 1 and calls a block in the **2nd** network. This block in turn also calls a block in the **2nd** network. In this block in the 1st network the specified interconnection is carried out.

Figure 2-19: Interconnection ID diagram



The interfaces can be connected with global tags of a tag table
(e.g. inputs, outputs and flags) as well as with global tags of a global data block.

Figure 2-20: Connecting the interfaces 1

	A	F	G	H
1	**Connection-ID**	**Inputs/Outputs**	**Name of the global variable (from a tag table or global data block)**	**Data type**
2				
3				
4	1.5.0.0	Input:Input_1	power_on	Bool
5	1.2.1.1	InOut:UDT_Motor	DB_Verschaltung.Motor_1	"Mtr"
6	1.2.1.2	InOut:UDT_Motor	DB_Verschaltung.Motor_2	"Mtr"
7	1.2.1.3	InOut:UDT_Ventil	DB_Verschaltung.Ventil_1	"Vlv"
8	1.2.1.4	InOut:UDT_Ventil	DB_Verschaltung.Ventil_2	"Vlv"
9	1.2.1.5	InOut:UDT_Msg_Err	DB_Verschaltung.Msg_Error	"Msg_Err"
10	1.2.2.1	InOut:UDT_Motor	DB_Verschaltung.Motor_1	"Mtr"
11	1.2.2.2	InOut:UDT_Motor	DB_Verschaltung.Motor_2	"Mtr"
12	1.2.2.3	InOut:UDT_Ventil	DB_Verschaltung.Ventil_1	"Vlv"
13	1.2.2.4	InOut:UDT_Ventil	DB_Verschaltung.Ventil_2	"Vlv"
14	1.2.2.5	InOut:UDT_Msg_Err	DB_Verschaltung.Msg_Error	"Msg_Err"
15	1.3.1.1	InOut:UDT_Motor	DB_Verschaltung.Motor_1	"Mtr"
16	1.3.1.2	InOut:UDT_Motor	DB_Verschaltung.Motor_2	"Mtr"
17	1.3.1.3	InOut:UDT_Ventil	DB_Verschaltung.Ventil_1	"Vlv"
18	1.3.1.4	InOut:UDT_Ventil	DB_Verschaltung.Ventil_2	"Vlv"
19	1.3.1.5	InOut:UDT_Msg_Err	DB_Verschaltung.Msg_Error	"Msg_Err"
20	1.3.2.1	InOut:UDT_Motor	DB_Verschaltung.Motor_1	"Mtr"
21	1.3.2.2	InOut:UDT_Motor	DB_Verschaltung.Motor_2	"Mtr"
22	1.3.2.3	InOut:UDT_Ventil	DB_Verschaltung.Ventil_1	"Vlv"
23	1.3.2.4	InOut:UDT_Ventil	DB_Verschaltung.Ventil_2	"Vlv"
24	1.3.2.5	InOut:UDT_Msg_Err	DB_Verschaltung.Msg_Error	"Msg_Err"
25	126.1.1.0	InOut:UDT_Ventil	var_Ventil_1	"Vlv"
26	126.1.2.0	InOut:UDT_Ventil	var_Ventil_2	"Vlv"
27	126.1.3.0	InOut:UDT_Ventil	var_Ventil_3	"Vlv"
28	126.2.0.0	Input:Input_1	DBI_EnergyManagement.StatusStruct.Automatic	Bool

Notes on inputting interconnections

Table 2-7

Input box	Special feature
Interconnection ID	- Always four digits "W.X.Y.Z." - W: Number of the OB - X OBs Network number in the OB (1st level) - Y: Network number of the block (2nd level) - Z: Network number of the block - (3rd level) - a block which is called in OB1 in the 3rd network receives the interconnection ID 1.3.0.0
inputs/outputs	- Input:<Name> - InOut:<Name> - Output:<Name>
Name of the global tag	- Elements of an array cannot be interconnected e.g. database.Arr[0] - the whole array can be interconnected. e.g. database.Arr - Constants cannot be interconnected e.g. "5.0".
Data type	- enclose user-defined data types with quotation marks e.g. "Vlv"

If several inputs / outputs are to be interconnected, further entries can be created according to the same pattern (columns I to K) in the following 245 columns (columns L - IV) of the corresponding line.

Example 7: Connection of a library bound FB of the 4th level with a global tag of a global data block in a 3rd level FB

Thus, the interconnection ID links the information to the call of a block and the connection of its interfaces. The **interconnection ID 1.2.2.1** shown in the figure means that the OB "*Main*" has the block number 1 and calls a block in the **2nd** network. This block in turn also calls a block in the **2nd** network. In this block in the **1st** network the specified interconnection is carried out.

This corresponds to the call of the library bound FB "Mtr_FB" (4th level) in the first network of the FB "Tank_FB" (3rd level), which has already been described in the spreadsheet "Program blocks".

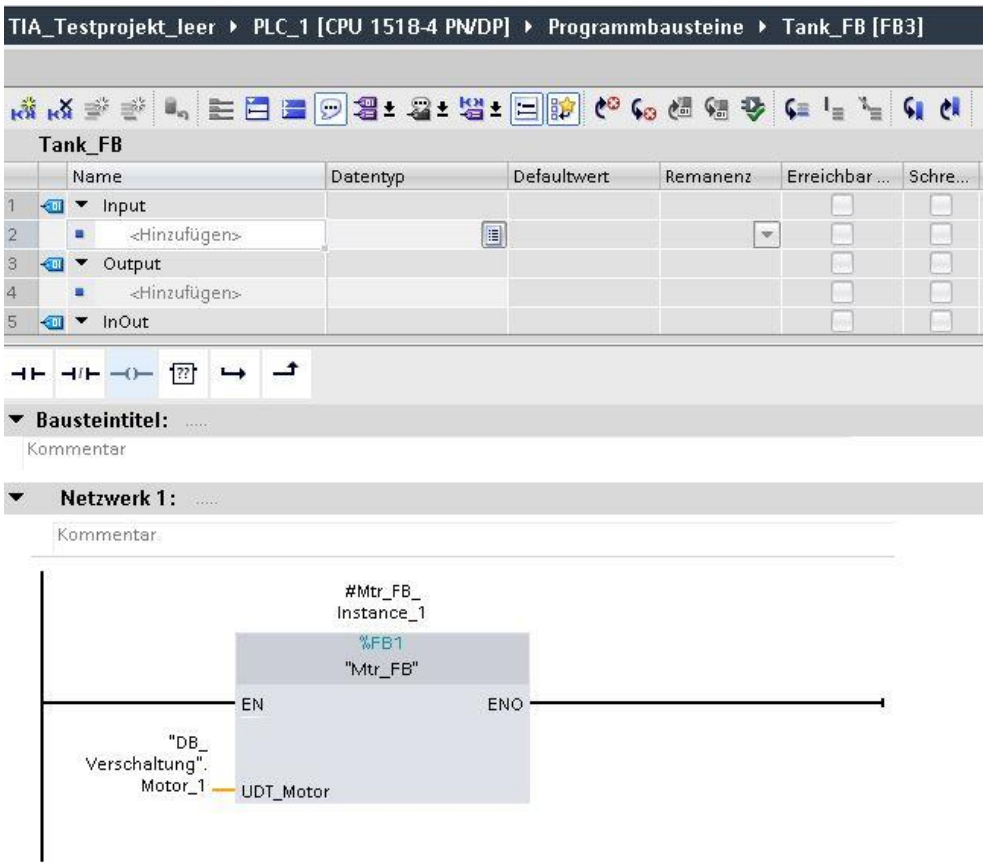
Figure 2-21 describes that the InOut interface of the called FB is called "UDT_Motor" and is connected with the global tag "Motor_1" of the global data block "DB_Interconnection". The data type of this tag or the interface is the PLC data type "Mtr".

Figure 2-21: Connecting the interfaces 3

	A	F	G	H
1	Connection-ID	Inputs/Outputs	Name of the global variable (from a tag table or global data block)	Data type
2				
3				
10	1.2.2.1	InOut:UDT_Motor	DB_Verschaltung.Motor_1	"Mtr"

After generation, this example corresponds to the figure below.

Figure 2-22: Generation of interconnections in the TIA Portal 1

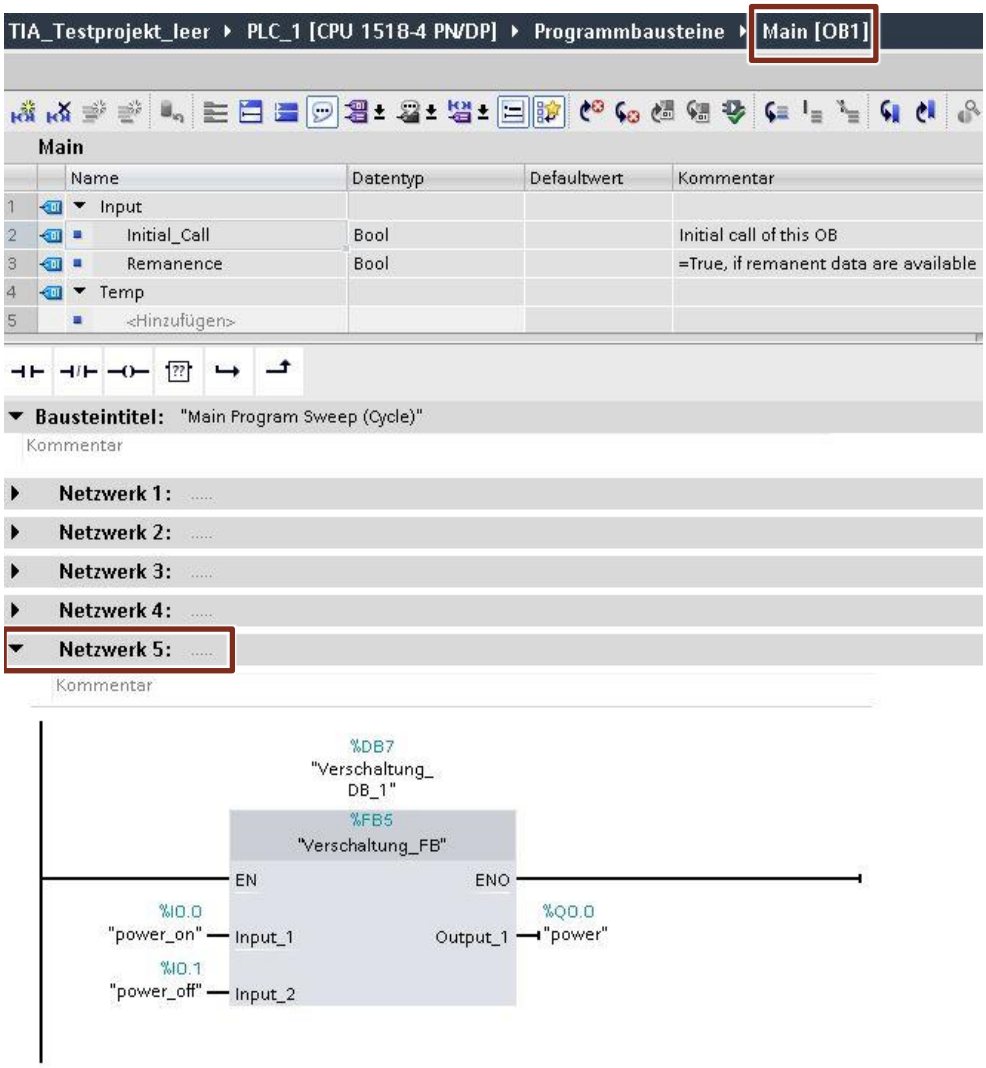


Example 8: Connection of a library bound FB of the 2nd level with 3 global tags of a tag table in OB1

If several inputs / outputs are to be interconnected, further entries can be created according to the same pattern (columns I to K) in the following 245 columns (columns L - IV) of the corresponding line.

Figure 4.8-5 illustrates the wiring of an FB in the 5th network of the OB1 "Main". The interconnection ID is therefore 1.5.0.0. This corresponds to line 10 of Figure 2-20 highlighted. As described in this line, the two inputs and the output of the block "Interconnection_FB" of the 2nd level are connected with global tags of a tag table.

Figure 2-23: Generation of interconnections in the TIA Portal 2



2.3 Generation

2.3.1 Generation in general

As soon as the generation is started via the "Start generation" button on the "BaseSettings" spreadsheet, the code generator accesses the TIA portal. A message that appears in the TIA Portal when using Openness for the first time can be confirmed with "Yes, all". During the generation, it is recommended to unfold the project navigation to observe the process. The end of the generation is indicated in the code generator by the color change of the bar from red to green. The process must not be interfered with during the generation procedure.

2.3.2 Generating a complete project

The generation procedure corresponds to variant 1 in section 4.4.1.

The "name of the project" is not considered in this case. A new project is created during each generation process and an S7 controller (S7-1518) is automatically

inserted into the project. Start the generation by clicking on "Start generation". The TIA Portal opens and creates a new empty project with the name "ExampleProject". The projected elements are inserted into this project. Before you can generate the project again, the previously opened project must be closed.

2.3.3 Generating parts of a project

The generation procedure corresponds to variant 2 in section 4.4.2.

Open your template project in the TIA Portal. Create all elements that are not generated by the code generator. Open the "BaseSettings" spreadsheet and adapt the "Project name" to the name of the opened project. This name connects the code generator to the open project in the TIA Portal. Enter the name of the configured S7-1500 CPU into the field "Name of the PLC". Start the generation by clicking on the "Start generation" button. The configured elements are now inserted into the project.

2.3.4 Extra functions

The spreadsheet "AdditionalFeatures" is not used for the described generation process of the PLC program. It provides some additional functions of the Openness interface for use in Excel, for example to save and close the project in the TIA portal.

2.4 Error handling

2.4.1 Log files

If the configuration has an error, a log file is created in the "Temp\Log" folder during generation. This folder is created in the current path of the Excel file. Warnings and errors are documented in the log.

2.4.2 Code generator does not connect to the TIA Portal

If a new project opens during generation according to variant 2, wait until an error message is issued. Close the code generator and the TIA Portal. Open both again. Open the template project again. Now the generation can be started again.

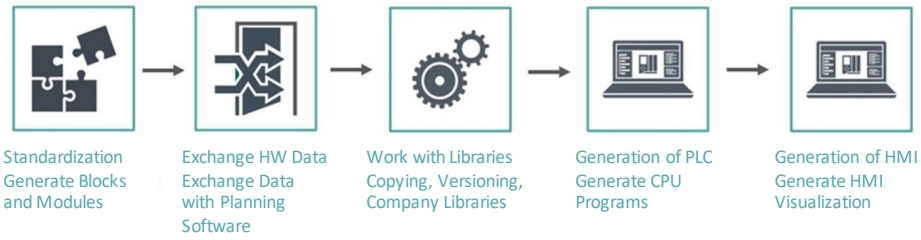
3 Useful information

3.1 Basics

3.1.1 Automatic code generation

The generation of the PLC program is part of the digital workflow to increase efficiency and is based on the standardization of PLC blocks and the subsequent storage of standardized objects in global libraries.

Figure 3-1: Automatic code generation



3.1.2 TIA Portal Openness

In order to generate the PLC program instead of programming it, Siemens AG provides the open interface TIA Portal Openness. TIA Portal Openness is part of the scope of delivery of STEP 7 V15 / V15.1 / V16 and offers the possibility to automate the engineering. A specially programmed application can remotely control the TIA portal via the Openness API. TIA Portal Openness contains API functions that are used for program-controlled creation and modification of projects and project data as well as for remote control of the TIA Portal functions. On the other hand, TIA Portal Openness includes functions for exporting and importing data in XML file format. The TIA Portal functionality is stored in libraries (DLLs) that allow tested and released access to the objects and functions of the TIA Portal.

3.1.3 Entering Users in the TIA Openness Group

The code generator works on the basis of the Openness interface of the TIA portal. If the user of the PC is not yet registered in the user group "Siemens TIA Openness", you must carry out an initial setup of the Openness interface via the system control. Depending on the configuration of your computer, you may have to log in with administrator rights to extend the user group.

Note

Please refer to the manual for detailed instructions on how to enter the user in the TIA Openness group:
https://support.industry.siemens.com/cs/ww/en/view/109477163

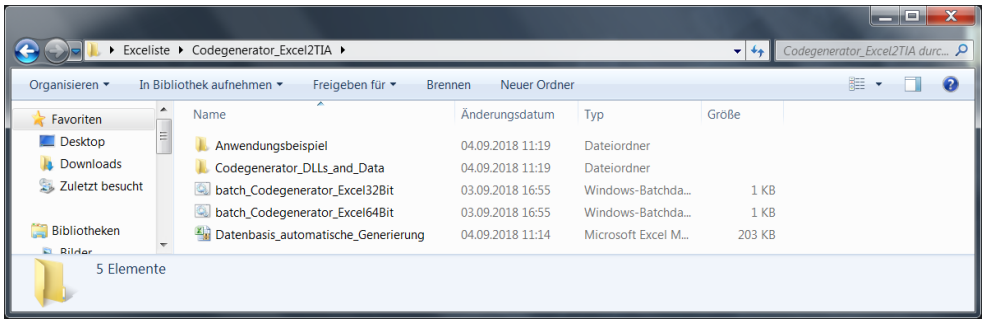
3.2 Alternative installation with earlier versions

If you are working with previous versions, please install the software as follows.

3.2.1 Unzip data

Unzip the content of the "Codegenerator_Excel2TIA_Vx.x.zip" file (previous versions) to an empty directory of your choice. Check whether the unpacked content corresponds to the image.

Figure 3-2: File structure



3.2.2 Determine Excel version

To find out the version of your Excel installation, open Excel. In Excel it is documented under "File > Help" whether it is a 32 or 64 bit installation.

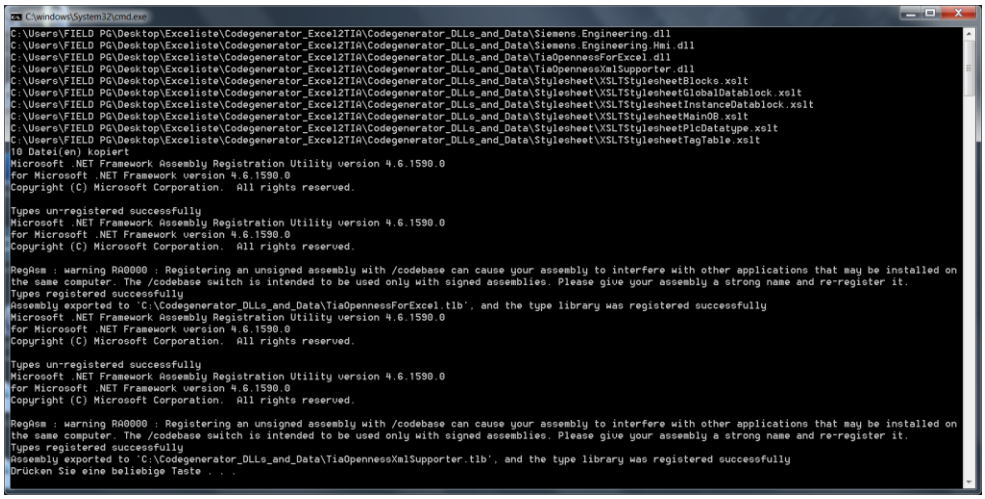
Alternative: If you are using Microsoft Excel for Office 365, proceed as follows: Open "File > Account > About Excel". You will be shown whether it is a 32-bit or 64-bit installation.

Figure 3-3: Determine Excel version

3.2.3 Installation

To install the necessary files, you have to run the "batch_Codegenerator_Excel<xx>Bit" corresponding to your version of Excel in the unzipped folder as administrator.

Figure 3-4: Installation execution



After the commands in the command line have been executed automatically, you can press any key or close the window. The code generator, the Excel file "Database_automatic_generation", can now be used anywhere in the file system.

4 Appendix

4.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:
https://support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:
www.siemens.com/industry/supportrequest

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that’s tailored to the customer’s specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:
www.siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:
https://support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:
https://support.industry.siemens.com/cs/ww/en/sc/2067

4.2 Links and literature

Table 4-1

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/109770550
\3\	SIMATIC automation of projects with scripts https://support.industry.siemens.com/cs/ww/en/view/109755218

4.3 Change documentation

Table 4-2

Version	Date	Modification
V3.0	09/2019	First version
V3.1	08/2021	TIA Portal V16 support; support of Office 32-/64-bit