SIEMENS

Ingenuity for life

Basics on HMI Faceplates

SIMATIC Comfort Panels, Runtime Advanced and WinCC (TIA Portal)

https://support.industry.siemens.com/cs/ww/DE/view/68014632

Siemens Industry Online Support

Warranty and Liability

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens’ products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.industry.siemens.com.

Table of Contents

Preface

Objective of the document

It is the objective of this document to provide you with all the useful information on the topic of "creating HMI faceplates" and how to use HMI faceplates as a user.

Validity

- Software version as of WinCC Comfort V14
- All SIMATIC HMI operator panels that support faceplates. Information on the function scope of the SIMATIC HMI operator panels is available in the Online Help of WinCC (TIA Portal) or FAQ http://support.automation.siemens.com/WW/view/en/40227286. \3\

1 Basics

The Basics chapter describes individual terms relevant for the application.

1.1 What is a faceplate?

A faceplate is a configured group of displays and operator objects. The "data exchange" with the display and operator objects contained in the faceplate is performed via an interface at the created faceplate.

The properties of the used display and operator objects are assigned to the individual objects in the "faceplate editor".

Faceplates can be managed and modified in a central library.

1.2 How to create and open a faceplate?

Faceplates can be created and opened in two ways.

- Create and open in the "pictures" editor.
- Create and open via the project library.

Respective details are described in chapter **2.1**.

1.3 What does the "pictures" editor refer to?

In this application, the "pictures" editor refers to the editor in which you create and edit your HMI pictures.

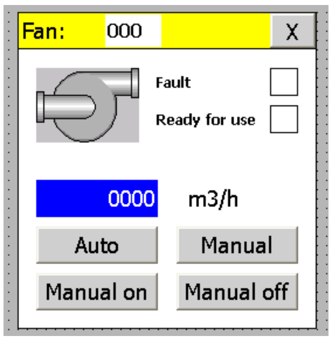
In the "pictures" editor, the following familiar task cards are available to you:

- Tools:
  - display and control objects
- Animations:
  - templates for dynamic configuration.
- Layout:
  - tools for adjusting the representation
- Libraries:
  - managing the project library and the global libraries.

1.4 What is a picture object?

In this application, a picture object refers to a compilation of several individual display and operator objects, such as IO fields and buttons, whose properties can be edited **individually**. The example in this application is the "control panel" shown in the figure below.

Figure 1-1



Control panel consisting of:

• buttons

• IO fields

• text fields

1.5 Difference between picture object and faceplate?

When looking at a "control panel" of compiled objects, then you can initially not know whether this "control panel" has been created as a "conventional" picture object or as a faceplate.

Only looking at the properties of the faceplate reveals the individual differences.

The example described below shows the principle difference between a control panel which was

- created as a "conventional" picture object.
- created as a faceplate.

The described task highlights the advantages of a faceplate.

1.5.1 Control panel created as "conventional” picture object

The task is to create a uniform control panel for 8pprox.. 20 fan drives. The template is the **picture object** available in Figure 1-2.

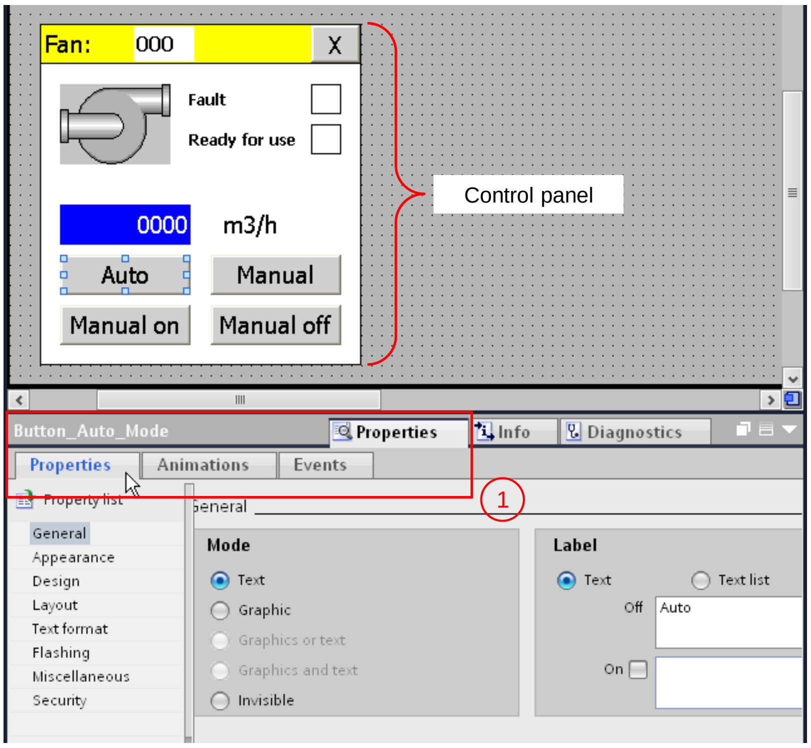
The picture object consists of a number of individual objects. For each used object, you can define "Properties”, "Animations” and "Events” (1).

The properties such as "Label”, "Layout” etc. are specified for each object.

In the "Events” or "Animations” tab, various system functions are configured at the parameterized objects such as "SetBit” etc.

Depending on the used system function (e.g. "SetBit”), they must be interconnected with an HMI tag.

Figure 1-2



Problem

- For each of the 20 control panels you need to adjust the HMI tags for the respective drive at the objects to which you have configured a system function.
  - This may easily lead to errors when selecting the tags for the individual drives.
  - The assignment is very time consuming.

- You have configured all control panels in the project. Your customer subsequently requires a further button in the control panel.
  - You need to call and adjust each of the 20 control panels individually. Highly time consuming.
- You have configured all control panels in the project. Your customer subsequently requests a further function or animation to be added.
  - You need to call and adjust each of the 20 control panels individually. Highly time consuming.
- Your customers use HMI operator panels with different display sizes.
  - You need to create control panels of different sizes in order to use them for the different display sizes. You need to adjust font and dimensions of the individual objects. Highly time consuming.

1.5.2 Control panel created as "faceplate”

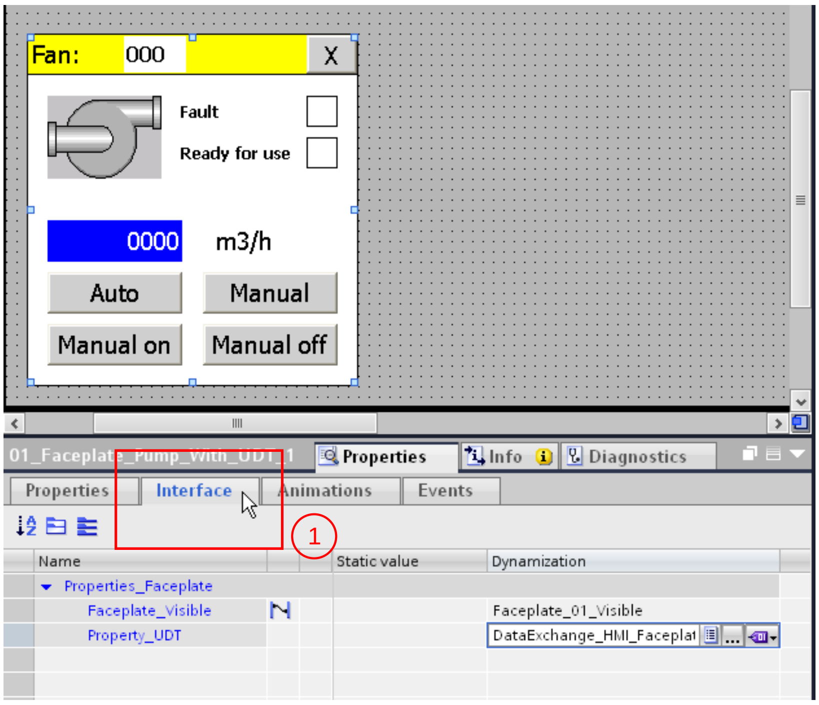
The task is to create a uniform control panel for 10pprox.. 20 fan drives. The template is the **faceplate** available in Figure 1-3.

All properties and events are defined in the faceplate editor and have been assigned to a user data type (HMI UDT). With a user data type (HMI UDT) you summarize a number of different tags.

The data exchange (1) is performed via the "Interface” tab.

Note The additional "Interface” tab identifies the "picture object” as a faceplate.

Figure 1-3



Name	Static value	Dynamization
Properties_Faceplate		
Faceplate_Visible		Faceplate_01_Visible
Property_UDT		DataExchange_HMI_Faceplat

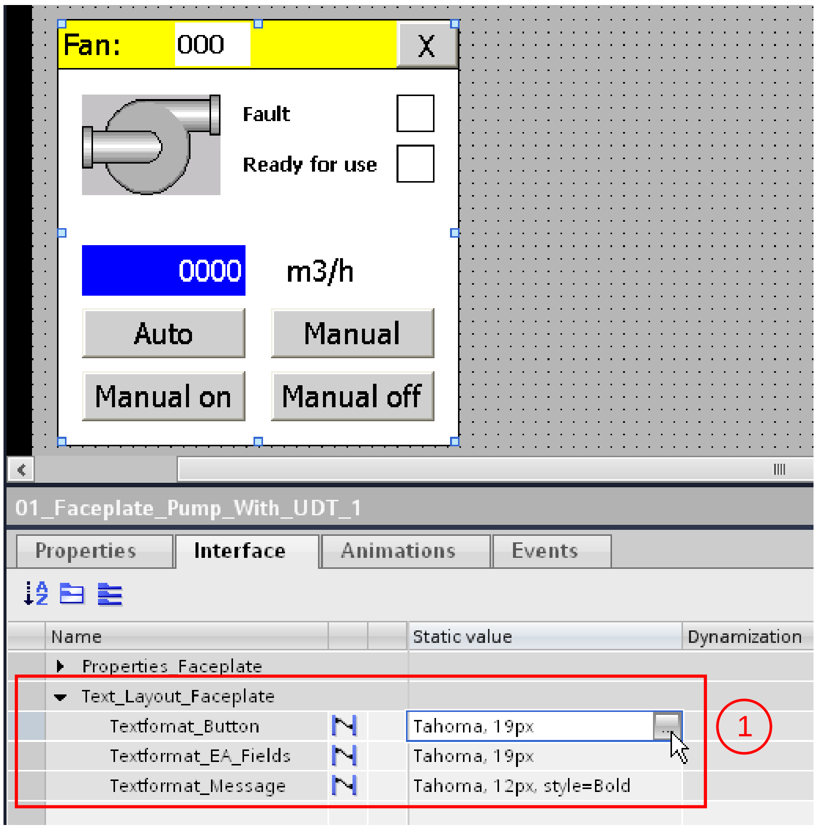
Advantages of using a faceplate

- Instead of many individual HMI tags only one tag per drive is required in this case. The used tag has one structure.
  - Due to the simple structure an error during tag assignment is less likely. Less time is consumed.
- You have configured all control panels in the project. Your customer subsequently requires a further button in the control panel.
  - You can change the used faceplate centrally and expand it by the button. The changes are subsequently adopted for all respective faceplates. The configuration workload is reduced.

1.5 Difference between picture object and faceplate?

- You have configured all control panels in the project. Your customer subsequently requests a further function or animation to be added.
  - You can change the used faceplate centrally and expand it by the function or animation. The changes are subsequently adopted for all respective faceplates. The configuration workload is reduced.
- Your customers use HMI operator panels with different display sizes.
  - When creating the faceplates you have the option to label the size of the "text format" or "layout", for example, as "parameter" (static tag). Later, you can specify the font size individually for all objects via the "Interface" tab (1). The faceplate can therefore be used for different display sizes. The configuration workload is reduced.

Figure 1-4



Name	Static value	Dynamization
▸ Properties_Faceplate		
▾ Text_Layout_Faceplate		
Textformat_Button	Tahoma, 19px	
Textformat_EA_Fields	Tahoma, 19px	
Textformat_Message	Tahoma, 12px, style=Bold	

Note Details for the configuration process are available in the document "Configuration Examples for HMI Faceplates". (68014632_Faceplates_EngineeringExamples_en).

1.6 What is an "HMI UDT” (user data type)?

Note As of WinCC (TIA Portal) V13 SP1 and when using an S7-1200 or S7-1500 controller, you do no longer need an "HMI UDT”.

More information is available in chapter 1.7 and 1.8.

User data types, abbreviated as "HMI UDT”, summarize a number of different tags which form a logical unit (in WinCC flexible 2008 formerly referred to as "structure”).
The advantage is generating a "group” of tags once, which you can then use in several faceplates.

Example:
The different states of a motor can be described with 5 individual tags, for example.

- "Off” state
- "On” state
- "Fault” state
- "Manual” state
- "Automatic mode” state

These tags (user data type elements) are summarized as a "group”. The functionality is the same as familiar from STEP 7 programming when creating a structure in a data block.

Details for creating a user data type are described in chapter 3.

How to create and open a user data type?

User data types are created and opened via the project library.

Respective details are described in chapter 3.

1.7 What is a PLC data type ("PLC UDT”)?

PLC data types, in short "PLC UDT”, are data structures defined by you that you can use several times in the program. The structure of a PLC data type is made up of several components. The type of component is specified when declaring the PLC data type.

The advantage is generating a "group” of tags once, which you can then use in several faceplates.

Example:
The different states of a motor can be described with 5 individual tags, for example.

- "Off” state
- "On” state
- "Fault” state
- "Manual” state
- "Automatic mode” state

These 5 tags are summarized as a "group”. The functionality is the same as familiar from STEP 7 programming when creating a structure in a data block.

Details for creating a PLC data type are described in chapter 4.

1.8 Difference between "HMI UDT” and "PLC UDT”

The main difference between an "HMI UDT” and a "PLC UDT” is that

- an "HMI UDT” can **only** be used in faceplates.
- the data structure of a "PLC UDT” needs only be created once. As opposed to the "HMI UDT”. Here, you need to store the same data structure exactly twice in the program.
- when changes are made in the respective data block, these must then be updated manually in the "HMI UDT”. This may cause errors when assigning the name/ tag type.
- "PLC UDT” handling is much easier in connection with faceplates.

1.9 Project library and global libraries

The created faceplates are stored and edited in the project library. If you wish to reuse the created faceplates across different projects, then save the completed faceplates in "Global libraries”.

Respective details are described in chapter 2.9.

2 Functions and Properties of the Faceplate

In this chapter, the functions and properties of a faceplate and the individual parameters of the faceplate editor are described.

If you are not yet familiar with the functions and properties, such as the parameters in the "faceplate editor", then thoroughly read this chapter.

2.1 Creating faceplates

There are two options for creating a faceplate.

- Creating a faceplate via the "pictures" editor.
- Creating a faceplate via the "faceplate editor".

The individual points are explained below using an example.

If you wish to recreate the chapter, then it is sufficient if you initially use only a button as an object, for example.

2.1.1 Creating a faceplate via the "pictures" editor

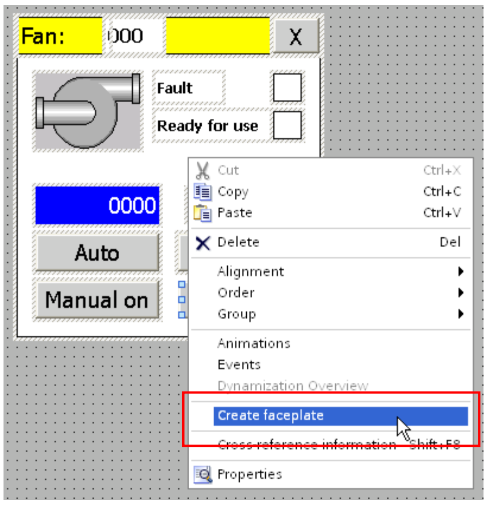
Creating a faceplate via the "pictures" editor is always an option if you have already created a completely configured "picture object", for example. You can access the already existing "picture objects".

In order to create a faceplate via the "pictures" editor, please proceed as follows.

- Generate a picture and add all required objects to this picture or call up an existing "picture object".
- Select all objects in the picture which you require for the faceplate.
- Right-click on one of the selected objects. A context menu opens.
- Select the "Create faceplate" item from the opened context menu.

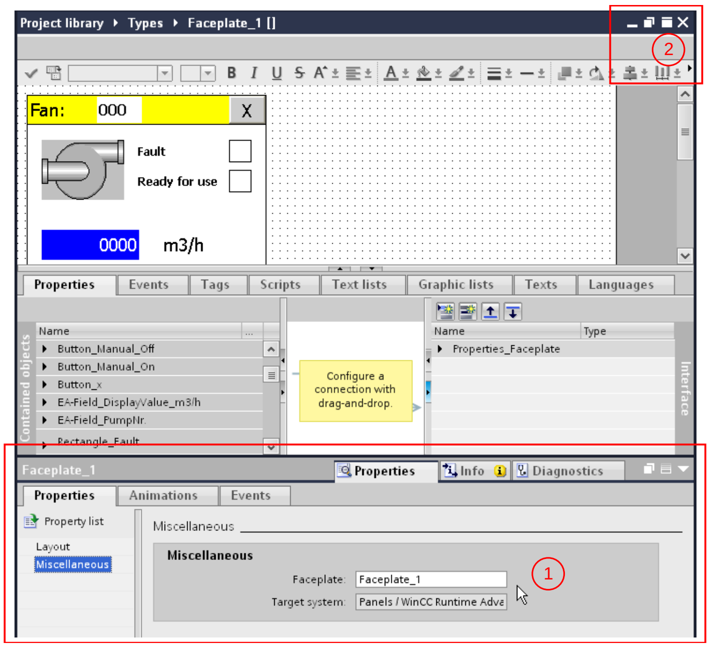
The following figure shows the context menu.

Figure 2-1



- The faceplate editor for creating / editing the faceplate opens.
  In the inspector window (1) at "Properties > Property list > Miscellaneous” you can adjust the name of the faceplate. "Target system” displays the type of Runtime for which the faceplate is suitable.

Figure 2-2



Faceplate: Faceplate_1

Target system: Panels / WinCC Runtime Adva

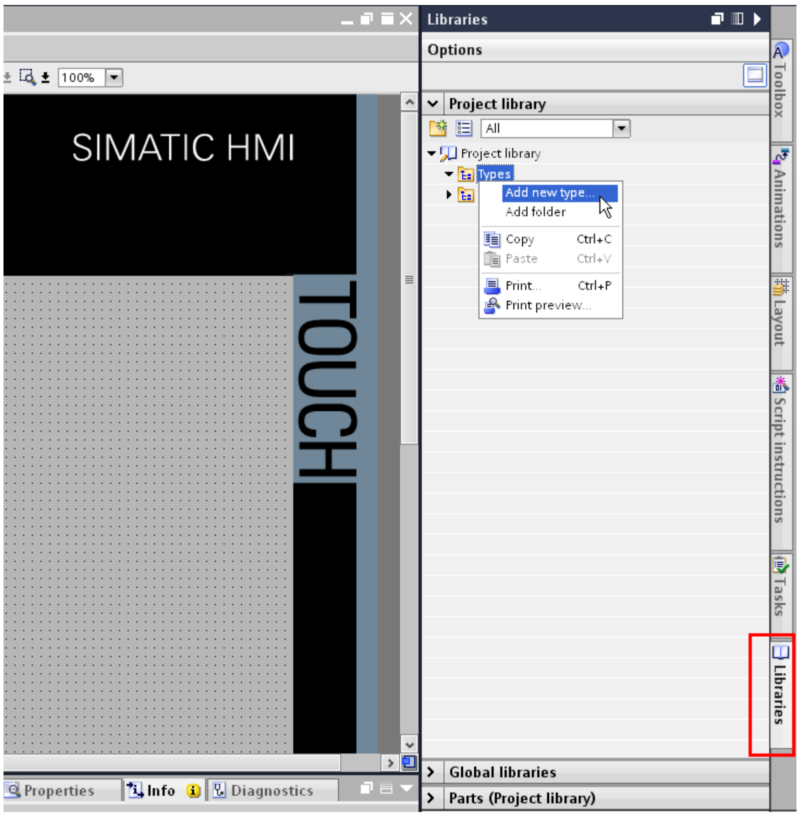
- First you close the created faceplate via the "Close” "**X**” button (2).

2.1.2 Creating a faceplate via the faceplate editor

In order to create a faceplate via the faceplate editor, please proceed as follows.

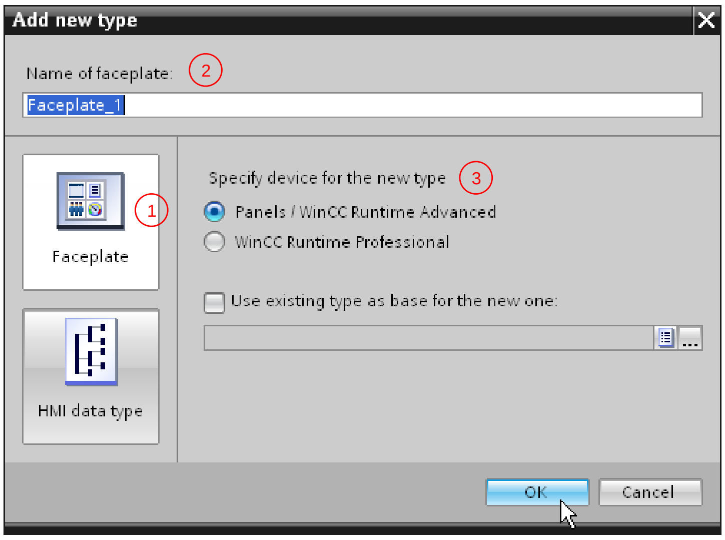
- Select the "Libraries" task card.
- Open the "Project library" palette.
- Right-click on the "Types" folder. A context menu opens.
- Select "Add new type…" in the context menu.
  The "Add new type…" window opens.

Figure 2-3



Display of the "Add new type" window.

Figure 2-4



Name of faceplate: Faceplate_1

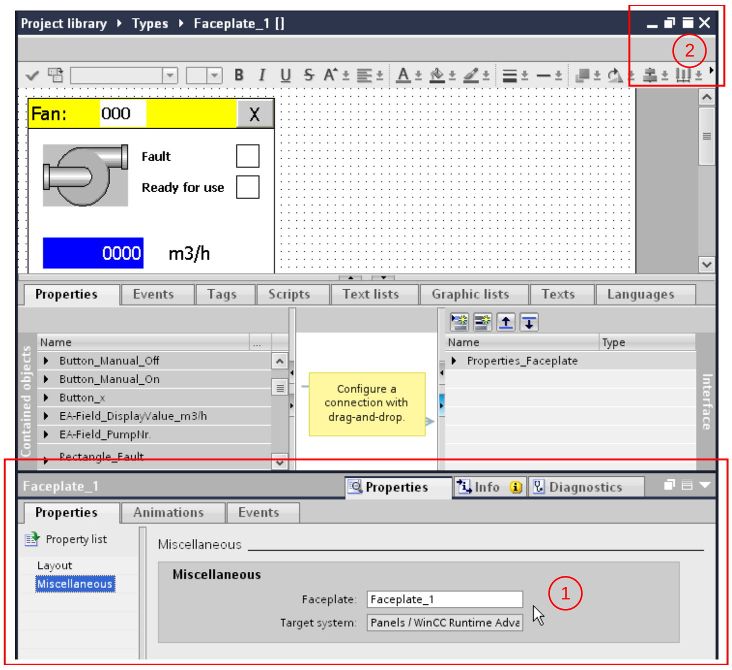
Specify device for the new type
[✓ Panels / WinCC Runtime Advanced]
WinCC Runtime Professional

Use existing type as base for the new one:

- On the left side, click on the "HMI faceplate" button (1).
- Assign a unique name to the faceplate (2).
- In "Specify device for the new type" you activate the radio button "Panels / WinCC Runtime Advanced" (3).

- Confirm the entries with "OK".
  The faceplate editor for creating / editing the faceplate opens.
- Now add all objects you require into the "workspace" of the picture. The procedure is the same as in the "pictures" editor.

Figure 2-5



Faceplate: Faceplate_1
Target system: Panels / WinCC Runtime Adva

- First you close the created faceplate via the "Close" "**X**" button (2).

2.2 Opening a faceplate

There are two options for opening a faceplate.

- Open the faceplate via the "pictures" editor
- Open the faceplate via the "Project library".

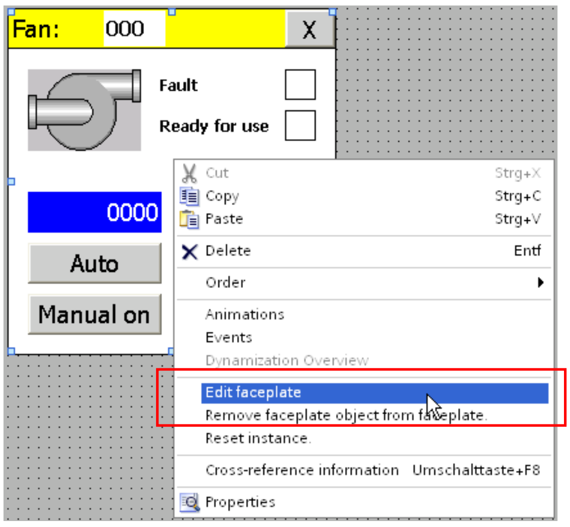
The individual points are explained below using an example.

2.2.1 Open the faceplate in the "pictures" editor

To open an existing faceplate via the "pictures" editor, please proceed as follows.

- Call the picture which contains the faceplate.
- Right-click on the faceplate. A context menu opens.
- Select the "Edit faceplate" item from the opened context menu.
  The faceplate editor for creating / editing the faceplate opens.

Figure 2-6

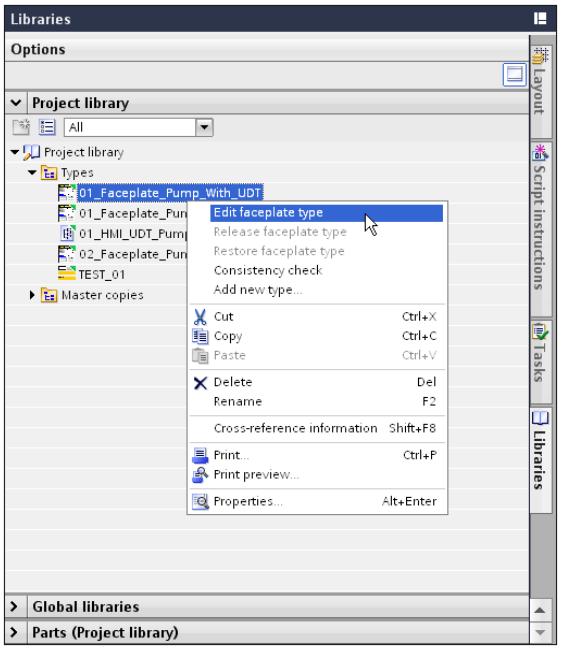


2.2.2 Opening the faceplate via the Project library

To open an existing faceplate from the Project library, please proceed as follows:

- Select the "Libraries” task card.
- Open the "Project library” palette.
- In the "Types” folder you use the right mouse button to select the faceplate you wish to open. A context menu opens.
- Select "Edit faceplate type” in the context menu.
  The faceplate editor for creating / editing the faceplate opens.

Figure 2-7



2.3 Releasing or restoring the faceplate

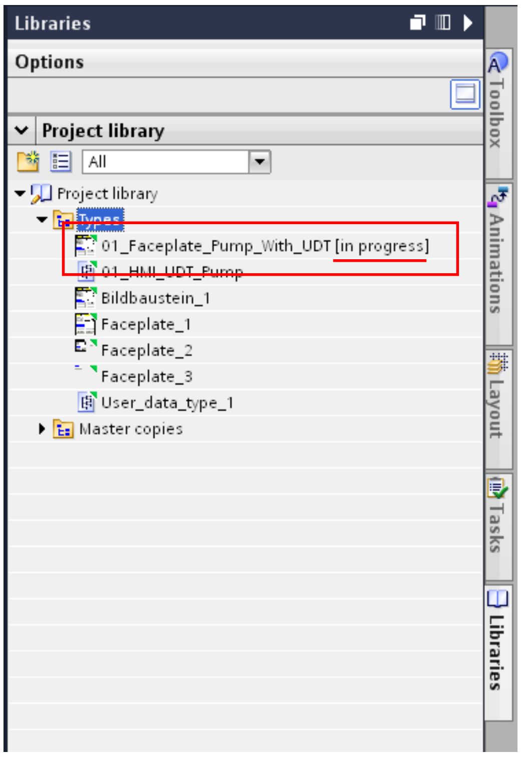
Faceplate status

When editing a faceplate, the following entry "Name of the faceplate **[in progress]**" appears next to the name of the faceplate.

In the header of the faceplate editor you can also see the current status of the opened faceplate.

The "status information" [in progress] shows you that the faceplate on hand was edited and this modification has not yet been released. For this purpose, please refer to chapter 2.3.2.

Figure 2-8



What happens with the faceplates used in the project if the respective faceplate is currently edited?

When editing a faceplate used in the project, then the modifications performed at this faceplate only become effective after the respective faceplate has been explicitly released (see chapter 2.3.1).

You can, for example, interrupt editing a faceplate and save and close the configuration. After opening the configuration, you can continue editing the faceplate. The status of the edited faceplate is then always still "[in progress]".

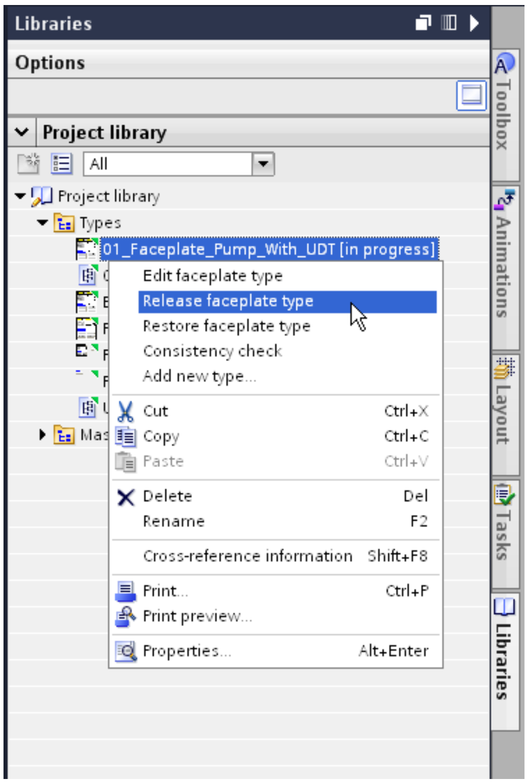
2.3.1 Releasing the faceplate

When the faceplate is released, it receives a version designation (see chapter 2.3.3). After the release, all faceplates of this type are updated and point to the new version of the faceplate. The old version is deleted.

To release a faceplate, please proceed as follows:

- Select the "Libraries" task card.
- Open the "Project library" palette.
- In the "Types" folder you use the right mouse button to select the faceplate you wish to release. A context menu opens.
- Select "Release faceplate type" in the context menu.
  The faceplate is updated.

Figure 2-9



Example:
Using a faceplate in several pictures. Subsequently, you edit this faceplate and add a further button. As long as you don't "release" the edited faceplate, the faceplates existing in the pictures will not change. The performed modification only becomes effective after "releasing" the edited faceplate. The faceplate used in the pictures now contains the added button.

2.3.2 Restoring the faceplate

You can, for example, interrupt editing a faceplate and save and close the configuration. After opening the configuration, you can continue editing the faceplate.

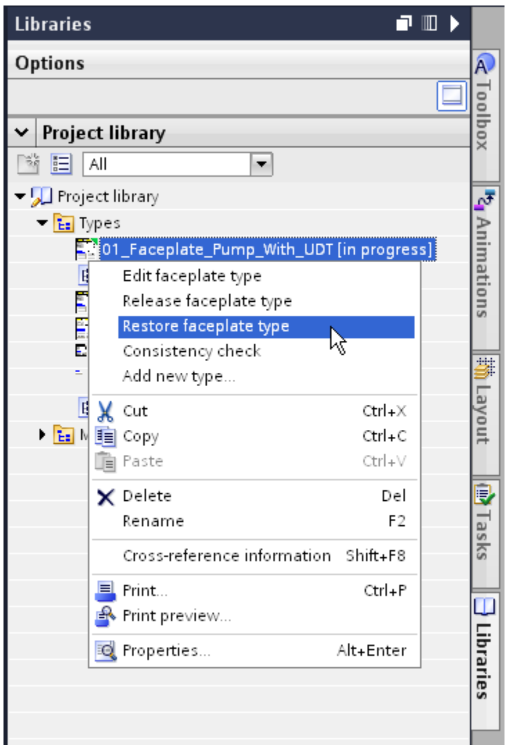
If you now wish to restore the "old" status of the faceplate, please use the "Restore faceplate type" function.

All modifications since the last release at the faceplate will be rejected. The faceplate will then be released again.

To restore a faceplate, please proceed as follows:

- Select the "Libraries" task card.
- Open the "Project library" palette.
- In the "Types" folder you use the right mouse button to select the faceplate you wish to restore. A context menu opens.
- Select "Restore faceplate type" from the context menu.
  The modifications are rejected. The faceplate is released again.

Figure 2-10



Example:
You perform the modifications at a faceplate. You notice that you have edited the "wrong" faceplate. To restore the "old" status, please use the "Restore faceplate type" function. The modifications are rejected. The respective faceplate in the pictures will not be modified.

2.3.3 Version designation of a faceplate

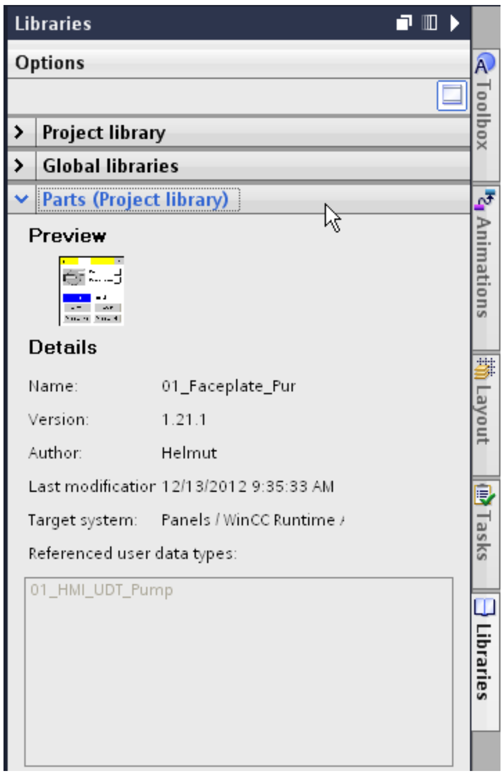
Upon releasing a faceplate, the faceplate receives a version designation. After the release, all faceplates of this type are updated and point to the new version of the faceplate. The old version is deleted.

The current version of a faceplate can be displayed as follows.

Requirement:
The faceplate is opened in the faceplate editor.

- Select the "Libraries” task card.
- Open the "Parts (Project library)” palette.
- The data of the faceplate currently being edited are displayed.

Figure 2-11



Name:	01_Faceplate_Pur
Version:	1.21.1
Author:	Helmut
Last modificatior	12/13/2012 9:35:33 AM
Target system:	Panels / WinCC Runtime /

Referenced user data types:

01_HMI_UDT_Pump

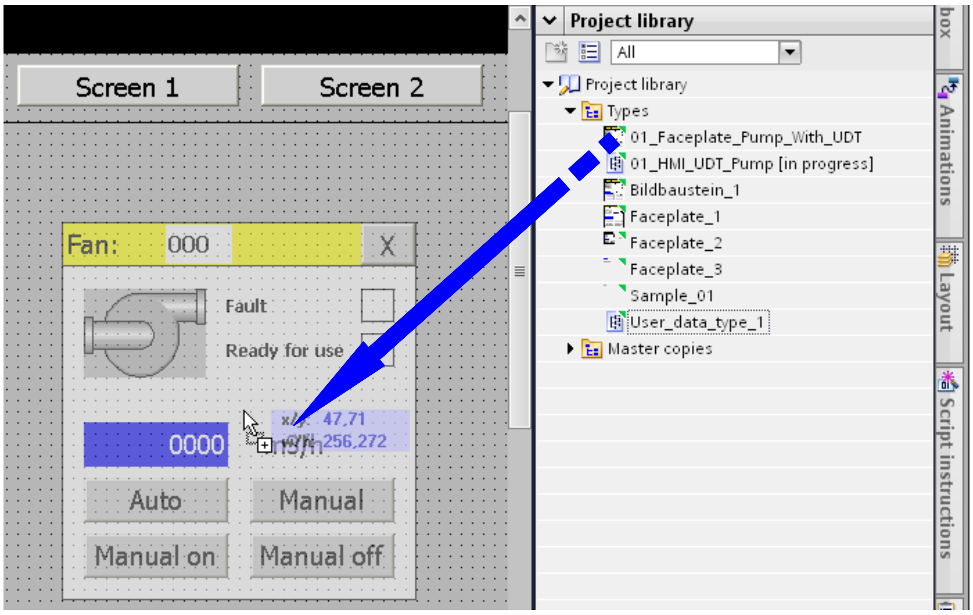
2.4 Calling a faceplate or inserting it into a picture

Faceplates can be called via the "Libraries” task card and be inserted into a picture.

To add a faceplate to a picture, please proceed as follows:

- Select the "Libraries” task card.
- Open the "Project library” palette.
- Open the "Types” folder.
  The available faceplates are displayed in the "Types” folder.
- Select the desired faceplate and drag it to the HMI screen via "drag&drop”.

Figure 2-12



2.5 Faceplate editor

The subsequent figure shows the faceplate editor in which you can create and edit the faceplate.

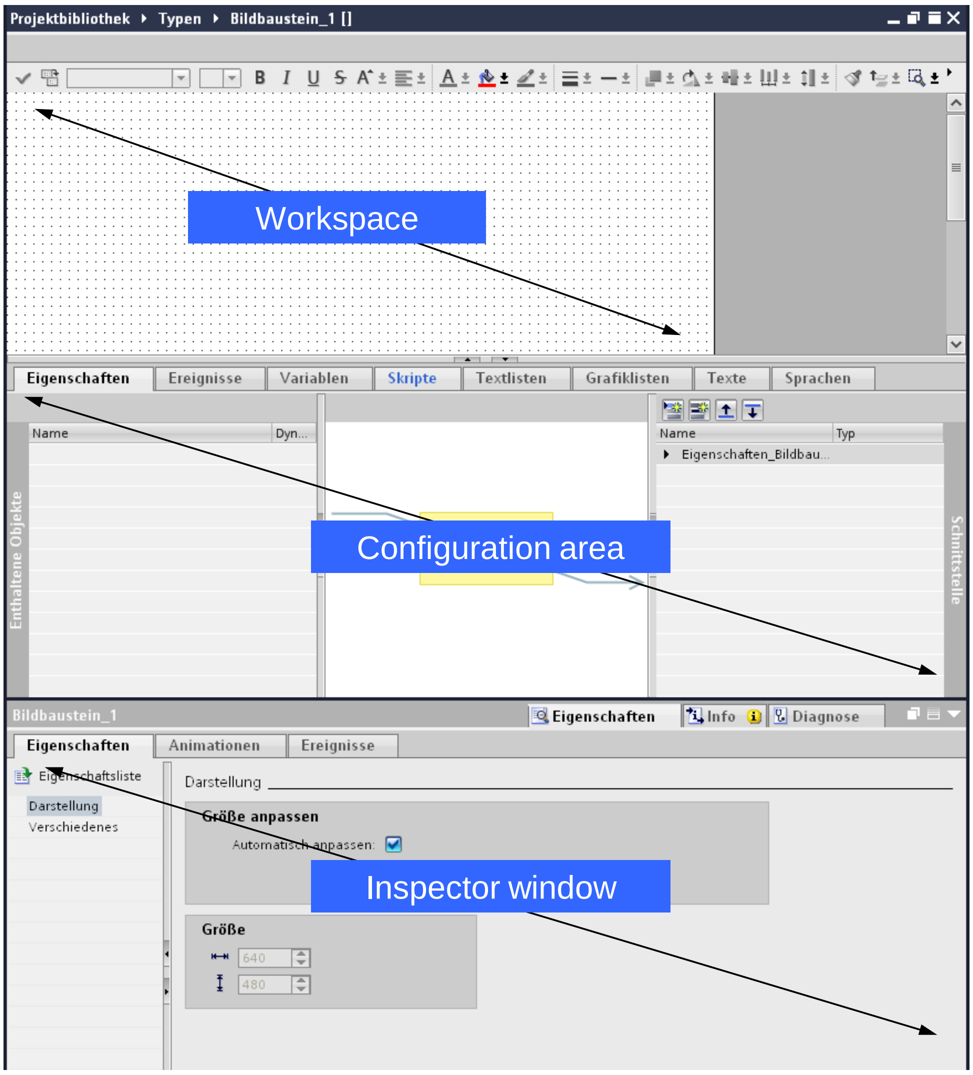
In the further course of the application, the individual areas such as

- workspace
- configuration area
- inspector window

are explained in greater detail.

Several examples illustrate the effects of the configured function on the created faceplate.

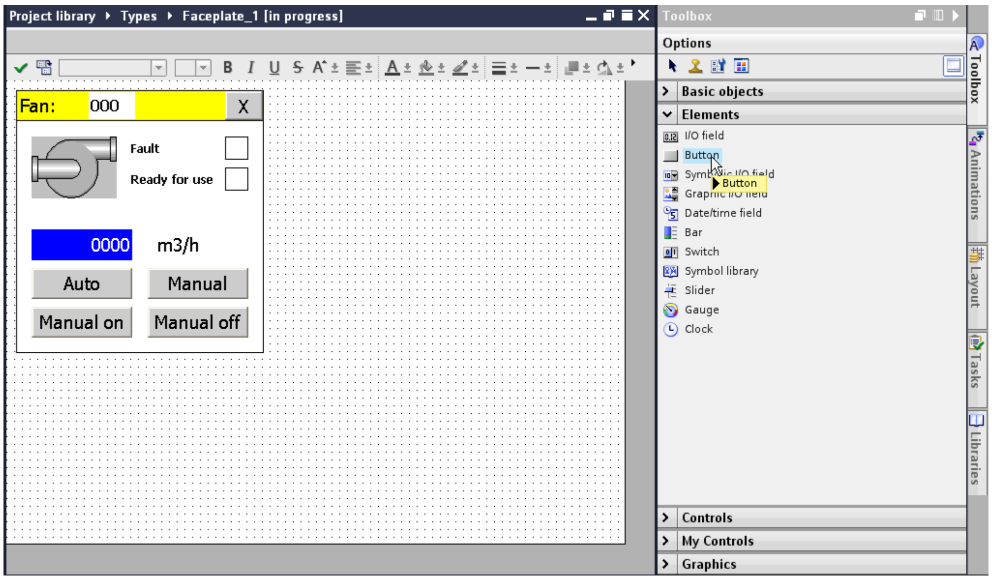
Figure 2-13



2.5.1 Workspace

In the workspace area you place the objects for the faceplate as familiar from the "pictures" editor. You remove the objects or add new objects into the workspace from the "Tools" task card.

Figure 2-14



Tip:

After you have inserted an object, you assign a name to the object which enables identifying the object easily. This procedure facilitates the work for the subsequent configuration.

Example:

You have inserted a button. The system automatically assigns a name to this button (e.g. "Button_1").

Change the name e.g. from "Button_1" to "Button_Automatic_Mode".

In the "Inspector window" you change the properties of the button. In the "Properties > Property list > Miscellaneous" tab you can adjust the name of the object.

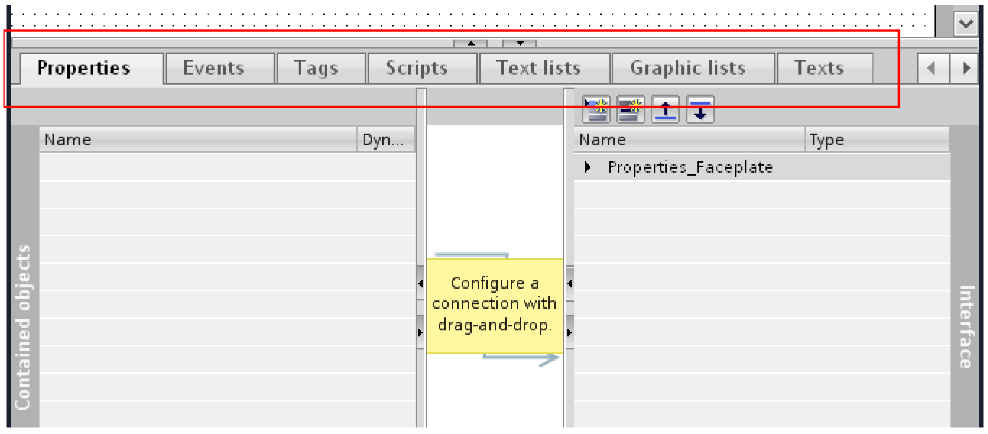
In the further course of this application you will see what this procedure is useful for.

2.5.2 Configuration area

The configuration area contains various tabs.

Details on the individual tabs are described in chapter 2.7.

Figure 2-15



2.5.3 Inspector window

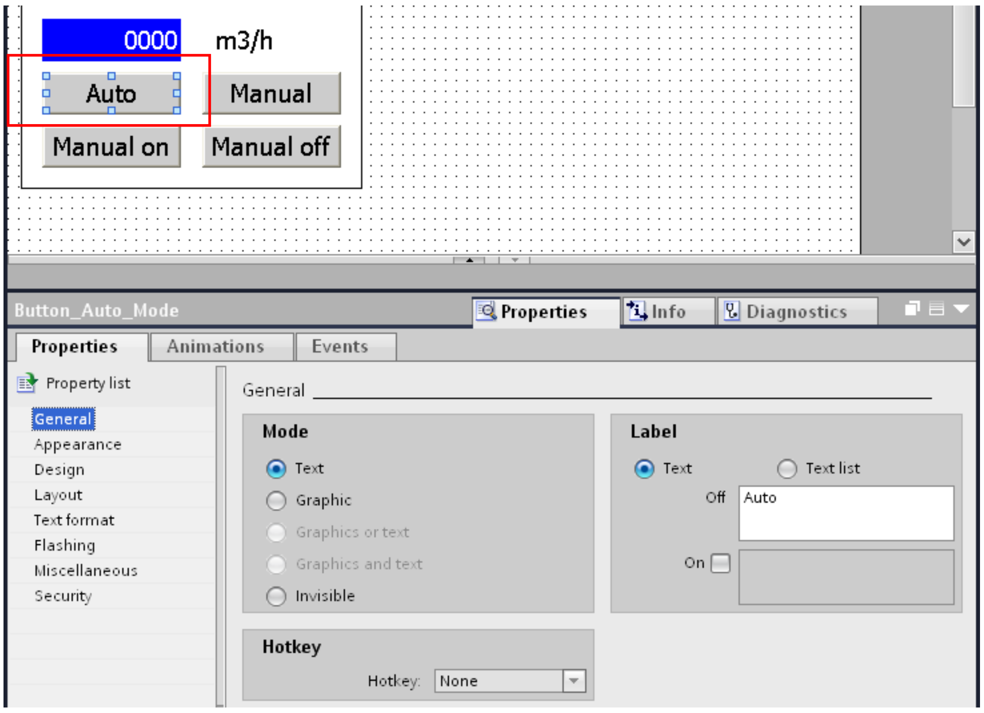
The properties of the selected object are displayed in the inspector window. It corresponds to the same view as familiar from the "pictures” editor.

Properties, animations and events which you configure here in the faceplate at the objects, cannot be edited in the "pictures” editor.

As opposed to the "Configuration area” (see previous chapter) where the given properties, animations and events could be transferred to the faceplate as "Parameter” or be modified afterwards.

Settings in the inspector window are always advisable if, for example, you wish to assign a fixed text, layout or function to an object.

Figure 2-16



Example 1 (assigning a text):
You have configured a button for the "automatic mode”. The displayed text at the button shall always be called "Auto”.
In this case, you specify the name of the button in the inspector window at the "Properties > Property list > Miscellaneous”.

- **Result:**
  The "Auto” text is firmly assigned to the button and cannot be changed later on at the faceplate in the "Interface” tab.
- **Alternative:**
  Alternatively, you have the option to configure the name of the button in the configuration area as "static” property. In the "Interface” tab of the faceplate you can later on adjust the name individually.

Example 2 (assigning a function):
You have configured a button for the "automatic mode”.
In the "Press” property you have assigned the "SetBit” function in the inspector window.
The respective tag has been configured as "Parameter” in the configuration window.

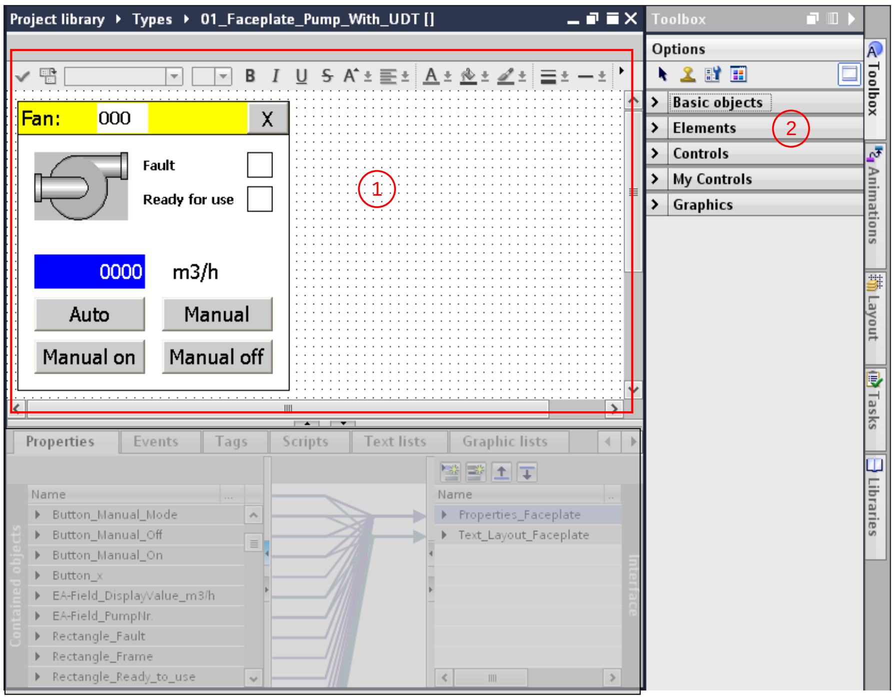
- **Result:**
  When "pressing” the button, the "SetBit” function is executed. The function cannot be changed later on in the "Interface” tab at the faceplate.
  The respective tag, however, can be assigned individually to the used function.
- **Alternative:**
  Alternatively, in the configuration area you have the option to configure the "Press” event to the button, for example.
  In the "Interface” tab of the faceplate you can assign a system function individually to the "Press” event. You are therefore not tied to a fixed function, such as "SetBit”.

2.6 Parameters in the workspace

The workspace of the faceplate editor contains the same parameters and settings options as familiar from the "pictures” editor (1).

First of all, you need the "Toolbox” task card and the contained "Basic objects” and "Elements” palettes (2).

Figure 2-17

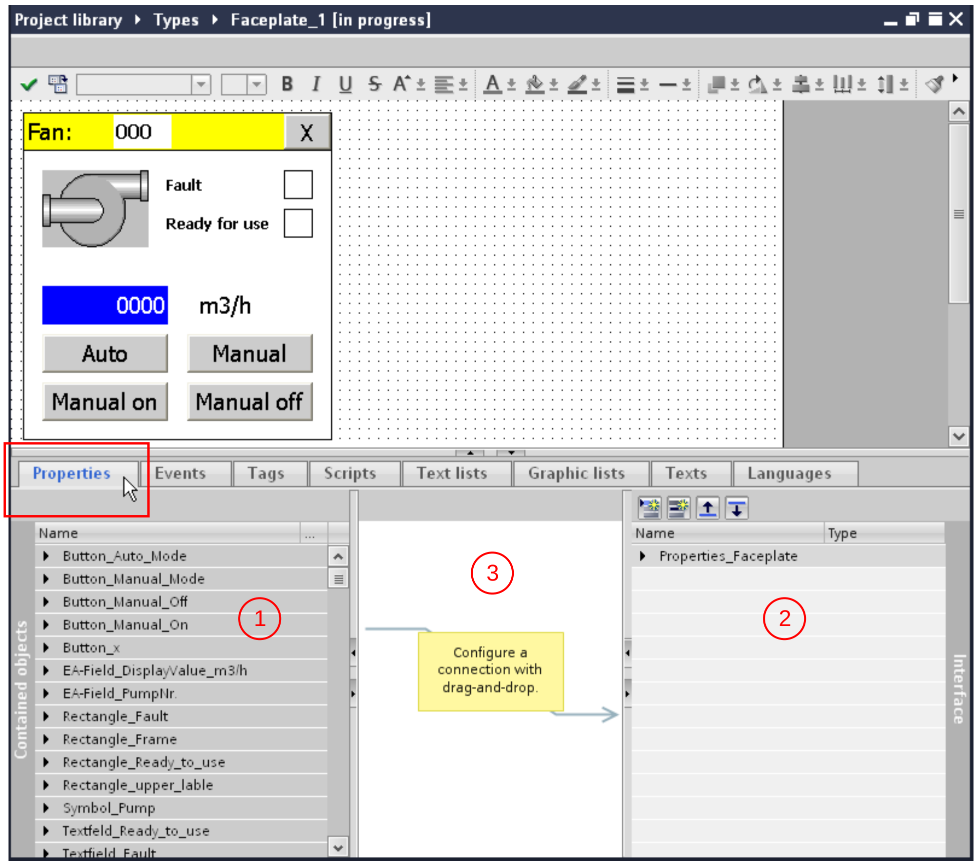


2.7 Parameters in the configuration area

Below, the individual tabs in the configuration area are described.

2.7.1 "Properties” tab

Figure 2-18



The "Properties” tab contains two lists as well as a "graphic area".

- "Contained objects” list (1)
- "Interface” list 2
- "Graphic” area (3)

"Contained objects" list (1)

The "Contained objects" list contains all objects located within the workspace.

Each of these objects has object-specific properties, which are already familiar from the "pictures" editor (a button has different properties than, for example, a rectangle).

You can use the symbol next to the objects for "opening/closing" the property to navigate through the properties of the objects.

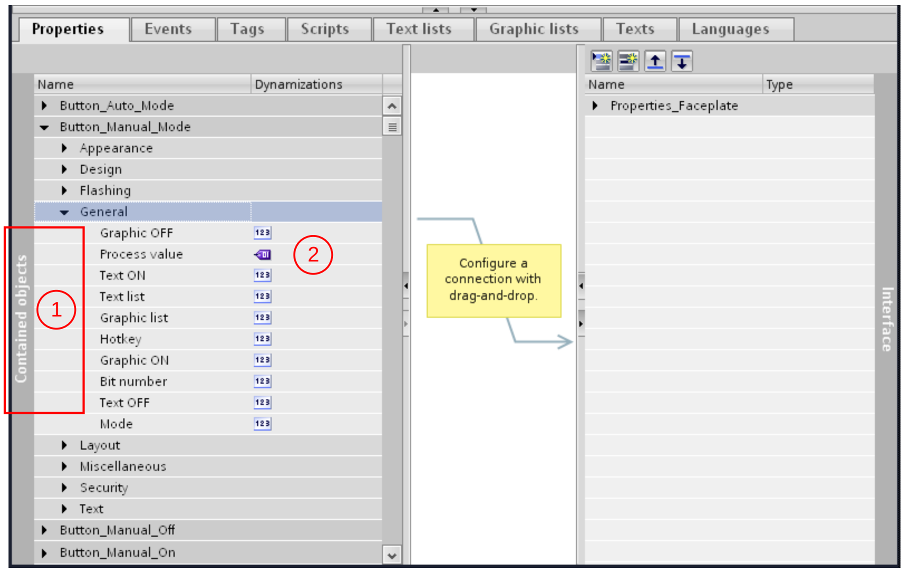
Properties of the objects

In the "Dynamizations" column (2) you can see from the symbols whether the available property has a "static" or a "dynamic" property.

- Dynamic properties (apparent from the symbol):
  Values can later be assigned to the faceplate as tags or via a text or graphics list.

- Static properties (apparent from the symbol):
  Values are later assigned to the faceplate as a constant value. Exception: In a trend view, this specifies the trends to be configured.

Figure 2-19

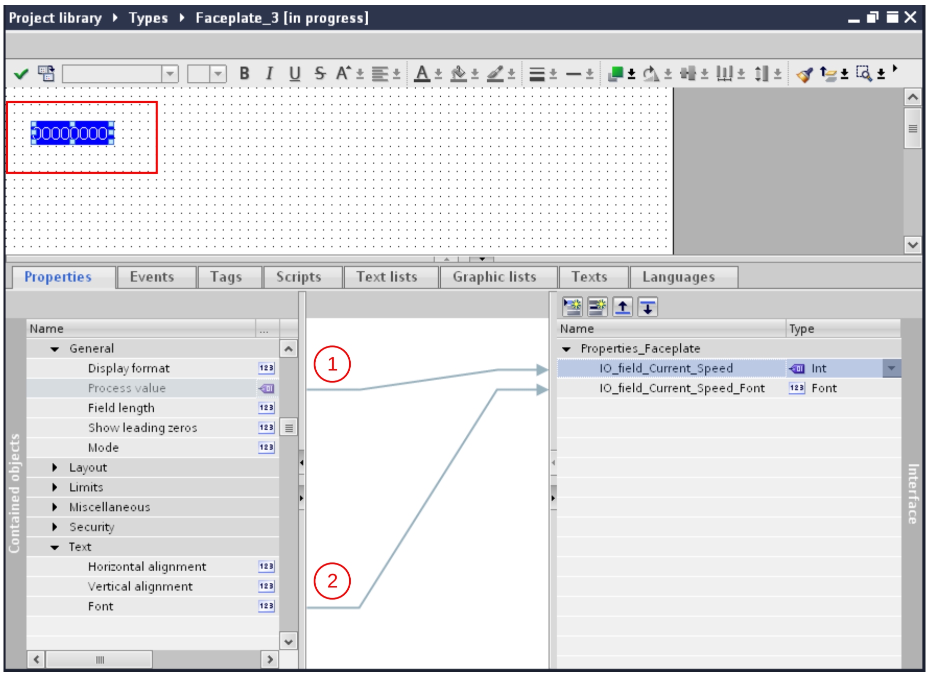


The following example shall explain this:

Example: I/O field in a faceplate
The "Process value" property has a "dynamic" property (1).
Through the configuration, a tag can be assigned to the IO field at the faceplate in the "Interface" tab.

The "Font" property has a "static" property (2).
Through the configuration, a font size can be individually assigned to the IO field at the faceplate in the "Interface" tab.

Figure 2-20

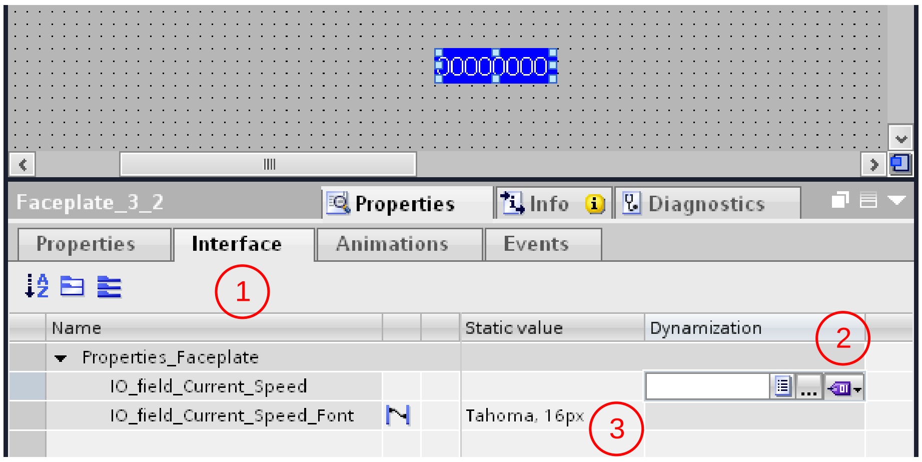


Name
General
Display format
Process value
Field length
Show leading zeros
Mode
Layout
Limits
Miscellaneous
Security
Text
Horizontal alignment
Vertical alignment
Font

Name	Type
Properties_Faceplate	
IO_field_Current_Speed	Int
IO_field_Current_Speed_Font	Font

The figure below shows the created faceplate (in this case only an IO field), which was called in the "pictures" editor.

Figure 2-21



Name		Static value	Dynamization
Properties_Faceplate			
IO_field_Current_Speed			
IO_field_Current_Speed_Font		Tahoma, 16px	

In the "Interface" tab (1) you see the two configured properties of the faceplate. A tag can be assigned to the "dynamic" interface (2). Only a fixed value can be assigned to the "static" interface (3).

"Interface” list

A faceplate is initially a self-sufficient "picture object”. From "outside” there is no access to the display and control objects used in the faceplate.

Selected properties, animations and events of the used display and control objects are listed in the "Interface” list. The configuration of the faceplate then occurs via these fixed properties and tags in the "pictures” editor.

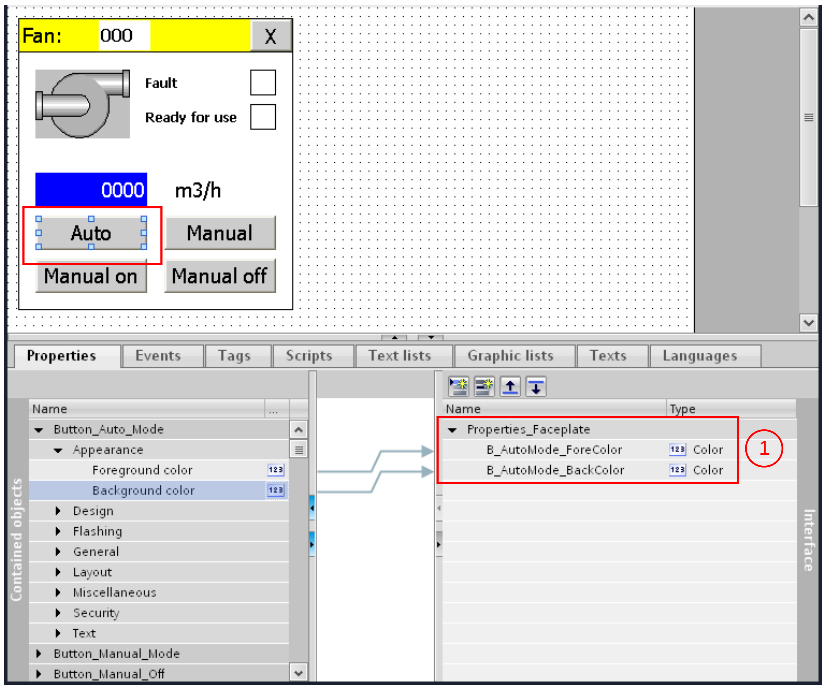
You can assign **one** property to **one** generated property or tag in the "Interface” list.
However, it is also possible to assign **several** properties to **one** generated property or tag in the "Interface” list.

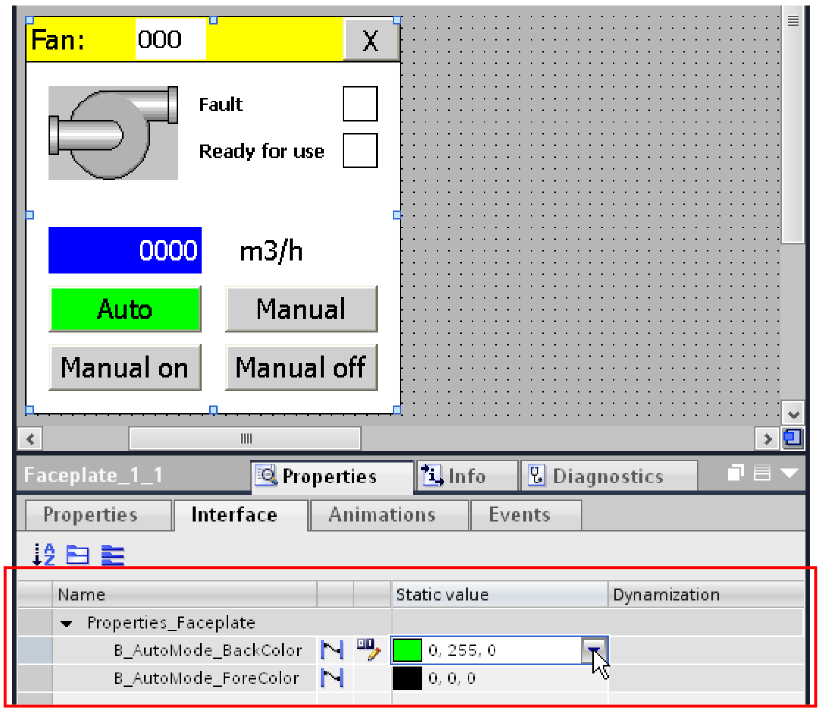
The examples below shall illustrate the interface functionality.

Example 1: button, configurable color

The foreground and background color of the "Auto” button shall be configurable.

Table 22-23

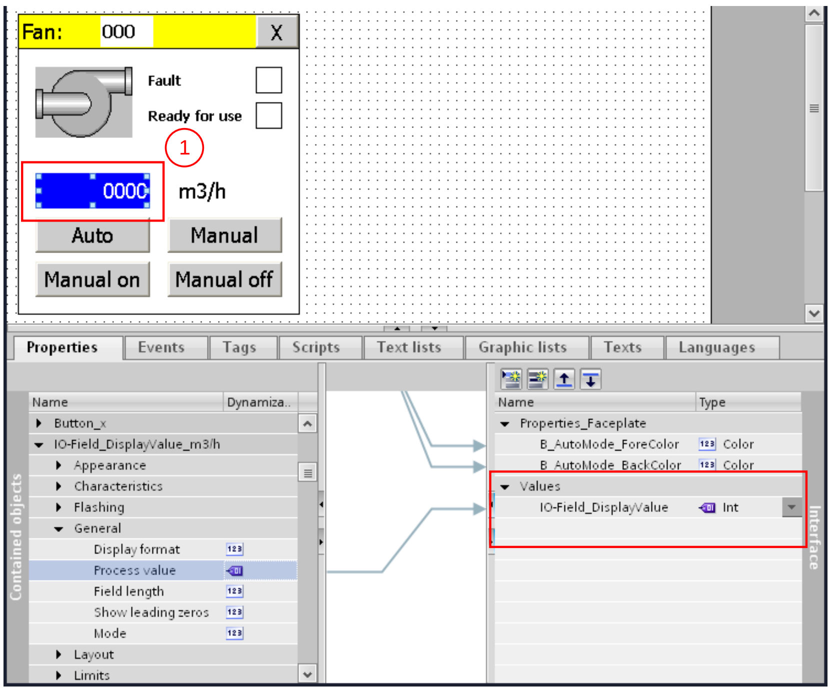
No.	Action
1.	**Configuration view:** From the "Contained objects” list, the properties selected from the "Button_Auto_Mode” button are connected with the "Properties” in the "Interface” list via drag&drop (1). The data type is given automatically. The "Properties” in the "Interface” list were created beforehand. 

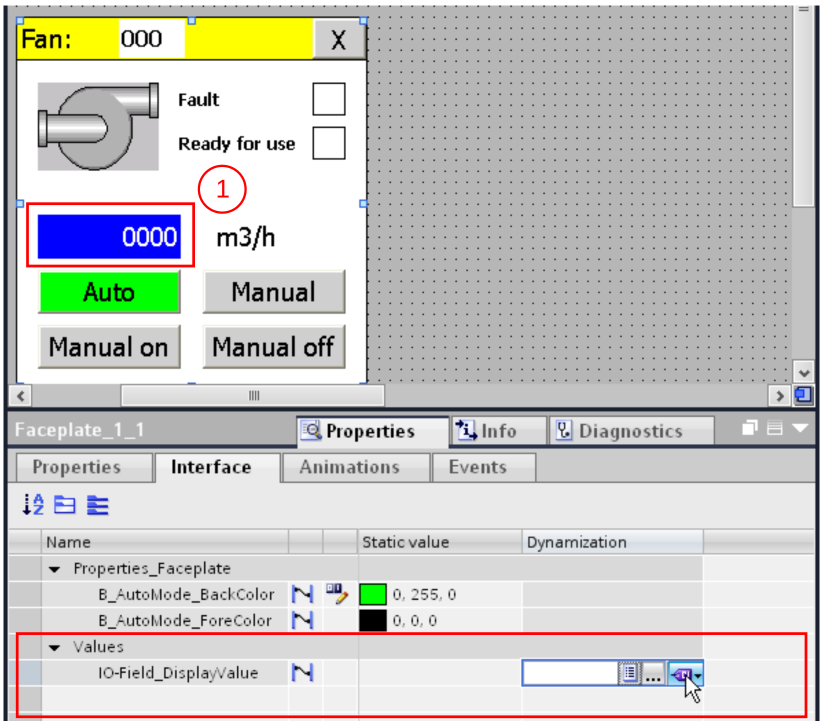
No.	Action
2.	**View of the faceplate in the "pictures" editor:** The color for the "Auto" button can be assigned via the two configured properties in the "Interface" tab.  Fan: 000 X Fault Ready for use 0000 m3/h Auto Manual Manual on Manual off Faceplate_1_1 Properties Info Diagnostics Properties Interface Animations Events Name | Static value | Dynamization Properties_Faceplate B_AutoMode_BackColor | 🟩 0, 255, 0 B_AutoMode_ForeColor | ■ 0, 0, 0

Example 2: IO field, tag assignment

The tag of the IO field for the output of the current value for the flow rate shall be configurable (1).

Table 24-1

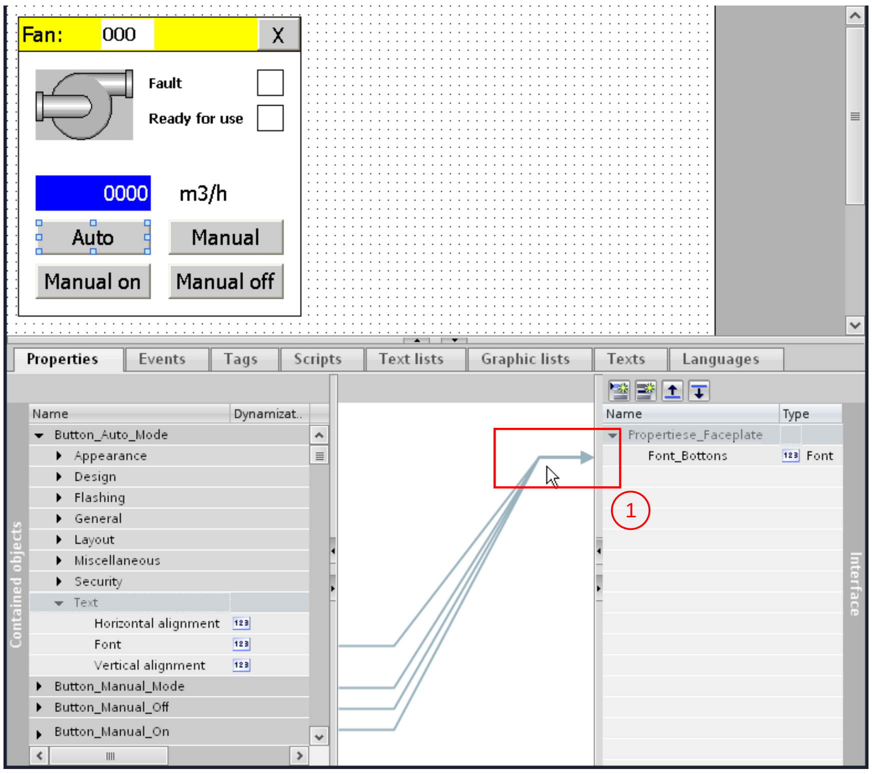
No.	Action
1.	**Configuration view:** From the "Contained objects" list, the selected property from IO field "IO-Field_DisplayValue_m3/h" is connected with the "Tag" created in the "Interface" list via drag&drop. The data type must be assigned / adjusted accordingly. The "Tag" in the "Interface" list was created beforehand. 

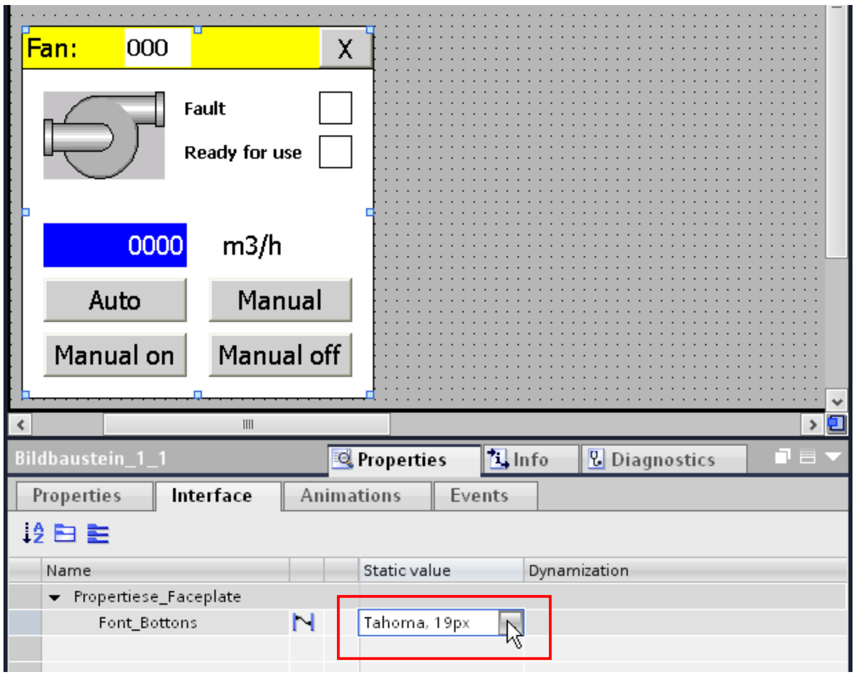
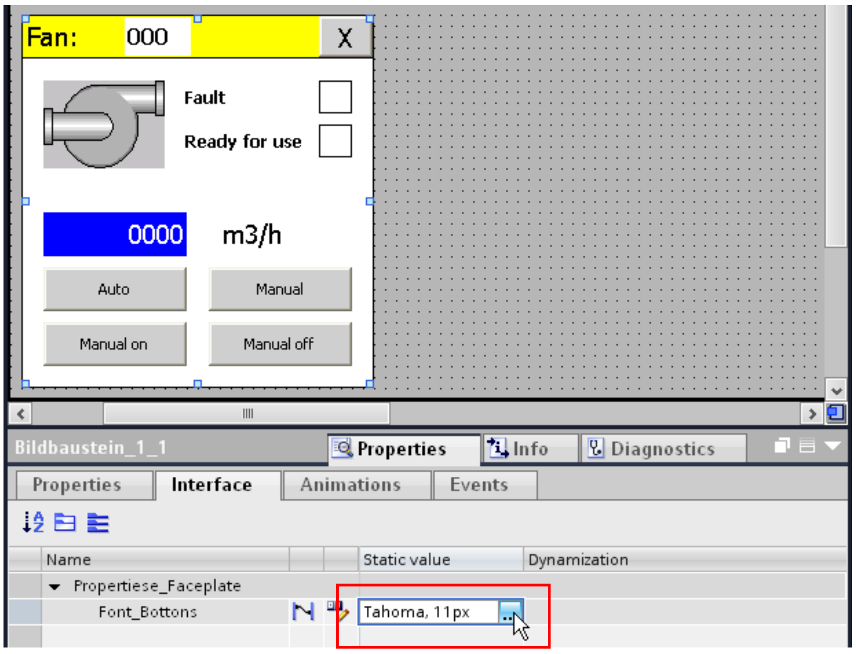
No.	Action
2.	**View of the faceplate in the "pictures" editor:** An HMI tag can be assigned to the IO field for the output of the current flow rate via the "IO-Field_DisplayValue" tag in the "Interface" tab (1).  Fan: 000 X Fault Ready for use 1 0000 m3/h Auto Manual Manual on Manual off Faceplate_1_1 Properties Info Diagnostics Properties Interface Animations Events Name / Static value / Dynamization Properties_Faceplate B_AutoMode_BackColor: 🟩 0, 255, 0 B_AutoMode_ForeColor: ■ 0, 0, 0 Values IO-Field_DisplayValue

Example 3: multiple assignment

The font of the four buttons shall be configurable.

Table 2-2

No.	Action
1.	**Configuration view:** From the "Contained objects" list, the selected properties of the four buttons are connected with **one** generated "Properties" in the "Interface" list via drag&drop (1). The data type is given automatically. The "Property" in the "Interface" list was created beforehand.  Fan: 000 X Fault Ready for use 0000 m3/h Auto Manual Manual on Manual off Properties Events Tags Scripts Text lists Graphic lists Texts Languages Name Dynamizat.. Button_Auto_Mode Appearance Design Flashing General Layout Miscellaneous Security Text Horizontal alignment Font Vertical alignment Button_Manual_Mode Button_Manual_Off Button_Manual_On Contained objects Name Type Propertiese_Faceplate Font_Bottons Font 1 Interface

No.	Action
2.	**View of the faceplate in the "pictures" editor:** In the "Interface" tab, the font for the buttons can be assigned via "Font_Buttons" in "Properties". In this example: "Tahoma **19px**"  Bildbaustein_1_1 — Properties / Interface / Animations / Events Name: Propertiese_Faceplate > Font_Bottons — Static value: Tahoma, 19px
3.	For comparison. In this case, the font "Tahoma **11px**" was selected.  Bildbaustein_1_1 — Properties / Interface / Animations / Events Name: Propertiese_Faceplate > Font_Bottons — Static value: Tahoma, 11px

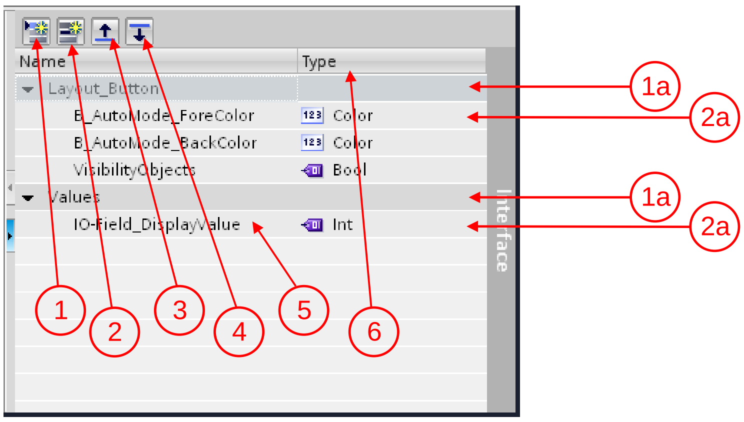
Defining the properties in the "Interface" list:

The figure below shows the "Interface" list and the respective parameters.

General:
Depending on the added event and the resulting "Type", the "Interface" list refers to a

- Property
  - Property (apparent from the symbol). Values are later assigned to the faceplate as a fixed value.
- Tags
  - Tag (apparent from the symbol). Values can later be assigned to the faceplate as tags or via a text or graphics list.

Figure 2-25



Name	Type
Layout_Button	
B_AutoMode_ForeColor	Color
B_AutoMode_BackColor	Color
VisibilityObjects	Bool
Values	
IO-Field_DisplayValue	Int

Adjusting a name:
If you wish to edit the name of a category, property or tag, then right-click on the name to be changed. A dialog box opens. In the dialog field, you select the option you wish to perform.

"Add category" (1)

- In order to create a new category, click on the "Add category" button (1). This function gives you the option to structure the used properties or tags according to "topics" (1a). If necessary, adjust the name accordingly. The name is displayed at the faceplate in the "Interface" tab.

"Add property or tag" (2)

- In order to create a new property or tag, click on the "Add property" button (2). The name is displayed at the faceplate in the "Interface" tab.
  Tip: assign a name via which you can easily recognize the given property or tag (2a).
  Example: instead of "Button_01", rather use "Button_AutoMode".

Adjusting the sequence (3), (4)

- The buttons enable you to adjust the sequence of the given categories, properties and tags afterwards.

Name (5)

- Name of the added category, property or tag.

Type (6)

- Use it to select the data type.

  Note:
  If you connect "static values" from the "Contained objects" list with the "Interface" list via "drag&drop", the data type is assigned automatically. If you assign the "Properties" from the inspector window of the "Interface" list, you need to specify the respective data type and adjust it to your specification.

"Graphic" area

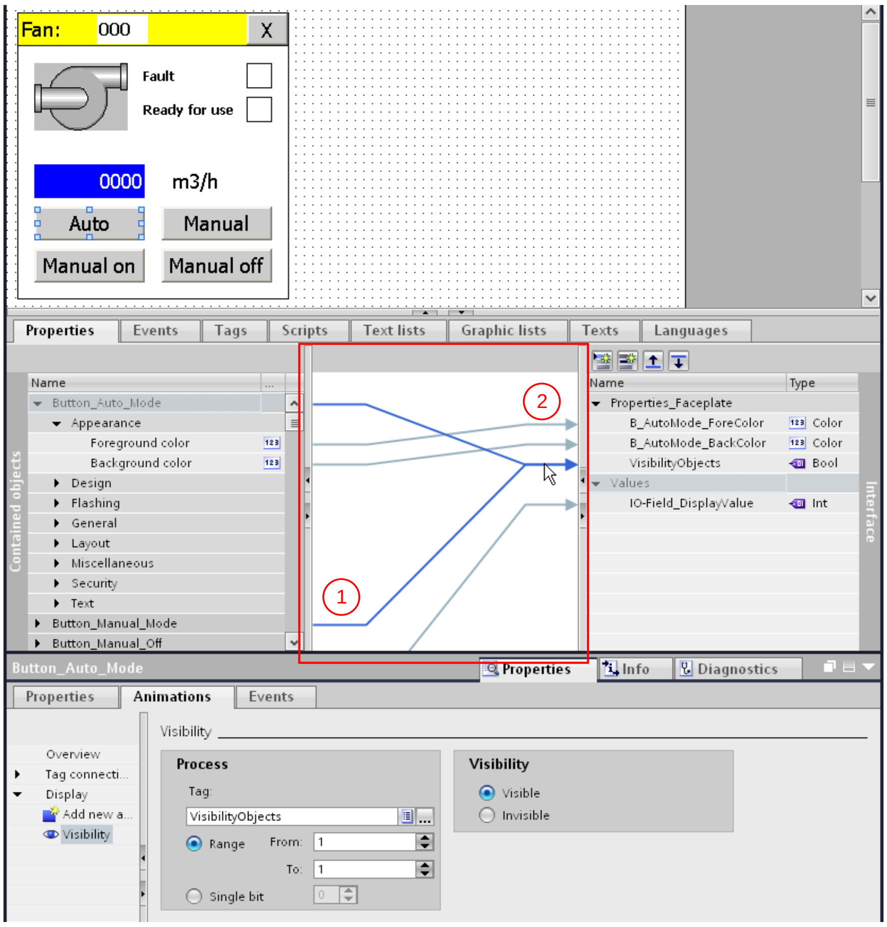
The graphic area shows the configured connections between the "Contained objects" list and the "Interface" list.
Furthermore, function assignments made between the inspector window and the "Interface" list are displayed.

Clicking a line enables you to see which properties are connected with each other. This facilitates the assignment among one another, in particular for extensive assignments.

Example:
The "Visibility" function is configured in the inspector window of the "Manual" and "Auto" buttons. The respective tag is stored in the "Interface" list. This assignment automatically allocates the graphic lines directly to the "Name" of the interface (1).

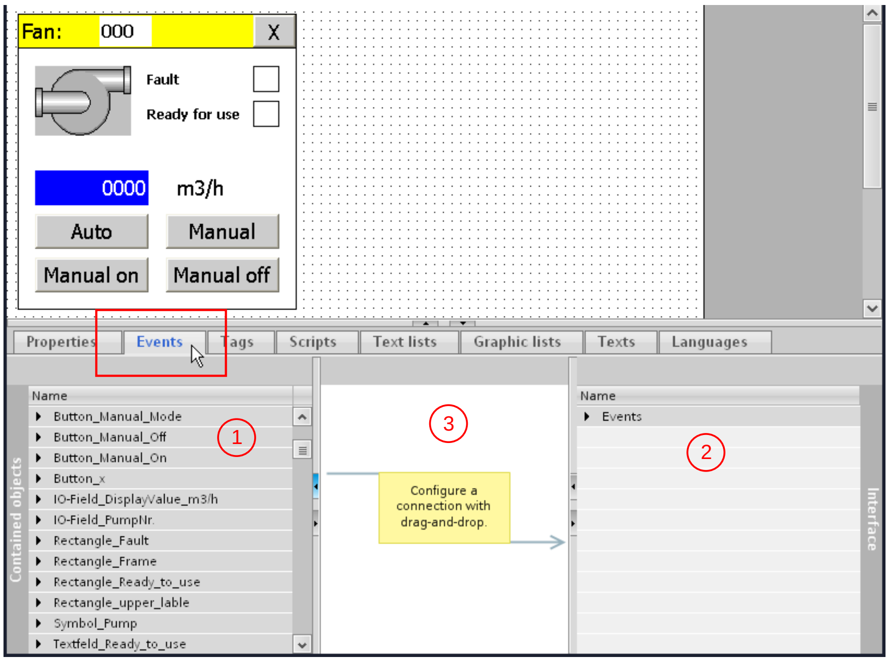
As opposed to the graphic line allocation between the "Contained objects" list and the "Interface" list. Here, there is a direct graphical line allocation between the used "Property" from the "Contained objects" list and the respective "Property" from the "Interface" list (2).

Figure 2-26



2.7.2 "Events” tab

Figure 2-27



Like the previous "Properties” tab, the "Events” tab contains two lists as well as a "graphic” area.

- "Contained objects” list (1)
- "Interface” list 2
- "Graphic” area (3)

"Contained objects" list

Upon calling the "Events" tab, the properties of the objects in the "Contained objects" list change.

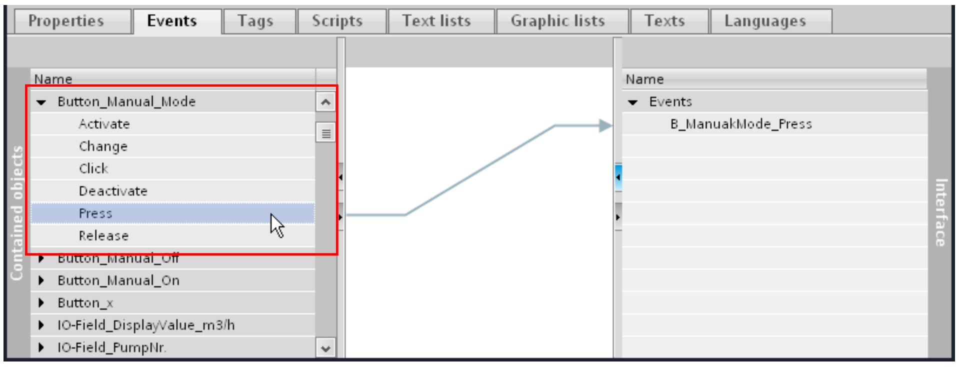
At objects such as a button, the known events such as "Press", "Click", "Release" etc. are available. Furthermore, the view in the "Interface" list changes.

You can use the symbol for "opening/closing" to navigate through the properties of the objects.

In the displayed example, the names of the individual objects have been individually adjusted / assigned beforehand. Through this name assignment, the used objects can be easily identified.

The figure below shows a detailed extract of the possible "Events" of the "Button_Manual_Mode" button.

Figure 2-28



Example:
As the event, the "Press" function was selected for the "Button_Manual_Mode" button and assigned to the "Interface" list.
In the "Events" tab of the "pictures" editor a function can be assigned individually at the faceplate. The function is carried out when "pressing" the "Auto" button. Please refer to the below description of the "Interface" list.

"Interface" list

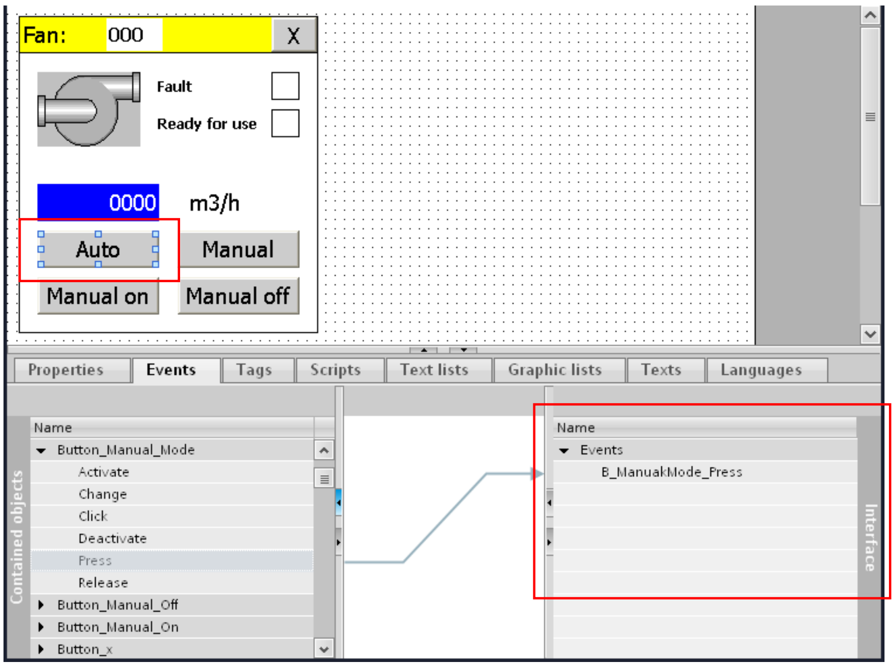
The "Interface" list is used to direct selected events from the "Contained objects" list, such as "Press", "Click", "Release" etc., "outwards" via an "Interface".

That is:
you have called a faceplate in the "pictures" editor Functions shall be assigned to the control objects contained in the faceplate. The properties configured in the "Interface" list are used for the functions to be assigned to the control objects contained in the faceplate.

Example:
A function shall be configurable to the "Button_Manual_Mode" button for the "Press" event.

Configuration view:
From the "Contained objects" list, the selected property of the "Press" event is assigned to the "Interface" list via drag&drop.

Figure 2-29

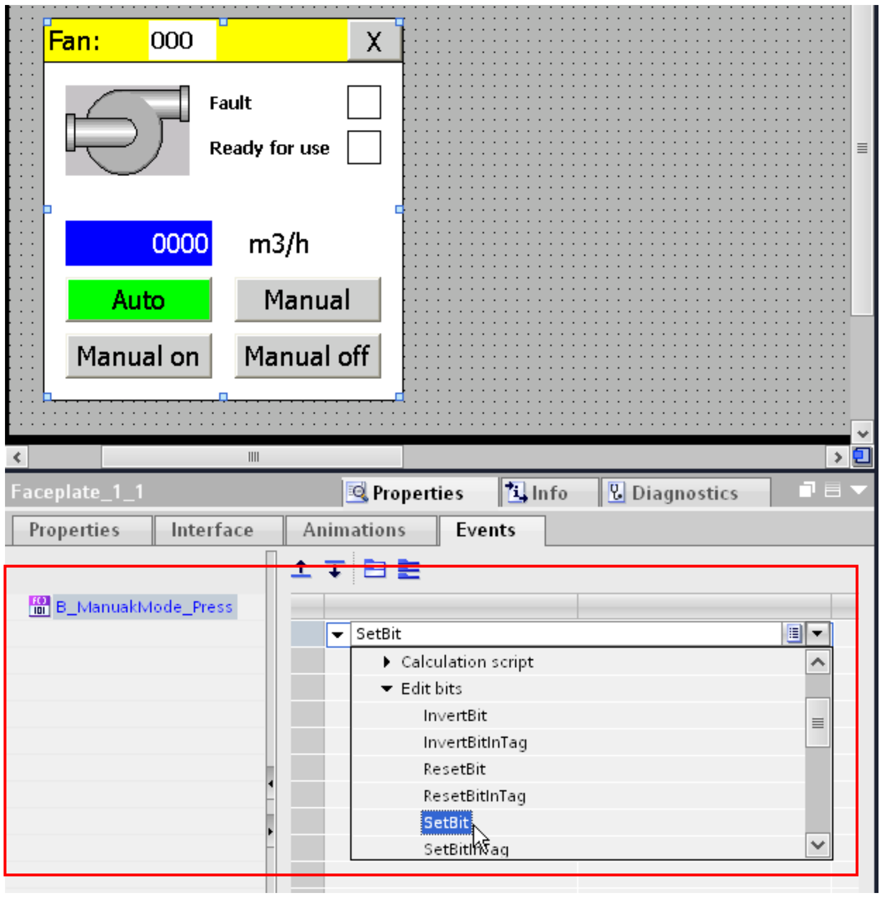


Name
- Button_Manual_Mode
  - Activate
  - Change
  - Click
  - Deactivate
  - Press
  - Release
- Button_Manual_Off
- Button_Manual_On
- Button_x

Name
- Events
  - B_ManuakMode_Press

View of the faceplate in the "pictures" editor:
Any function can be assigned to the "Manual" button via the "Events" tab. In this case "SetBit".

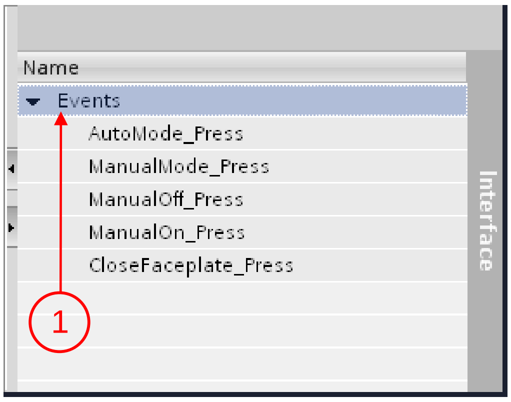
Figure 2-30



Defining the properties in the "Interface" list:

The following figure shows the "Interface" list. The "Interface" list has no specific properties.

Figure 2-31



Events (1)

- Here, the name of the added events from the "Contained objects" list is displayed.
  The events from the "Contained objects" list are directly assigned to the "Interface" list via drag&drop.

  The name which you specify here is displayed at the faceplate in the "Events" tab.
  Tip: assign a name via which you can easily recognize the given property.

  Example: instead of "Press", rather use "AutoMode_Press".

Note:
The sequence of the listed properties cannot be adjusted afterwards.

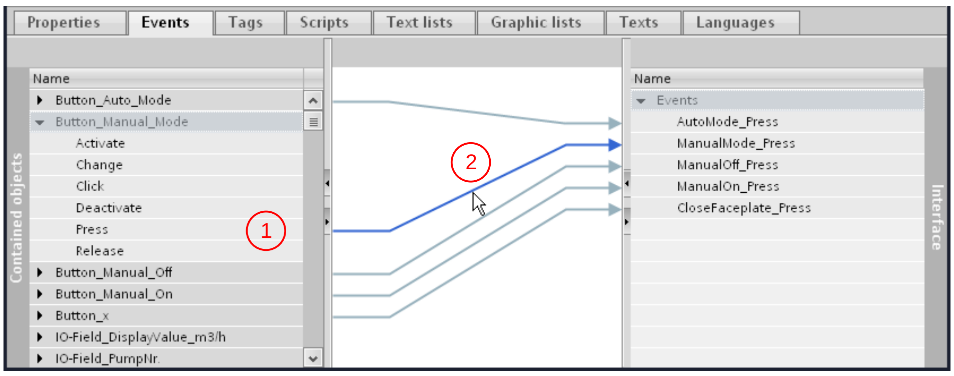
"Graphic" area

The graphic area shows the configured connections between the "Contained objects" list and the "Interface" list.

Clicking a line enables you to see which properties are connected with each other. This facilitates the assignment among one another, in particular for extensive assignments.

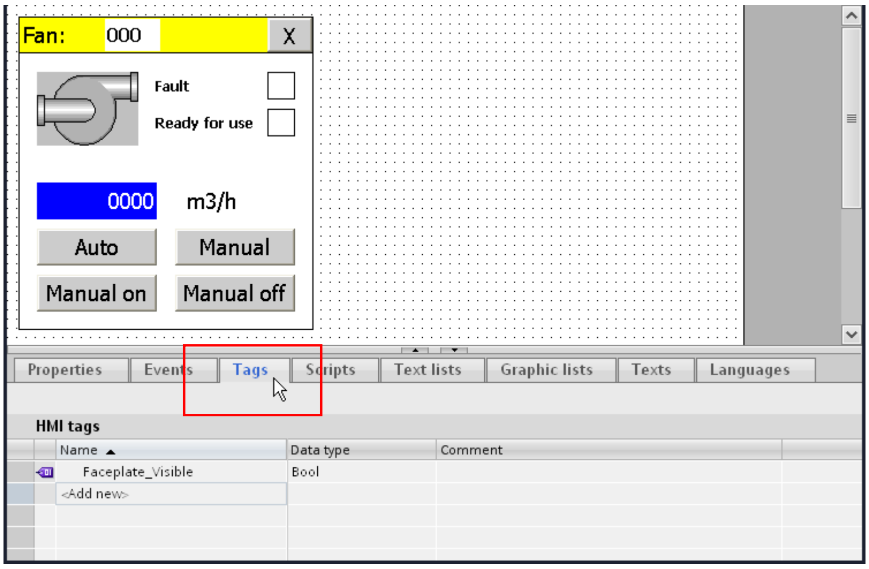
Example:
The "Press" event has been configured at the buttons (1). The names in the "Interface" list have been adjusted according to the assigned property. Clicking one of the lines (2) enables you to see the appropriate assignment.

Figure 2-32



2.7.3 "Tags” tab

Figure 2-33



HMI tags

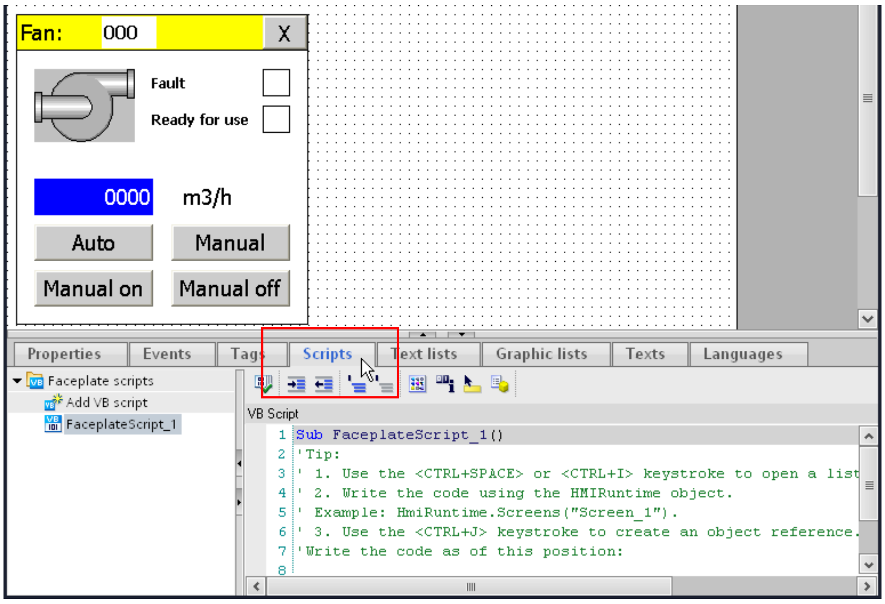
Name	Data type	Comment
Faceplate_Visible	Bool	
<Add new>		

The "Tags” tab contains tags which are only available within the faceplate. The tags are connected directly to the objects contained in the faceplate.

Note You cannot access the tag tables contained in the project tree of your project via the faceplate editor.

2.7.4 "Scripts” tab

Figure 2-34



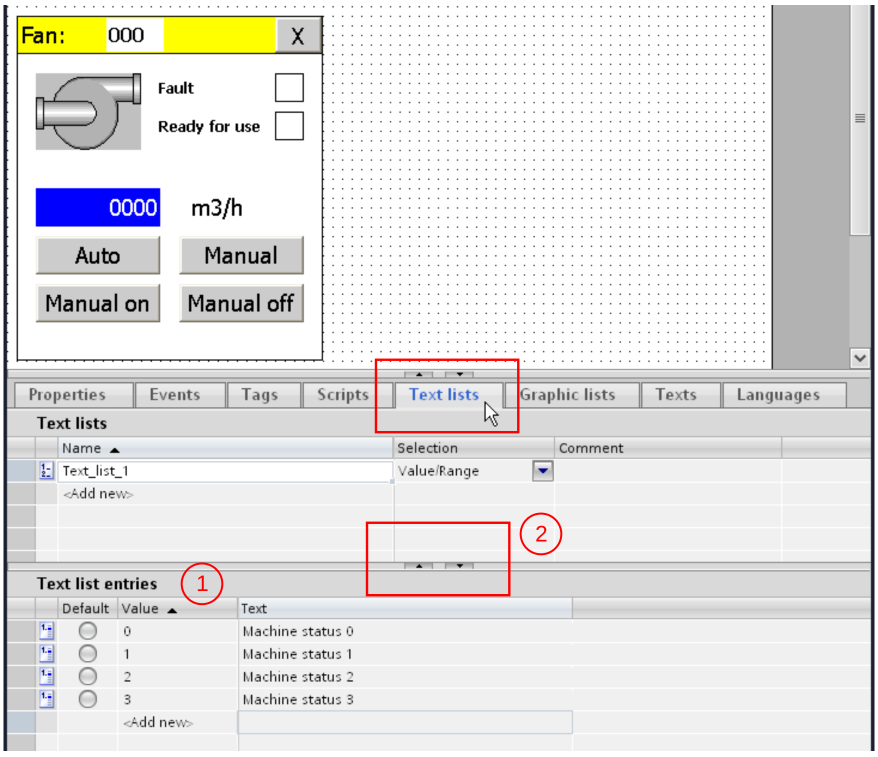
```
Sub FaceplateScript_1()
'Tip:
' 1. Use the <CTRL+SPACE> or <CTRL+I> keystroke to open a list
' 2. Write the code using the HMIRuntime object.
' Example: HmiRuntime.Screens("Screen_1").
' 3. Use the <CTRL+J> keystroke to create an object reference.
'Write the code as of this position:
```

The "Scripts” tab contains scripts which are only available within the faceplate. In the script you call up the system functions or program new functions, e.g. for converting values. Programming is performed in VB script code.

Note You cannot access the scripts contained in the project tree of your project via the faceplate editor.

2.7.5 "Text lists” tab

Figure 2-35



Text lists

Name	Selection	Comment
Text_list_1	Value/Range	
<Add new>		

Text list entries

Default	Value	Text
○	0	Machine status 0
○	1	Machine status 1
○	2	Machine status 2
○	3	Machine status 3
	<Add new>	

In the "Text lists” tab you create text lists for the faceplate and edit them. The created text lists are only available within the faceplate. You interconnect the text lists of the faceplate directly to a symbolic I/O field.

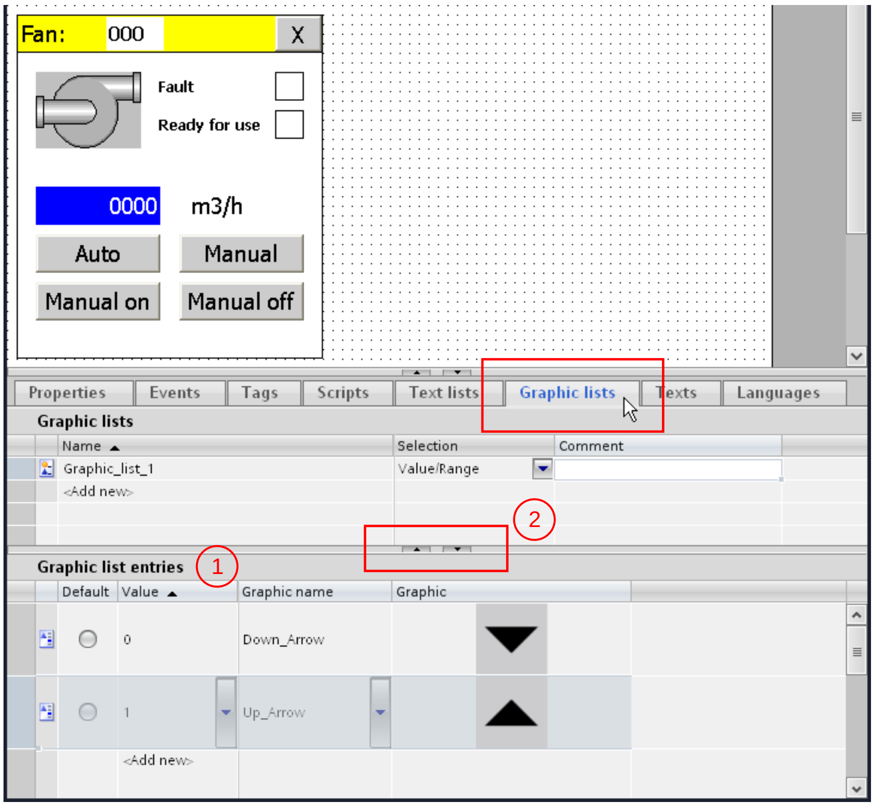
The entries for the text lists are entered in the separate "Text list entries” window (1).

If you do not wish to display the "Text list entries” window, open the window via the arrow key (2).

Note You cannot access the text lists contained in the project tree of your project via the faceplate editor.

2.7.6 "Graphic lists” tab

Figure 2-36



In the "Graphic lists” tab you create graphic lists for the faceplate when required. These graphic lists are only available within the faceplate. You interconnect the graphic lists of the faceplate directly to a graphic I/O field.

The entries for the graphic lists are entered in the separate "Graphic list entries” window (1).

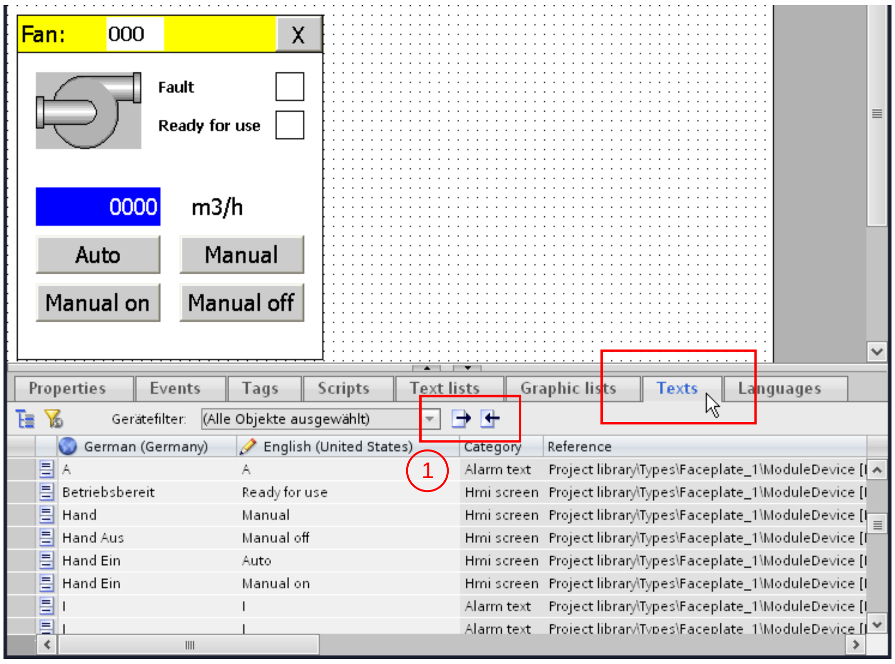
If you do not wish to display the "Graphic list entries” window, open the window via the arrow key (2).

Note

You cannot access the graphic lists contained in the project tree of your project via the faceplate editor.

2.7.7 "Texts” tab

Figure 2-37



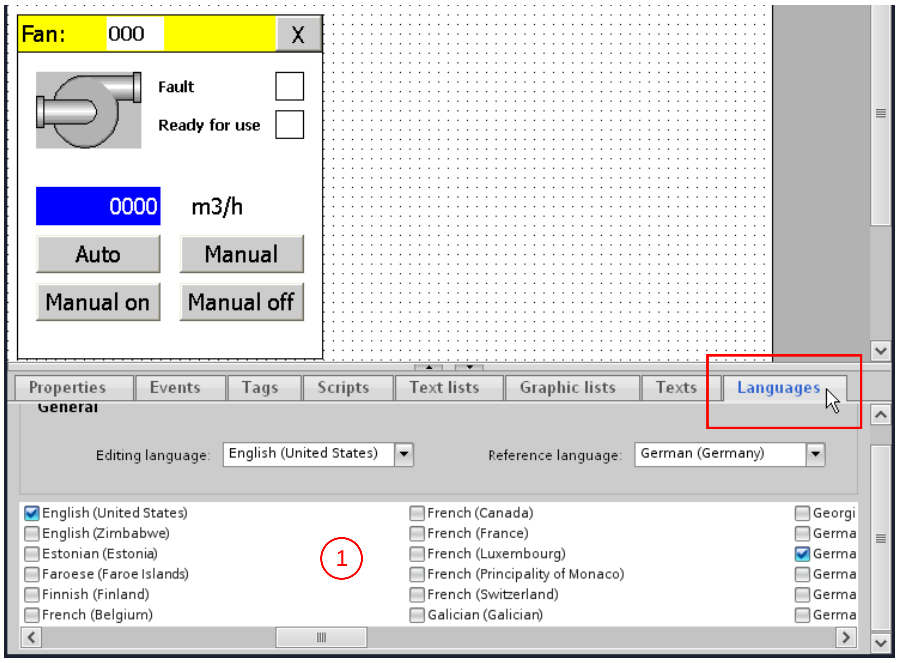
German (Germany)	English (United States)	Category	Reference
A	A	Alarm text	Project library\Types\Faceplate_1\ModuleDevice [
Betriebsbereit	Ready for use	Hmi screen	Project library\Types\Faceplate_1\ModuleDevice [
Hand	Manual	Hmi screen	Project library\Types\Faceplate_1\ModuleDevice [
Hand Aus	Manual off	Hmi screen	Project library\Types\Faceplate_1\ModuleDevice [
Hand Ein	Auto	Hmi screen	Project library\Types\Faceplate_1\ModuleDevice [
Hand Ein	Manual on	Hmi screen	Project library\Types\Faceplate_1\ModuleDevice [
I	I	Alarm text	Project library\Types\Faceplate_1\ModuleDevice [
I	I	Alarm text	Project library\Types\Faceplate_1\ModuleDevice [

The "Texts” tab only contains texts which are used in the faceplate. For multi-language configurations, the known functions such as "Import project texts” and "Export project texts” are available (1). Alternatively, you enter the multi-language texts directly into the existing editor.

Note You cannot access the text list contained in the project tree of your project via the faceplate editor.

2.7.8 "Languages” tab

Figure 2-38



Editing language: English (United States)
Reference language: German (Germany)

- [✓] English (United States)
- [ ] English (Zimbabwe)
- [ ] Estonian (Estonia)
- [ ] Faroese (Faroe Islands)
- [ ] Finnish (Finland)
- [ ] French (Belgium)
- [ ] French (Canada)
- [ ] French (France)
- [ ] French (Luxembourg)
- [ ] French (Principality of Monaco)
- [ ] French (Switzerland)
- [ ] Galician (Galician)
- [ ] Georgi
- [ ] Germa
- [✓] Germa
- [ ] Germa
- [ ] Germa
- [ ] Germa

In the "Languages” tab you assign the language to be available for the generated faceplate. The languages are independent of the language selected in the project.

Using this functionality, you can also use a faceplate in projects with different languages.

When using the faceplate, it must be ensured, that the stored languages of the faceplate are also used in the project in which the faceplate is called.

Example:
In your project, you have selected the languages "German(Germany)” and "English(**United States**)”.

The faceplate was generated in the languages "German(Germany)” and "English(**Great Britain**)”.

When using the faceplate in the project with the languages "German(Germany)” and "English(**United States**)” and selecting the language "English(**United States**)”, then no texts are displayed at the faceplate since only the language "English(**Great Britain**)” is stored at the faceplate.

Note

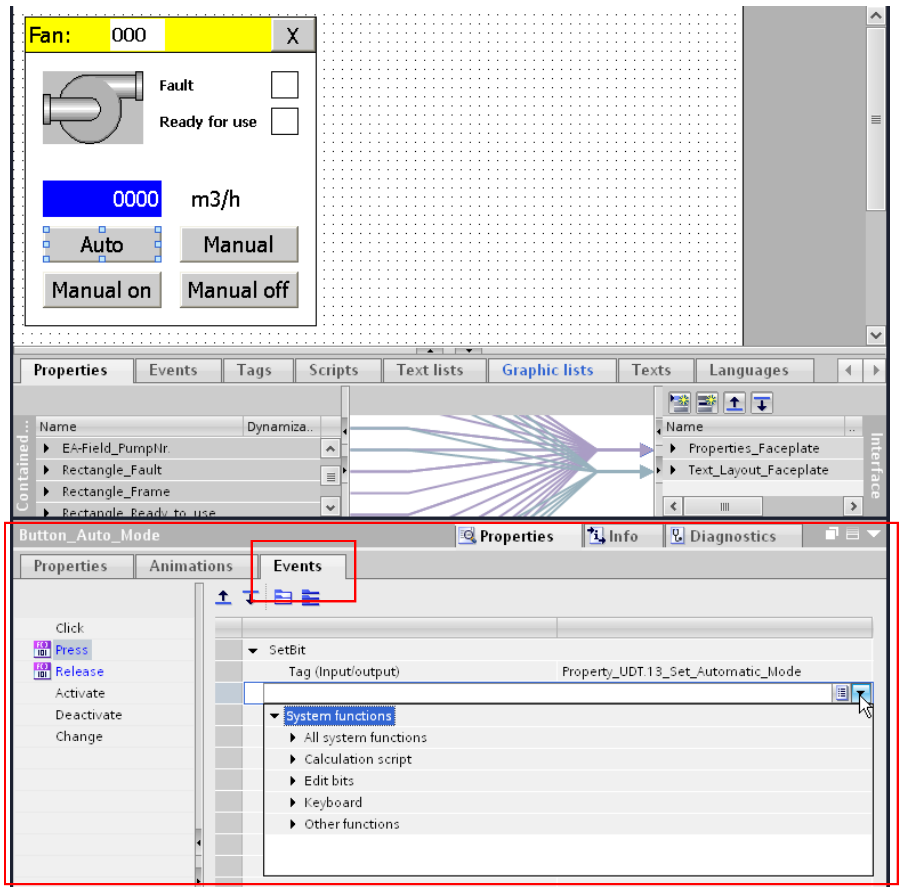
Settings in the project languages in the faceplate editor are independent of the project languages of the user project.

2.8 Parameters in the inspector window

The inspector window in the faceplate editor mainly contains the same parameters and settings options as familiar from the "pictures" editor.

Deviations occur in the "Properties > Events" tab. Functions such as "Activate screen", for example, are not available here.

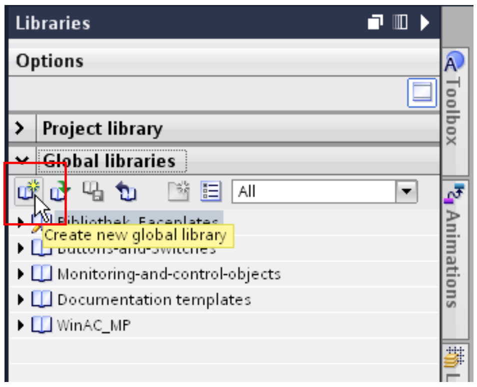
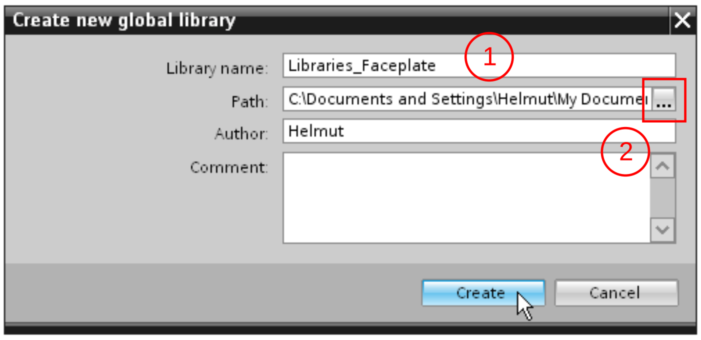
Figure 2-39

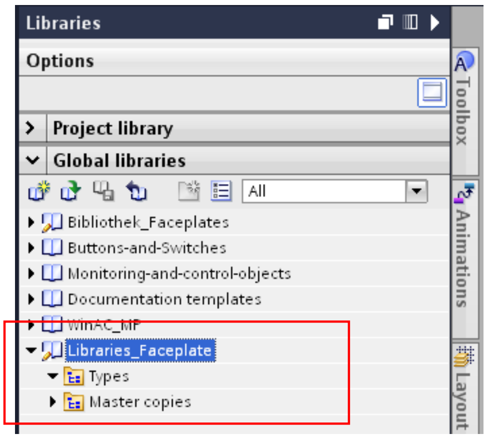
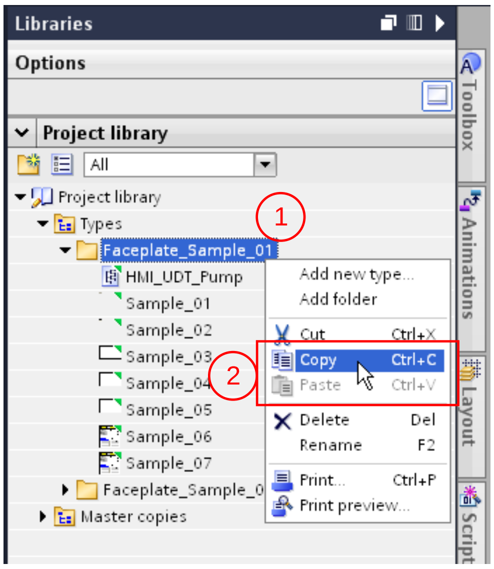


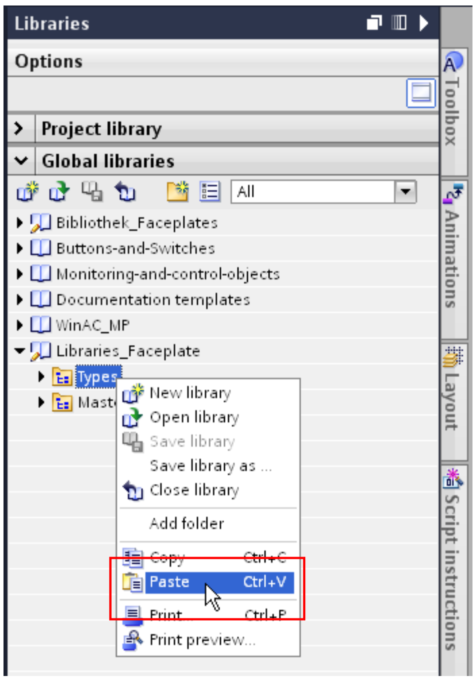
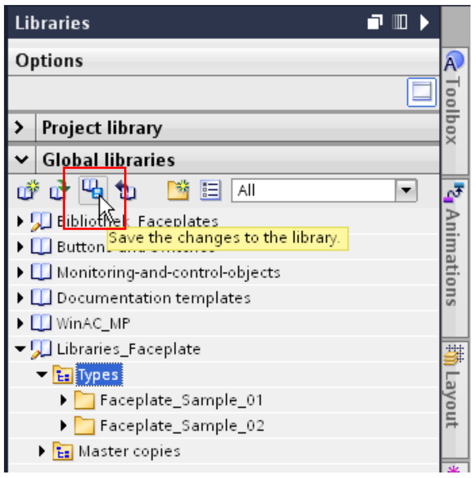
2.9 Saving faceplates in "Global libraries”

The created faceplates are stored and edited in the project library. If you wish to reuse the created faceplates across different projects, save the completed faceplates in the "Libraries” task card and there in the "Global libraries” palette.

Table 2-40

No.	Action
1.	**Create a new "Global library”** • Open the "Libraries” task card and there the "Global libraries” palette. • Click on the first icon. The new global library is created via this icon.  Libraries Options Project library Global libraries All Create new global library Bibliothek_Faceplates Buttons-and-switches Monitoring-and-control-objects Documentation templates WinAC_MP Toolbox Animations • A window opens in which you specify the name (1) as well as the storage path (2) etc.. Close the entry with the "Create” button.  Library name: Libraries_Faceplate Path: C:\Documents and Settings\Helmut\My Docume Author: Helmut Comment:

No.	Action
	View of the newly created "Global library” The following picture shows the view of the newly created global library "Libraries_Faceplate” with the subfolders "Types” and "Master copies”.  Libraries Options Project library Global libraries All Bibliothek_Faceplates Buttons-and-Switches Monitoring-and-control-objects Documentation templates WinAC_MP Libraries_Faceplate Types Master copies Toolbox Animations Layout
2.	**Copying faceplates from the project library** • Go to the "Project library” palette. • In the "Types” folder you select an individual block or the entire folder with the contained faceplates (1). • Right-click on the selected entry. A context menu opens. Here you select the "Copy” option (2).  Libraries Options Project library All Project library Types Faceplate_Sample_01 HMI_UDT_Pump Sample_01 Sample_02 Sample_03 Sample_04 Sample_05 Sample_06 Sample_07 Faceplate_Sample_0 Master copies 1 2 Add new type... Add folder Cut Ctrl+X Copy Ctrl+C Paste Ctrl+V Delete Del Rename F2 Print... Ctrl+P Print preview... Toolbox Animations Layout Script

No.	Action
3.	**Insert the copied faceplates into global library "Libraries_Faceplate"** • Go back to the "Global libraries" palette and open the global library "Libraries_Faceplate". • Select the "Types" folder via the right mouse button. A context menu opens. Here you select the "Insert" option. The previously copied folder with the faceplates is copied to the "Types" folder.  Libraries Options Project library Global libraries All Bibliothek_Faceplates Buttons-and-Switches Monitoring-and-control-objects Documentation templates WinAC_MP Libraries_Faceplate Types Mast New library Open library Save library Save library as ... Close library Add folder Copy Ctrl+C Paste Ctrl+V Print... Ctrl+P Print preview... Toolbox Animations Layout Script instructions
4.	**Save the changes in "Global libraries"** • Click on the third icon. The changes saved in the library via this icon.  Libraries Options Project library Global libraries All Bibliothek_Faceplates Save the changes to the library. Monitoring-and-control-objects Documentation templates WinAC_MP Libraries_Faceplate Types Faceplate_Sample_01 Faceplate_Sample_02 Master copies Toolbox Animations Layout Continuing information on the topic of "Libraries" is available in the online help.

3 Functions and Properties of an HMI UDT

This chapter describes the steps for creating / editing an HMI UDT user data type.

Note As of WinCC (TIA Portal) V13 SP1 and when using an S7-1200 or S7-1500 controller, you do no longer need an "HMI UDT”.
For this purpose, please refer to chapter 4.

If you are not yet familiar with the properties of a user data type, then thoroughly read this chapter.

Introduction

Later on, a tag must be assigned to each configured function of a faceplate via the interface. Depending on the scope of the faceplate, this may be quite extensive.

Using an HMI UDT, only assigns **one** tag to the faceplate via the interface.

General

A user data type (HMI UDT) always includes

- a data block which has the same structure as the user data type.
- an HMI tag for which the created user data type has been selected as "Data type”. The data exchange is then performed via this tag between the PLC and the user data type used in the faceplate.

For the configuration of a user data type it makes sense to initially create the data block with the required structure.

You can assign respectively one data block to a user data type. However, you can also use a data block which contains several structures and assign it to the individual user data types.

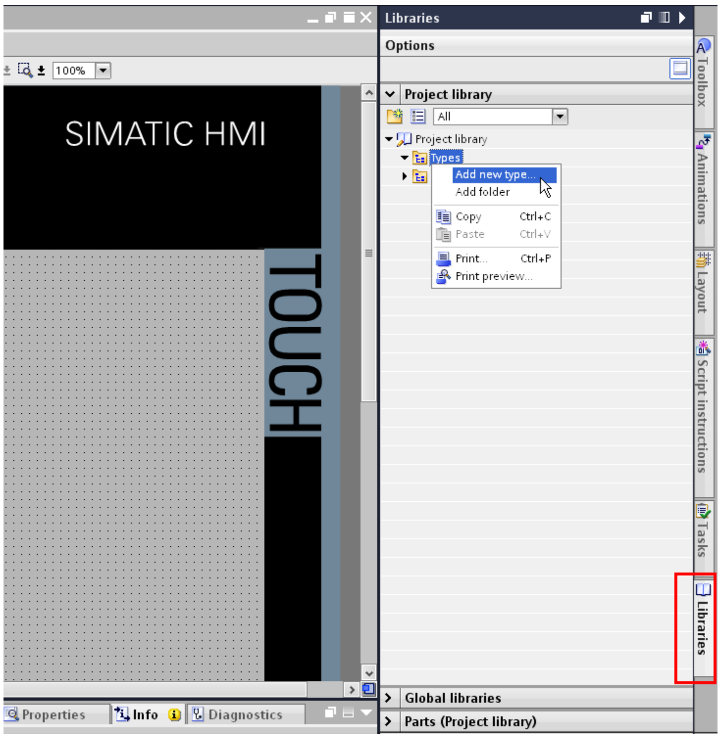
Note An HMI UDT can only be used in connection with a faceplate.

3.1 Creating a user data type

To create a new user data type, please proceed as follows:

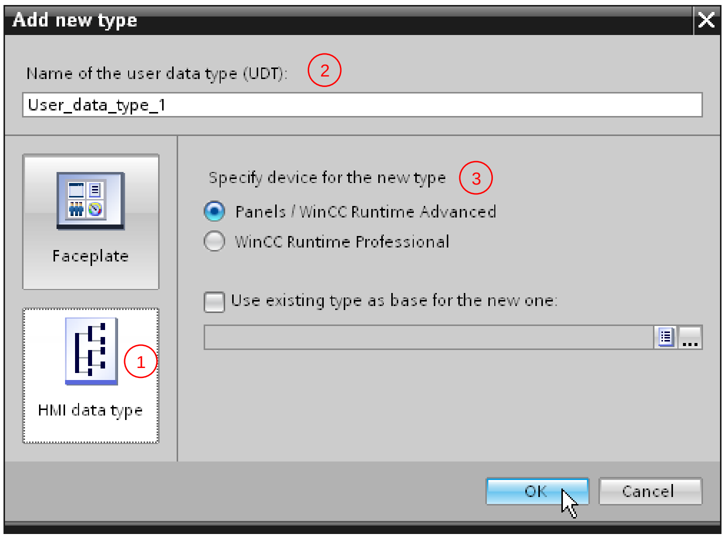
- Select the "Libraries" task card.
- Open the "Project library" palette.
- Right-click on the "Types" folder. A context menu opens.
- Select "Add new type…" in the context menu.
  The "Add new type…" window opens.

Figure 3-1



Display of the "Add new type” window.

Figure 3-2



Name of the user data type (UDT): User_data_type_1

Specify device for the new type

[✓ Panels / WinCC Runtime Advanced]

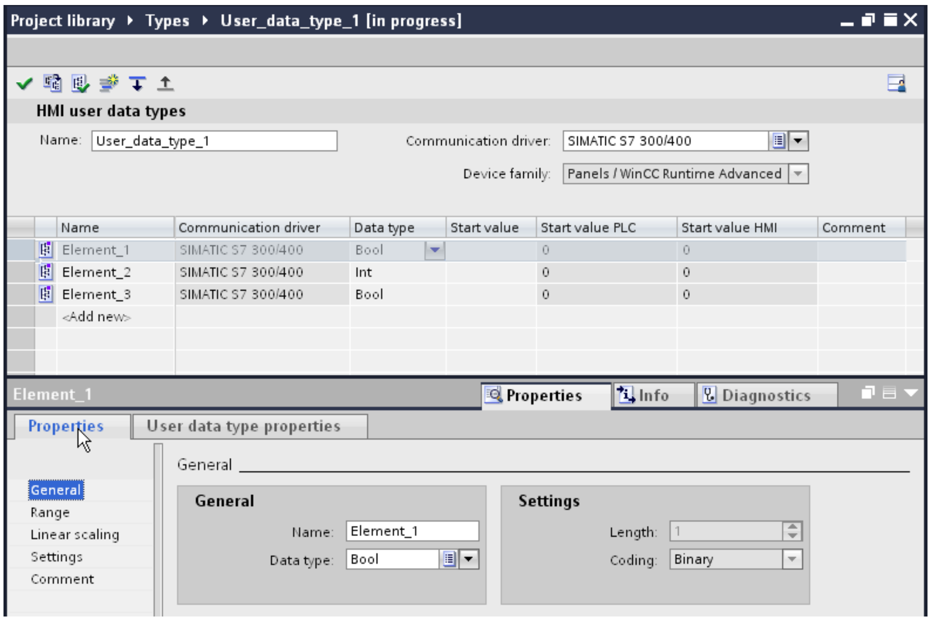
WinCC Runtime Professional

Use existing type as base for the new one:

- On the left side, click the "HMI UDT” type (1).
- Assign a unique name to the user data type (2).
- In "Specify device for the new type” you activate the radio button "Panels / WinCC Runtime Advanced” (3).
- Confirm the entries with "OK”.
  The editor for creating / editing the user data type opens.

View of the opened editor.

Figure 3-3



Name: User_data_type_1

Communication driver: SIMATIC S7 300/400

Device family: Panels / WinCC Runtime Advanced

Name	Communication driver	Data type	Start value	Start value PLC	Start value HMI	Comment
Element_1	SIMATIC S7 300/400	Bool		0	0	
Element_2	SIMATIC S7 300/400	Int		0	0	
Element_3	SIMATIC S7 300/400	Bool		0	0	
<Add new>						

General

General

Name: Element_1

Data type: Bool

Settings

Length: 1

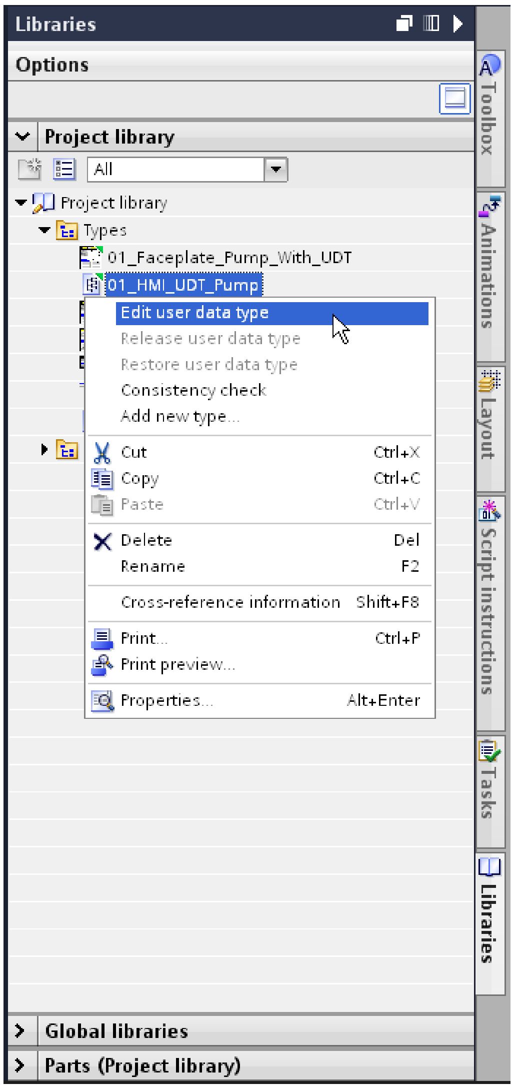
Coding: Binary

3.2 Editing/opening a user data type

To process an existing user data type, please proceed as follows:

- Select the "Libraries” task card.
- Open the "Project library” palette.
- In the "Types” folder, you use the right mouse button to select the user data type you wish to open. A context menu opens.
- Select "Edit user data type” in the context menu.
  The editor for creating / editing the user data type block opens.

Figure 3-4



3.3 Configuring the user data type

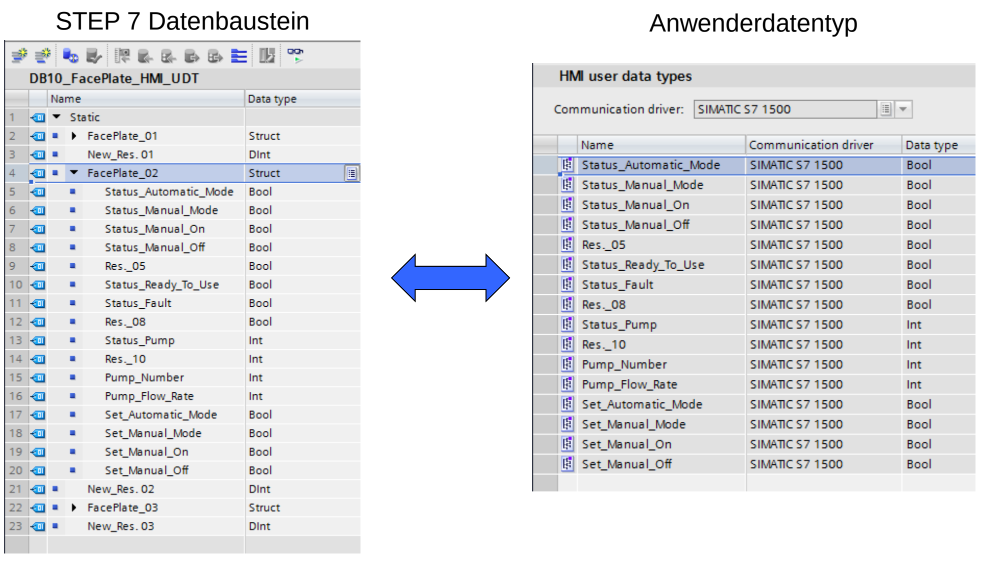
General

Considerations during the generation process:

- Only **one** communication driver can be assigned to a user data type.
  - **Example:**
    You have a configuration with a faceplate. This faceplate uses a user data type. The communication driver configured in the user data type is an S7-300/S7-400 controller.
    If you wish to use this faceplate in a project with an S7-1500 controller, the communication driver must be adjusted in the user data type.
- Prior to configuring a user data type, all "Functions” a faceplate should have should be known. It facilitates the later assignment to the selected properties in the faceplate. However, you can also adjust a user data type later on at any time.
- The date type in STEP 7 programming must have the same "Structure” as the created user data type.
- The "data types” used in the data block of STEP 7 programming must correspond to the data types in the user data type.

Example view between a DB with data type "Struct” and a user data type. The figure illustrates the structure of both files.

Figure 3-5



DB10_FacePlate_HMI_UDT

	Name	Data type
1	Static	
2	FacePlate_01	Struct
3	New_Res. 01	DInt
4	FacePlate_02	Struct
5	Status_Automatic_Mode	Bool
6	Status_Manual_Mode	Bool
7	Status_Manual_On	Bool
8	Status_Manual_Off	Bool
9	Res._05	Bool
10	Status_Ready_To_Use	Bool
11	Status_Fault	Bool
12	Res._08	Bool
13	Status_Pump	Int
14	Res._10	Int
15	Pump_Number	Int
16	Pump_Flow_Rate	Int
17	Set_Automatic_Mode	Bool
18	Set_Manual_Mode	Bool
19	Set_Manual_On	Bool
20	Set_Manual_Off	Bool
21	New_Res. 02	DInt
22	FacePlate_03	Struct
23	New_Res. 03	DInt

HMI user data types

Communication driver: SIMATIC S7 1500

Name	Communication driver	Data type
Status_Automatic_Mode	SIMATIC S7 1500	Bool
Status_Manual_Mode	SIMATIC S7 1500	Bool
Status_Manual_On	SIMATIC S7 1500	Bool
Status_Manual_Off	SIMATIC S7 1500	Bool
Res._05	SIMATIC S7 1500	Bool
Status_Ready_To_Use	SIMATIC S7 1500	Bool
Status_Fault	SIMATIC S7 1500	Bool
Res._08	SIMATIC S7 1500	Bool
Status_Pump	SIMATIC S7 1500	Int
Res._10	SIMATIC S7 1500	Int
Pump_Number	SIMATIC S7 1500	Int
Pump_Flow_Rate	SIMATIC S7 1500	Int
Set_Automatic_Mode	SIMATIC S7 1500	Bool
Set_Manual_Mode	SIMATIC S7 1500	Bool
Set_Manual_On	SIMATIC S7 1500	Bool
Set_Manual_Off	SIMATIC S7 1500	Bool

Note:

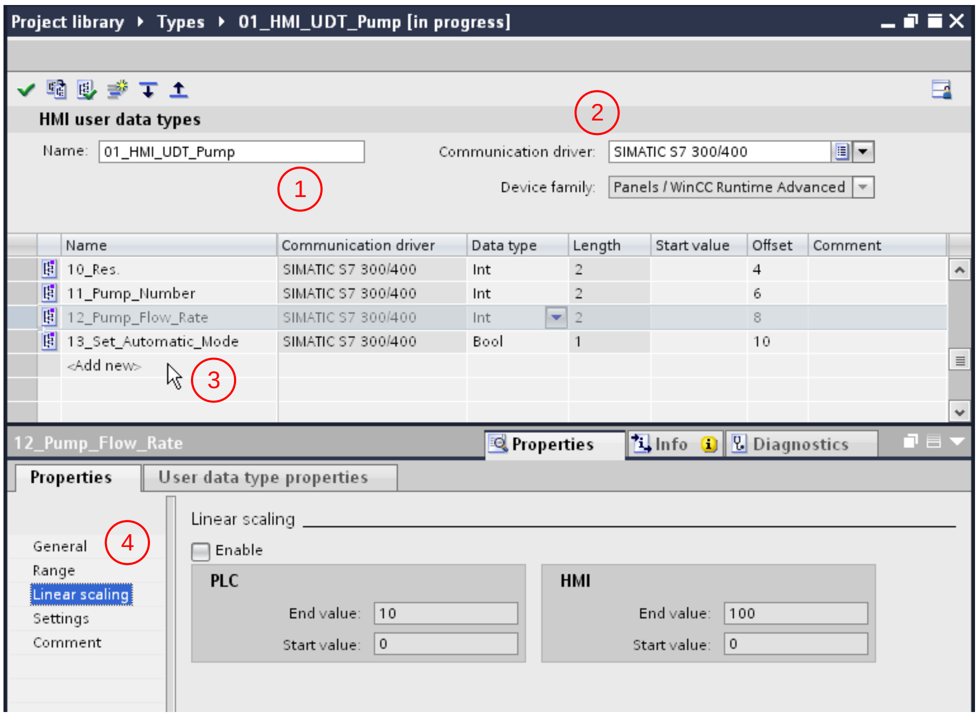
For each faceplate you use in the project, you need to provide a data area (structure).
For each faceplate you can create a separate data block or, as shown in the figure, provide one data block for several faceplates.

The user data type needs only be created once. It can be used for each faceplate that has the same functionality.

Creating tags

The process for adding tags to the user data type editor is similar as for entering tags into a "Tag table”.

Figure 3-6



Project library ▸ Types ▸ 01_HMI_UDT_Pump [in progress]

HMI user data types

Name: 01_HMI_UDT_Pump

Communication driver: SIMATIC S7 300/400

Device family: Panels / WinCC Runtime Advanced

Name	Communication driver	Data type	Length	Start value	Offset	Comment
10_Res.	SIMATIC S7 300/400	Int	2		4	
11_Pump_Number	SIMATIC S7 300/400	Int	2		6	
12_Pump_Flow_Rate	SIMATIC S7 300/400	Int	2		8	
13_Set_Automatic_Mode	SIMATIC S7 300/400	Bool	1		10	
<Add new>						

12_Pump_Flow_Rate

Properties | User data type properties

Linear scaling

Enable

	PLC	HMI
End value:	10	100
Start value:	0	0

- (1) Here you can adjust the name of the user data type.
- (2) Here you select the communication driver (e.g. S7 300/400). The set communication type applies for all elements. After a communication driver has been unselected, an internal tag cannot be used as data type.
- (3 Double-click on "Add” in the "Name” column and specifythe data type etc.. Replace the standard entry in the "Name" column with a meaningful name. The name should be a reference to the tag names used in the data block.
- (4) Here you can define special properties such as "linear scaling” for the tag. In comparison with "Tag table”, the possible properties are limited.

After all tags have been added, close the editor via the button "Close” "**X**”.

For the created user data type to be available in the project, it must first be released. For more information, please refer to chapter 3.4.

3.4 User data type

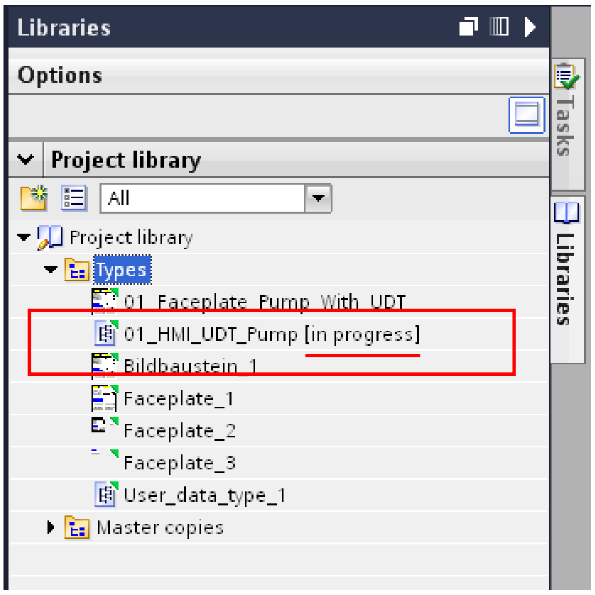
User data type status

When editing a user data type, the following entry "Name of the faceplate **[in progress]**" appears next to the name of the user data type.

In the header of the user data type editor you can also see the current status of the opened user data type.

The status information "[in progress]" shows you that the user data type on hand was modified.

Figure 3-7



Project library
Types
01_Faceplate_Pump_With_UDT
01_HMI_UDT_Pump [in progress]
Bildbaustein_1
Faceplate_1
Faceplate_2
Faceplate_3
User_data_type_1
Master copies

What happens with a user data type used in the project if the respective user data type is currently edited?

When editing a user data type, the modifications at the user data type used in the project only become effective after the respective user data type has been explicitly released (see chapter 3.4.1).

You can, for example, interrupt editing a user data type and save and close the configuration. After opening the configuration, you can continue editing the user data type. The status of the edited user data type then still is "[in progress]".

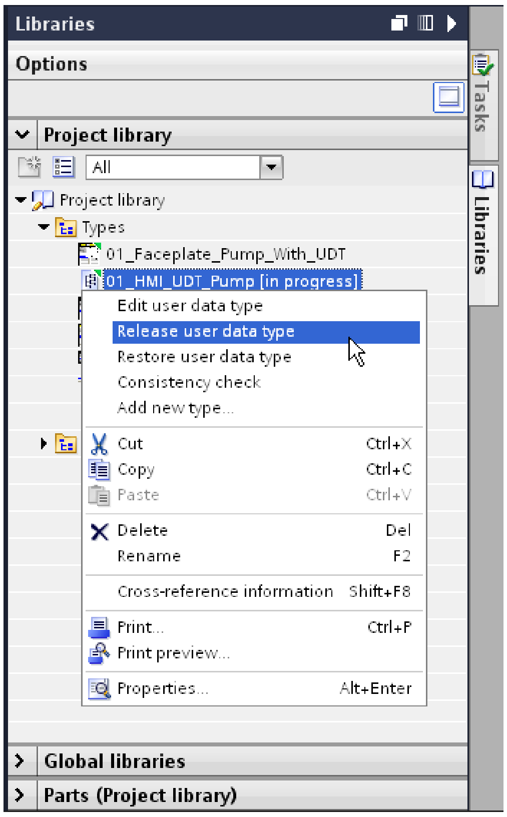
3.4.1 Releasing the user data type

After you have created or edited a user data type, it is initially not applicable in the project. A user data type must first be released for it to be available in the project.

To release an existing user data type, please proceed as follows:

- Select the "Libraries" task card.
- Open the "Project library" palette.
- In the "Types" folder you use the right mouse button to select the user data type you wish to release. A context menu opens.
- Select "Release user data type" in the context menu.
  The user data type is updated and can now be used in the project.

Figure 3-8



3.4.2 Restoring the user data type

You can, for example, interrupt editing a user data type and save and close the configuration. After opening the configuration, you can continue editing the user data type.

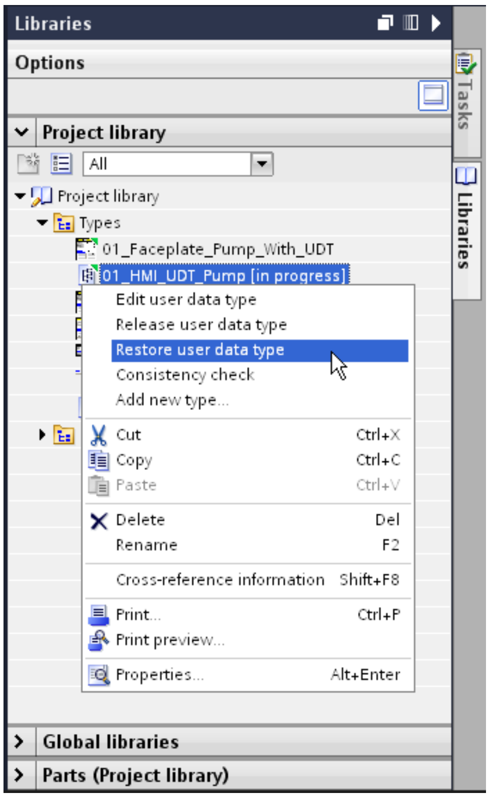
If you now wish to restore the "old" status of the user data type, please use the "Restore user data type" function.

All modifications since the last release at the user data type will be rejected. The user data type is then released again.

To restore a user data type, please proceed as follows:

- Select the "Libraries" task card.
- Open the "Project library" palette.
- In the "Types" folder you use the right mouse button to select the user data type you wish to restore. A context menu opens.
- Select "Restore user data type" in the context menu.
  The modifications are rejected. The user data type is released again.

Figure 3-9

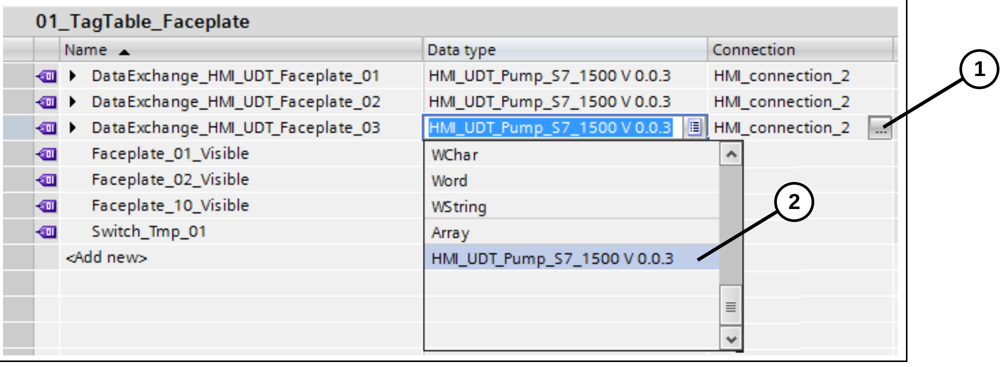


3.5 Assigning an HMI UDT to an HMI tag

The data exchange between the PLC and the user data type used in the faceplate requires an HMI tag.

Please refer to the below description.

Table 3-10

No.	Action
1.	**Creating an HMI tag** • In the HMI configuration, you open the tag folder and create a new tag in the tag table. • Assign a control connection to the tag that the faceplate uses to communicate with the PLC (1). • In the drop-down list of the "Data type" column you select the user data type used in the faceplate (HMI UDT) (2). 

01_TagTable_Faceplate

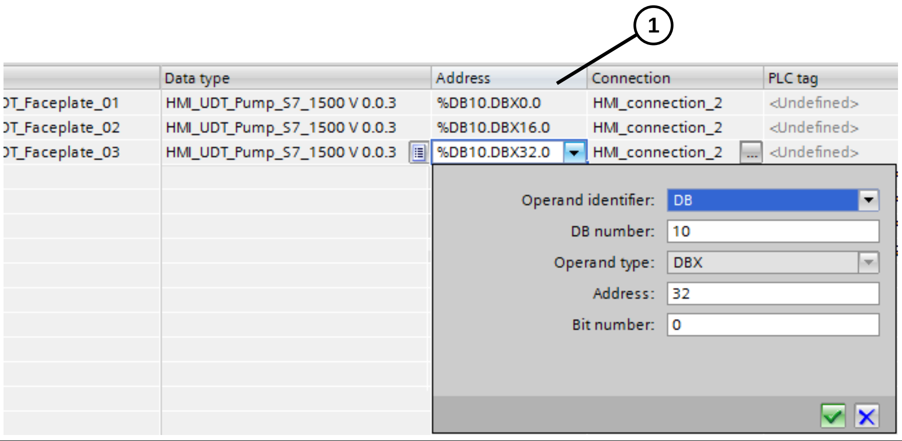
Name	Data type	Connection
DataExchange_HMI_UDT_Faceplate_01	HMI_UDT_Pump_S7_1500 V 0.0.3	HMI_connection_2
DataExchange_HMI_UDT_Faceplate_02	HMI_UDT_Pump_S7_1500 V 0.0.3	HMI_connection_2
DataExchange_HMI_UDT_Faceplate_03	HMI_UDT_Pump_S7_1500 V 0.0.3	HMI_connection_2
Faceplate_01_Visible		
Faceplate_02_Visible		
Faceplate_10_Visible		
Switch_Tmp_01		
<Add new>		

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>2.</td><td>

Assigning a start address to the HMI tag

For each faceplate, a data area ("structure”) has been assigned in a data block. For this purpose, refer to chapter 3.3.
For the data exchange you need to specify the respective start address.

- In the "Address” column, you specify the start address of the data area (1). Data block "DB10” displayed in chapter 3.3 contains the structure for three faceplates.
  - The start address for the first faceplate is "DB10.DBX**0**.0”
  - The start address for the second faceplate is "DB10.DBX**16**.0”
  - The start address for the third faceplate is "DB10.DBX**32**.0”.



	Data type	Address	Connection	PLC tag
)T_Faceplate_01	HMI_UDT_Pump_S7_1500 V 0.0.3	%DB10.DBX0.0	HMI_connection_2	<Undefined>
)T_Faceplate_02	HMI_UDT_Pump_S7_1500 V 0.0.3	%DB10.DBX16.0	HMI_connection_2	<Undefined>
)T_Faceplate_03	HMI_UDT_Pump_S7_1500 V 0.0.3	%DB10.DBX32.0	HMI_connection_2	<Undefined>

Operand identifier: DB
DB number: 10
Operand type: DBX
Address: 32
Bit number: 0

</td></tr>
</table>

4 Functions and Properties of a PLC Data Type

This chapter describes the steps for creating / editing a PLC data type ("PLC UDT”).

Note As of WinCC (TIA Portal) V13 SP1 and when using an S7-1200 or S7-1500 controller, you do no longer need an "HMI UDT”.

4.1 Introduction

Later on, a tag must be assigned to each configured function of a faceplate via the interface. Depending on the scope of the faceplate, this may be quite extensive.

Using a PLC data type, only assigns **one** tag to the faceplate via the interface.

Chapter 3 describes how to use a user data type (HMI UDT). This is the case, for example, when using a SIMATIC S7-300/400 controller.

The disadvantage of an HMI UDT is:

- The HMI UDT can only be used for faceplates.
- When changes are made in the respective data block, these must then be updated manually in the HMI UDT. This may cause errors when assigning the name/ tag type.

As of WinCC (TIA Portal) V13 SP1 and in connection with a SIMATIC S7-1200 or S7-1500 controller, you do **no longer** need a user data type. From this version on, you can use a **PLC data type**.

The advantage of a PLC data type is:

- The data type needs only be created once.
- The created data type can also be used in further applications – not only in faceplates.
- Modifications in the data structure are automatically updated in all locations of usage.

4.2 Creating PLC data types

General

Considerations during the generation process:

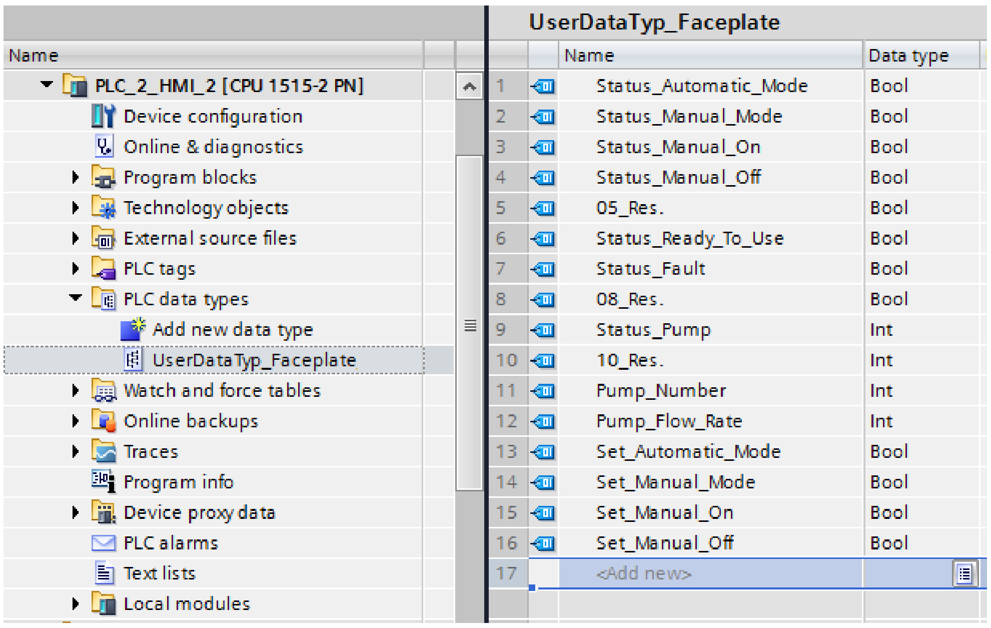
Prior to configuring a PLC data type, all "Functions" a faceplate should have should be known. It facilitates the later assignment to the selected properties in the faceplate. However, you can also adjust a PLC data type later on at any time.

Creating a PLC data type

- In the project navigation, you select the PLC controller for performing the data exchange with the faceplate later on.
- Select the PLC data types folder and add a new "data type".
- Enter all the required tags into the table:

Note If a "data structure" already exists in an existing data block, you can copy it and add it directly into the PLC data type.

Figure 4-1



UserDataTyp_Faceplate

	Name	Data type
1	Status_Automatic_Mode	Bool
2	Status_Manual_Mode	Bool
3	Status_Manual_On	Bool
4	Status_Manual_Off	Bool
5	05_Res.	Bool
6	Status_Ready_To_Use	Bool
7	Status_Fault	Bool
8	08_Res.	Bool
9	Status_Pump	Int
10	10_Res.	Int
11	Pump_Number	Int
12	Pump_Flow_Rate	Int
13	Set_Automatic_Mode	Bool
14	Set_Manual_Mode	Bool
15	Set_Manual_On	Bool
16	Set_Manual_Off	Bool
17	<Add new>	

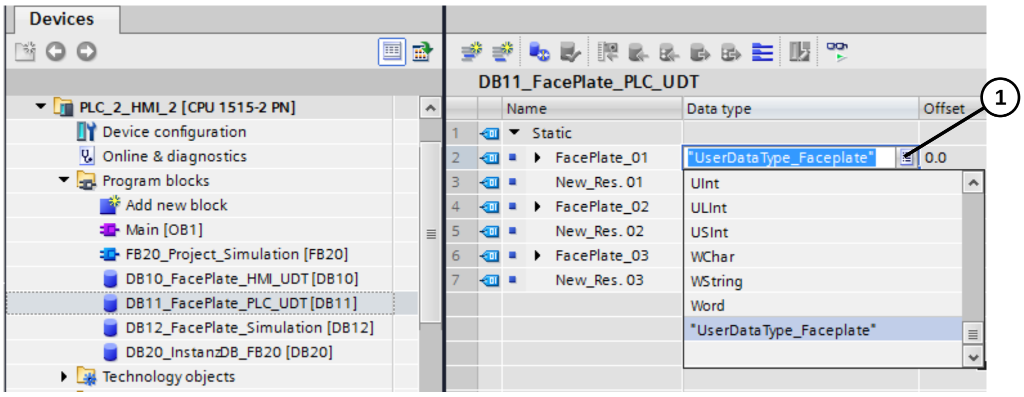
4.3 Inserting a PLC data type into a data block

It is assumed, that a "PLC data type" already exists.

- Open an existing data block or create a new data block.
- Create a new tag. In the "Data type" column you select the created "PLC data type" (1).
  In this example, "UserDataTyp_Faceplate".
- For each control station you add a further tag.

Note You can also use the created "PLC data type" for "faceplates" that have previously been using "HMI-UDTs". The prerequisite is, that both data types have the same structure/setup.

Figure 4-2



- PLC_2_HMI_2 [CPU 1515-2 PN]
  - Device configuration
  - Online & diagnostics
  - Program blocks
    - Add new block
    - Main [OB1]
    - FB20_Project_Simulation [FB20]
    - DB10_FacePlate_HMI_UDT [DB10]
    - DB11_FacePlate_PLC_UDT [DB11]
    - DB12_FacePlate_Simulation [DB12]
    - DB20_InstanzDB_FB20 [DB20]
  - Technology objects

DB11_FacePlate_PLC_UDT

	Name	Data type	Offset
1	Static		
2	FacePlate_01	"UserDataType_Faceplate"	0.0
3	New_Res.01	UInt	
4	FacePlate_02	ULInt	
5	New_Res.02	USInt	
6	FacePlate_03	WChar	
7	New_Res.03	WString	
		Word	
		"UserDataType_Faceplate"	

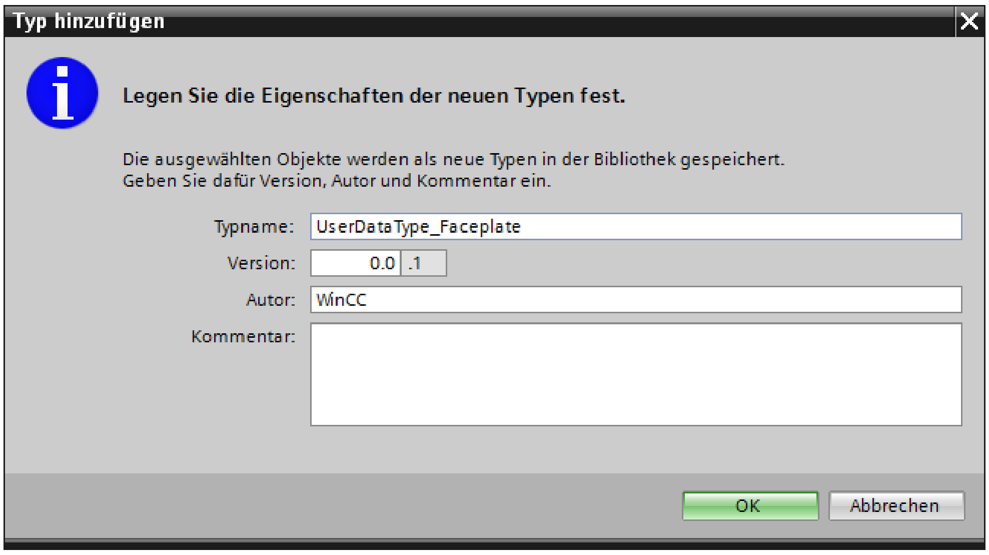
4.4 Integrating a PLC data type into a faceplate

To use a PLC data type in a faceplate, proceed as follows:

- Select the "Libraries" task card.
- Open the "Project library" palette.
- Open the "Types" folder.
- In the project navigation, you select the PLC controller for performing the data exchange with the faceplate later on.
- Select the PLC data types folder and mark the existing PLC data type.
  In this example, "UserDataTyp_Faceplate".
- Drag and drop the selected PLC data type into the opened "Project library > Types". A window opens in which you can change the name of the added PLC data type. Confirm the entries with "OK".

4.4 Integrating a PLC data type into a faceplate

Figure 4-3



Typ hinzufügen

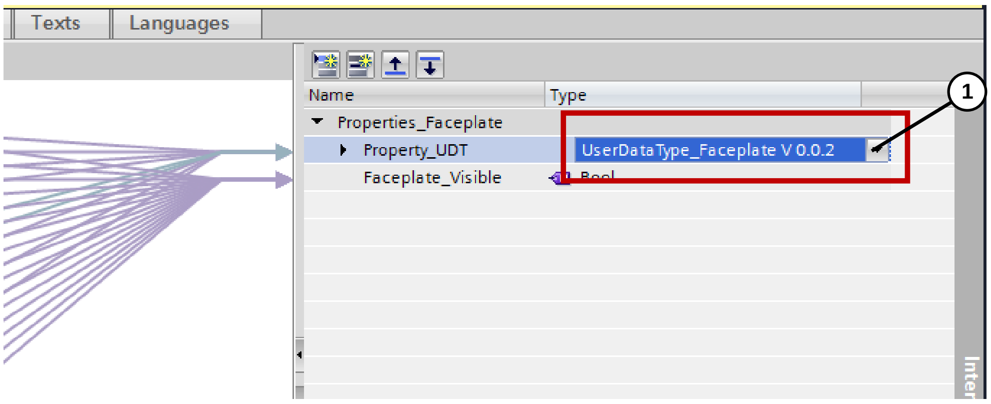
Legen Sie die Eigenschaften der neuen Typen fest.

Die ausgewählten Objekte werden als neue Typen in der Bibliothek gespeichert.
Geben Sie dafür Version, Autor und Kommentar ein.

Typname:	UserDataType_Faceplate
Version:	0.0 .1
Autor:	WinCC
Kommentar:	

- Open an existing faceplate or create a new faceplate.
- In the "Properties > Interface” tab of the configuration area, you add a tag and select the created PLC data type (1) in "Type” via the dropdown menu.
  In this example, "UserDataTyp_Faceplate”.

Figure 4-4



Name	Type
Properties_Faceplate	
Property_UDT	UserDataType_Faceplate V 0.0.2
Faceplate_Visible	Bool

4.5 Editing/opening a PLC data type

To process an existing user PLC data type, please proceed as follows:

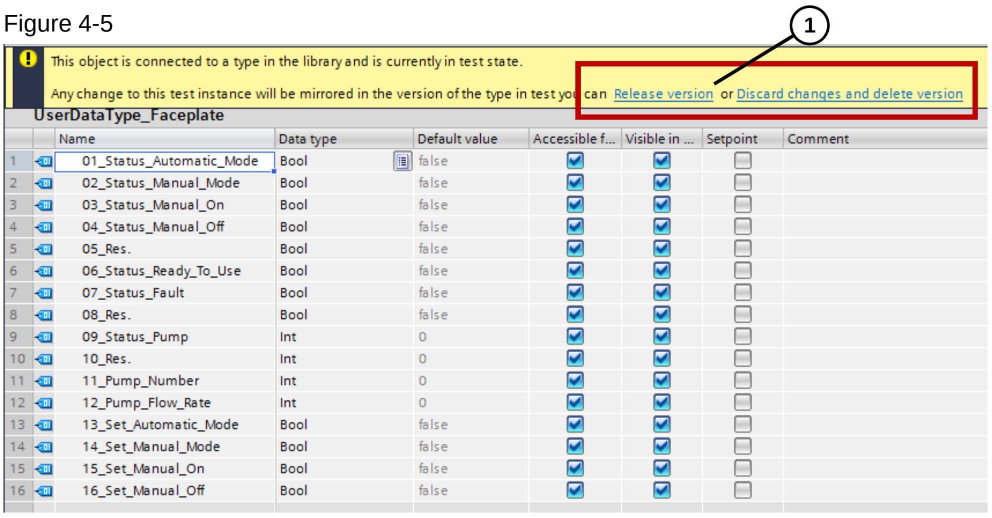
"PLC data type" not yet added into a library

- Right-click on the PLC data type to be edited to select it.
- Select the "Open" item from the context menu.
- Make the required changes and close the PLC data type again.

"PLC data type" added to a library

- Right-click on the PLC data type to be edited to select it.
- Select the "Edit type" item from the context menu.
- Make the required changes.
- Prior to closing the editor, select one of the following options.
  - Release version
  - Discard changes and release version

Figure 4-5



This object is connected to a type in the library and is currently in test state.

Any change to this test instance will be mirrored in the version of the type in test you can Release version or Discard changes and delete version

UserDataType_Faceplate

	Name	Data type	Default value	Accessible f...	Visible in ...	Setpoint	Comment
1	01_Status_Automatic_Mode	Bool	false	✓	✓	☐	
2	02_Status_Manual_Mode	Bool	false	✓	✓	☐	
3	03_Status_Manual_On	Bool	false	✓	✓	☐	
4	04_Status_Manual_Off	Bool	false	✓	✓	☐	
5	05_Res.	Bool	false	✓	✓	☐	
6	06_Status_Ready_To_Use	Bool	false	✓	✓	☐	
7	07_Status_Fault	Bool	false	✓	✓	☐	
8	08_Res.	Bool	false	✓	✓	☐	
9	09_Status_Pump	Int	0	✓	✓	☐	
10	10_Res.	Int	0	✓	✓	☐	
11	11_Pump_Number	Int	0	✓	✓	☐	
12	12_Pump_Flow_Rate	Int	0	✓	✓	☐	
13	13_Set_Automatic_Mode	Bool	false	✓	✓	☐	
14	14_Set_Manual_Mode	Bool	false	✓	✓	☐	
15	15_Set_Manual_On	Bool	false	✓	✓	☐	
16	16_Set_Manual_Off	Bool	false	✓	✓	☐	

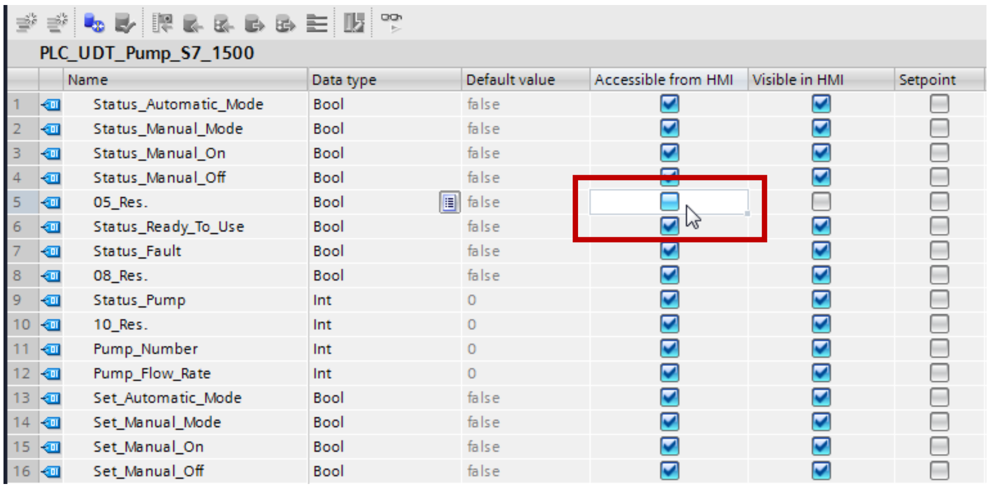
4.6 Tips for the configuration

4.6.1 Reducing power tags

When creating a PLC data type, not all of the created tags necessarily need to be used in the HMI configuration.

Use the option of deactivating no longer required tags for the HMI operator panel. This enables reducing the number of used power tags considerably.

Figure 4-6



PLC_UDT_Pump_S7_1500

	Name	Data type	Default value	Accessible from HMI	Visible in HMI	Setpoint
1	Status_Automatic_Mode	Bool	false	✓	✓	☐
2	Status_Manual_Mode	Bool	false	✓	✓	☐
3	Status_Manual_On	Bool	false	✓	✓	☐
4	Status_Manual_Off	Bool	false	✓	✓	☐
5	05_Res.	Bool	false	☐	☐	☐
6	Status_Ready_To_Use	Bool	false	✓	✓	☐
7	Status_Fault	Bool	false	✓	✓	☐
8	08_Res.	Bool	false	✓	✓	☐
9	Status_Pump	Int	0	✓	✓	☐
10	10_Res.	Int	0	✓	✓	☐
11	Pump_Number	Int	0	✓	✓	☐
12	Pump_Flow_Rate	Int	0	✓	✓	☐
13	Set_Automatic_Mode	Bool	false	✓	✓	☐
14	Set_Manual_Mode	Bool	false	✓	✓	☐
15	Set_Manual_On	Bool	false	✓	✓	☐
16	Set_Manual_Off	Bool	false	✓	✓	☐

4.6.2 Change-over from HMI UDT to PLC data types

You have an existing configuration with a faceplate in which an "HMI UDT" is used. This "HMI UDT" shall be replaced by a PLC data type.

The least workload occurs if the "PLC data type" has the same structure as the HMI UDT. In this case, you can replace the HMI UDT 1:1 with the PLC data types.

If the "HMI UDT" is replaced by a PLC data type with a different structure to the "HMI UDT", the connections to the individual objects must be recreated in the faceplate editor.

Note As of WinCC (TIA Portal) V13 SP1 and in connection with a SIMATIC S7-1200 or S7-1500 controller, PLC data types can be used.

Procedure

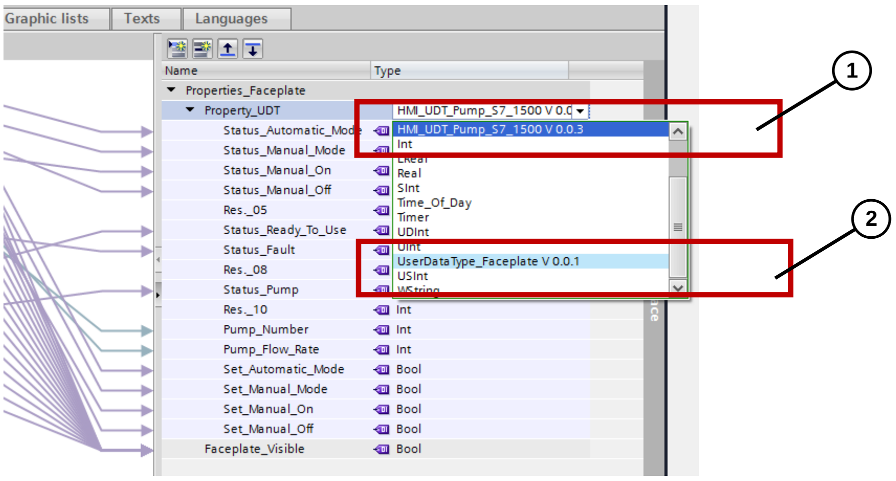
- Open the data block that has the structure of the "HMI UDT" and copy the respective structure.
- Open or create a PLC data type.
- Add the previously copied "data structure" into the PLC data type.
- Drag and drop the created PLC data type into the project library.

4.6 Tips for the configuration

- Open the faceplate for which the "HMI UDT" shall be replaced by the PLC data type.
- In the configuration area in the "Properties > Interface" tab, you open the tag with the "HMI UDT".
- Select the created PLC data type (2) in "Type" via the dropdown menu. In this example, "UserDataTyp_Faceplate".
  If all of the data match, all assigned "connections" are adopted.

Note The name of the tag must not start with a numeral.

Figure 4-7



Name	Type
Properties_Faceplate	
Property_UDT	HMI_UDT_Pump_S7_1500 V 0.0
Status_Automatic_Mode	
Status_Manual_Mode	
Status_Manual_On	
Status_Manual_Off	
Res._05	
Status_Ready_To_Use	
Status_Fault	
Res._08	
Status_Pump	
Res._10	Int
Pump_Number	Int
Pump_Flow_Rate	Int
Set_Automatic_Mode	Bool
Set_Manual_Mode	Bool
Set_Manual_On	Bool
Set_Manual_Off	Bool
Faceplate_Visible	Bool

5 Links & Literature

Internet link specifications

This list is by no means complete and only presents a selection of suitable information.

Table 5-1

	Topic	Title
\1\	Link to this document	https://support.industry.siemens.com/cs/ww/en/view/68014632
\2\	Siemens Industry Online Support	Industry Online Support https://support.industry.siemens.com
\3\	FAQ	What are the functional differences between the different SIMATIC panels? https://support.industry.siemens.com/cs/ww/en/view/40227286
\4\	FAQ	How can you create faceplates in WinCC (TIA Portal) with user authorizations? https://support.industry.siemens.com/cs/ww/en/view/57434982
\5\	Application	Application: Sample blocks for STEP 7 and WinCC flexible https://support.industry.siemens.com/cs/ww/en/view/36435784
\6\	Application	Faceplates for the Visualization of Sentron PAC Power Meters. https://support.industry.siemens.com/cs/ww/en/view/67318600

6 History

Table 6-1

Version	Date	Modifications
V1.0	01/2013	First version
V2.0	06/2015	Update to WinCC V13 SP1 (new function – PLC data types)
V2.0	08/2017	Update to WinCC V14