SIEMENS

SIMATIC Visualization Architect

Application Manual

12/2017
Online help printout

Legal information

Warning notice system

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

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

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

NOTICE

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

⚠ WARNING

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Document order number: Online help printout
℗ 12/2017 Subject to change

Copyright © Siemens AG 2017.
All rights reserved

Table of contents

Security information

1

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.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, visit

http://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 under

http://www.siemens.com/industrialsecurity.

Network drive

Ensure that network drives are protected from unauthorized access in your network infrastructure and computers.

Communication via Ethernet

In Ethernet-based communication, end users themselves are responsible for the security of their data network. Proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to overload of the device.

See also

http://support.automation.siemens.com (http://www.siemens.com/industrialsecurity)

http://www.siemens.com/industrialsecurity (http://www.siemens.com/industrialsecurity)

Basics

2

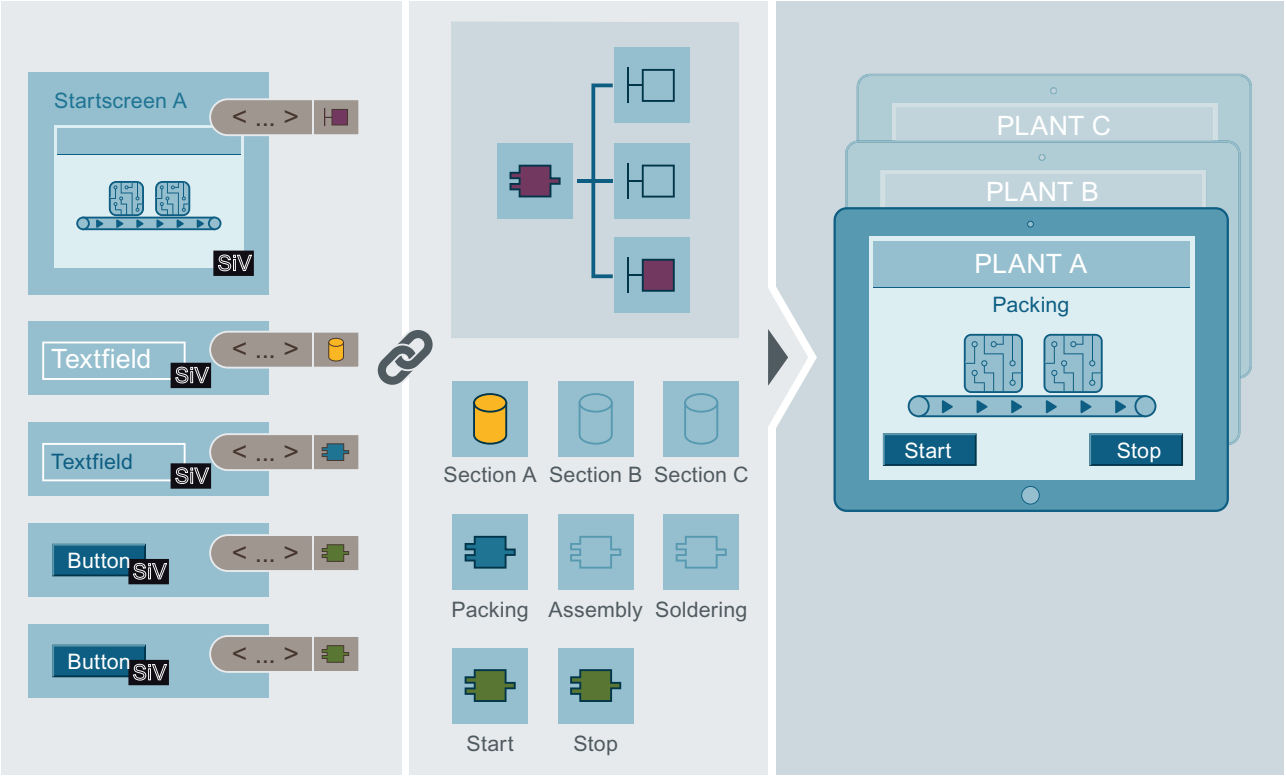
2.1 Introduction

What is SiVArc?

SiVArc (SIMATIC WinCC Visualization Architect) is an option package in the TIA Portal.

With SiVArc you generate the visualization for multiple HMI devices and PLCs from program blocks and generation templates.

You use generation rules to specify which HMI objects are generated for which blocks and devices.



Functional scope

You can generate the following HMI objects from controller data with SiVArc:

- Screens, faceplates and a selection of display and operating elements
- External tags
- HMI text lists

Without reference to the control program, you can generate a selection of objects from your WinCC project library with SiVArc to your WinCC project or use them as instance.

Use generation templates from the project library or global library for the generation.

SiVArc can simultaneously generate the visualization for multiple HMI devices, multiple PLCs and device proxies. While generating the visualization with SiVArc, you can continue working with TIA Portal in a second instance. With SiVArc and the TIA Portal option "TIA Portal Multiuser", you can also have different users work on a SiVArc project.

For further information regarding SiVArc automation task, refer to Siemens YouTube Channel (https://www.youtube.com/watch?v=txLWqFOmAIM).

For additional Information on SiVArc, please refer to Siemens Industry Online Support (https://support.industry.siemens.com/cs/ww/en/view/109751096).

See also

Supported objects in the user program (Page 84)

Configuring an HMI solution with SiVArc (Page 19)

Overview of SiVArc expressions (Page 87)

Picture legends (Page 231)

Creation of generation templates (Page 72)

2.2 Applications

Overview

You use SiVArc for automation solutions with a high degree of standardization.

SiVArc supports the configuration engineer during engineering with the following tasks:

- Automatic generation of the visualization including process connection
- Uniform layout of user interfaces
- Consistent naming of operating elements
- Structured storage of configuration data

SiVArc also offers support during the operating phase:

- Commissioning
  SiVArc helps during the commissioning, because a commissioning engineer can perform changes in the project at short notice using a generation matrix even without SiVArc expertise.
- Adaptations
  To apply changes to an entire project, you only have to adapt central templates with SiVArc.
- Plant maintenance
  The generation of specific individual devices, means for example that it is easy to exchange HMI devices.

SiVArc is also suitable to promote standardization in your project and continuously optimize your projects.

Advantages

The fundamental added value of SiVArc compared to conventional configuration of visualization consists of the following SiVArc principles:

- The generated visualization retains the reference to the SiVArc project. Adaptations and optimizations with SiVArc ensures high-performance and clearly structured database.
- The visualization is linked directly to the user program. Changes in the user program require only minimum adaptations in the WinCC project.
- Layout, design and the consistent designation in the display is centrally controlled across STEP 7 and WinCC.

Requirements on the configuration engineer

The following prior knowledge is required to use SiVArc:

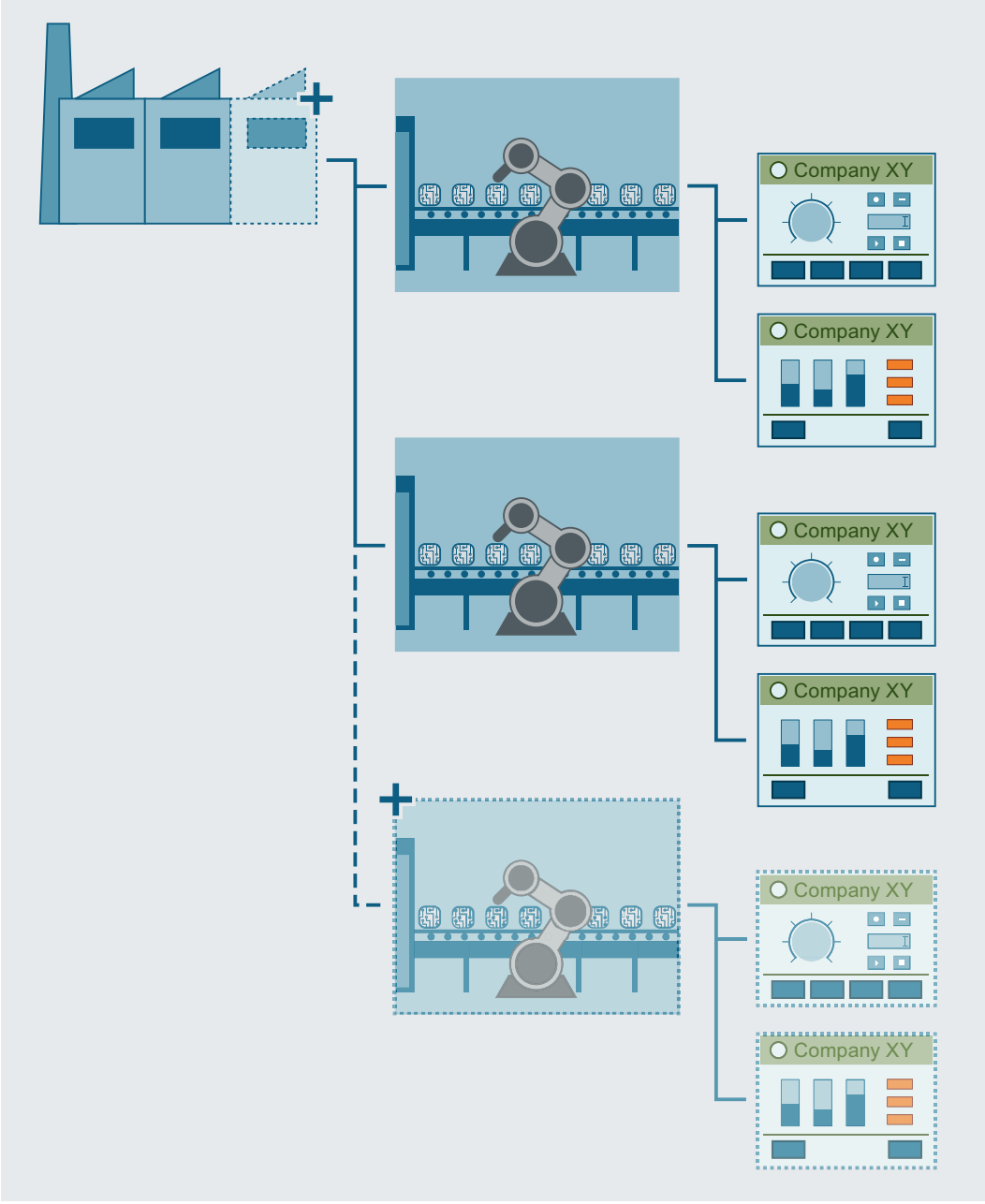
- You have configuration experience in STEP 7 and WinCC.
- You have a basic knowledge of Visual Basic Script (VBS).

2.3 Example: Using SiVArc to generate the visualization

Example scenario

An existing plant for printed circuit boards is to be expanded with a third production line. The company commissioned an external engineering office for the visualization of the expansion based on the existing control program and the existing visualization design. The task for the engineering firm includes the following requirements:

- The customer is new. There is no prior project.
- The new corporate design of the customer is to be incorporated into the visualization.
- The customer wants to optimize standardization within the company.



Designing a parameterization solution

The engineering firm decides to adapt an existing SiVArc sample project for implementation of the task. To achieve this, they assign a PLC programmer and a visualization expert the task of analyzing the user program and the visualization design.

Together, they define the following:

- Number of layout templates depending on the HMI devices used
- Required external tags

- Naming conventions for naming external tags
- Text sources in the user program which are used in the visualization
- Assignment of program blocks to generation templates
- Structure of SiVArc expressions in the generation templates
- Storage structures in the SiVArc project

Then the visualization expert determines the number and type of required generation templates, SiVArc rules and SiVArc tags.

Implementation of parameterization solution

The PLC programmer adapts the user program to the design solution:

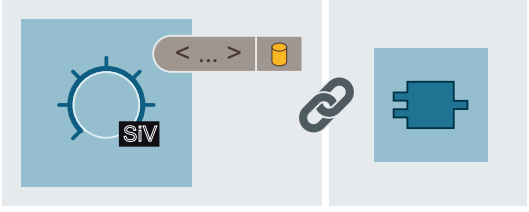
- Set PLC tags to "Accessible from HMI"
- Check the text sources in STEP 7 and make them consistent, if necessary
- Optimize the storage of the program blocks in the project tree
- Expand existing libraries

The visualization expert implements the corporate design of the customer for several HMI devices on screen templates.

Based on the screen templates, a positioning scheme is created for each device.

The generation templates of the example project for standard objects are adapted to the visualization design.

The generation templates and function blocks are linked to each other in the screen rules.



SiV	SiVArc generation template
< ... >	SiVArc property with referenced text source, e.g., a data block
	Function block

Result

A new, customer-specific and agile SiVArc project was created based on a SiVArc sample project. Further expansions of the plant and the user program now only require minimal interventions in the SiVArc project.

2.4 Example: Using SiVArc to generate tags

Example scenario

Plant builders often experience unplanned delays during commissioning. Analyses have revealed that existing naming conventions for tags are not consistently implemented. Recreating the tags places a heavy load on the storage volume of the HMI devices.

The company turns to an engineering firm to standardize the tag names and re-link them.

The downtime should be minimized and free space made available on the HMI devices.

Solution concept

The engineering firm analyzes the user program and sets the required tags to "Accessible from HMI".

Depending on the type of PLC tags, UDTs or arrays used, the engineer configures the synchronization of the tag names.

SiVArc starts the generation of the tags. SiVArc only generates the tags necessary for visualization.

The desired tag names are generated via SiVArc expressions based on the naming concept for tags.

Note

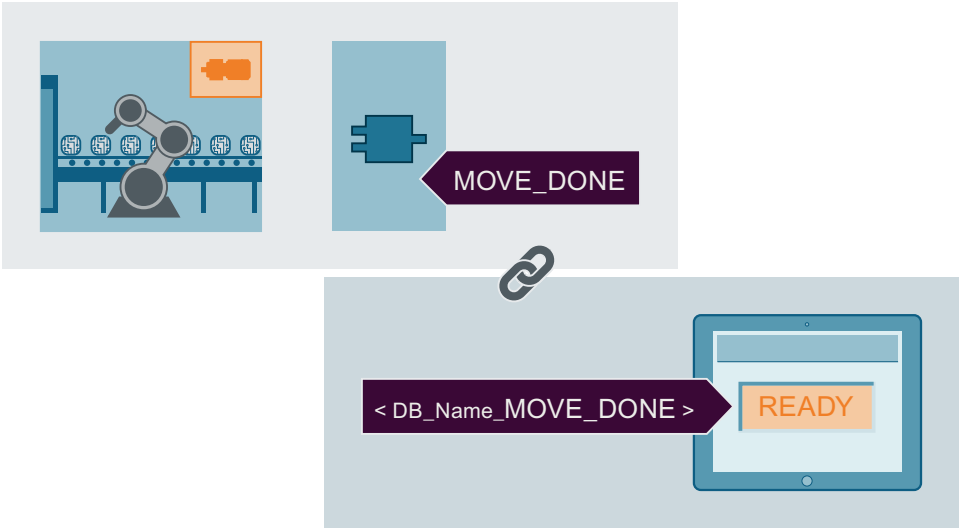
Tag names

WinCC supports fewer characters than STEP 7. If you use a character in the PLC tag name that is not supported by WinCC, this character is deleted when the name of the external tag is generated. The result may be multiple tags with the same name. This generates an error because SiVArc does not generate tags with the same name.

Only use characters supported by WinCC when assigning names for PLC tags.

Result

The required tags have been uniformly named. The reference to the PLC tags can be read at the point of interconnection in the WinCC project.



Function block

Only the genuinely necessary tags are included in the WinCC project. Further processing and continuous adaptation of the tags is possible with SiVArc.

See also

Principle of the tag generation (Page 149)

2.5 Configuring an HMI solution with SiVArc

Introduction

Configuring HMI solutions with SiVArc requires a standardized project. The more standardized a project, the easier and more effective the use of SiVArc to create the visualization.

Requirement

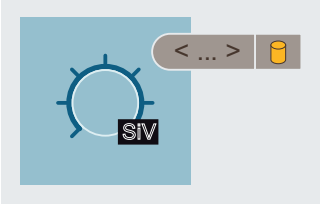
- The plant is a standard facility.
- The structured user program is created.
- A visualization and operating concept is created.
- Standard blocks in the user program are accessible via libraries.

- Faceplates for standard applications are accessible via libraries.
- The project is standardized and transferable.

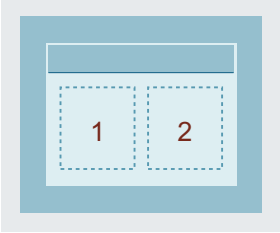
Procedure

To generate HMI solutions with SiVArc, follow these steps:

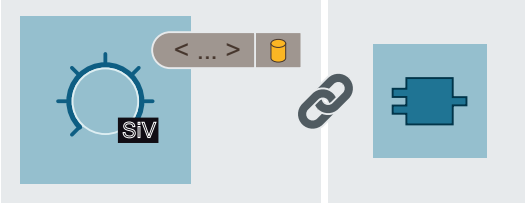
1. Design the layout.
   - Which HMI devices are in use?
   - How is the corporate design implemented in the screen?
   - How many generation templates for screens are required?
   - How many positioning schemes are required?
2. Define which external tags are going to be generated by SiVArc.
3. Create the generation templates for HMI objects and store them in the library.



4. Create the positioning schemes for screens and store them in the library.



5. You create screen rules to link the generation templates with the function blocks.

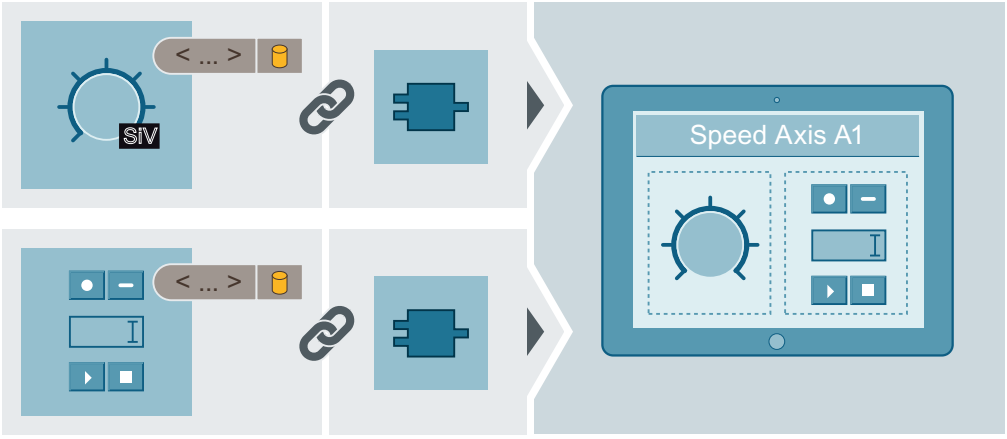


6. You create tag rules to control the storage of generated tags.
7. You create copy rules to copy collected HMI objects from the library into the project.
8. Define the text list entries.
9. Generate the full visualization or only for selected devices.

Result

The generated HMI objects are created in the project tree and marked as generated objects.

The generated screen objects are arranged according to their positioning schemes in the generated screens.



SiV SiVArc generation template

< ... > SiVArc property with referenced text source, e.g., a data block

Function block

Further processing

Note

Subsequent name changes of generated objects

If the name of a generated HMI object has been changed, the object is created and interconnected again at the next SiVArc generation. The renamed object remains available.

Change the names of generated objects only in the user program.

Restart the generation after each change to the SiVArc project or the user program.

SiVArc processes the changed information and replaces the existing generation and generates additional HMI objects, if necessary.

See also

Plan layout (Page 39)

Creation of generation templates (Page 72)

Creating rules (Page 132)

Generate visualization (Page 172)

Installation 3

3.1 Installing SiVArc

Introduction

The setup program of the "SiVArc" add-on package starts once the installation medium has been inserted into the respective drive.

You need a valid license to install SiVArc. Use the "Automation License Manager" to manage your license keys.

Note

Version compatibility

Your SiVArc version is only compatible with the corresponding version of STEP 7 and WinCC Professional or WinCC Advanced.

To upgrade your version of the TIA Portal, you must also upgrade your version of SiVArc and vice versa. If you uninstall WinCC or STEP 7, SiVArc is also uninstalled.

Select a side-by-side installation to work with different versions of the TIA Portal.

Note

Windows 10 system requirements

In order to be able to use the complete functionality of SiVArc in Windows 10, you need to install Windows 10 Enterprise Version 1703.

You can find additional information on the topic of "System requirements" in the online help of the TIA Portal.

Requirement

- STEP7 Professional V15 is installed.
- SIMATIC WinCC Professional V15 or SIMATIC WinCC Advanced V15 is installed.

Procedure

To install the "SiVArc" add-on package, follow these steps:

1. Place the installation data medium in the drive.
   To start Setup manually, double-click the "Start.exe" file in the Explorer.
2. Select an installation language and click "Next."
3. Select the product you want and click "Next."

4. To continue the installation, read and accept all license agreements and click "Next".
   If the TIA Portal security and permission settings prevent installation, the security settings dialog opens.
5. To continue the installation, accept the changes to the security and permission settings.
6. Check the selected installation settings in the overview.
7. Change your settings as required and then click "Install".
   Installation is started.
   The completion of the installation is displayed.
8. Reboot your PC if required or exit Setup.

Result

The "SiVArc" add-on package is installed on your PC.

How to handle existing SiVArc projects

With a basic installation, existing SiVArc projects can be opened in the TIA Portal, even without a SiVArc installation.

If you then open the project with SiVArc, all SiVArc functions are active again.

To upgrade a SiVArc project, you require a SiVArc installation.

A basic installation consists of the following software packages:

- STEP7 Professional
- SIMATIC WinCC Professional
  or
- SIMATIC WinCC Advanced

To remove the reference to SiVArc in your project, delete all SiVArc configurations from your project. When you open the project with a basic installation, information about the missing SiVArc-installation will no longer be output.

Elements and basic settings

4

4.1 SiVArc editors

4.1.1 "Screen rules" editor

Description

In the "Screen rules" editor, you define the screen rules according to which SiVArc HMI objects are generated in screens for various devices. A rule is made up as follows:

- Name
  Unique name of the screen rule
- Program block
  FB or FC that is invoked at any position in the user program.
- Screen object
  Master copy or type of the HMI object that is generated. The master copy or type must be stored in a library.
- Screen
  Generation template of the screen on which the HMI object is generated. The generation template must be stored in a library.
- Layout field
  Layout field that is included in the positioning scheme of the screen. Use the layout field to specify the positioning of the HMI object to be generated.
- Condition (optional)
  SiVArc expression that is evaluated when processing this screen rule. If no condition is specified, the screen rule is always executed. The condition applies collectively for a rule group. You can refine the condition for individual rules of the rule group.
- Comment (optional)
  Individual comment for screen rule

Display the following columns as required via the icons in the toolbar:

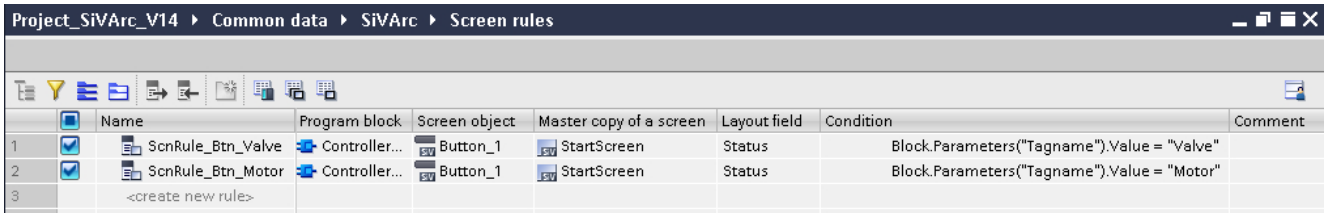
- PLC
  The screen rule is executed for the selected controllers. If you do not select any controller, the rule applies to all controllers in your project.
- HMI device
  The screen rule is executed for the selected HMI devices. If you select no HMI device, the rule applies to all HMI devices in your project.
- HMI device type
  If multiple HMI devices of the same type are available in your project, you can also select types of HMI devices. During generation it is checked and indicated whether a rule can be applied to an HMI device or to a controller.

If you want to generate a screen without a screen object for a program block, leave the "Screen object" field blank.

Example

You can use a program block to control a valve or motor. A button labeled "Open valve" or "Start engine" is to be generated depending on the use of the program block.

You need a screen rule for the valve symbol and for the motor symbol.



		Name	Program block	Screen object	Master copy of a screen	Layout field	Condition	Comment
1	☑	ScnRule_Btn_Valve	Controller...	Button_1	StartScreen	Status	Block.Parameters("Tagname").Value = "Valve"	
2	☑	ScnRule_Btn_Motor	Controller...	Button_1	StartScreen	Status	Block.Parameters("Tagname").Value = "Motor"	
3		<create new rule>						

When the program block is processed by SiVArc during generation of the HMI objects, SiVArc evaluates the condition of each screen rule. In this example, the use of the program block is defined by an input, for example `Block.Parameters("Tag name").Value = "Valve"`. In this case, the condition of the first screen rule applies, which then generates the button labeled "Open valve".

See also

SiVArc tags (Page 88)

Editing the view in the SiVArc editors (Page 229)

Exporting and importing SiVArc rules (Page 168)

Editing and managing SiVArc rules (Page 166)

4.1.2 "Tag rules" editor

Description

In the "Tag rules" editor, you define tag rules according to which the external tags generated by SiVArc are stored in structured form.

Double-click "Common data > SiVArc > tag rules" in the project tree to open the "Tag rules" editor.

A tag rule is contains the following elements:

- Name
  Unique name of the tag rule
- Index
  Specifies the order in which the rules are executed. You change the index using drag-and-drop in the table rows.

- Tag group
  Name of the tag group in which the external tag is generated
- Tag table
  Name of the tag table in which the external tag is generated
- Condition (optional)
  SiVArc expression that is evaluated when processing this tag rule
- Comment (optional)
  Individual comment for tag rule

See also

SiVArc object properties (Page 211)

Editing the view in the SiVArc editors (Page 229)

Exporting and importing SiVArc rules (Page 168)

Editing and managing SiVArc rules (Page 166)

"Copy rules" editor (Page 30)

Checking result (Page 184)

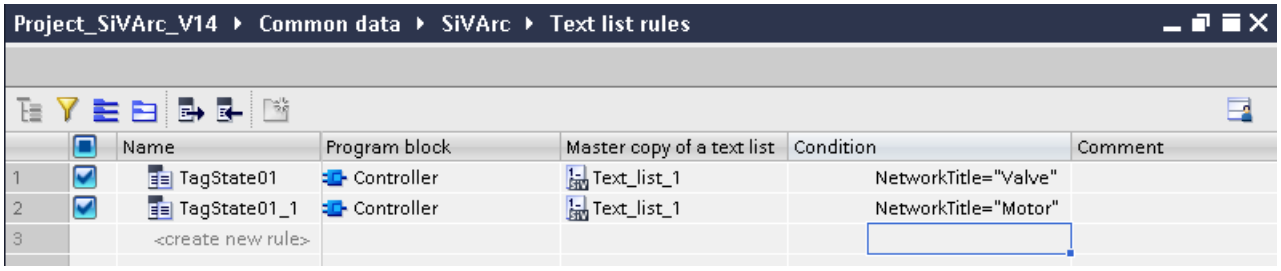
4.1.3 "Text list rules" editor

Description

In the "Text list rules" editor, you define SiVArc rules according to which text lists are generated for various devices. A text list rule is made up as follows:

- Name
  Unique name of the text list rule
- Program block
  FB or FC that is invoked at any position in the user program.
- Text list
  Master copies of text lists are saved in the "Text and Graphic Lists" editor during generation.

- Condition (optional)
  SiVArc expression that is evaluated when processing this text list rule. If no condition is specified, the text list rule is always executed.
- Comment (optional)
  Individual comment for text list rule



See also

Editing the view in the SiVArc editors (Page 229)

Exporting and importing SiVArc rules (Page 168)

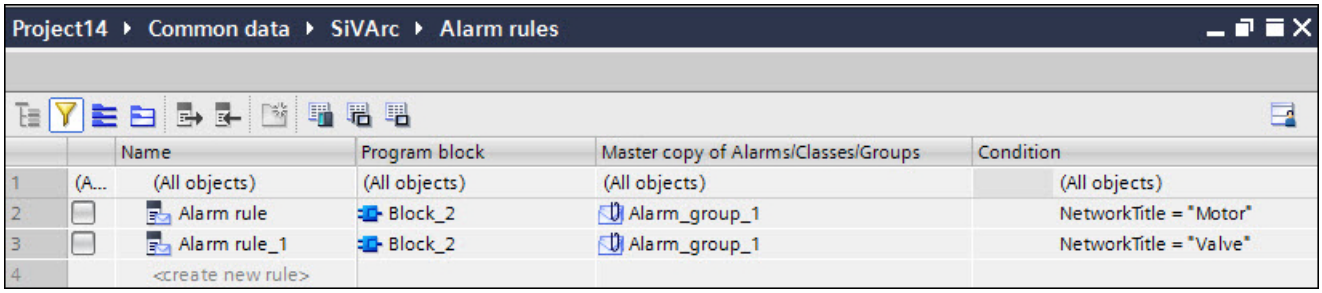
Editing and managing SiVArc rules (Page 166)

4.1.4 "Alarms rules" editor

Introduction

In the "Alarm rules" editor, you define SiVArc alarm rules according to which the alarms are generated for various HMI devices. The "Alarm rules" editor is made up of:

- Name - Unique name the alarm.
- Program block - Function block or function call that is invoked at any position in the user program.
- Master copy of Alarms/Classes/Groups - Master copies of alarms, classes or groups can be browsed and fetched.
- Condition (optional) - SiVArc expression that is evaluated when processing alarm rule. If no condition is specified, the alarm rule is always executed.
- Comment (optional) - Individual comment for alarm rule.



The following functionaities are avaialble in "Alarm rules" editor:

- Intellisense support
- Auto complete (drag the corners of the cell to auto complete values)
- Cut and copy/paste options

Requirements for alarms generation

- PLC built program blocks (FBs and FCs) are configured and connected to HMI device.
- Alarms are created and configured in "Alarms" editor in WinCC.
- Master copies (alarms or block types) are created and configured in global library.

You can configure and store alarm rules in the global library using the "Alarm rules" editor. During alarm rule configuration, the PLC blocks of a S7 device are mapped to the master copies stored in libraries, and conditions are defined. During SiVArc generation, based on the user defined conditions; the system processes the rules accordingly and generates the alarm objects.

Note

- The alarm objects are processed only after the rule satisfies the user defined condition.
- While choosing the program blocks, you can choose between PLC blocks or PLC types.

General notes

- You cannot delete the system generated alarm classes, but can only edit the alarm classes.
- SiVArc properties are completely tangential to the Engineering System properties. The options in the printout of the static value and printout of tags are similar to the Engineering System properties.
- You can also configure common alarm classes using SiVArc "Plug-in" editor.
- During SiVArc generation, if the resulting object name is similar to the default alarm class name, SiVArc displays an error.
- Alarm groups constituting similar classes are displayed in "Class group". SiVArc properties allows you to configure acknowledgement, assign status, ans suppression for alarm groups and sub-groups.
- Any newly added alarm classes will be automatically displayed in "Class group" area.
- In SiVArc, you can edit an existing alarm object.
- When identical alarm rules are created, system displays the following error:
  “Alarm rules for <’object name’>, is identical to <’object name’>. So, it is not considered for generation and can be deleted.”
- For enumeration type properties, cut, copy, paste will not be supported.

See also

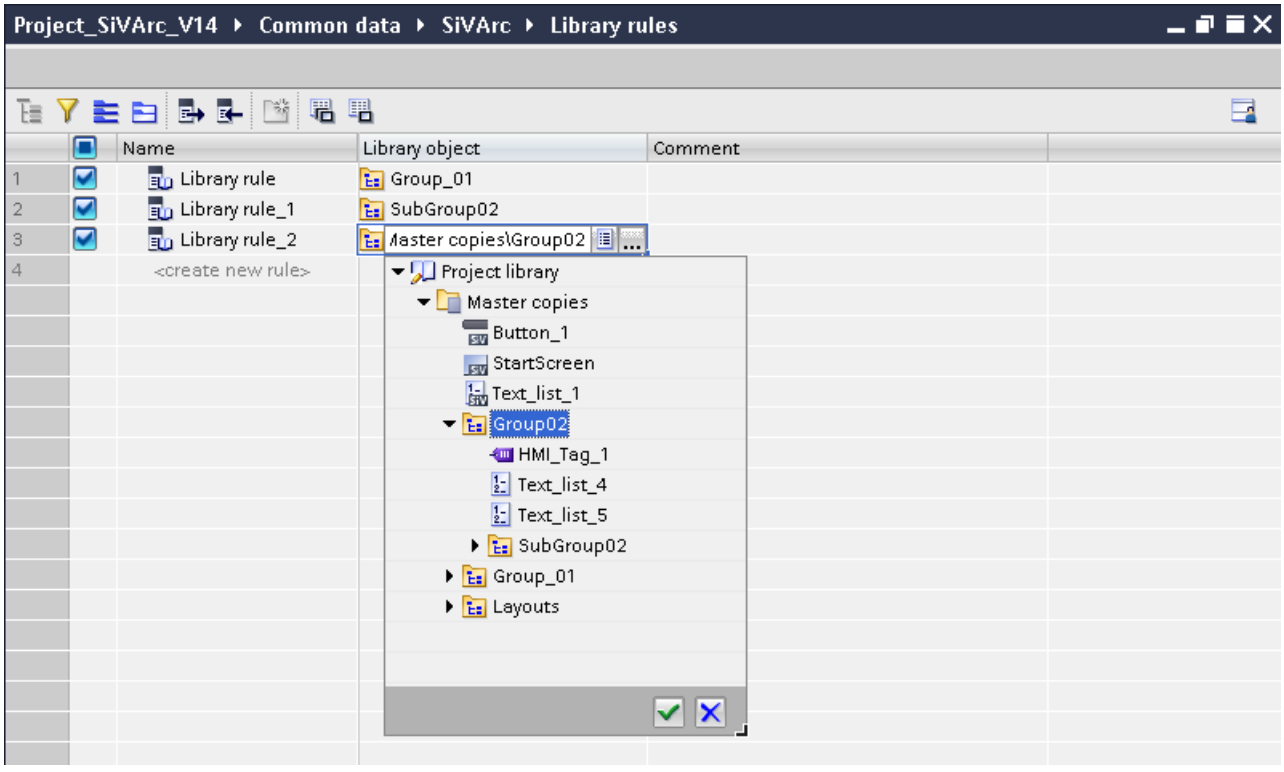
Creating alarm rules (Page 158)

4.1.5 "Copy rules" editor

Description

A rule is made up as follows:

- Name
  Unique name of copy rule
- Library object
  Master copy or type of an object that is generated or a library folder that contains library objects. The master copy or the type must be contained in the project library.
- Comment (optional)
  Individual comment for the copy rule



You show the following columns, when necessary, using the icons in the toolbar:

- HMI device
  The copy rule is executed for the selected HMI devices. If you select no HMI device, the rule applies to all HMI devices in your project.
- HMI device type
  If multiple HMI devices of the same type are available in your project, you can also select types of HMI devices. During generation it is checked and indicated whether a rule can be applied to an HMI device or to a controller.

See also

Editing the view in the SiVArc editors (Page 229)

Exporting and importing SiVArc rules (Page 168)

Editing and managing SiVArc rules (Page 166)

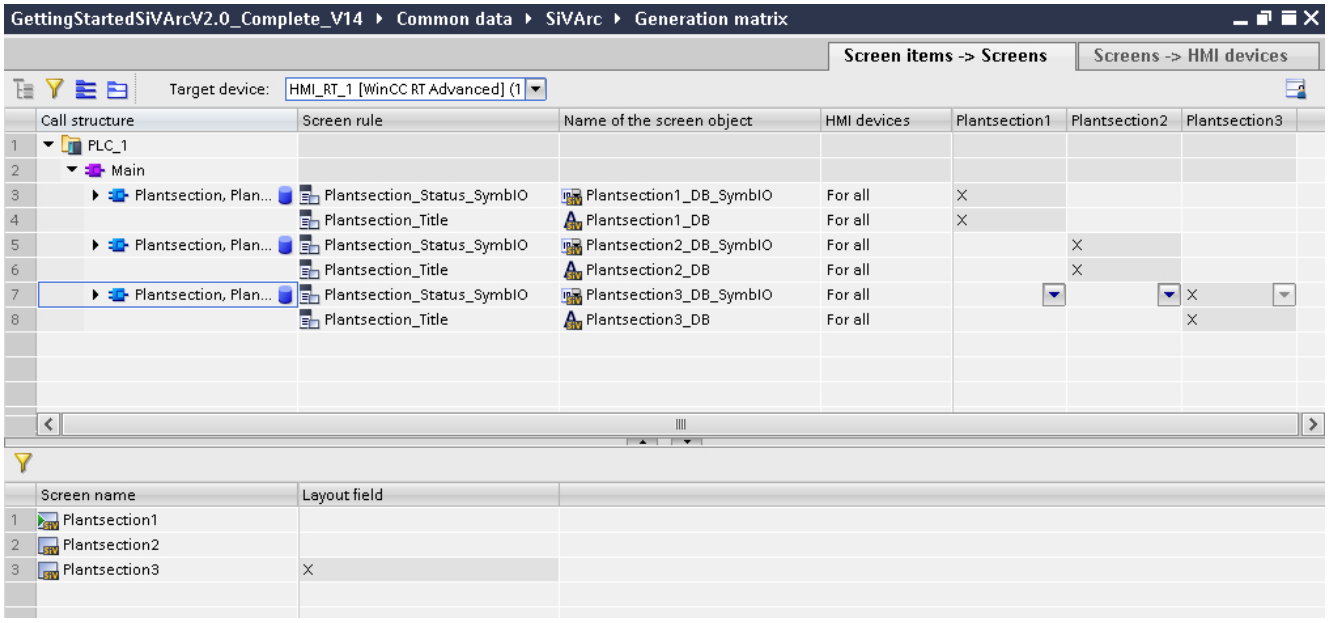
4.1.6 "Generation matrix" editor

Tab "Screen objects -> Screens"

In the toolbar of the editor, you select the HMI device for which the matrix is to be displayed under "Target device". SiVArc also displays the device type for all devices.

In this tab, assign a generated screen object to another screen. The tab contains the following columns:

- Call structure
  Shows for each line the block instances that are called in the user program and used for generating screen objects.
- Screen rule
  Shows the screen rules that were executed for each block instance.
- Name of the screen object
  Shows the generated screen object.
- HMI devices
  Lists for each screen object the HMI devices for which the screen object was generated.
- Screen columns
  A separate column is displayed for each screen. The columns are sorted alphabetically.
  - "X": Screen object is not positioned in a layout field.
  - "<Name of the layout field>": Screen object is contained in the specified layout field.

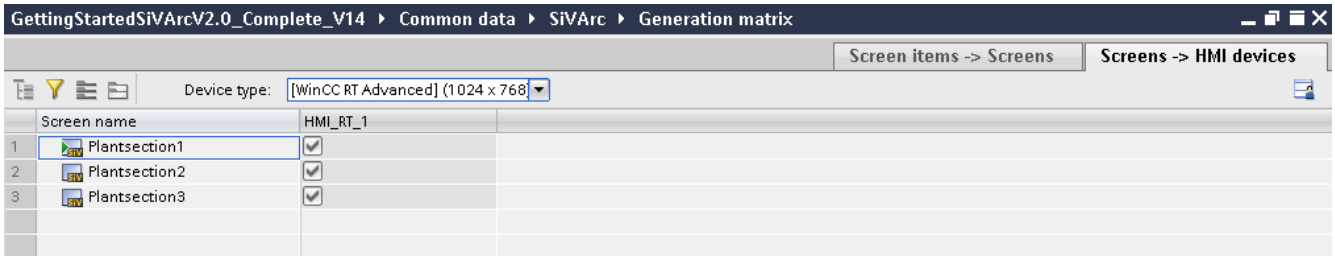


"Screens -> HMI devices" tab

In the toolbar of the editor, you select the HMI device type for which the matrix is to be displayed under "Device type". The editor then displays the screens of all HMI devices of this type.

On this tab, assign a generated screen to another HMI device. The tab contains the following columns:

- Screen
  Shows the generated screens.
- HMI devices
  Shows the HMI devices. A separate column is shown for each HMI device. The columns are sorted alphabetically.

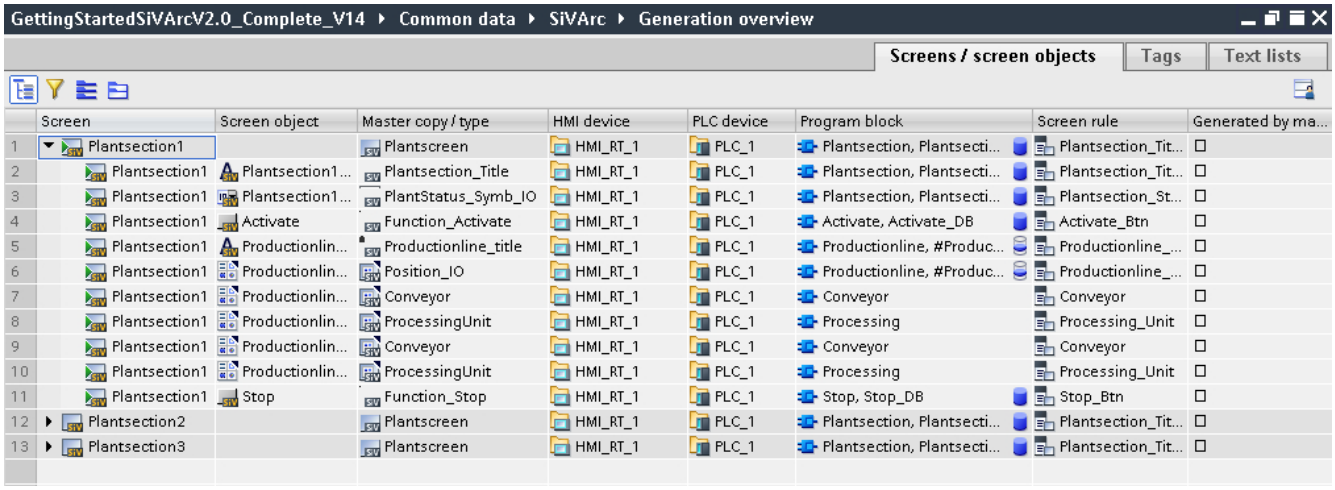


4.1.7 Generation overview

Description

After the initial generation of the visualization, all generated screen objects are listed in the generation overview. The generated objects are divided into the tabs "Screens/Screen objects", "Tags" and "Text lists".

The generation overview also displays, using various views, the relations between screen rules and generated objects after the generation. With the help of the generation overview, you plan and configure subsequent changes for an additional generation.



The contents of the generation overview are made up as follows:

"Screens/screen object" tab	"Tags" tab	"Text lists" tab
Name of the screen/screen object Unique name of the object	Name Name of generated tag table/generated tags	Text list/text list entry Name of the text list and its text list entries
Master copy/type Name of the generation template of the object	Data type Data type of the generated tags. The name of the UDT data type is shown for the "UDT" data type (PLC data type).	Master copy/type Name of the generation template for the text list
HMI device Name of the HMI device for which the object was generated	HMI device Name of the HMI device for which the external tags were generated	HMI device Name of the HMI device for which the text list was generated
PLC device Name of the PLC for which the object was generated	PLC device Name of the controller for which the tags were generated.	PLC device Name of the controller for which the text list was generated
Program block FB or FC for which the object was generated	Program block DB for which the tag was generated	Text Text that contains the text list entry
Screen rule Screen rule which defined the generation of the object	PLC tag Name of the PLC tag for which the external tag was generated.	Rule name Name of the text list rule which specified the generation of the text list
Date Time stamp on which the object was generated.	Tag table Name of the tag table in which the tags were generated	Network Name of the network which was evaluated during the generation
Generated by matrix Object was created in a downstream generation using the generation matrix.	Tag folder Name of the folder in the project tree in which the tag tables and tags were generated	Program block FB or FC for which the text list was generated

"Screens/screen object" tab	"Tags" tab	"Text lists" tab
Layout field If the object was generated in a layout field, the name of the field is displayed here.	Tag rule Tag rule which specified the storage structure of the generated tags	Call structure Call path in the cycle OB "Main1", which specified the generation of the text list
Call structure Path of the evaluated block in the call hierarchy in the user program (OB1)	---	---

4.2 SiVArc in the WinCC editors

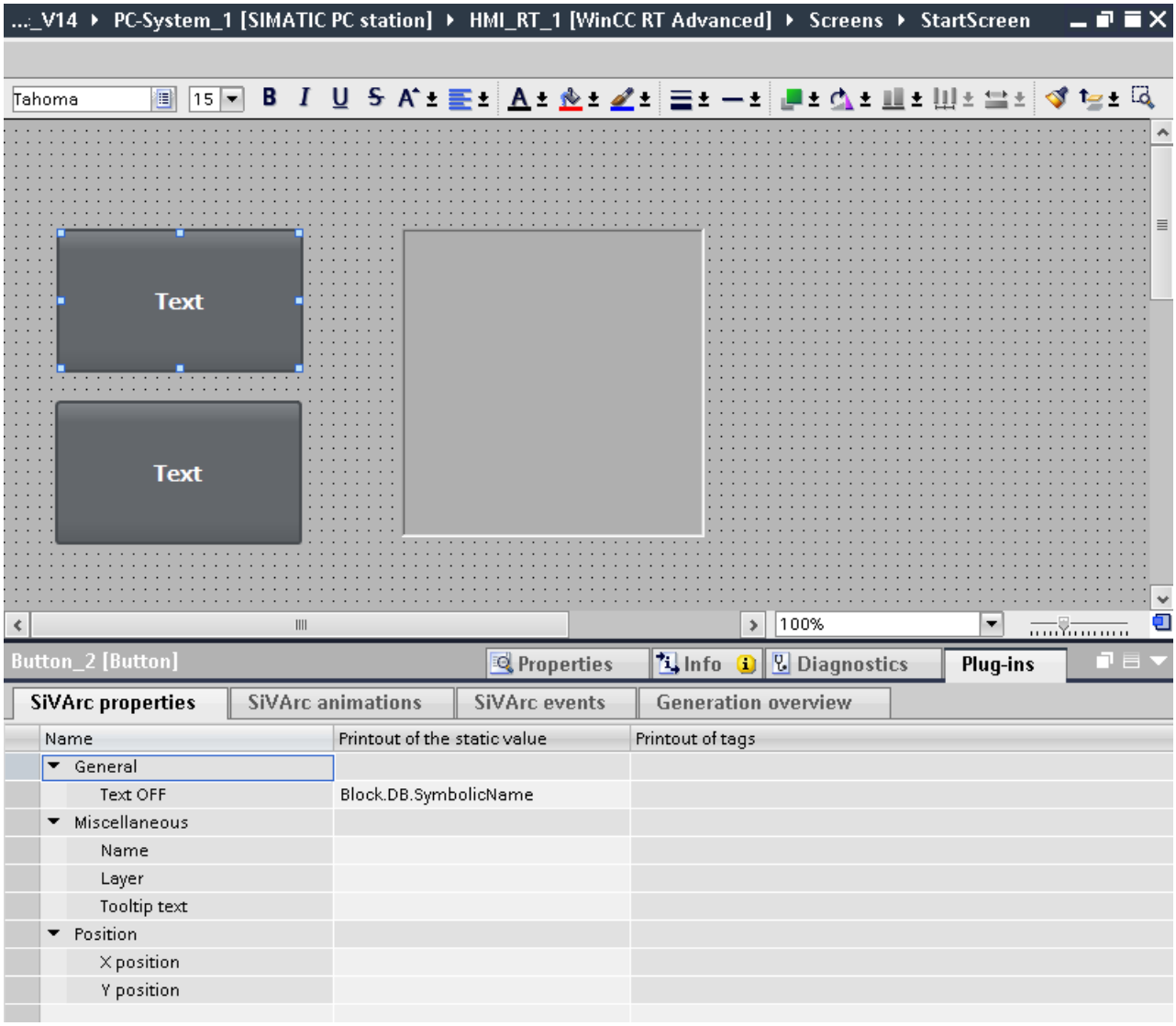
4.2.1 "SiVArc properties" tab

Description

A SiVArc property is an object property that you configure either statically or dynamically with a SiVArc expression.

In the "SiVArc properties" tab, you can configure the properties of a text list, a screen or a screen object with SiVArc expressions. You then store the configured object in the project library. The SiVArc expressions are evaluated during generation of the visualization.

The "SiVArc properties" tab is only available for objects supported by SiVArc.



Layout

The tab contains the three columns:

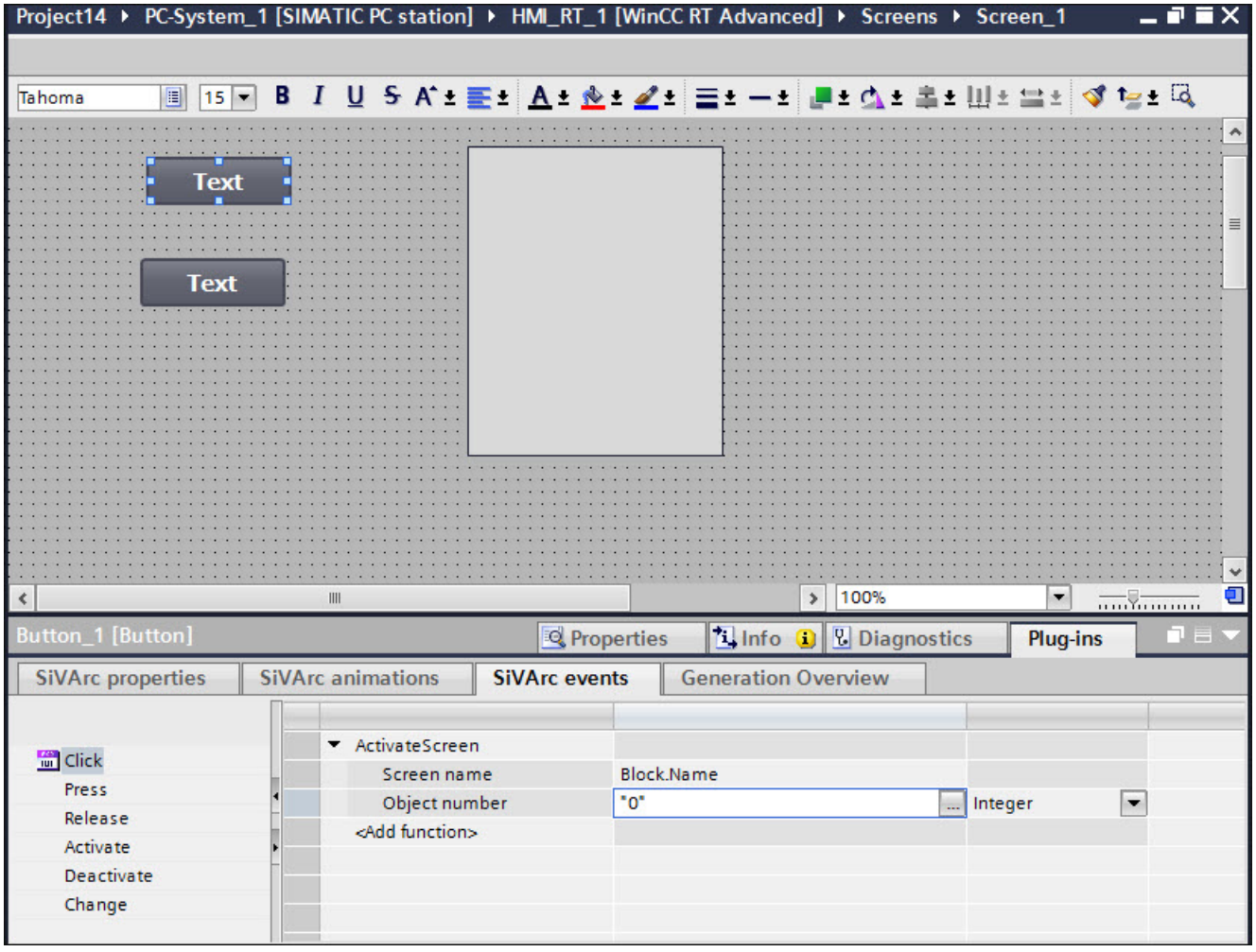
- Name
  This column lists the available properties.
- Expression of the static value
  In this column, you assign a property with a fixed value or a SiVArc expression that returns a string or a number.
  Fixed values are entered in every instance of this master copy when generating the visualization. Pay attention, for example, with the "Name" property that the uniqueness of the object name is ensured when it is used multiple times in an screen.
- Expression of the tag
  In this column, you assign a property with a tag name or a SiVArc expression that returns a tag name.

4.2.2 "SiVArc events" tab

Description

In the "SiVArc events" tab, you can configure a function list to an event of a generation template of a screen or screen object. You add system functions or a script to the function list.

You can configure the parameters of the system function or script with SiVArc expressions.



Layout

Column 1: Select the function or script in column 1.

Column 2: Enter a SiVArc expression in column 2.

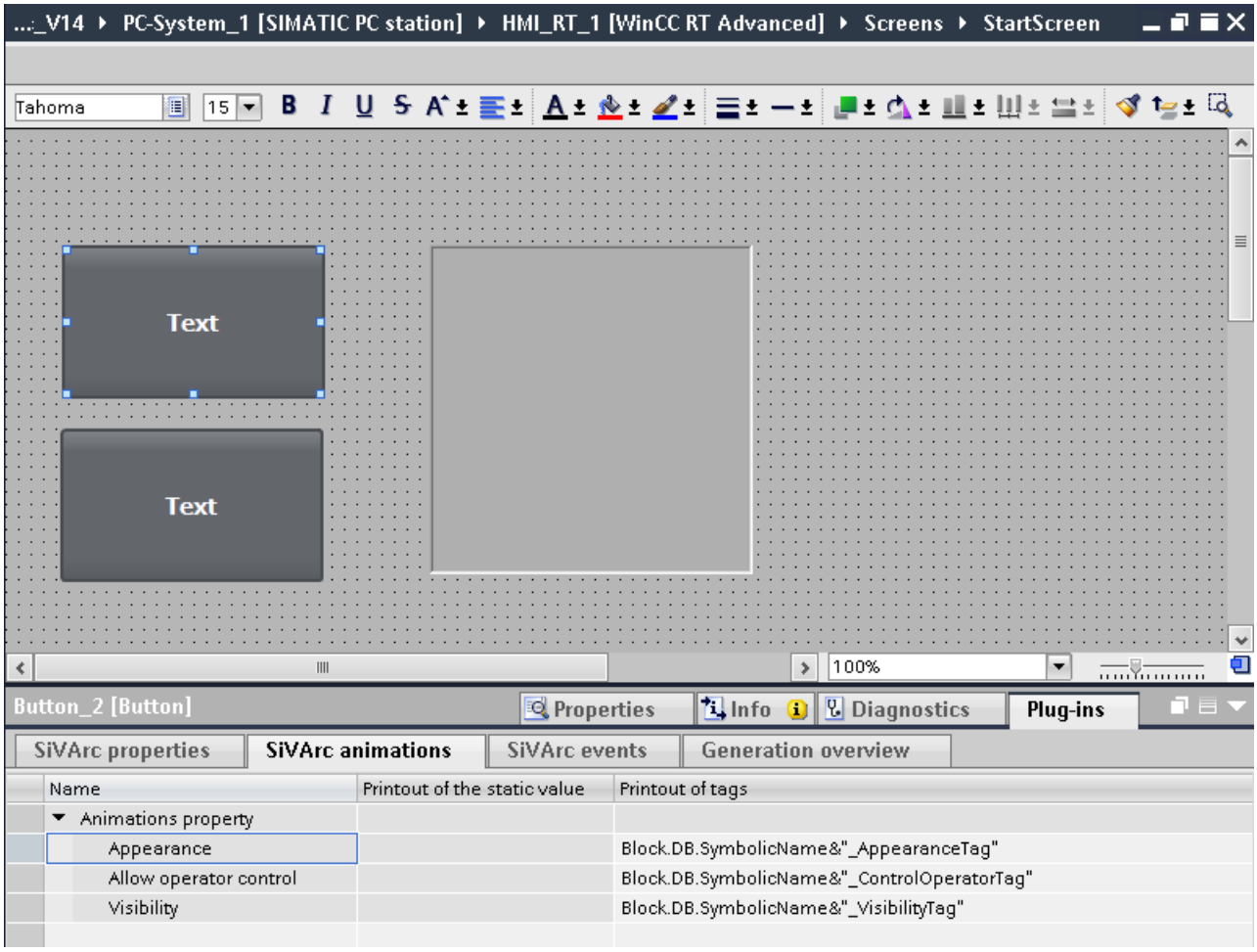
Column 3: Once you have selected a script, select a data type in column 3.

4.2.3 "SiVArc animations" tab

Description

Animations configured on the screen object are listed in the "SiVArc animations" tab.

The "SiVArc animations" tab is only available for HMI objects supported by SiVArc.



Layout

The "SiVArc animations" tab contains the following columns:

- Name
  The animations configured under "Properties > Animations" are listed in this column.
- Expression of the static value
  This column cannot be edited for animations.
- Expression of the tag
  In this column, you configure the process tags for the animation with a SiVArc expression. The SiVArc expression must return a tag name.

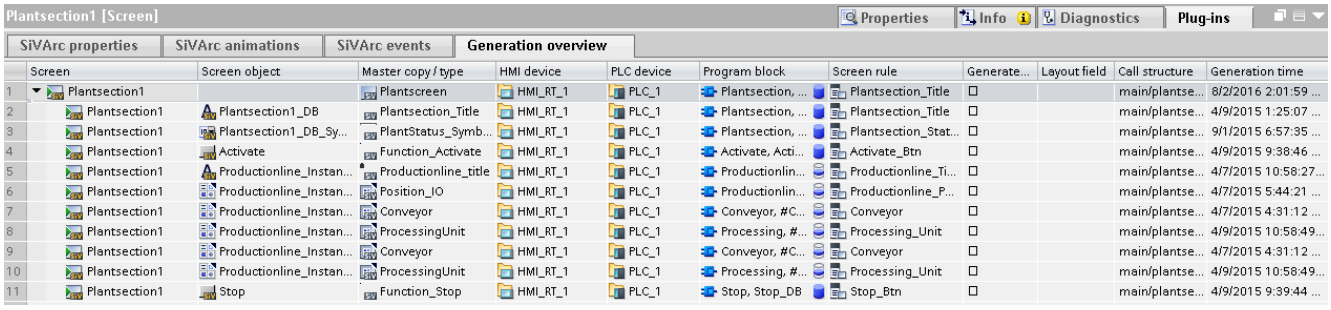
4.2.4 "Generation overview" tab

Description

After the first generation, the "Generation overview" tab is displayed in the Inspector window of a generated screen. The number of displayed objects is limited to the display and operating objects generated in the selected screen.

With the following exceptions, the "Generation overview" tab contains the same editing options as the "Generation overview" SiVArc editor:

- Filter function
- Sorting function
- "Open all" and "Expand all" buttons



	Screen	Screen object	Master copy / type	HMI device	PLC device	Program block	Screen rule	Generate...	Layout field	Call structure	Generation time
1	Plantsection1		Plantscreen	HMI_RT_1	PLC_1	Plantsection, ...	Plantsection_Title	☐		main/plantse...	8/2/2016 2:01:59 ...
2	Plantsection1	Plantsection1_DB	Plantsection_Title	HMI_RT_1	PLC_1	Plantsection, ...	Plantsection_Title	☐		main/plantse...	4/9/2015 1:25:07 ...
3	Plantsection1	Plantsection1_DB_Sy...	PlantStatus_Symb...	HMI_RT_1	PLC_1	Plantsection, ...	Plantsection_Stat...	☐		main/plantse...	9/1/2015 6:57:35 ...
4	Plantsection1	Activate	Function_Activate	HMI_RT_1	PLC_1	Activate, Acti...	Activate_Btn	☐		main/plantse...	4/9/2015 9:38:46 ...
5	Plantsection1	Productionline_Instan...	Productionline_title	HMI_RT_1	PLC_1	Productionlin...	Productionline_Ti...	☐		main/plantse...	4/7/2015 10:58:27...
6	Plantsection1	Productionline_Instan...	Position_IO	HMI_RT_1	PLC_1	Productionlin...	Productionline_P...	☐		main/plantse...	4/7/2015 5:44:21 ...
7	Plantsection1	Productionline_Instan...	Conveyor	HMI_RT_1	PLC_1	Conveyor, #C...	Conveyor	☐		main/plantse...	4/7/2015 4:31:12 ...
8	Plantsection1	Productionline_Instan...	ProcessingUnit	HMI_RT_1	PLC_1	Processing, #...	Processing_Unit	☐		main/plantse...	4/9/2015 10:58:49...
9	Plantsection1	Productionline_Instan...	Conveyor	HMI_RT_1	PLC_1	Conveyor, #C...	Conveyor	☐		main/plantse...	4/7/2015 4:31:12 ...
10	Plantsection1	Productionline_Instan...	ProcessingUnit	HMI_RT_1	PLC_1	Processing, #...	Processing_Unit	☐		main/plantse...	4/9/2015 10:58:49...
11	Plantsection1	Stop	Function_Stop	HMI_RT_1	PLC_1	Stop, Stop_DB	Stop_Btn	☐		main/plantse...	4/9/2015 9:39:44 ...

Working with SiVArc

5

5.1 Plan layout

5.1.1 Positioning schemes

5.1.1.1 Overview for positioning of generated objects

Introduction

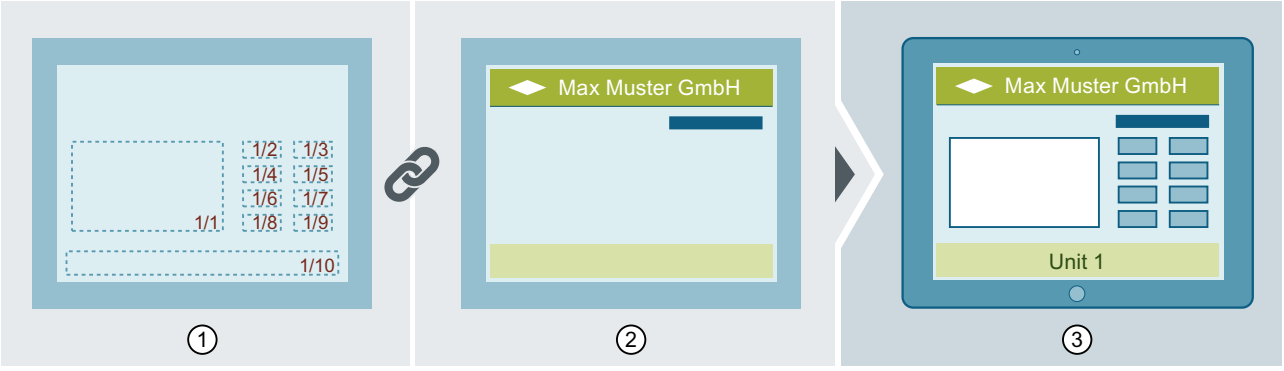
With SiVArc the layout of your process pictures is divided up into two tasks:

- Graphical design of the process pictures as well as display and operating objects as in WinCC.
- Positioning of the generated display and operating objects

SiVArc provides the methods described below to control the positioning of generated objects. It is true for all methods that manual position changes made after the first generation are retained for all subsequent generations.

Controlled positioning

You use positioning schemes to control the position at which the display and operating objects are generated in the process picture. To do so, you combine a generation template of a screen with the positioning fields of a defined scheme.



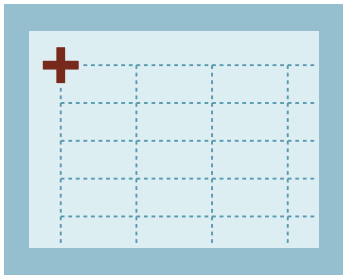
① User-defined positioning scheme with defined screen areas
② Generation template for a screen
③ Generated process screen

Free positioning

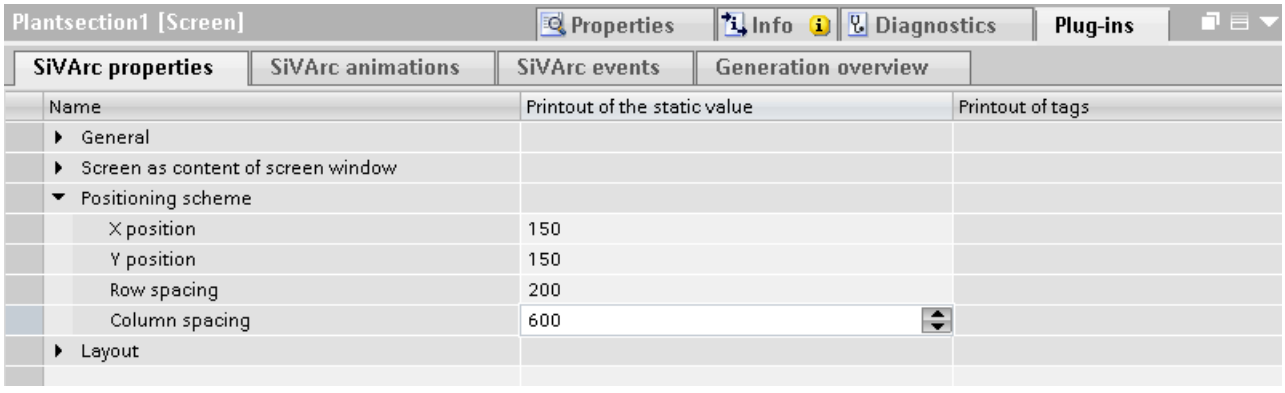
As an alternative to controlled positioning you can position the generated objects manually.

To do so, the SiVArc positioning scheme is stored for each generation template for screens. The SiVArc positioning scheme is a configurable grid.

The figure below shows the start position of the grid:

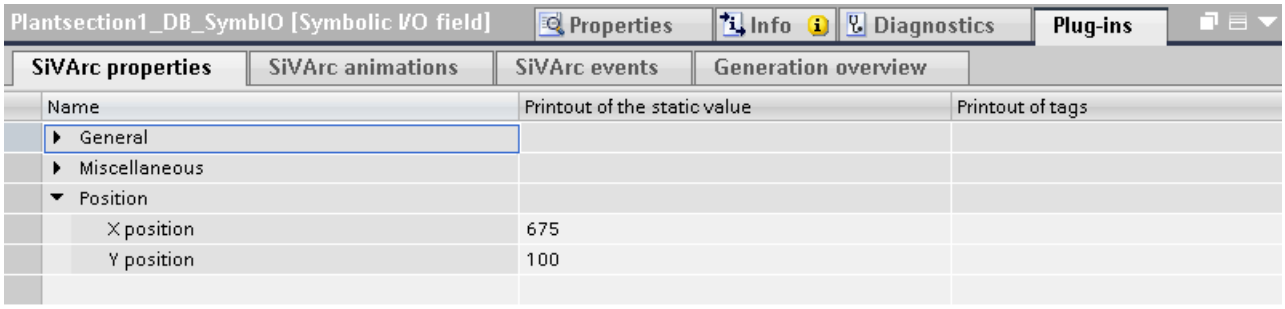


You set the start position of the grid along with the line and column spacing in the template.



After the generation, you can move the objects as needed. These positions are also retained during the next generation.

In addition you can define fixed position coordinates for individual objects:



Priority of the positioning methods

If a separate positioning scheme was stored for a display and operating object in the screen rules, all other specifications on the position are ignored during generation.

If you do not save a separate positioning scheme, the generated display and operating objects are arranged according to the fixed positioning or the SiVArc positioning scheme.

Generated objects with fixed positioning or with a positioning scheme cover display and operating objects already existing at a configured position.

SiVArc processes the individual positioning methods with the following priority:

1. Positioning scheme
2. Fixed positioning (SiVArc property)
3. Fixed positioning (WinCC property)
4. Free positioning

When you generate a display and operating object using a positioning scheme and position it in the screen, a fixed positioning configured at the object is ignored. The SiVArc positioning scheme is ignored as well.

See also

Positioning according to defined schemes (Page 41)

Fixed positioning of the generated object (Page 50)

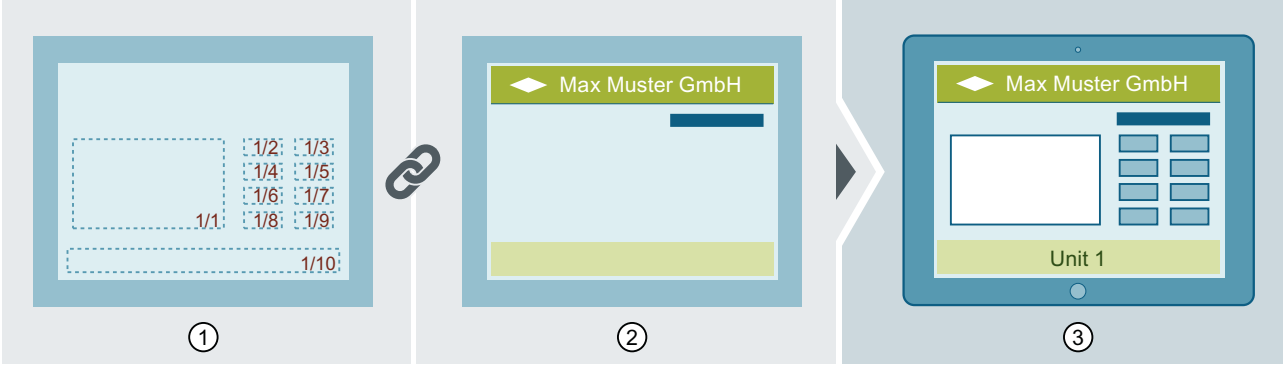
Free positioning (Page 51)

Designing a layout (Page 58)

5.1.1.2 Positioning according to defined schemes

Definition

A user-defined positioning scheme is a pixel-precise configurable grid. You can organize this grid into different areas. You assign the scheme to a screen. Display and operating objects are then generated by SiVArc directly in the desired area.



① User-defined positioning scheme with defined screen areas
② Generation template for a screen
③ Generated process screen

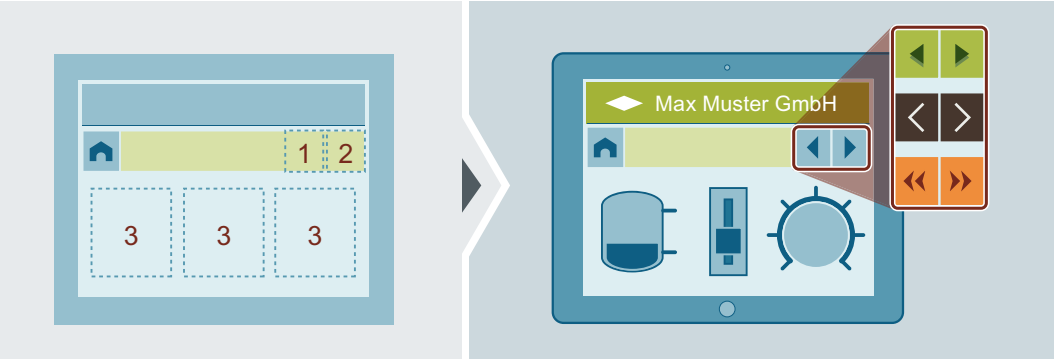
Use

Use your own positioning scheme from the library if your project requires a pixel-precise and standardized positioning of the display and operating objects.

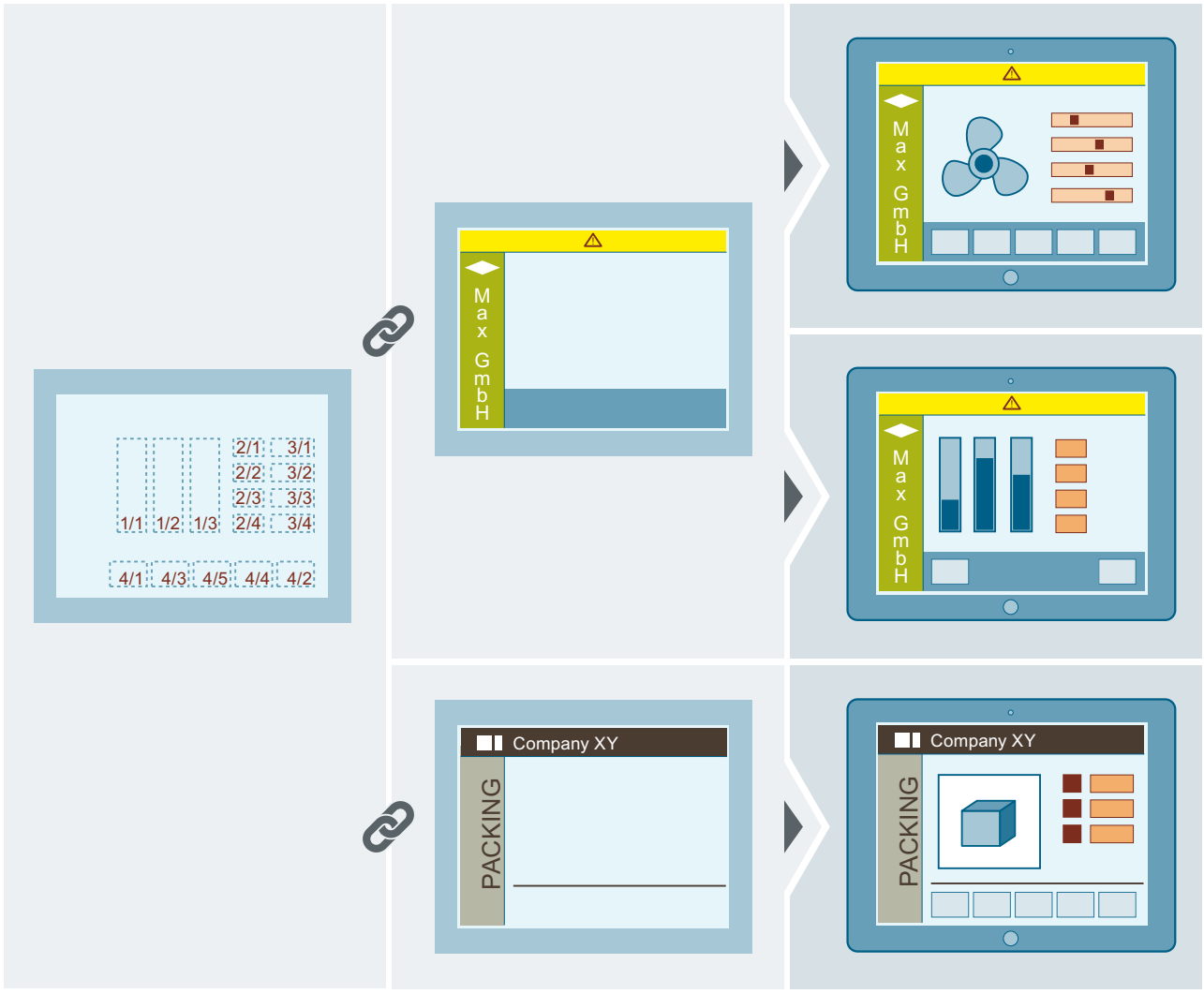
Advantages

A user-defined positioning scheme offers multiple benefits:

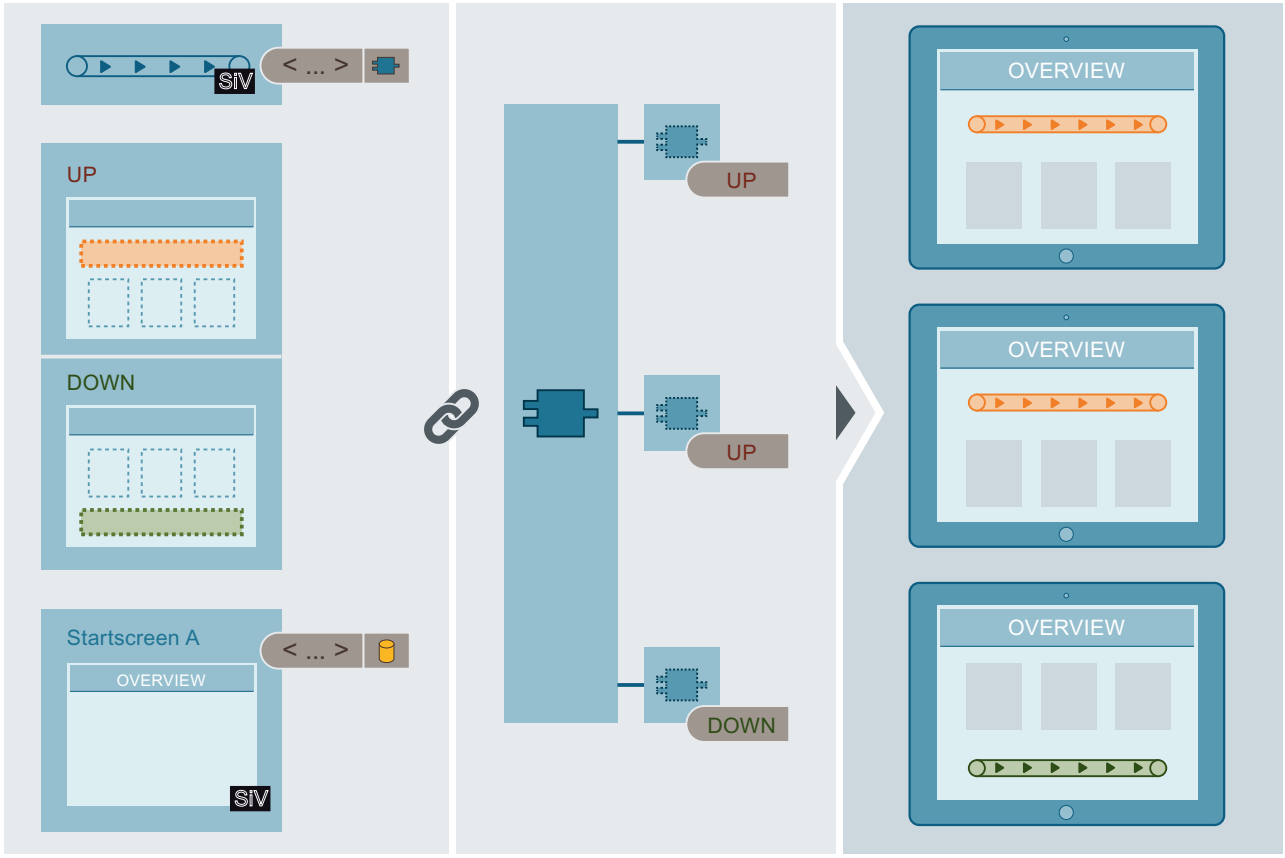
- Greater standardization in the project
  You use your own positioning schemes to control and manage the arrangement of the generated objects on different HMI devices. You can also place navigation buttons individually at the desired position within the scheme.



- Separation of layout and positioning
  You can also use existing positioning schemes across projects due to the separation of layout and positioning.


2/1
3/1
2/2
3/2
2/3
3/3
1/1
1/2
1/3
2/4
3/4
4/1
4/3
4/5
4/4
4/2
Max GmbH
Max GmbH
Max GmbH
Company XY
PACKING
Company XY
PACKING

- Dynamic positioning scheme
  A positioning scheme can be dynamically assigned to the screens, for example, using conditions in the user program. This reduces the number of screen rules in the SiVArc project.
  The figure below shows the screen rule for a conveyor belt graphic that is either positioned at the top or bottom in the generated screen depending on a SiVArc tag on the network:



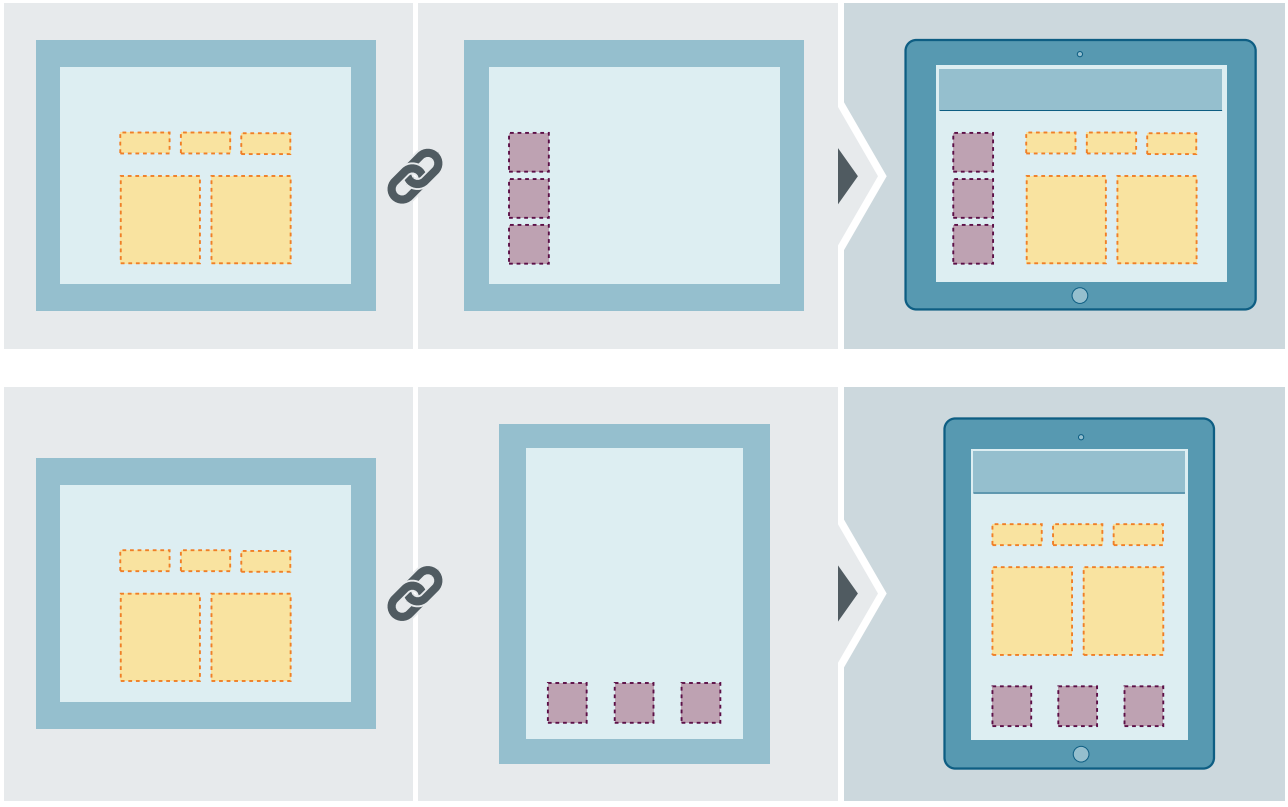
SiV — SiVArc generation template

< ... > — SiVArc property with referenced text source

Function block

Instantiated function block

- Combined positioning scheme
  An existing positioning scheme is combined with another scheme. This way you can divide up a display area into modules, for example, that you can combine at random as versions for different HMI devices.
  The figure below shows a positioning scheme to which you can assign other different schemes:



  If a module exists in the higher-level positioning scheme that has the same name as a lower-level scheme, the lower-level area is ignored.

- Preview of the positioning
- Positioning can be planned before the first generation
- Low error susceptibility
- Central availability of layout versions

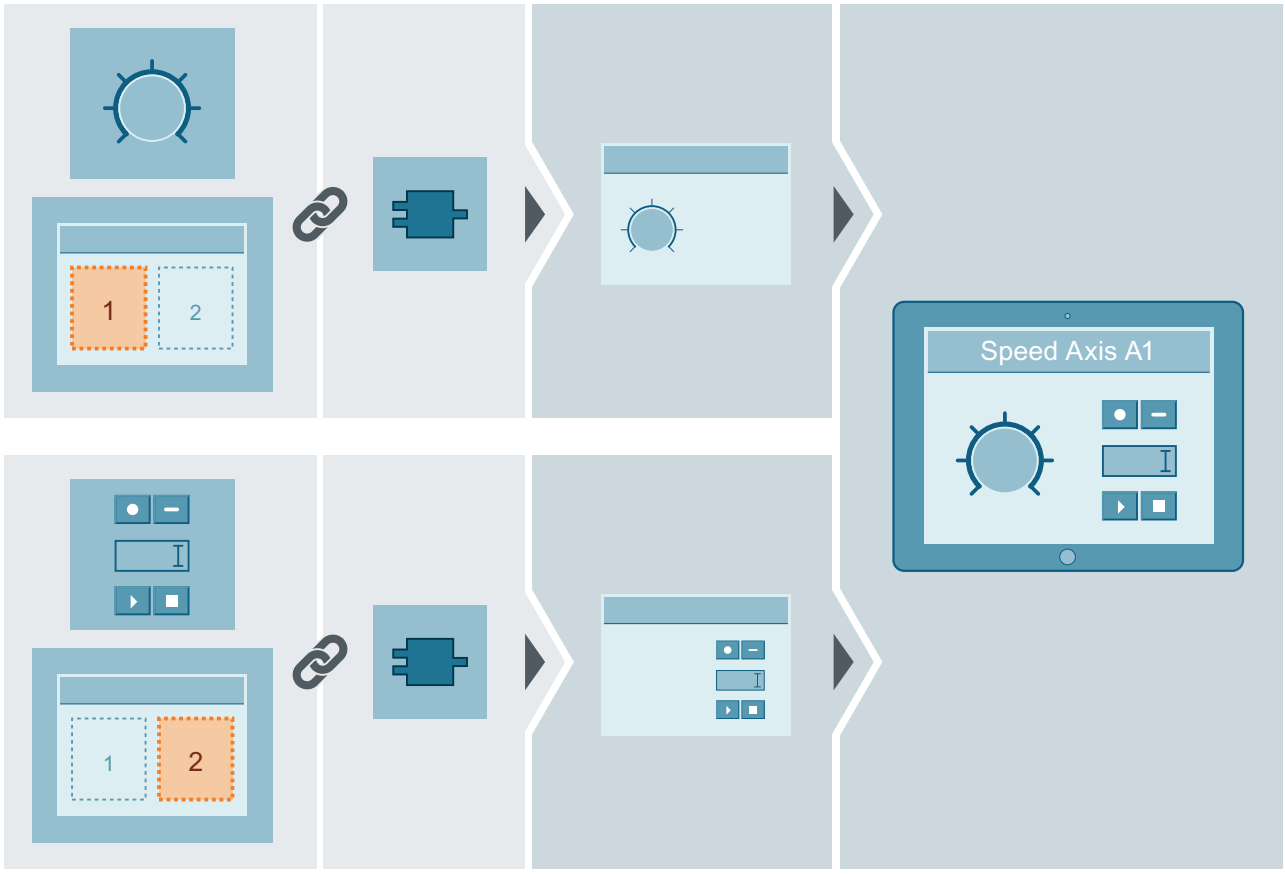
Note

Positioning schemes for pop-up screens

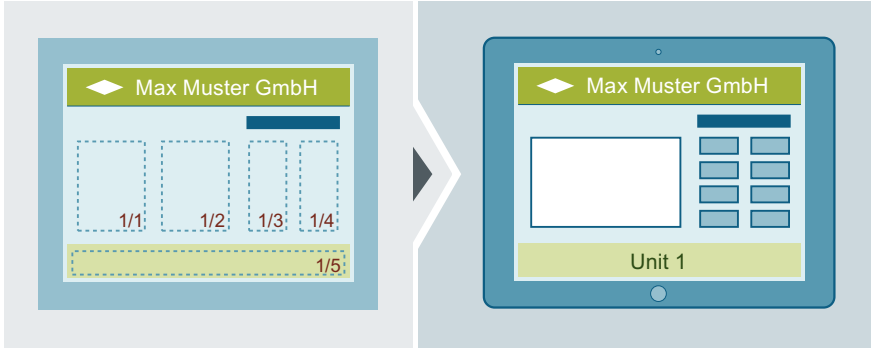
A positioning scheme for a pop-up screen cannot be used for any other display and operating object.

Operating principle

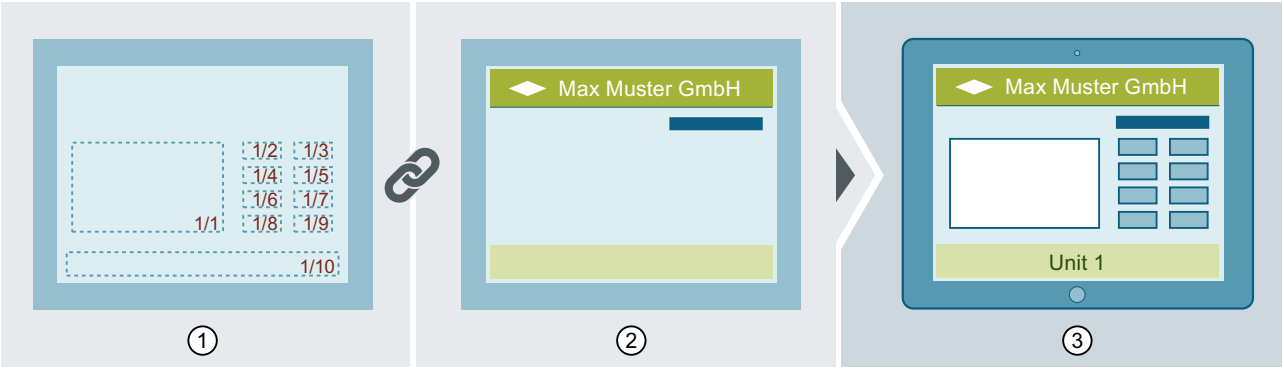
You can attach a screen rule to a display and operating object which defines the area of the positioning scheme in which the object is going to be placed. If you are using a combined scheme, you can interconnect all layout fields contained therein with generation templates regardless of the nesting depth of the scheme.



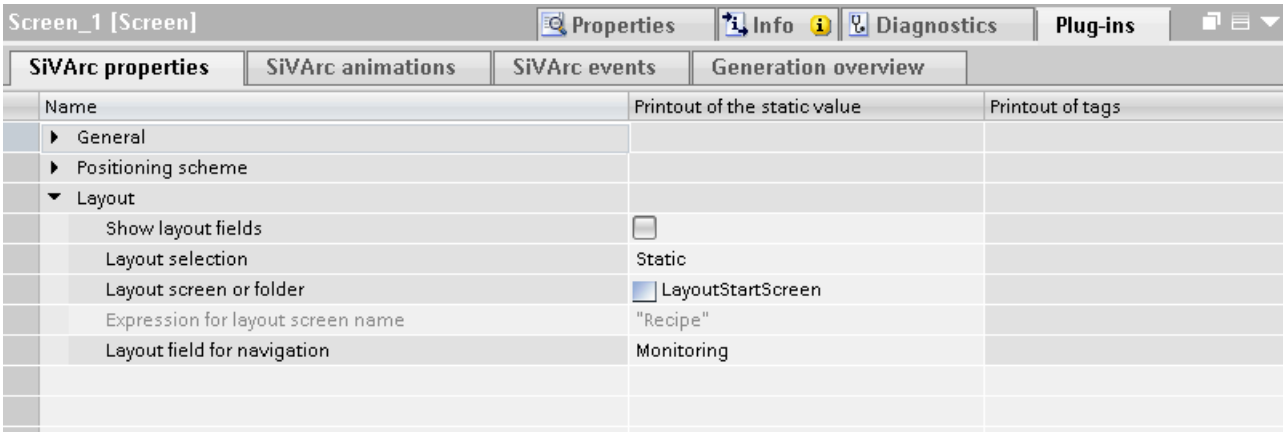
You can use a positioning scheme in a screen rule as you would a generation template for a screen. Layout and positioning are then contained in the same template:



To separate layout and positioning assign the positioning scheme of a generation template for a screen permanently or dynamically:



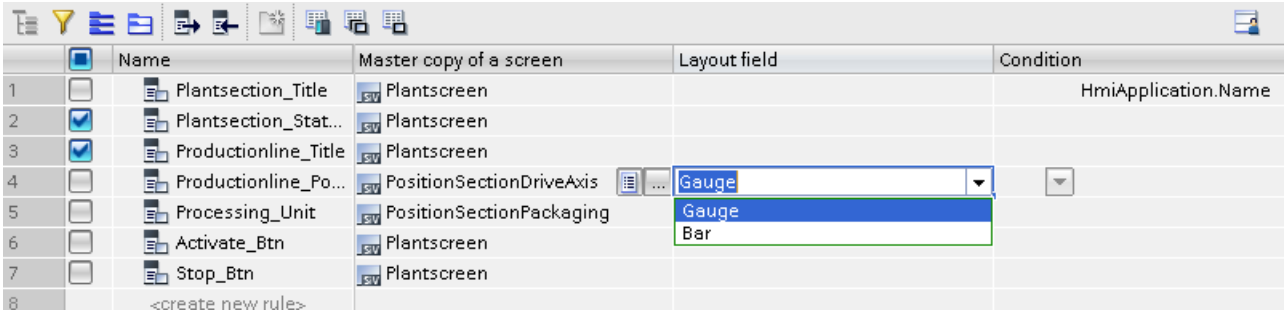
The figure below shows the associated setting in the SiVArc properties of a generation template for screens:



Layout fields

In the screen rule you select the area of the positioning scheme in which the object is generated. SiVArc generates the screen object in this area into the layout field with index 1. The next generated object is generated into the field with index 2, and so on.

The figure below shows the configuration of the layout fields in the screen rules:



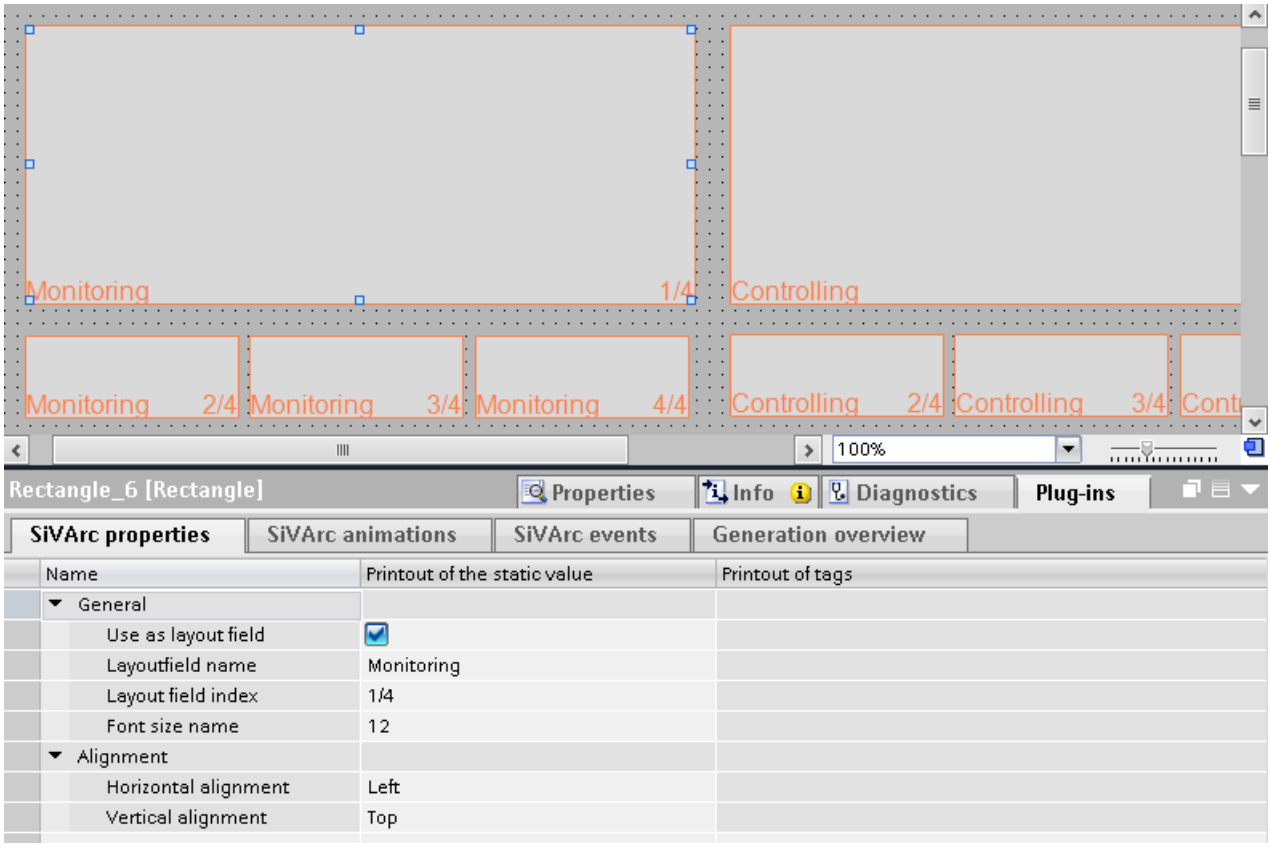
Layer assignment

When you have assigned a fixed layer to a master copy and have used your own positioning scheme during generation, the HMI object is generated in the layer that was specified in the positioning scheme.

Structure

A user-defined positioning scheme consists of a screen that contains layout fields for generated display and operating objects. You assign the positioning scheme to a generation template and thus create a process screen.

By giving the layout fields identical names, you group those layout fields into a logical unit. Layout fields are filled in the order of the index within a logical unit.



Subsequent changes

If you manually change the position of a generated display and operating object, this change is retained at the next generation. This is true even if the position has been defined with its own positioning scheme. Even if you change the positioning scheme, the manually configured position is retained after the next generation.

See also

Overview for positioning of generated objects (Page 39)

Overflow mechanisms (Page 53)

Creating user-defined positioning scheme (Page 60)

Example: Using a dynamic layout (Page 67)

Example: Using a combined layout (Page 69)

Example: Using generated screen navigation (Page 70)

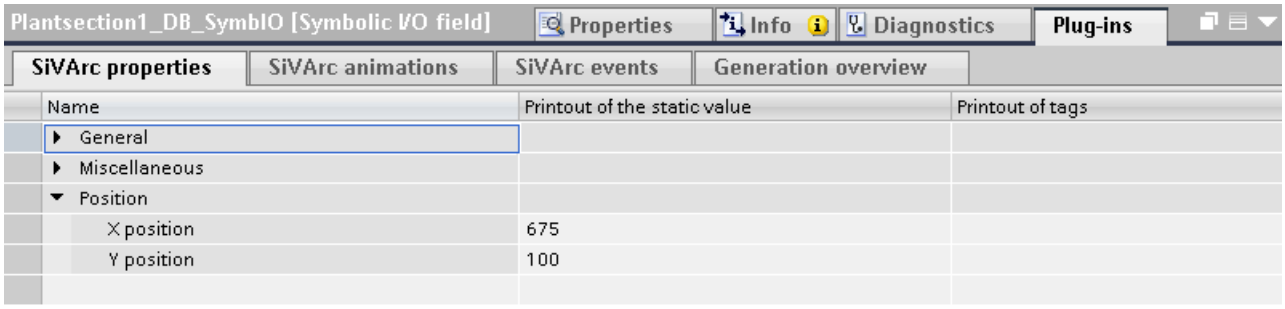
Picture legends (Page 231)

5.1.1.3 Fixed positioning of the generated object

Select a fixed position if you want to always anchor specific objects at the same position in the screen, for example, for standard objects.

The fixed positioning depends on the screen resolution. In an HMI device with high resolution the display and operating object is displayed further up and to the left than in an HMI device of the same size with a lower resolution.

Define the coordination of the object individually and independently of the positioning scheme in the SiVArc properties of a generation template for a display and operating object.



Note

Unchangeable fixed positioning of screen objects

For screen objects with fixed positioning, a manual change of the position is ignored at the next generation.

See also

Overview for positioning of generated objects (Page 39)

5.1.1.4 Free positioning

Overview

A grid is stored on the generated screen and used to arrange the screen objects during generation. The grid can be configured.

During initial generation, the objects are generated in the grid on the screen. You then arrange the generated objects individually. The new layout is retained for each subsequent generation.

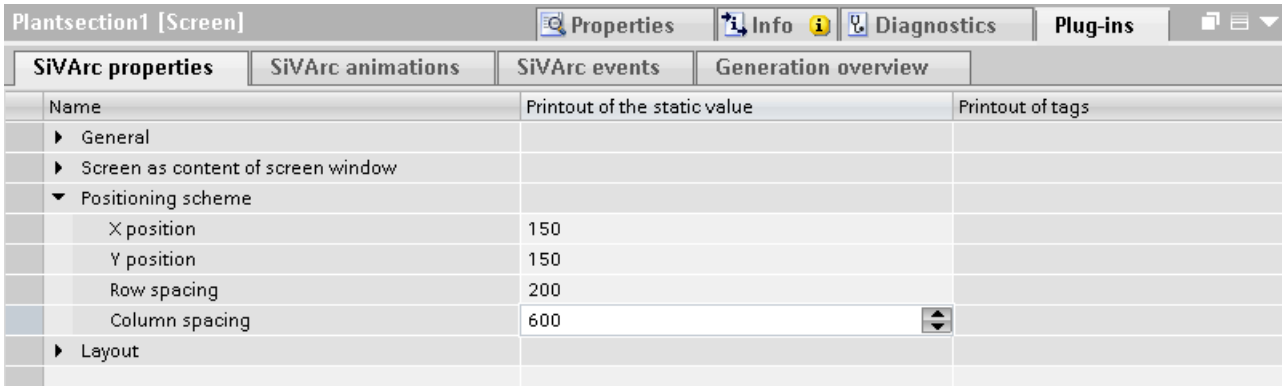
This method has the following advantages:

- It is not necessary to plan the layout extensively beforehand.
- After each generation, you can further adjust the layout and add more definitions.
- The layout develops together with the SiVArc project.

This procedure is very suitable for smaller individual and development projects. When the project becomes larger, the post-editing requirements increase.

Structure and filling of the positioning scheme

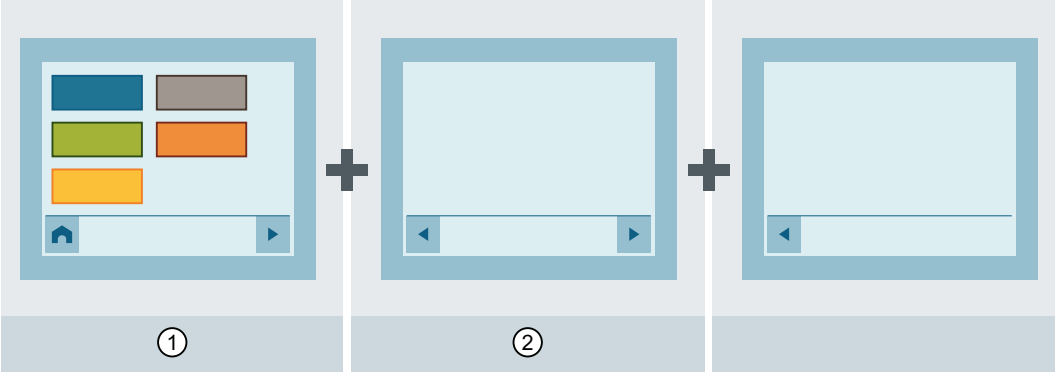
You configure the positioning scheme of the objects in the SiVArc properties of the screen.



After the initial generation, the HMI objects are positioned depending on the positioning scheme. The positioning scheme is based on the start position of the first object and the distances in the x and y position.

If no screen objects are assigned to overflow screens, the screen objects are arranged by default in the base screen after initial generation of the visualization.

The figure below shows the default arrangement of the screen objects in the base screen.



① The generated screen objects are positioned column-by-column in each screen from top to bottom and left to right. The screen objects always have the same distance to each other.

② If overflow screens have been generated for a screen, SiVArc automatically inserts navigation buttons with configured screen changes.

See also

Generating visualization (Page 181)

Overview for positioning of generated objects (Page 39)

Overflow mechanisms (Page 53)

Example: Using a layout with free positioning (Page 64)

5.1.1.5 Nesting depth

Nesting depth in screen layers

You set the nesting depth of the objects to be generated in the SiVArc master copy by means of the layer hierarchy. This setting is retained during generation.

Note

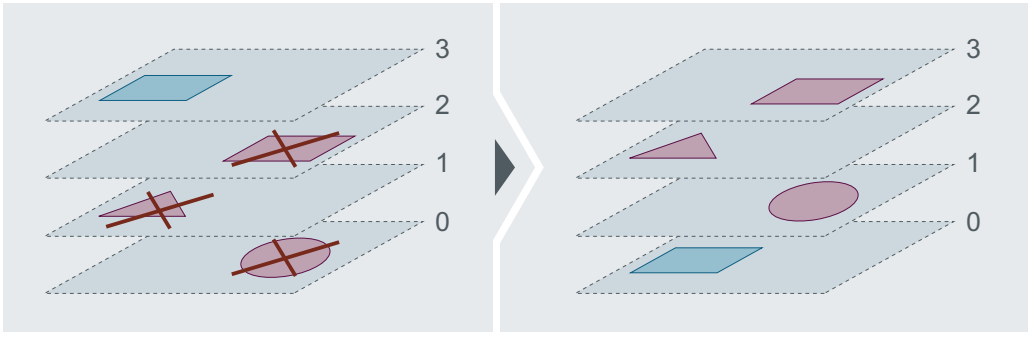
Changing the layer

When you change the layer assignment of objects in the generated screen, this assignment is retained during a subsequent generation.

Nesting depth within a screen layer

The following applies within the same layer in the generated screen:

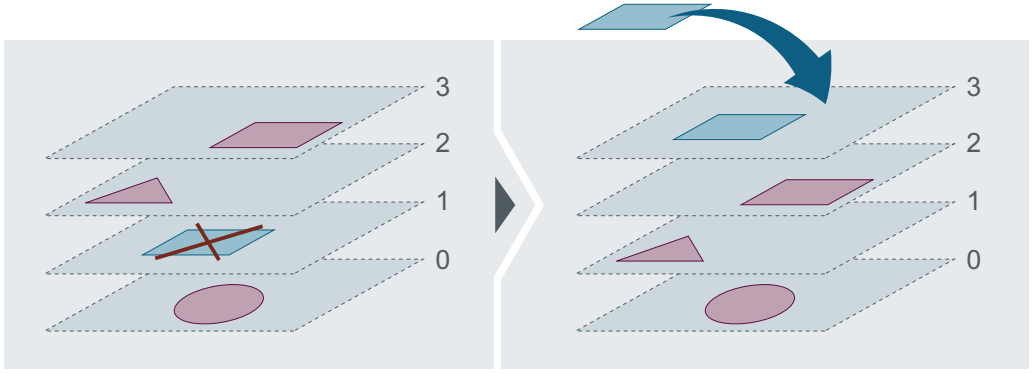
- When you delete the generated objects and then manually insert objects, the objects are also generated over the manually inserted objects in the nesting depth during the next generation.



Generated display and operating object

Display and operating object manually inserted into the generated screen

- If you arrange a manually inserted object in the generated screen at a specific depth and then delete it, this previous arrangement is not relevant for SiVArc. During the next generation, the screen objects are arranged in the lowest position in the layer. If you insert the deleted object once again manually, it is located in the highest position.



Generated display and operating object

Display and operating object manually inserted into the generated screen

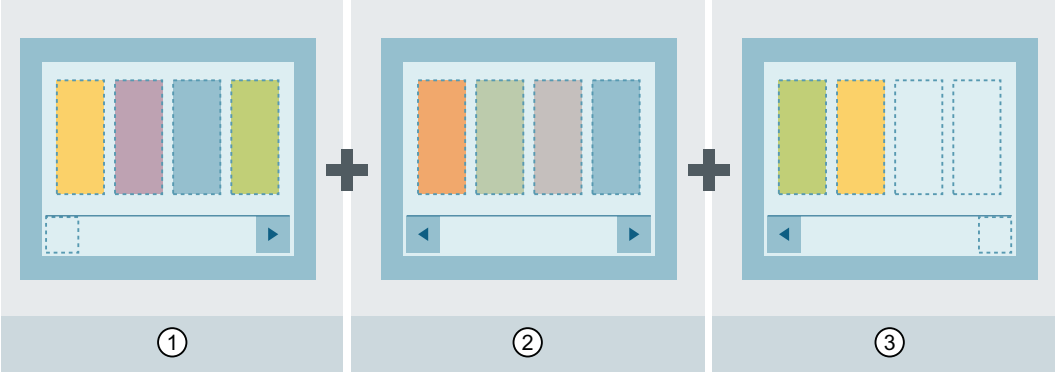
5.1.2 Overflow mechanisms

Definition

Overflow screens are screens generated when there is insufficient space on a screen for the number of generated screen objects. Depending on the positioning scheme used, overflow screens are generated differently. The overflow screens are generated for each instance of a generation template.

Overflow screens based on a defined positioning scheme

If the configured layout fields are not sufficient for all generated display and operating elements, overflow screens are generated on the basis of the positioning scheme.



① Base screen
② First overflow screen with positioning scheme of the base screen
③ Second overflow screen with positioning scheme of the base screen

Filling overflow screens manually

If you do not use your own positioning scheme and specify the number of overflow screens as decimal number in the SiVArc properties of the screen generation template, the screen objects are only arranged in the base screen. To limit the generation of overflow screens, formulate a condition under the "Number of overflow screens". Disable the option "Evaluate number of overflow screens as bit mask".

After the first generation, you move the screen objects to the required positions in the overflow screens.

The modified positions of the screen objects are retained for each additional generation.

Note

Copying generated display and operating elements to an overflow screen

Note the following when you define the number of overflow screens as decimal number:

When you manually copy objects generated with SiVArc from a base screen to an overflow screen, this change is retained for a renewed generation. The copy is then treated together with the HMI object on the base screen like an object generated by SiVArc and has a reference to SiVArc.

Requirement: The name of the copy must match the name of the original.

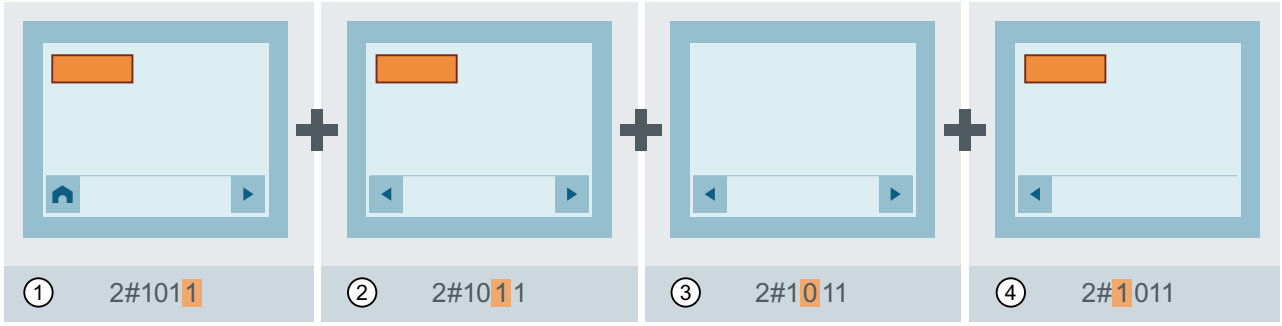
Overflow screens controlled with bit mask

You define the bit mask at the program block or in the generation template of the screen. To do so, use a static value or a tag.

If you specify the distribution of overflow screens as bit mask, the screen objects are arranged in the base screen and in overflow screens.

You define the following with the bit mask:

- Number of overflow screens
  The number of bit positions in the bit mask defines the number of overflow screens. The first position in the bit mask corresponds to the screen of the master copy. The second position corresponds to the first overflow screen, the third position to the second overflow screen, etc. The bit mask is limited to 31 overflow screens. An overflow screen is not generated when you use bit mask 2#0.
- Overflow screens with screen objects
  If the screen object of the used screen rule is to be generated to an overflow screen, set the corresponding bit in the bit mask to 1.
  Example: You are using bit mask 2#1011. Three overflow screens are created during generation. The screen object of the used screen rule is generated as follows:



① Base screen with generated screen object
② First overflow screen
③ Second overflow screen
④ Third overflow screen

As soon as a layout field is configured in the screen, configurations for overflow screens contained in the screen are no longer evaluated during generation.

Note

Copying or moving objects to an overflow screen

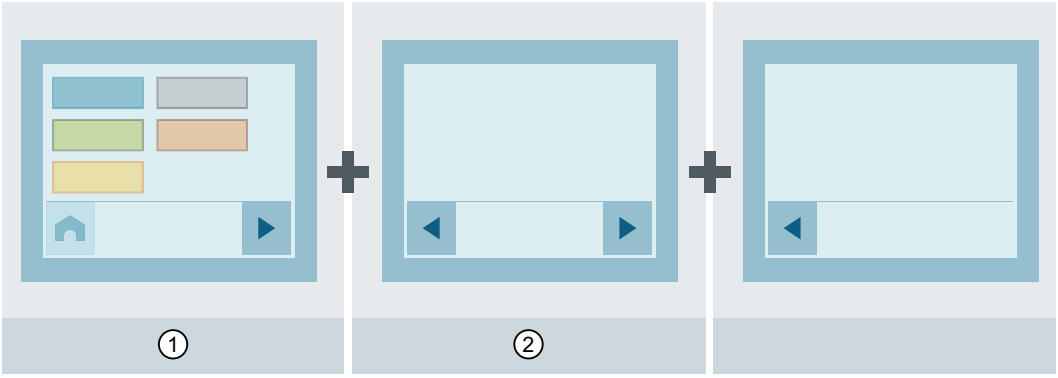
Note the following when you configure overflow screens with a bit mask:

When you copy or move generated display and operating elements to an overflow screen or base screen, these objects are treated as manually created objects in case of a new generation. The display and operating element is created again during the next generation process.

Navigation buttons

If SiVArc generates overflow screens, navigation buttons for moving to the previous screen and the next screen are generated.

If you freely position the generated display and operating elements, the original screen of the generation template is generated as base screen. The base screen is connected to the first overflow screen with a navigation button.



① Generated screen of the master copy (base screen)

② First overflow screen with automatically generated navigation buttons with configured screen changes

In order to dispense with navigation buttons, you can disable the "Navigation buttons" selection in the generation template of the screen.

Note

You can store master copies in the library for the navigation buttons.

For more information, refer to the section "Generation templates in SiVArc (Page 72)"

Pop-up screens

Overflow screens are not generated for pop-up screens. An error message is output when more display and operating elements are generated than can be positioned. Display and operating elements that no longer fit on the pop-up screen are not generated.

See also

Free positioning (Page 51)

Example: Using overflow mechanisms (Page 64)

Configuring overflow screens without screen objects (Page 63)

5.1.3 Supported devices

Overview

SiVArc can be used with the following devices:

- PLCs
  - SIMATIC S7-1200
  - SIMATIC S7-1500
  - SIMATIC S7-1500 software controller
  - ET 200SP CPU
- Device proxies
  Device proxies are only used to generate external tags.
- HMI devices
  - HMI devices with WinCC RT Professional
  - HMI devices with WinCC RT Advanced
  - Comfort Panels
  - Mobile Panels 2nd Generation
  - Basic Panels

5.1.4 Designing a layout

Introduction

The positioning method you select depends on the size of an automation project and the requirements within the company.

The following project examples illustrate the operating principle and the purpose of the individual positioning methods.

Standardized large-scale projects

In this case we recommend using your own positioning schemes. These will offer you exact placement and optimal use of the display area for your different HMI devices. You can assign a separate positioning scheme to each HMI device. To implement a change in standardized form, you can, for example, set up vertical and horizontal alignment of HMI devices as separate positioning schemes.

Even if many objects are being generated, placement is still manageable because each object receives its placement information through the screen rules. If the display area is not sufficient, overflow screens with the same layout are created automatically. By modularizing the placement areas through combined positioning schemes, you achieve high reusability of your templates.

Smaller, individual projects

Free positioning requires very little planning on your part for the layout. Manual changes to the layout require less time in a small project.

When you position freely you also have the option of working with few generation templates for screens and expanding your visualization using overflow screens. You also equip overflow screens individually with objects when you use free positioning.

If you want to use a special concept for distributing the objects to the overflow screens, you can control the overflow screens using a bit mask. Because you can store the bit mask in the controller, you get a direct link from the controller to the screen layout. This link supports you, for example, when troubleshooting.

Development and test phase

Free positioning offers more advantages in this case. You remain flexible for a long time in positioning. Because you can move the generated objects individually, you can use many different HMI devices in the project. Overflow screens ensure clear representation of your process pictures. The screen navigation of the overflow screens is generated automatically.

An HMI device swap requires a new generation with manual positioning.

Standard process pictures

The fixed positioning depends on the HMI device. This means fixed positioning is only useful in projects with many identical HMI devices or for the top left area of an HMI device. Because fixed positioning is ignored by a positioning scheme, use fixed positioning only in combination with free positioning.

If changes are not expected and an HMI device is not swapped in a project, fixed positioning is the right choice.

Planning the screen layers

You control the positioning within the screen layers for the generated display and operating objects as in WinCC with the WinCC properties of the generation templates. The nesting depth defined by SiVArc is only relevant within one layer.

See also

Overview for positioning of generated objects (Page 39)

5.1.5 Creating user-defined positioning scheme

Requirement

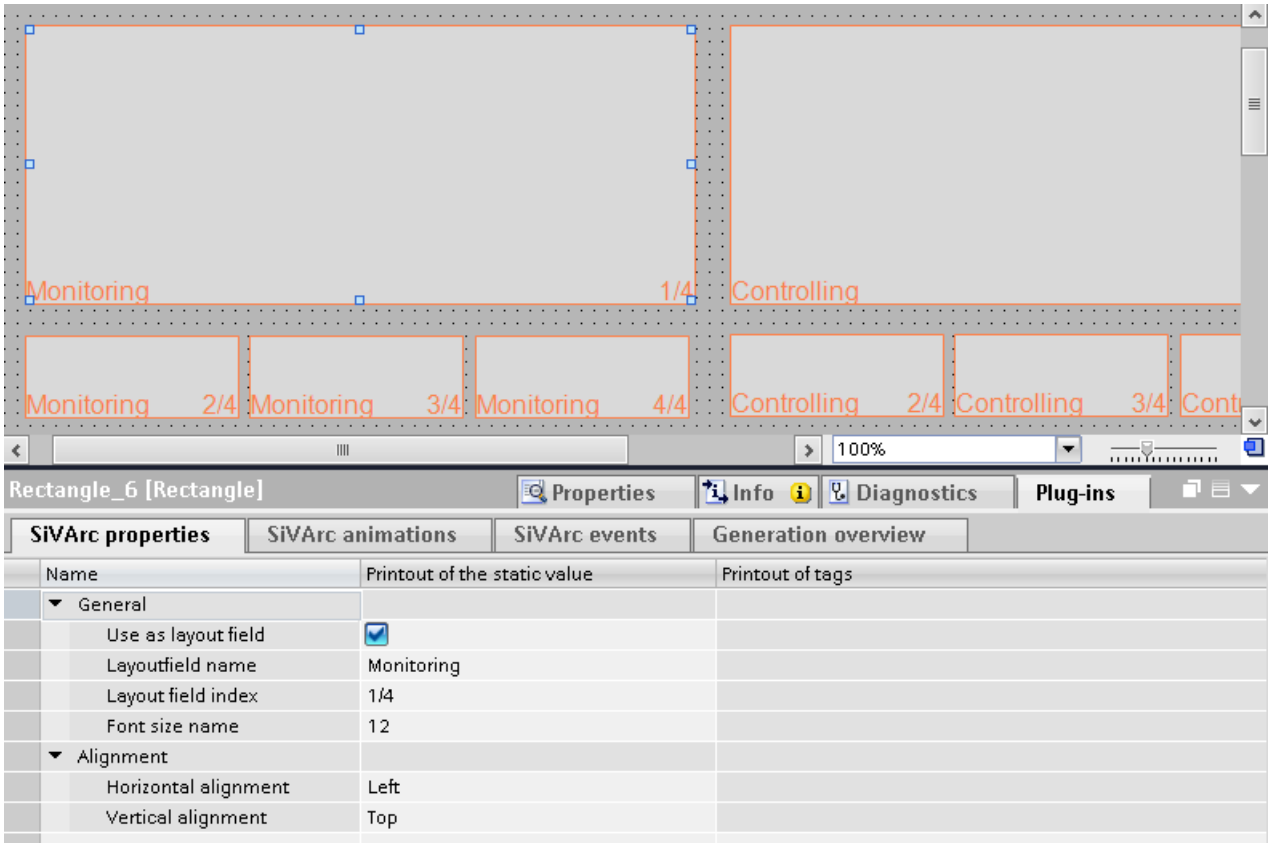
- The "Screens" editor is open.
- The "Overview" screen is created.

Procedure

To create a positioning scheme, follow these steps:

1. From the "Basic objects" group in the toolbox window, add multiple rectangles to the screen. Make sure that the rectangles for the generated display and operating objects are sufficiently large; otherwise, the display and operating objects overlap in the generated screen.
2. In the SiVArc properties of the rectangles, select "SiVArc properties > General > Use as layout field".

3. Define areas in the screen.
   - Give the same name to layout fields that belong to a logical unit, for example, "Monitoring" and "Controlling".
   - To do this, change the name of the layout field under "General > Layout field name".
   - Set up the font size under "General > Font size name".
   - Specify the border and font color of the layout fields in the WinCC properties under "Properties > Properties > Appearance".



4. If necessary, change the order of filling the fields under "General > Layout field index". The layout fields are shown with name and index.
5. Store the "Overview" screen as master copy in the library.
6. Delete the "Overview" screen in the project tree.

Index order

The index assignment follows the time sequence in which you edit the indices. If you subsequently assign a layout field to another logical unit, for example, the field gets the last index number of this unit regardless of the arrangement in the screen.

The index order automatically readjusts itself after each change.

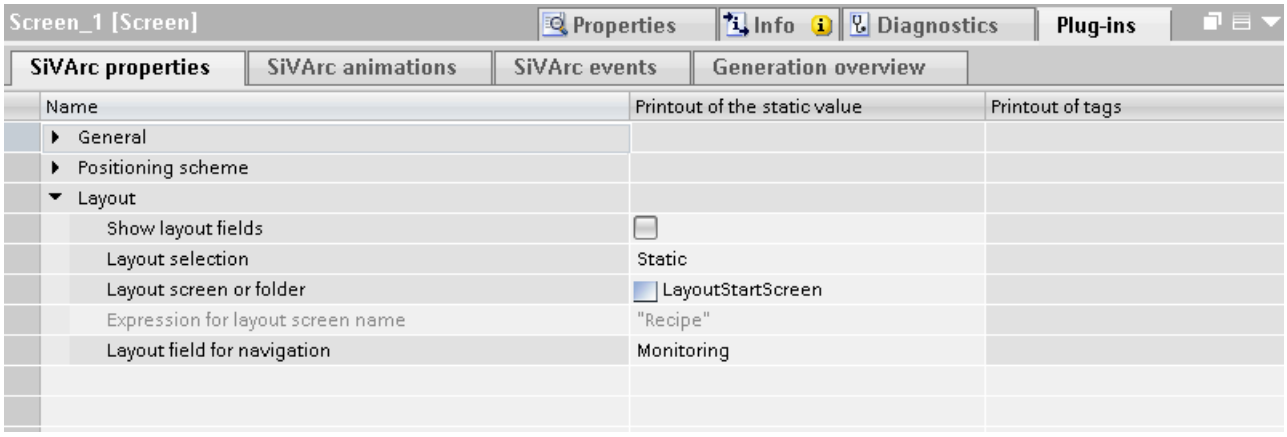
Result

The positioning scheme has been created. You can store the scheme of a screen generation template or use the scheme as screen in a screen rule.

Assign positioning scheme to a generation template permanently

To use a positioning scheme in a generation template, follow these steps:

1. Generate a new screen from the generation template in which you want to store the new positioning scheme.
2. Select the "Static" option under "Layout selection".
3. Under "Layout screen or folder", select the required positioning scheme.



4. Delete the generation template in the library.
5. Store the edited screen as generation template in the library.
6. Delete the screen in the project tree.

When you use the generation template in a screen rule, you also specify the layout field in the screen rule. SiVArc generates the screen object into this layout field in the field with index 1. The next generated object is generated into the field with index 2, and so on.

Note

Layer assignment

When you have assigned a fixed layer to a master copy and have used your own positioning scheme during generation, the HMI object is generated in the layer that was specified in the positioning scheme.

Displaying layout fields in the generated screen

To display the layout fields in the generated screen, select "SiVArc properties > Layout > Show layout fields" in the SiVArc properties of the screen.

See also

Positioning according to defined schemes (Page 41)

5.1.6 Configuring overflow screens without screen objects

Introduction

You can set up overflow screens with and without screen objects in the SiVArc project. When you configure overflow screens without screen objects, you move the generated display and operating objects to the overflow screens after the first generation. This position is then set for all subsequent generations.

Procedure

To configure overflow screens without screen objects, follow these steps:

1. Enter the required number of screens in the Inspector window under "Plug-Ins > SiVArc properties > General" for "Number of overflow screens".

   Note

   The overflow screens are generated for each instance of this master copy.

   To limit the generation of overflow screens, formulate a condition under the "Number of overflow screens".

2. Disable the option "Evaluate number of overflow screens as bit mask".
3. If necessary, enable the generation of navigation buttons.
4. Define one or more screen rules.
5. Start the generation.

SiVArc generates all screen objects into the generated base screen. After the first generation, you can move the generated screen objects to the required positions in the overflow screens. The modified positions of the screen objects are retained for each additional generation.

Note

Copying generated display and operating objects to an overflow screen

Note the following when you define the number of overflow screens as decimal number:

When you manually copy objects generated with SiVArc from a base screen to an overflow screen, this change is retained for a renewed generation. The copy is then treated together with the HMI object on the base screen like an object generated by SiVArc and has a reference to SiVArc.

Requirement: The name of the copy must match the name of the original.

Overflow screens and layout fields

As soon as a layout field is configured in the screen, the "Number of overflow screens" and "Evaluate number of overflow screens as bit mask" properties no longer have a function.

See also

Example: Using overflow mechanisms (Page 64)

Overflow mechanisms (Page 53)

5.1.7 Example: Using a layout with free positioning

Example scenario

A manufacturer of printed circuit boards upgrades the manufacturing plant and decides to go with a new generation of HMI devices with a larger display area.

Requirement

The engineering firm that will make the adjustments in the visualization develops matching positioning schemes for the new HMI devices for the display and operating elements generated by SiVArc.

Implementation concept

To develop the positioning schemes for SiVArc, the visualization engineer generated the existing project first without positioning scheme.

Then the configuration engineer rearranges the generated display and operating devices on the larger screens of the HMI devices. This arrangement is retained during each additional generation and the configuration engineer makes further adjustments as needed.

Only after the arrangement of the generated display and operating elements has been optimized does the configuration engineer create the positioning schemes for the new HMI devices.

See also

Free positioning (Page 51)

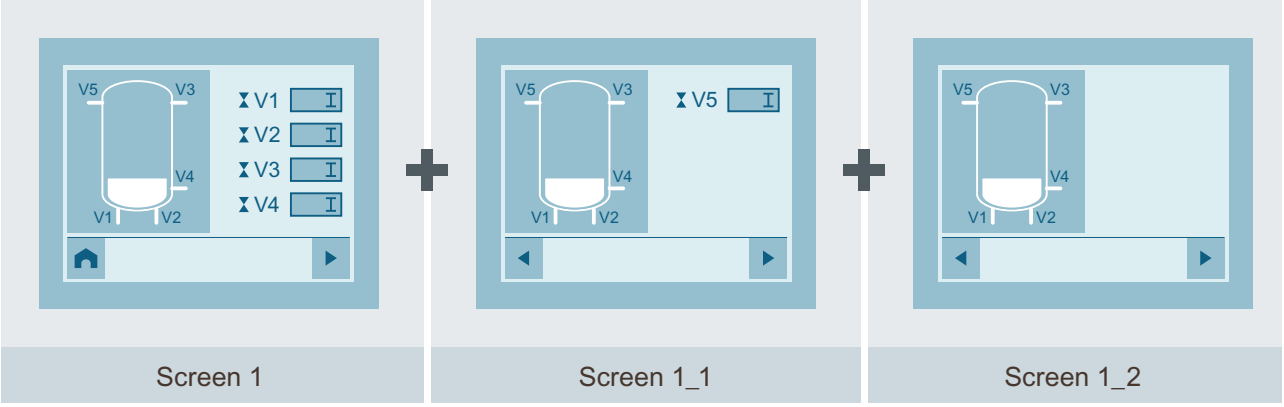
5.1.8 Example: Using overflow mechanisms

Example scenario overflow screen

A valve is added to an existing plant and visualized with a status output. The display of the associated HMI device is too small for additional display and operating elements.

Implementation concept

The additional display object is generated without any additional configurations in the first overflow screen.



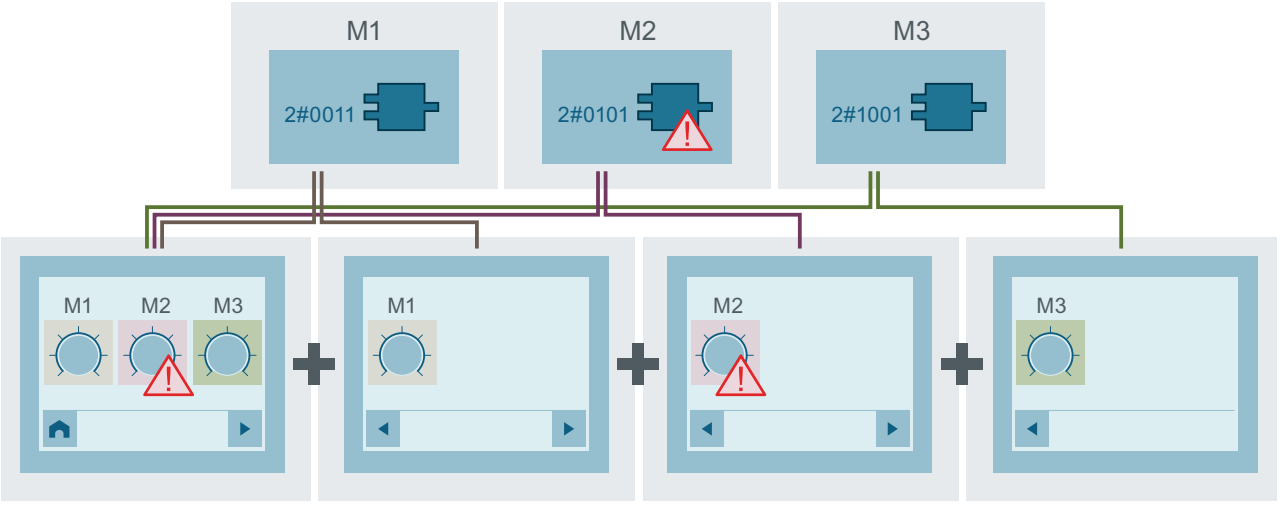
Example scenario overflow screen with bit mask

If an error occurs in a plant, navigation to the error source shall be ensured when an alarm is output.

Implementation concept

A bit mask for the generation of overflow screens is stored at an input in the controller. The distribution of the generated objects to the generated overflow screens is specified by the bit mask. The object of a screen rule is generated for each bit with the value 1.

If an error occurs at a bit, it is contained in the overflow screen of the same bit position in the process screen.



Evaluate number of overflow screens as bit mask

To configure overflow screens with bit mask, follow these steps:

1. Enter the required bit mask, for example, 11 (2#1011), for "Number of overflow screens" in the Inspector window under "Plug-Ins > SiVArc properties > General".
   or
   In the Inspector window under "Plug-Ins > SiVArc Properties > General" for "Number of overflow screens", select the block input at which the bit mask for overflow screens is set, for example, Block.Parameters("OVERFLOW_PIC").Value.
2. Enable the option "Evaluate number of overflow screens as bit mask".
3. If necessary, enable the generation of navigation buttons.
4. Define one or more screen rules.
5. Start the generation.

If you have entered a bit mask as number, three overflow screens are generated during generation in this example. The screen object of the used screen rule was generated in the first and third overflow screens and in the base screen.

If you have selected the block input, the value is processed at the parameter. If no valid value is set, the screen object of the used screen rule is only generated in the base screen and an error message is output.

Configuring bit mask for overflow screens as tag

To configure overflow screens with a bit mask which is saved in a tag, follow these steps:

1. In the Inspector window under "Plug-ins > SiVArc Properties > General", for "Number of overflow screens", enter the name of the SiVArc tag which was defined for the bit mask for overflow screens, for example, "SiVArcVariable".
2. Enable the option "Evaluate number of overflow screens as bit mask".
3. If necessary, enable the generation of navigation buttons.
4. Define one or more screen rules.
5. Start the generation.

The current value of the selected tag is processed during generation. If no tag is created, SiVArc generates the screen object of the used screen rule in the base screen.

See also

Overflow mechanisms (Page 53)

Configuring overflow screens without screen objects (Page 63)

5.1.9 Example: Using a dynamic layout

Example scenario

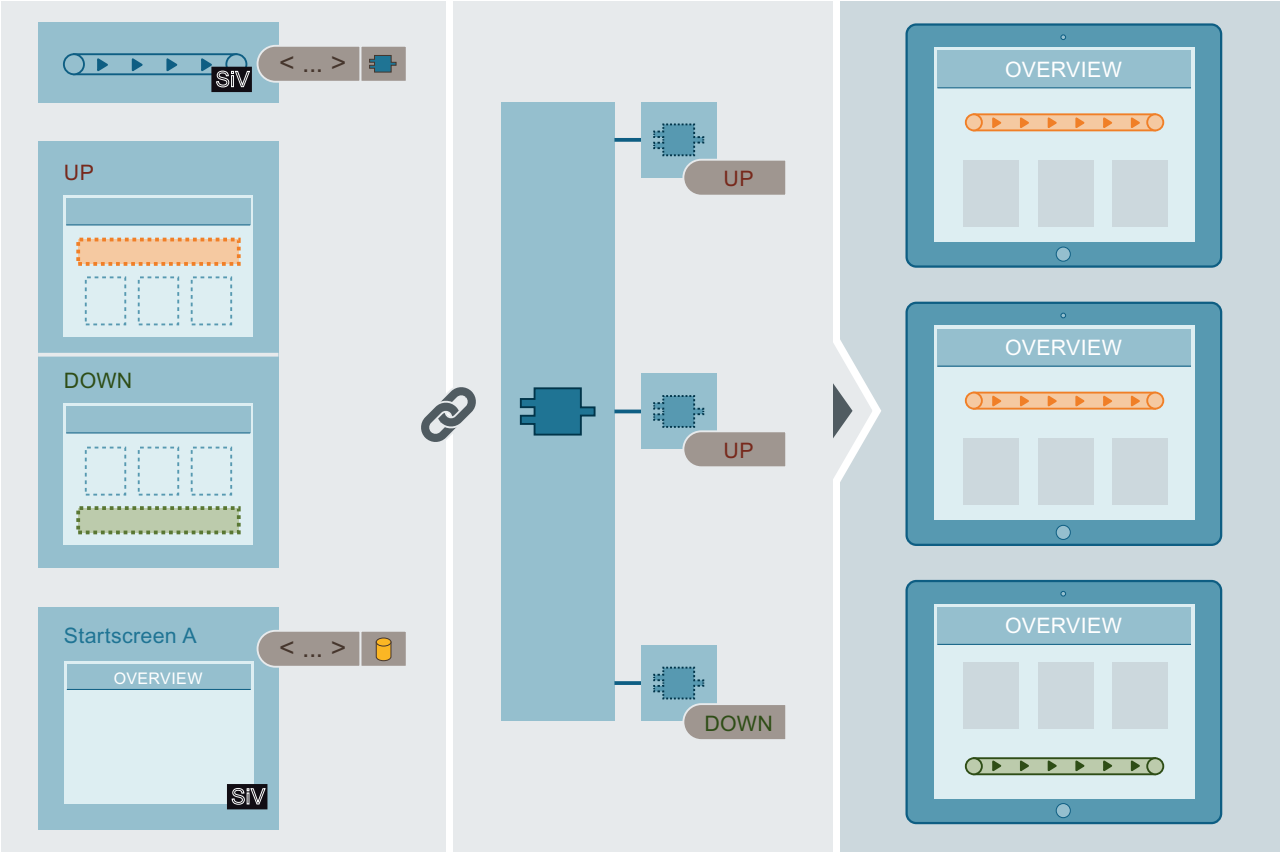
Two conveyor belts require a different screen arrangement but have the same operating objects.

Requirement

The number of screen rules shall be reduced.

Implementation concept

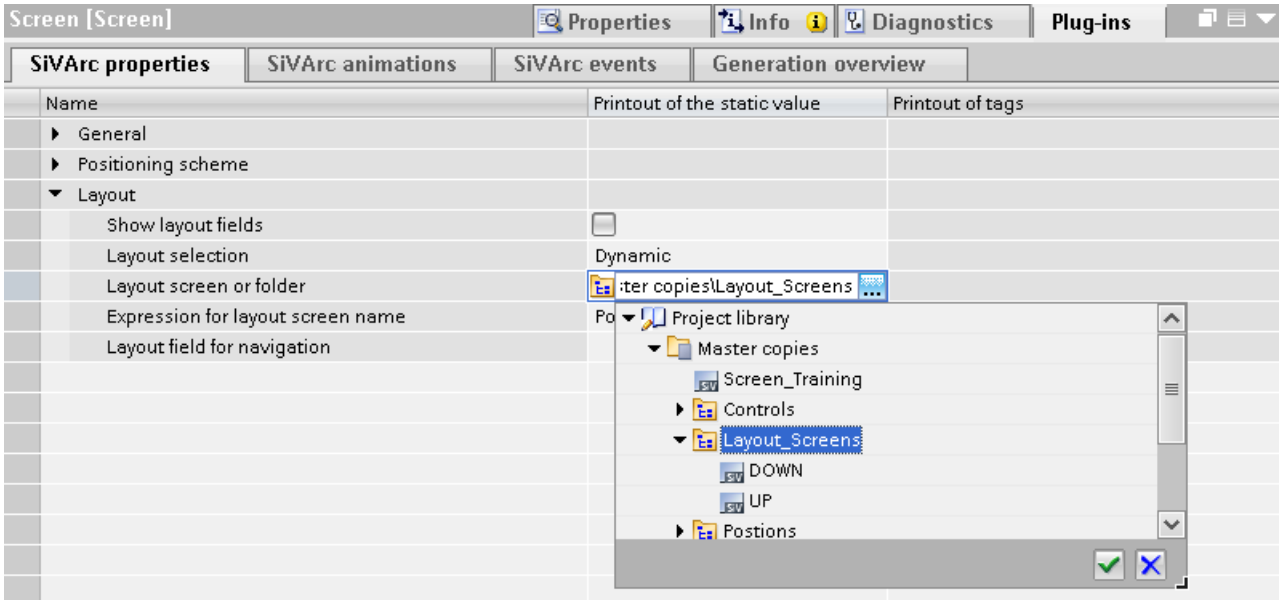
Two positioning schemes are created and stored in a library folder. A SiVArc tag is defined on the network. The SiVArc tag controls where the belt is to be arranged in the screen. The matching positioning scheme is selected during evaluation of the screen rule. One of the belts is arranged at the top of the screen area; the other is shown at the bottom.



Setting up a dynamic positioning scheme

If you want to assign a positioning scheme to a screen depending on specific conditions, assign a folder with positioning schemes to the generation template. Then you assign a SiVArc expression that returns the name of a positioning scheme contained in the selected folder.

1. Create multiple positioning schemes in a library folder.
2. Name the folder "Layout_Screens", for example.
3. Open the generation template of the screen to which you want to assign a dynamic positioning scheme.
4. Under "Layout selection" select the "Dynamic" mode in the SiVArc properties.
5. Under "Layout screen or folder", select the folder "Layout_Screens."



6. Configure a SiVArc expression under "Expression for layout screen name" that returns the name of a layout screen contained in the selected folder.
   You can define a SiVArc tag, for example, in the user program and use it as condition. You then assign the name of the positioning scheme required for this program block to the tag.
7. Store the edited screen as generation template in the library.
8. Delete the screen in the project tree.

Note

If you select the "Dynamic" mode for layout selection to generate multiple screen elements of a screen, not all dynamically assigned layout fields are displayed.

Even if you enable "SiVArc Properties > Layout > Show layout fields" in the SiVArc properties of the screen, only the layout field for the first generated screen element is displayed.

See also

Positioning according to defined schemes (Page 41)

Picture legends (Page 231)

5.1.10 Example: Using a combined layout

Example scenario

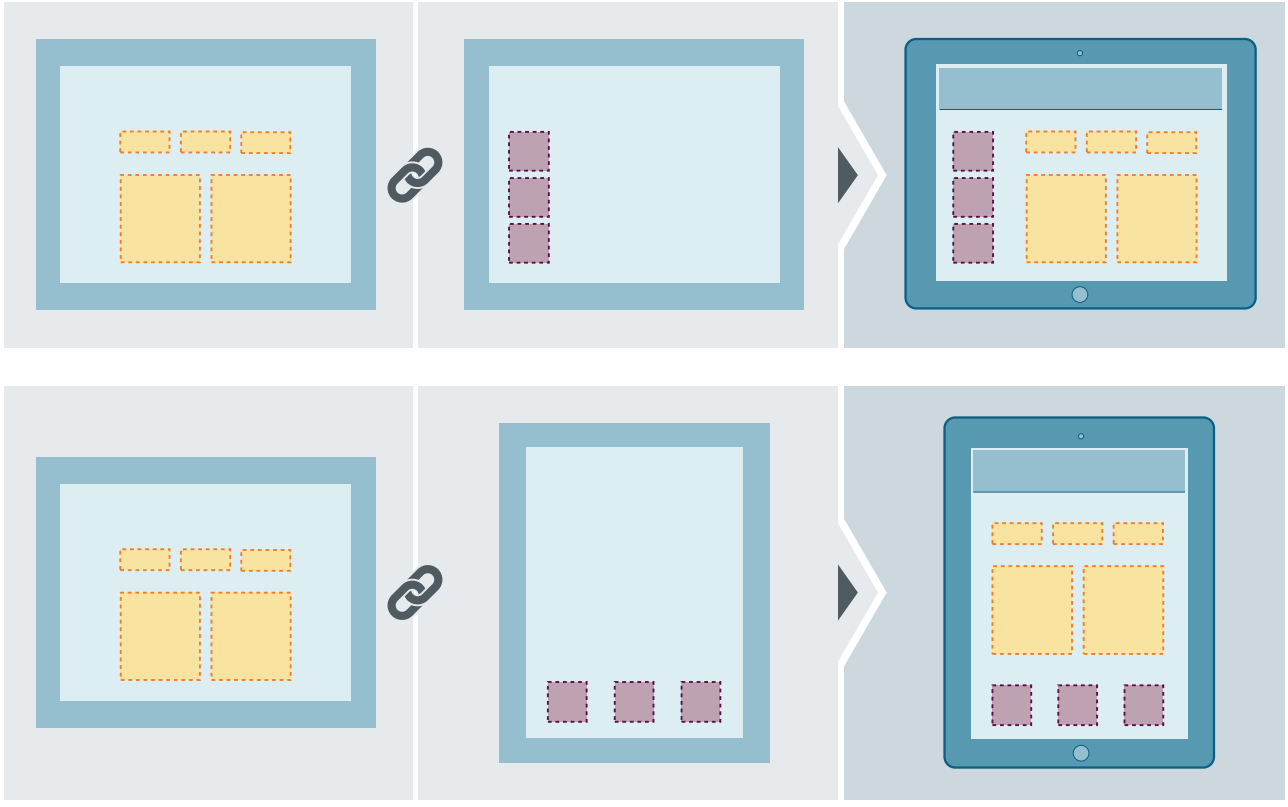
The overview screens of the production line for placing, soldering and packaging have a similar layout and differ from each other in the use of two different HMI devices (landscape format and portrait format). This results in a different partitioning of the screen.

Requirement

The layout of the process pictures is to be implemented with a high degree of reusability of the existing positioning schemes.

Implementation concept

The positioning scheme for the center screen objects of the production lines is combined with other positioning schemes. This way the display and operating objects are arranged at the bottom on the HMI device in portrait format. On the device in landscape format, the objects are arranged on the side.



See also

Positioning according to defined schemes (Page 41)

5.1.11 Example: Using generated screen navigation

Example scenario

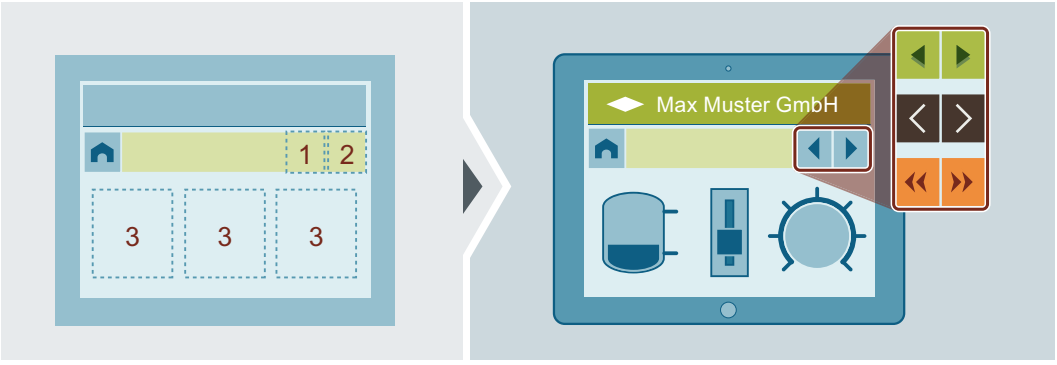
The screen navigation in an existing visualization object deviates from the new guidelines for corporate design and standardization in the company.

Requirement

Configuration of the screen navigation is to become more efficient and appealing.

Implementation concept

The engineering firm given the job decides to implement the new screen navigation with user-defined positioning schemes. This way the operating objects of the corporate design are used according to company-specific instructions.

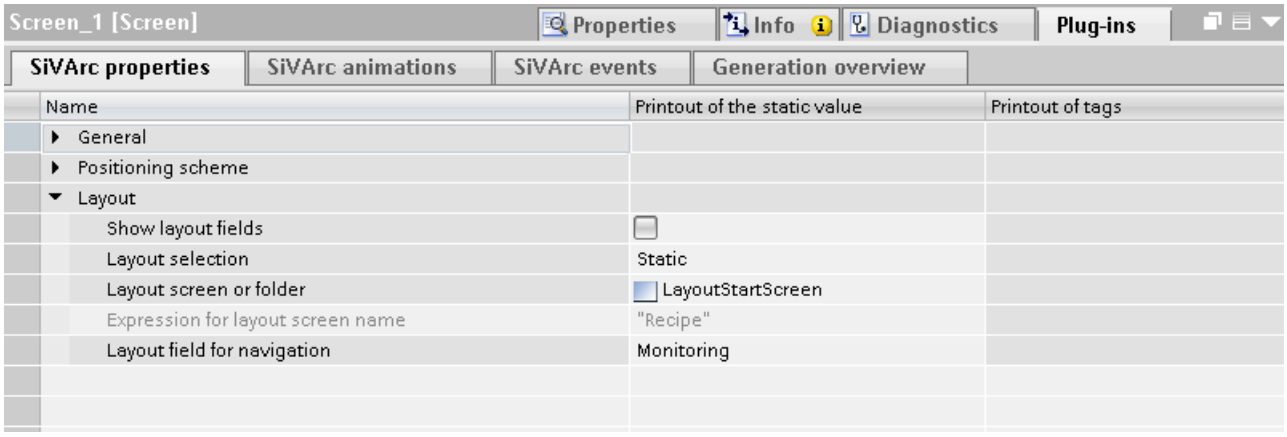


Automatic screen navigation is enabled in the overflow screens. Defined plant sections are thus shown in one single screen with overflow screens. Overflow screens reduce the number of generation templates for screens and standardize the display. Plus it ensures that all objects are generated in a visible area.

Company-specific screen objects are also stored for automatic screen navigation.

Using a layout field for navigation buttons

To implement the screen navigation with user-defined positioning schemes, select the layout field for navigation buttons, e.g. "Monitoring", in the SiVArc properties of the screen under "SiVArc properties > Layout >Layout field for navigation".



If overflow screens occur with this generation template during generation, the navigation buttons are placed in the "Monitoring" layout field.

Enabling automatic screen navigation in overflow screens

To enable automatic screen navigation in overflow screens, select the "Navigation buttons" option in the generation template of the screen.

See also

Positioning according to defined schemes (Page 41)

5.2 Creation of generation templates

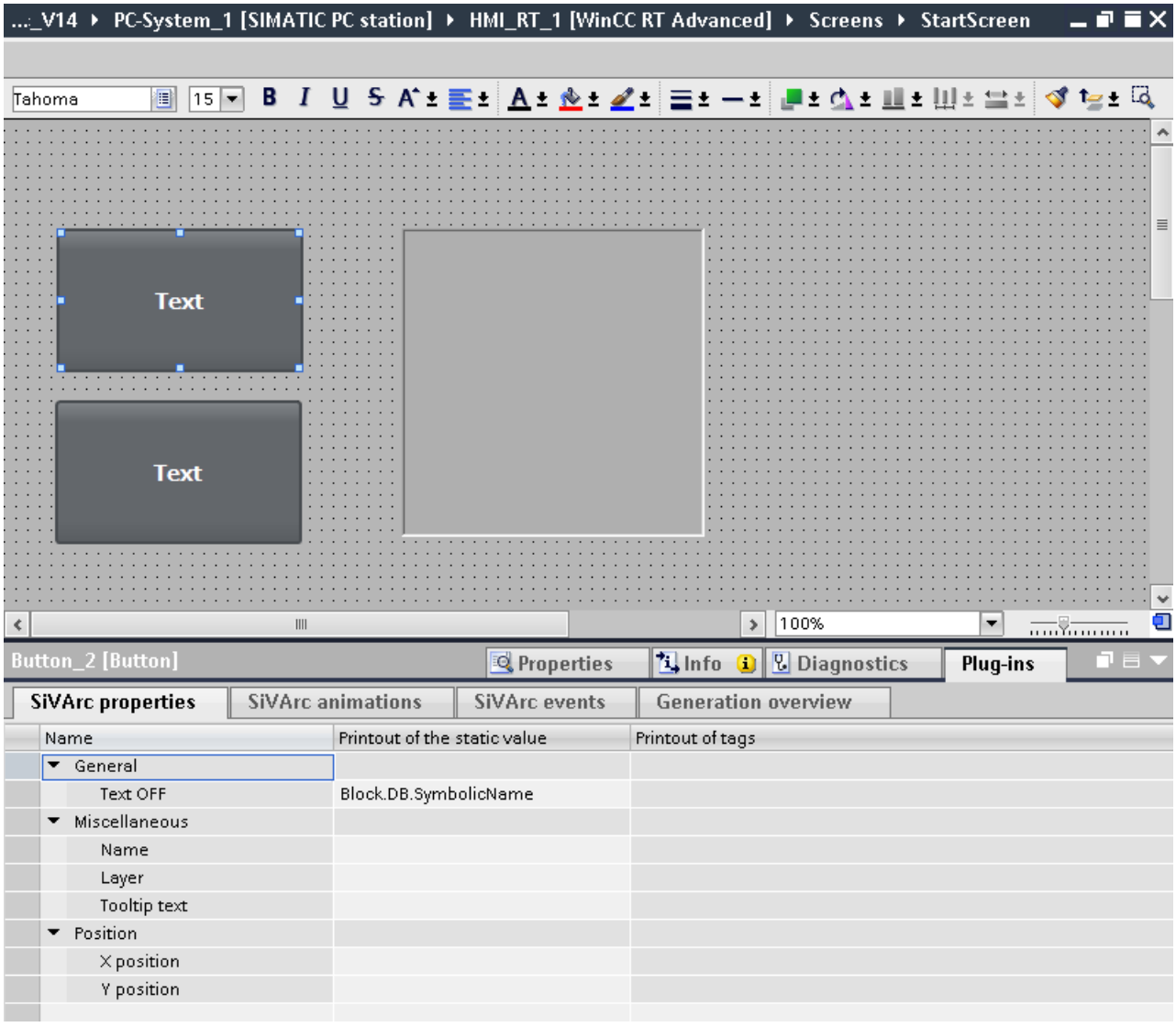
5.2.1 Generation templates in SiVArc

Definition

Generation templates are HMI objects from the library which are not only configured with fixed defined WinCC properties but also with SiVArc properties. A SiVArc property is an object property, which is first assigned as tag/expression. According to SiVArc mechanism, the SiVArc properties are only filled with texts, such as the object name, labels or a tag designation during the generation.

Operating principle

SiVArc properties can be static or dynamic. In the "SiVArc properties" tab, you can configure the properties of a generation template.



The tab contains the three columns:

- Name
  This column lists the available properties.
- Expression for the static value
  In this column, you assign a property with a fixed value or a SiVArc expression that returns a string or a number.
  Fixed values are entered in every instance of this master copy when generating the visualization. Pay attention, for example, with the "Name" property that the uniqueness of the object name is ensured when it is used multiple times in an screen.
- Tag expression
  In this column, you assign a property with a tag name or a SiVArc expression that returns a tag name.

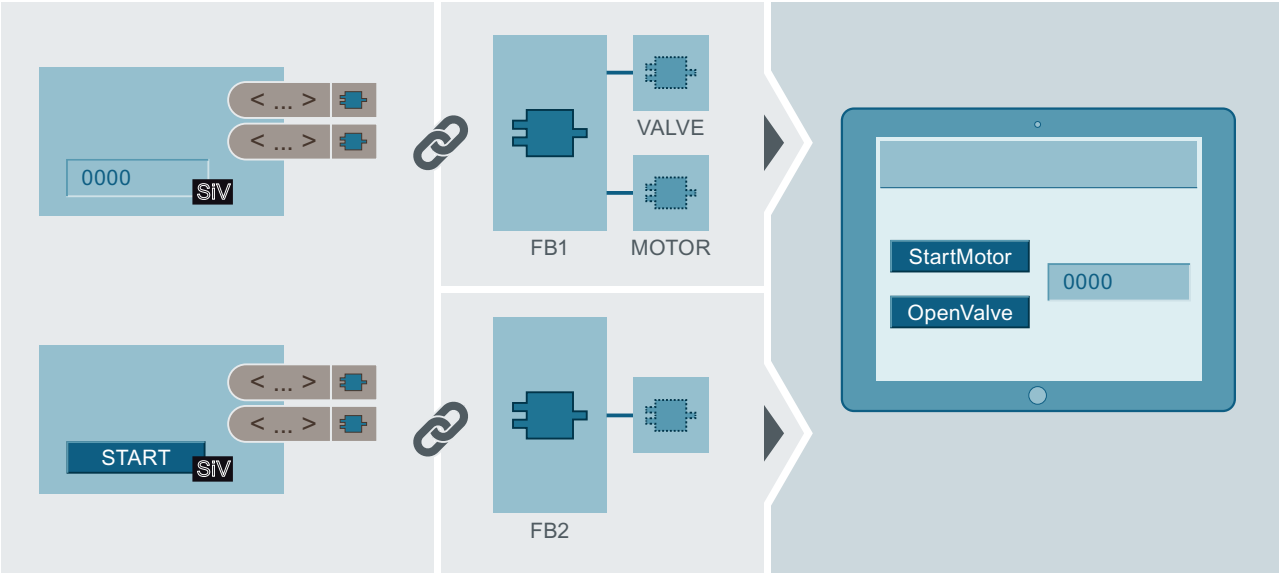
You then store the configured generation template in the project library and interconnect it later to a function block. You create a separate interconnection for each generation template. The SiVArc properties are evaluated during the generation of the visualization.

Generation templates for screens and screen objects

You generate one generation template per HMI object.

The "SiVArc properties" tab is only available for objects supported by SiVArc.

The following figure shows in schematic form the generation of screen objects from generation templates that refer to an instruction from STEP 7. The SiVArc properties are evaluated when the HMI objects are generated. Object properties, such as "Label" or "Name", are generated.



SiV	SiVArc generation template
< ... >	SiVArc property with referenced text source
	Function block
	Instantiated function block

Using types as generation template

Note the following when generating screen type instances and faceplate instances:

- If you use screen types for the SiVArc generation, all instances in the project are updated, even those not created by SiVArc.
- If you remove the connection of a generated instance to a screen type and change the screen, the change is still overwritten with a new instance of the screen type during the next generation.
- If you use screen types of the global library for the SiVArc generation, the screen type is added to the project library.
- SiVArc updates the screen type used to the latest type version in the project and in the libraries during generation.

Note

Using screen types as generation template

You use a screen type in the "Screen rules" editor only as a screen object. The screen type is therefore always displayed in a screen window.

Generation templates for text lists

With SiVArc, you create multilingual text list entries directly in the user program, for example, status texts for function blocks or interface descriptions for block parameters. During generation, you interconnect the texts with the corresponding display and operating objects. This allows you to generate descriptive texts for your project.

You can store text list entries in multilingual format at the block or derive these from a symbol table of a block parameter:

- Text list entries at the block
  You create the text list entries in the network or program block. To select the correct network, click a program block in any area of the network in the Inspector window. You can also use SiVArc expressions for the text list entries.
  Text definition and text list entry are linked in the text list master copy by identical names.
- Text list entries at the block parameter
  For individual parameters in the symbol table, you create a comment which is processed by SiVArc for the text list entry.

Note

Using text sources from STEP 7

Only one text source is processed within a text list. Therefore, use either texts from the block **or** texts from a symbol table for a text list.

Generation templates for automatically generated objects

The following objects are generated automatically with SiVArc:

- Screen window for displaying a screen within a screen
- Navigation buttons for overflow screens

You can customize the automatically generated objects using generation templates.

To do this, save the customized objects under "Master copies" in the project library.

Observe the following guidelines when storing the custom objects:

- The generation template for the screen window must be stored with the name "DefaultScreenWindowControl".
- The generation templates for the navigation buttons must be stored in a library with the names "NextButton" and "PrevButton". You can configure these buttons individually.

If you do not customize the generation templates, the default templates from the toolbox are used for generation.

Generation templates for positioning schemes

To specify the placement of screen objects in the generated screen, you specify a layout field group which you assign to the screen object in the screen rules.

Generation templates for screens using copy rule

In SiVArc, you can generate HMI screens through template generation using copy rules. Template screen generation using copy rules leverages the task of adding single or multiple template screens to multiple HMI devices configured in a project. The template screens are automatically copied from the project library or global library to HMI devices through copy rules. As part of SiVArc generation, the template screens for HMI devices are made available in the "Template" folder under "Screen management".

Generation of HMI screens using template screen property

In SIVArc, you can configure the HMI screens with template screens using the "SiVArc properties" under "Plug-ins". You must ensure the following:

- The "Template screen" name entered in the HMI screen's inspector window must be similar to the name of the template screen.
- The resolving expression for "Template screen" must match with the name of the existing template screen in the project.

After configuring the HMI screen with template screen, the HMI screen is placed in the "Master copies" folder under " Libraries". A screen rule is created, which makes use of the existing template screen, and is applied on the HMI screens during SiVArc generation. SiVArc generates the HMI screens with the same name as the configured template screen.

Note

Template screens using copy rule and SiVArc property is supported in Panels and RT Advaned devices.

See also

Requirements for a generation template (Page 90)

Example: Generating template screen using copy rule (Page 111)

Example: Generating HMI screen using template property (Page 112)

5.2.2 Supported HMI objects

HMI objects that can be generated with control data

SiVArc generates the following HMI objects, depending on the HMI device for which they are generated:

HMI object	Basic Panels	Comfort Panels/ Mobile Panels /2nd Generation RT Advanced	RT Professional
External tag[1]	x	x	x
Following master copies in a library:			

HMI object	Basic Panels	Comfort Panels/ Mobile Panels /2nd Generation RT Advanced	RT Professional
Bar	x	x	x
Screen[1]	x	x	x
Screen window	---	---	x
Template screen	-	x	-
I/O field	x	x	x
Graphic I/O field	x	x	x
GRAPH overview	---	x	x
Trend view	x	x	---
f(x) trend view	---	x	x
f(t) trend view	---	---	x
PLC code view	---	x	x
Pop-up screen[1]	---	x	---
ProDiag overview	---	x	x
Round button	---	---	x
Switch	---	x	---
Button	x	x	x
Slider	---	x	x
Symbolic I/O field	x	x	x
Text field	x	x	x
Text lists	x	x	x
Gauge	---	x	x
Following types in a library:			
Screen as screen window[1]	---	---	x
Alarms	x	x	x
Faceplates	---	x	x

[1]: Structured storage possible

Note

Depending on the used HMI device, the properties of the HMI objects supported by SiVArc may vary.

HMI objects that can be generated without control data

SiVArc generates or instantiates the following objects from types or master copies of a library:

HMI object	Basic Panels	Comfort Panels/ Mobile Panels/ 2nd Generation / RT Advanced	RT Professional
Screen	x	x	x
Tags			

HMI object	Basic Panels	Comfort Panels/ Mobile Panels/ 2nd Generation / RT Advanced	RT Professional
Internal tag	x	x	x
Tag table	x	x	x
Scripts			
C script	---	---	x
VB script	x	x	x
Text list	x	x	x

Properties with device dependent maximum values

The maximum values for individual properties are limited when generating the visualization for the following HMI devices:

Property	Basic Panels	Comfort Panels	Mobile Panels / 2nd Generation
Text Off (length)	320	500	500
ToolTip (length)	320	1000	1000
Text field (text)	320	32767	32767
Text list entry (text)	320	320	320

5.2.3 Sources for texts

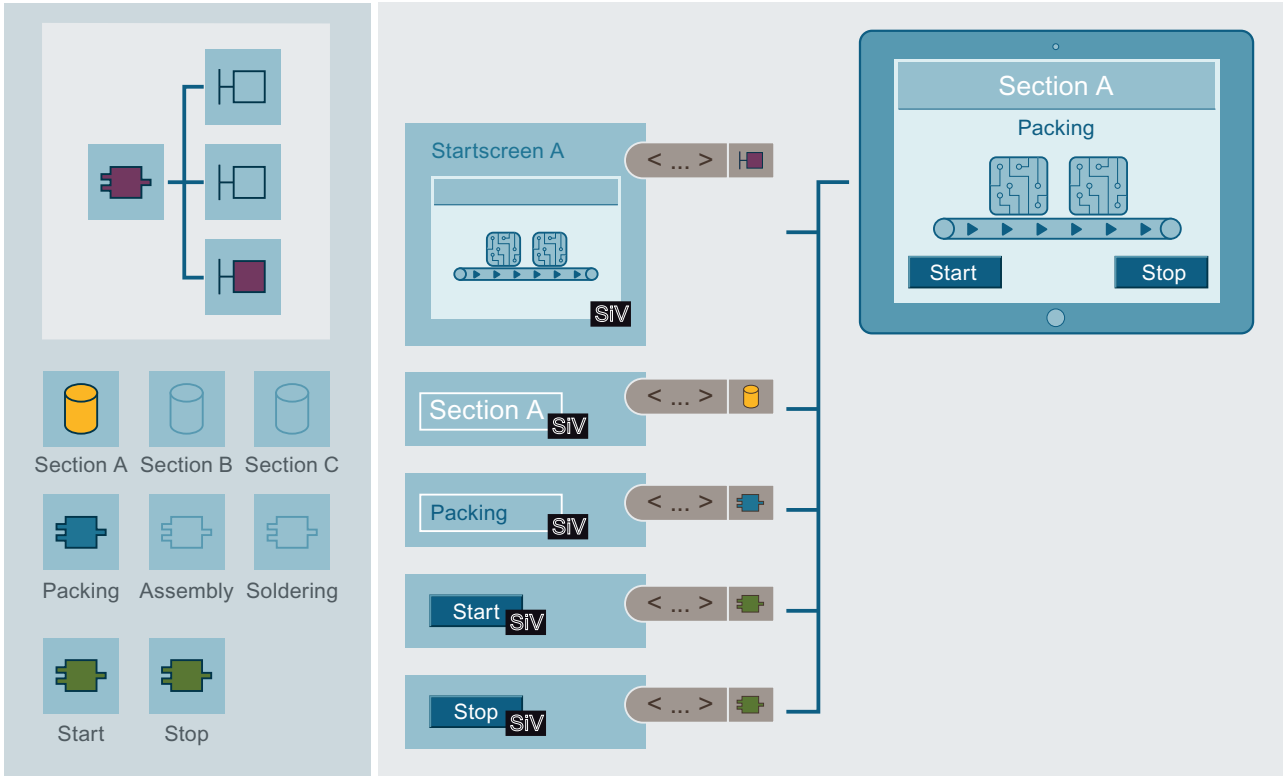
5.2.3.1 Overview of text sources in the SiVArc project

Definition

With SiVArc you access texts from STEP 7 and other TIA Portal editors for the visualization. Unlike the conventional WinCC configuration, the controller programmer creates these texts. Use these texts multiple times in the visualization using SiVArc.

The following screen shows how a screen is structured using SiVArc:

- Various text sources are available in STEP 7, for example, networks, data blocks or function blocks.
- A screen is made up of several generation templates. The SiVArc properties of the generation templates access text sources.
- During generation, SiVArc process the referenced text sources and fills the SiVArc properties of the HMI objects.



SiV	SiVArc generation template
< ... >	SiVArc property with referenced text source
	Main [OB1]
	Network in the user program
	Data block
	Process block
	Standard block

Advantages

Various texts are generated by SiVArc depending on which function block you link to a generation template. Therefore, a generation template can be used at various locations. The WinCC project can be easily adapted. The consistency of the texts is transferred from the user program in the visualization.

String functions

To maximize the reusability of generation templates or to optimize texts for the display, SiVArc provides various string functions, such as "Split", "Contains" or "Trim".

Text sources from STEP 7

A SiVArc property can refer to the following texts from STEP 7:

- Network
  - SiVArc texts and SiVArc tags
  - Network title
  - Network comment
- Data block
  - Symbolic name
  - Storage path in the project tree
  - Comment
  - Block number
  - TagPrefix
  - Type (IDB, MDB)
- Function block
  - Comment
  - Parameter value
  - Storage path in the project tree
  - Block number
  - Title
  - Type version of a library type

Text sources from hardware data

A SiVArc property can access the following properties of a HMI device:

- Runtime software
  - Name
  - Type
- HMI device
  - Name
  - Type

Text sources from libraries

A SiVArc property can access the following properties of library objects:

- Name
- Storage path in the library

See also

Influence of the user program on a generation template (Page 98)

SiVArc expression (Page 87)

5.2.3.2 SiVArc texts

Introduction

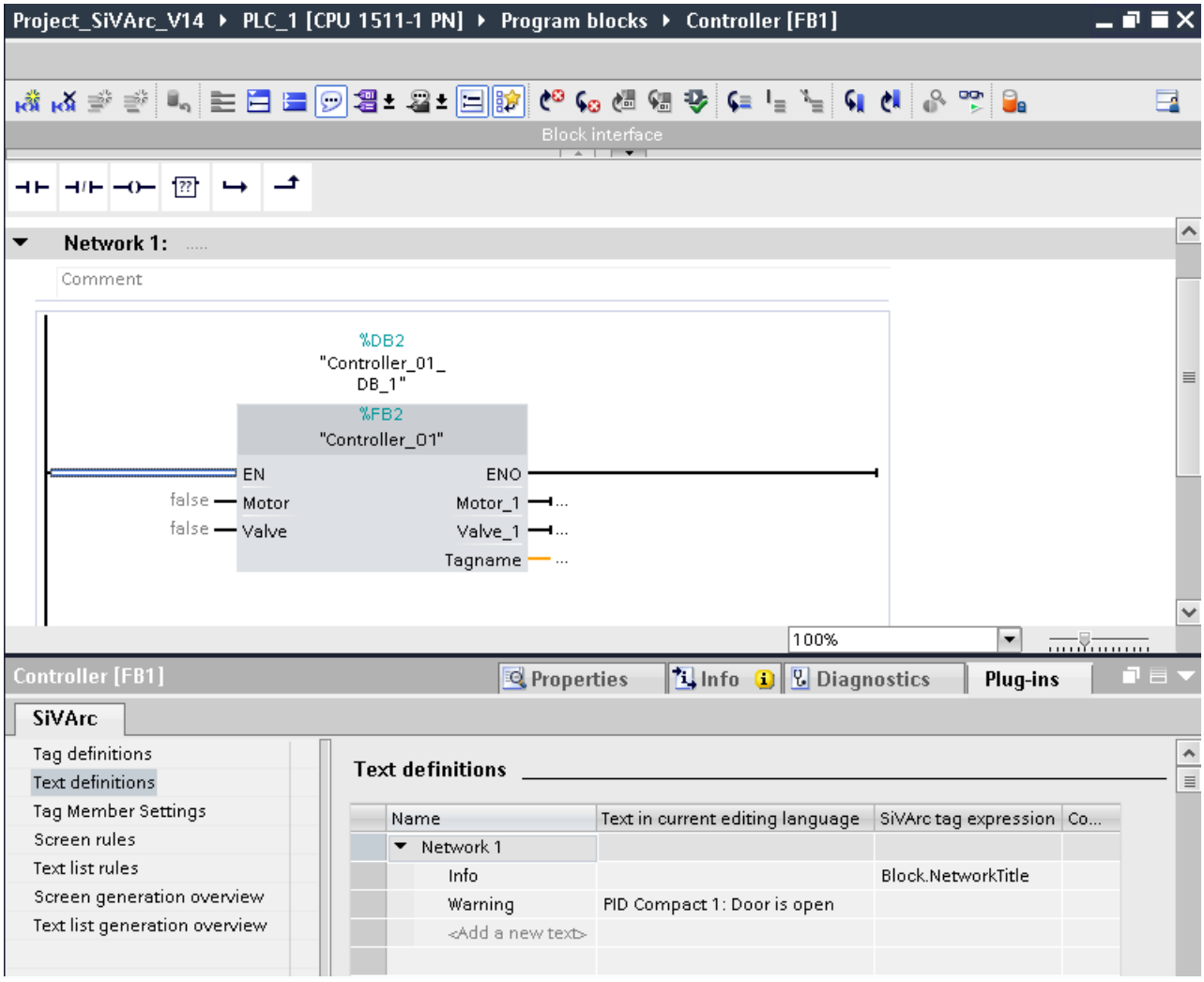
With SiVArc you can define texts as text list entries for the generation of your visualization. This functionality is integrated in the user program in STEP 7 and is available in any network title and block.

To define SiVArc texts, select the "Plug-ins" tab in the Inspector window of the required network title or block title.

Definition

SiVArc texts are text list entries created in STEP 7. When generating the visualization, they are used to generate text list entries in the HMI. Text definition and text list entry are linked by means of the name.

When the program block is used in a text list rule, the SiVArc texts are generated as text list entries in a text list.



You can specify SiVArc texts statically or dynamically:

- Static: Assign a text as text definition. You can also configure this text in multiple languages.
- Dynamic: Specify a SiVArc expression as text definition.

When you specify a text and a SiVArc expression, the SiVArc expression is used.

See also

SiVArc tags (Page 88)

Influence of multilingualism on a generation template (Page 101)

5.2.4 Supported objects in the user program

Program blocks

SiVArc supports the following program blocks:

- Function block (FB)
- Function (FC)
- Data block (DB)
  - Global DBs
  - Instance DBs

FBs and FCs are called in the user program. Only FBs and FCs are used in screen rules. You can also use FBs and FCs as master copies and types from the library.

Languages for program blocks

SiVArc supports the following programming languages for program blocks:

- STL
- FBD
- LAD

Technology objects

SiVArc supports the following technology objects:

- PID Control:
  - Compact PID
  - PID basic functions

See also

Supported data types for PLC tags (Page 227)

5.2.5 SiVArc scripting

Definition

SiVArc scripting is a scripting language derived from VBS that is used exclusively in the TIA Portal with the SiVArc option.

SiVArc scripting can address text sources in the TIA Portal. By doing so, SiVArc scripting links the user program and visualization in the TIA Portal.

You use SiVArc scripting to build SiVArc expressions in the generation templates. During generation, SiVArc evaluates all SiVArc expressions. Numerous consistent HMI objects are created from a template in this way.

"SiVArc expressions" editor

When you click in a table row of a SiVArc editor to program a SiVArc expression, a multi-line editor opens. The "SiVArc expressions" editor supports you with various functionalities:

- Autocomplete
  If you enter a letter or a character, the "SiVArc expressions" editor suggests potential operators, SiVArc object properties, properties and functions that begin with this letter or are compatible with this character.
- Syntax highlighting
  Keywords in the "SiVArc expressions" editor are highlighted using different colors. Unknown words are marked as such. The table shows the preset colors for the most important entries. You can change the default settings under "Options > General > Script / text editors".

Color	Meaning	Example
Blue	Operators And, Or, Xor, Not Boolean If function	And True If
Black	Other operators Character Other functions	+ , TrailNum
Dark cyan	String	"SG_NR"
Red	Unknown elements	$

- Error display
  The "SiVArc Expressions" editor highlights errors in the script and displays the causes of errors as tooltips.

You can change SiVArc expressions already created by selecting the expression and using commands from the shortcut menu.

You can copy or cut one or more expressions and paste them to the "SiVArc properties" tab of another HMI object.

Formulation rules

Note the following rules for the formulation of SiVArc expressions:

- An empty SiVArc expression returns an empty string.
- Mark string constants with quotation marks
- All characters are generally allowed in string constants.

- If you are using a string in quotes or backslashes, place a backslash in front as an the escape character:
  `\"`
  `\\.`
- A line break within a string constant is declared with `\n`.
- Only the following keywords (SiVArc objects) are allowed for the absolute call of a program block.
  - `Block`
  - `StructureBlock`
  - `ModuleBlock`
  - `SubModuleBlock`

  To address properties of the program block, link a SiVArc object through a point with SiVArc object properties, e.g. `ModuleBlock.SymbolicName` for addressing the symbolic name.

Input of data as binary code

To input data in binary code, use the prefix "2#", e.g. 2#00000101, to show that Bit 0 and Bit 2 of a tag are set.

If you use binary codes, observe the following:

- When necessary, you use all operators with the binary code, e.g. `2#1010 + 2#1111` = 25
- When necessary. you use binary code and SiVArc tags within an expression, e.g. `VAR_1 Or 2#11100` = 29
- When necessary, you use binary code and other constant values, e.g. `25 * 2#11100` = 700
- A binary code can contain up to 32 bits.
- You can also specify binary formatting using the "Format" function. To do this, use "`b`" as second operand.

Additional information

You can find detailed information about the structure of SiVArc expressions in the reference.

See also

Reference (Page 191)

5.2.6 SiVArc expression

5.2.6.1 Overview of SiVArc expressions

Definition

A SiVArc expression is a function that returns a text. The selected properties of generated HMI objects are filled with these texts.

The SiVArc expression accesses text sources via SiVArc objects. The SiVArc objects address blocks in the program call in STEP 7, in the HMI device or library data.

In contrast to the ES or runtime scripting, the SiVArc expression permanently links data and structures from other editors of the TIA Portal to the WinCC configuration. Changes and adaptations in the user program, the library or on the HMI devices directly effect the visualization.

Syntax elements of a SiVArc expression

The SiVArc expression is formed according to SiVArc scripting.

The following syntax elements are possible in a SiVArc expression:

- SiVArc objects
- SiVArc object properties
- SiVArc tags
- Boolean values `True` / `False`
- Strings
- Numbers
- Operators
- Predefined functions
- If conditions

Configurations with SiVArc expressions

SiVArc expressions are use for the following configurations:

- Formulate conditions for generating HMI objects

  Note

  Pay attention to the correct spelling of names in the formulation and addressing conditions.

  An error message is output only during the generation.

- Dynamic generation of properties, events and animations for the visualization
- Configure storage location and storage structure of external tags
- Configure storage structures of generated HMI objects

See also

SiVArc tags (Page 88)

SiVArc object properties (Page 211)

"Format" function (Page 214)

Structure of SiVArc expressions (Page 96)

5.2.6.2 SiVArc tags

Definition

SiVArc tags are user-defined tags. You can create multiple tags for the organization block "Main (OB1)" and for each network.

You define the tag name and the required value.

You use a SiVArc tag to store instance-specific information about a program block in the user program. You use SiVArc tags in SiVArc expressions and conditions.

Note

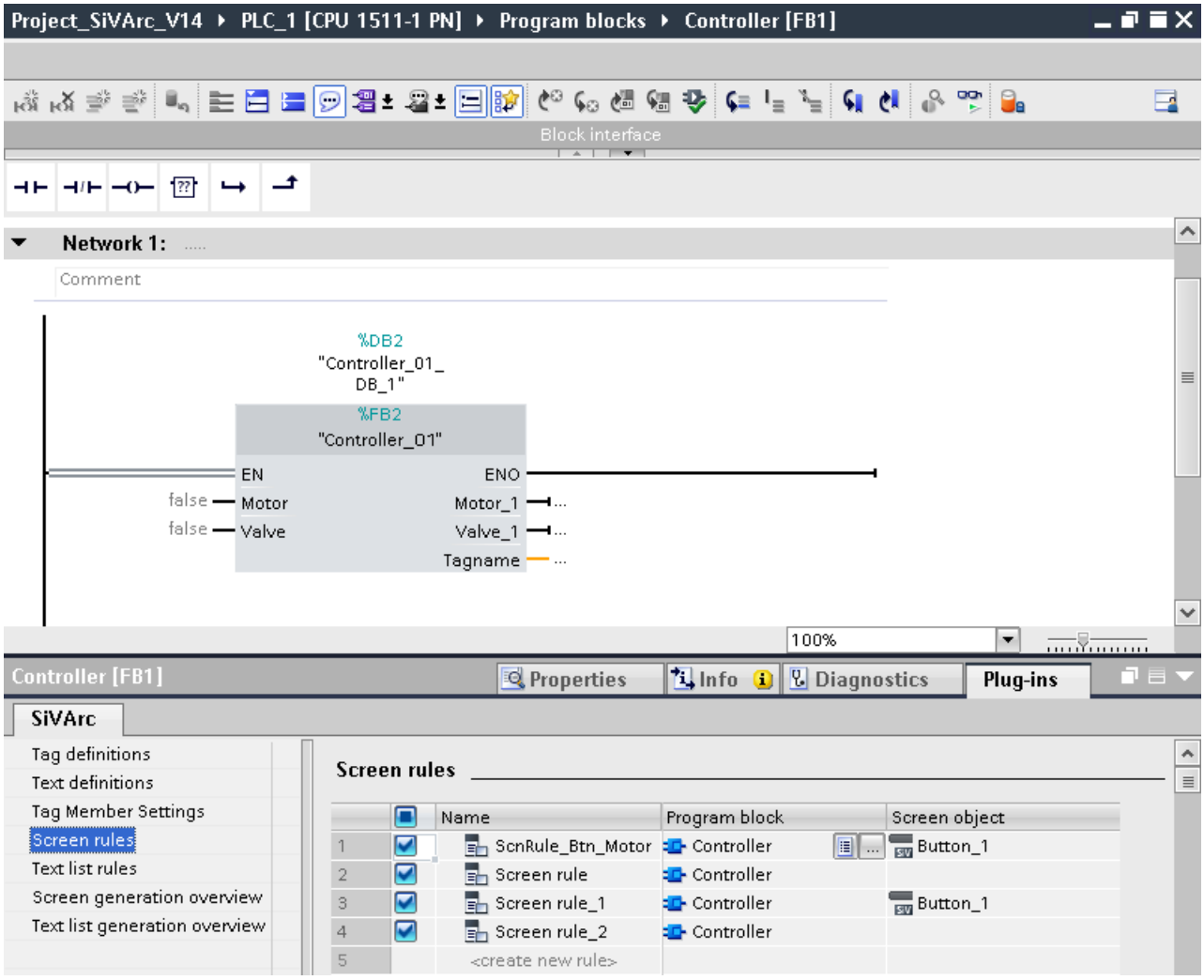
Using SiVArc tags in screen rules

If you use SiVArc tags in screen rules to evaluate instance-specific information, use the `IsDefined("Variablenname")` function. In this way, you query whether a SiVArc tag is present. This avoids generation errors due to a non-existent tag.

Creating and using a SiVArc tag

You create a SiVArc tag in the network or program block.

To select the correct network, click a program block in any area of the network in the Inspector window.



Based on the display in the Inspector window, you decide whether the tag should be created in this network.

You use SiVArc tags as follows:

- On the network
  The tag definition is valid in this network.
- At the program block
  The tag definition is valid in all networks in this program block. Through the tags, you address all program blocks that are called from the corresponding program block.
  If you use SiVArc tags in a program block, the SiVArc tag must be located in the calling block.
  Example:
  A SiVArc tag is defined in FB1. FB1 calls FB2. To enable access to the SiVArc tag, define a screen rule for FB2.

Note

Prioritizing SiVArc tags

If you use multiple SiVArc tags with the same name, the entry used is the one SiVArc found most recently. For example, if a SiVArc tag has the same name for a network and for a program block comment, SiVArc uses the tag value from the network comment.

See also

Overview of SiVArc expressions (Page 87)

"Screen rules" editor (Page 25)

SiVArc texts (Page 82)

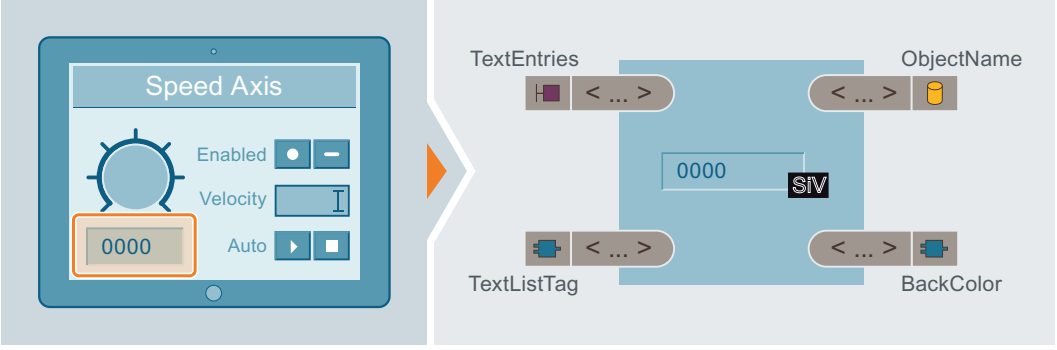
5.2.7 Requirements for a generation template

Optimized reusability

The most important requirement for a generation template is high reusability. You achieve the best reusability by using the type-instance concept from the library.

If blocks in the user program are made available and used throughout the company via the library, for example, it is practical to already assign a set of generation templates to the block type from the library.

You can then configure these generation templates with SiVArc properties so that each instance in the user program and each instance of the library type in the project tree can be visualized with this set of generation templates. There should as many variations as possible.



SiV — SiVArc generation template

< ... > — SiVArc property with referenced text source

Generation templates as types

Individual generation templates can even be created as a type and thus retain their direct relationship with the generated HMI objects.

Because HMI objects generated from types are type instances, the rules of the type/instance concept of the libraries apply.

Note

Type version

SiVArc always uses only the latest version of a type. If instances of the FC or FB type are not up-to-date in the project, SiVArc aborts the generation.

Update all types in your project before each SiVArc generation.

If you use types as generation templates, you assess the specific task and the type of application of an HMI object. Master copies are better suited for navigation buttons, for example.

Rules for using types in screen rules

If you use types, the following rules apply:

- If a type from the global library is used, SiVArc generates a copy of the type in the project library with the generation.
- As soon as SiVArc expressions are edited at the type, a new SiVArc generation is required.
- Other changes to the type are automatically updated in the instances used, even in instances of the type of generated by SiVArc .

  Note

  Simultaneous use of types and instances

  If you define screen rules yourself for an instance of a type in the project and for the type itself, SiVArc processes the type twice.

  Ensure that SiVArc processes either the instance or the type.

Device dependency

The availability of screen objects and display sizes depends on the HMI device. When creating generation templates, be aware of the devices for which the generation template can be used. You create different positioning schemes that you link to your generation template to control the arrangement of the generated HMI objects for different HMI devices.

You can also set up overflow screens in the generation template to ensure correct positioning for HMI devices of different size.

Parameterization of generation templates

To use a generation template as often as possible, it must be consistent in the naming and labeling. As many properties as possible should be linked to the suitable places of use in the user program.

In addition, a generation template ideally takes into account the storage structures in the project and the multilingualism of a project. To achieve this, you use structured SiVArc object properties such as "`<Object>.FolderPath`" and expressions that are configurable as multilingual such as "`DB.Comment`".

Dynamic resizing

Always change the size of screen windows, faceplates and text fields manually.

Although dynamic resizing is supported by SiVArc, it can lead to undesirable effects, for example, overlapping of the screen objects.

Changing generation templates

To apply changes and optimizations of a generation templates during the next generation, store the changed generation template once again in the library. The names of the generation templates are referenced in the screen rules. An updated generation template must therefore be stored in the library using the same name as the original generation template. Otherwise, the associated screen rule is invalid.

Advantages

The maintenance and optimization of generation templates help you to work efficiently with SiVArc. Your SiVArc project remains agile and easily adaptable to other STEP 7 user programs that work with standardized structures and naming conventions.

In this way, for example, you already set up your SiVArc project while creating the user program. The standardization in your company can be established and maintained with SiVArc, even for multilingual projects.

On the other hand, generation templates are individually adaptable to implement even less standardized projects with SiVArc.

See also

Generation templates in SiVArc (Page 72)

5.2.8 Parameterization concept

5.2.8.1 Example: Creating a parameterization concept

Introduction

To automatically generate as many HMI objects as possible with SiVArc, there are different approaches and options.

Example scenario

An engineering firm is tasked with deriving generation templates based on a completed user program. The project is very extensive and has a high degree of standardization.

After analysis of the project, the engineers decide to derive as many texts from the modular user program for visualization as possible and work with SiVArc to minimize customization and expansion work.

Because the user program is modularly structured and standardized, the number of SiVArc configurations can be minimized:

- Minimum number of generation templates
- Minimum number of SiVArc rules

When the next expansion of the project is pending, the engineering firm can generate the expanded visualization with a few adaptations in the SiVArc project.

Advantages

A clear and transparent HMI project is created by the structured formulation of the expressions and consistent assignment of instructions and HMI objects. Changes in the plant or in the user program can be implemented quickly and reliably. SiVArc simplifies recurring tasks in this way. Errors can be avoided in this way.

Furthermore, you can implement corporate standards more easily.

5.2.8.2 Assignment of block and generation template

Introduction

The standardization of the user program can be mapped in the SiVArc configuration. The better the user program is formed in terms of structure, modularity and standardization, the higher the application quality of the SiVArc configuration.

The optimum basic relation between an HMI object and a function module is derived from the modules of the user program.

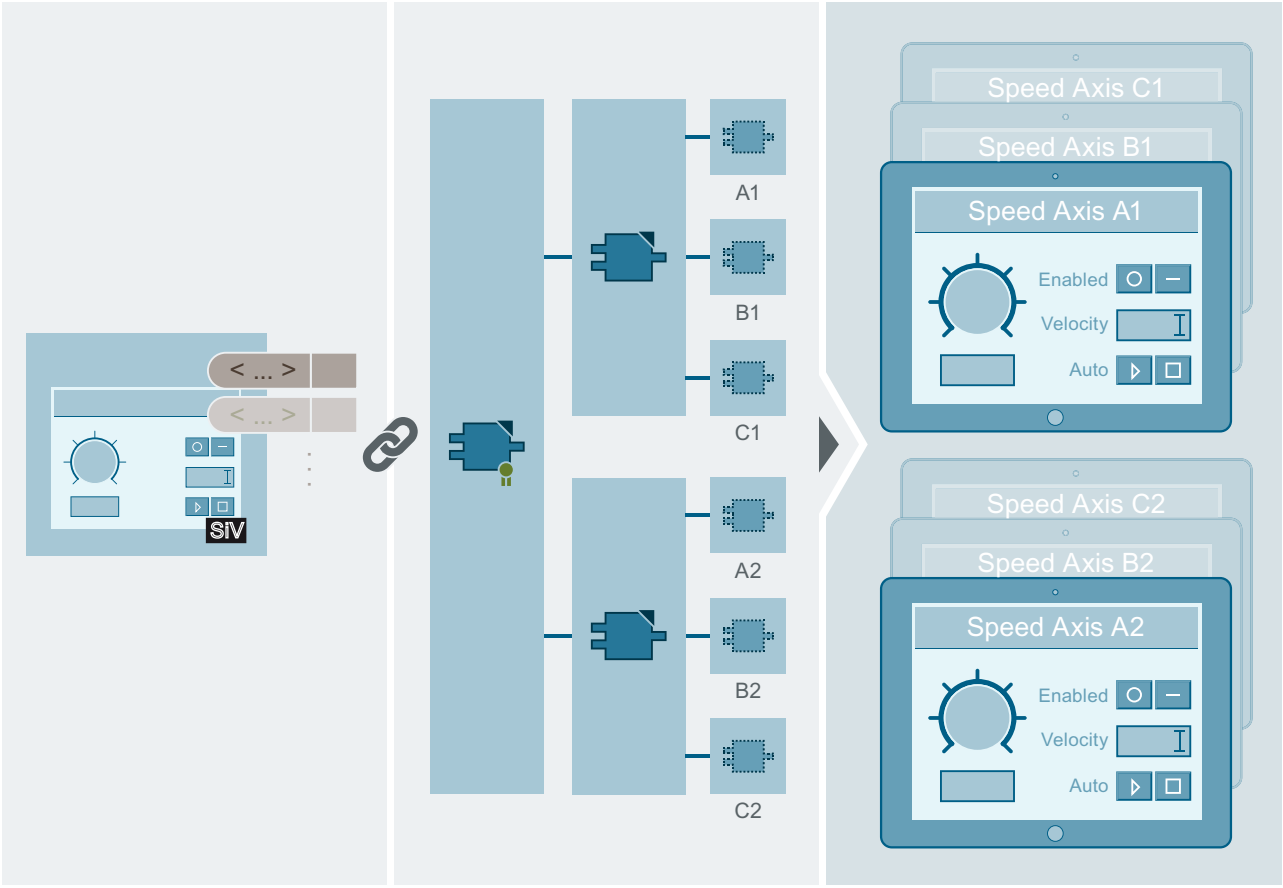
Example scenario

The user program uses the same instruction to control all virtual speed axes. This instruction is stored as a type in the library. Two type instances are used in STEP 7 and instantiated in the user program.

SiVArc concept

The following concept is derived from the example:

- The control of the speed-controlled axis is mapped in a faceplate.
- This faceplate is configured as a generation template for SiVArc.
  Names and interconnections are configured with the help of SiVArc properties.
- The SiVArc properties access texts that are only defined at the respective place of use in the user program, for example, the name of the speed-controlled axis in the network title.
- The screen rule links the library type of the instructions to the faceplate generation template.
- During generation, SiVArc runs through all instances of the block type and all its instantiated calls in the user program.



SiVArc generation template

Library type of the instruction

Instance of the library type in STEP 7

Instantiated type instance in a network in the Main OB

Result

A faceplate is generated with the texts from the user program for each call of a library type instance.

All speed axes can be visualized in the project with just one screen rule and one generation template. The relationship between the HMI object and the structure of the user program is optimally implemented.

5.2.8.3 Structure of SiVArc expressions

Concept of the SiVArc expressions in WinCC

To ensure uniqueness and clarity in the WinCC project, you use SiVArc expressions to access, for example, the data blocks of the instruction instances or the network titles. This means you should keep in mind uniqueness and consistency when naming data blocks and network titles.

The following table shows you, for example, how to use the symbolic name "SG01_FB" of a function block in the generated HMI object with SiVArc expressions.

SiVArc expression	Result
`"MyBlock"`	MyBlock
`"My\"Block"`	My"Block
`Block.SymbolicName`	SG01_FB
`"MyBlock_"&Block.SymbolicName`	MyBlock_SG01_FB
`"MyBlock_"&Block.SymbolicName&"_An"`	MyBlock_SG01_FB_An

Example: Unique HMI object names

To uniquely assign generated HMI objects that visualize a process that is used multiple times in a project to a process use, name the generated objects with the path call.

Example for a text field that labels a conveyor belt:

- Object name: Productionline_Instance_1_Dispatchunit_Instance_1_Conveyor
- Text: Dispatching

To do this, use SiVArc objects (keywords) in the SiVArc expression that address an instruction of the first three call levels in the call hierarchy.

Example: SiVArc expression for the object name of the text field:

- ```
  ModuleBlock.DB.SymbolicName&"_"&SubModuleBlock.DB.SymbolicName&"_Conveyor
  ```

Naming rules for each level of the call hierarchy are defined in the user program. Different HMI objects are provided for each process display.

SiVArc object	Function type	Name of the instruction	Symbol Name of the data block	Name of the generation template
StructureBlock	Main function	"Plantsection"	"Plantsection_1_Instanz_1_DB"	Label 01
ModuleBlock	Support function	"ProductionLine"	"ProductionLine_Instanz_1_DB"	Label 02
SubModuleBlock	Standard function	"DispatchUnit"	"DispatchUnit_Instance_1_DB"	Label 03
Block	Referenced instruction	"Initialize"	"Initialize_Instance_1_DB"	Button
  ```

- Principle of SiVArc expression for the object name of a generated text field

  ```
  ModuleBlock.DB.SymbolicName&"_"&SubModuleBlock.DB.SymbolicName&"_<
  Name_Generiervorlage>
  ```

- Generated object name
  ProductionLine_Instance_1_DispatchUnit_Instance_1_Label_03
- Generated label:
  DispatchUnit
- SiVArc expression for the label
  `Split("SubModuleBlock.DB.SymbolicName","_" (1)`

Example: Unique trigger tags

To interconnect trigger tags uniquely in the HMI object, make sure that names of the PLC tags and the runtime settings for synchronizing the tags are consistent.

The tag name from the symbolic name of the DB and the name of the PLC tag is formed in WinCC:

- PLC tag name in the DB of the second call level
  Activate
- Symbolic name of the DB
  Plantsection01
- Generated HMI tag name
  Plantsection01_Activate

The relevant instruction is located on the second call level.

- SiVArc expression of the tag name
  `StructureBlock.DB.SymbolicName&_Activate`

Example for labels

- If uniqueness is not required in a label, keep the instruction name short and concise so that it can be displayed on a button, for example:
  - Stop
  - Activate

  In the SiVArc expression in the generation template you assign the label directly via the name of the instruction:
  - SiVArc expression: `Block.Title`
- When a short and concise name is not possible, use string functions:
  - Name of the relevant data block: Plantsection1_DB
  - SiVArc expression: `Split(StructureBlock.DB.SymbolicName,"_",0)`
  - Generated label: "Plantsection1"

See also

Overview of SiVArc expressions (Page 87)

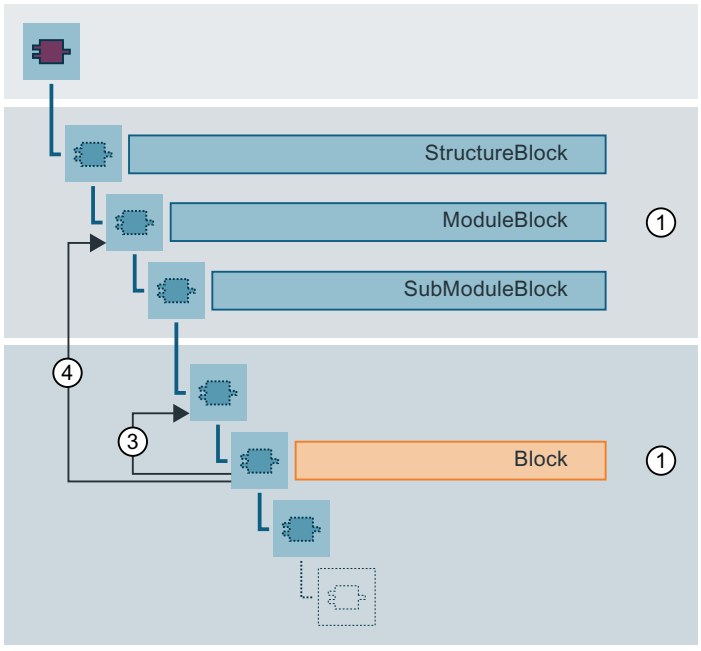
5.2.9 Influence of the user program on a generation template

Introduction

When you generate HMI objects with SiVArc, SiVArc evaluates all calls of program blocks in the user program. The user program is executed from top to bottom. If other program blocks are called in a program block, SiVArc executes the program blocks of the lower hierarchy levels first.

Addressing program block properties

The following figure shows the relationship between the call hierarchy of program blocks and the access to the properties of program blocks:



The blocks of the first three priority levels are mapped via SiVArc objects. Use the SiVArc objects for absolute addressing of these blocks.

The SiVArc object `Block` always shows the program block that is currently being executed by SiVArc, regardless of its position within the call hierarchy.

From a lower hierarchy level, you can address program blocks from each level above this. The addressing method depends on the current position in the call hierarchy. In this figure, SiVArc is currently executing a program block in the fifth level of the hierarchy.

① You can reach a higher-level block without a SiVArc object only through its relative addressing.

Beginning at the block currently being evaluated by SiVArc, enter a dot "." before each hierarchy level:

In this example, you address the name of the higher-level block as follows:

```
.Name
```

② You can reach a higher-level block with SiVArc object either through its relative or absolute addressing:

In this example, you address the block of the second hierarch level as follows:

- Relative: `...Name`
- Absolute: `ModuleBlock.Name`

Note

Working in the "SiVArc expressions" editor

Relative addressing is not supported by "SiVArc expressions" editor. To address a block relatively, enter the address directly in the input field of the SiVArc property.

Example

In an 8-level call hierarchy, you address hierarchy level 8 from an FB as follows:

- Address blocks from priority levels 1 - 3 with a SiVArc object or relatively, e.g. `StructureBlock.Version`or `.......Version`
- Address blocks from call levels 4 - 7 relatively, e.g. `...Version` (hierarchy level 5)

Use SiVArc object properties to address the properties of a program block.

Examples for access to program blocks

The following examples show how the properties of a program block are addressed in the respective call hierarchy:

Example	Standard call	Relative access to the calling block	Absolute access to the higher-level block on call level 1
Access to block names	`Block.Name`	`.Name`	`StructureBlock.Name`
Access to symbolic name of the DB	`Block.DB.SymbolicName`	`.DB.SymbolicName`	`StructureBlock.DB.SymbolicName`
Access to the value of a block parameter	`Block.Parameters("<Name Parameter>").Value`	`.Parameters("<Name Parameter>").Value`	`StructureBlock.Parameters("<Name Parameter>").Value`
Access to the comment of a tag that is assigned to the block parameter.	`Block.Parameters(<Name Parameter>).AssignedTag.Comment`	`.Parameters(<Name Parameter>).AssignedTag.Comment`	`StructureBlock.Parameters(<Name Parameter>).AssignedTag.Comment`
Access to the path of the addressed block	`Block.FolderPath` `ModuleBlock.FolderPath`	`.FolderPath` Maps the call hierarchy	`StructureBlock.FolderPath`
Access to the path of the instance DB of the addressed block The instance DB can be a single instance or multi-instance.	`Block.DB.FolderPath` Note: With `DB.FolderPath` , you only reference blocks that have a DB.	`.DB.FolderPath`	`StructureBlock.DB.FolderPathTagNaming.SeparatorChar`

If you use the standard call with the SiVArc object `Block`, the program block that is currently being executed in a SiVArc expression is addressed.

See also

SiVArc object properties (Page 211)

Overview of text sources in the SiVArc project (Page 79)

5.2.10 Influence of multilingualism on a generation template

Project languages and Runtime languages

You can optimize and efficiently implemented the multilingual configuration of the SiVArc functionality even down to the generation template.

If you configure multilingual properties with multilingual SiVArc properties in a generation template, for example, a corresponding string is generated for each runtime language.

The language settings of the TIA Portal and generation template with multilingual SiVArc objects define which multilingual texts are generated in the generated objects.

Language settings of the TIA Portal

You can generate a SiVArc project in all project languages. To do so, activate the required project languages as Runtime languages.

SiVArc uses the default generation language when multilingual SiVArc object properties are set at a monolingual property. The default generation language depends on the HMI device:

- HMI device with RT Advanced
  Runtime language at top of the list under "Runtime settings > Language & font > Runtime language and font selection"
- HMI device with RT Professional
  Runtime language that is set as "Runtime language for single-language objects" under "Runtime settings > Language & font > Runtime language and font selection"

If a project language is not set as Runtime language, the multilingual properties in the project are generated with the value from the master copy for this project language. The SiVArc expression for this property is not evaluated in this project language.

If a value is not set in a multilingual tag, the empty string is generated as property value for this language.

Multilingual SiVArc objects

You work with the following SiVArc objects when you configure a multilingual SiVArc project:

- Multilingual properties
- Multilingual SiVArc object properties
- SiVArc texts for text list entries

Multilingual WinCC properties

SiVArc supports the following multilingual properties.

The expressions of these properties are evaluated individually by SiVArc for each Runtime language. If an expression includes multilingual SiVArc object properties, the evaluation results in different values for the respective Runtime languages.

HMI object	Property
Bar	Title Tooltip Unit
Screen	Display name
Screen window	Title
Text field	Text
I/O field	Info text
Graphic I/O field	Tooltip
Switch	Title TextOFF TextON Tooltip
Round button	Text Tooltip
Button	TextOFF
Gauge	Title Unit

The expression is evaluated in the default generation language for all other properties for which you can use a SiVArc expression.

Multilingual SiVArc object properties

The following SiVArc object properties are configurable in multiple languages:

- `Title`
- `SymbolComment`
- `DB.Comment`
- `NetworkTitle`
- `NetworkComment`

Using SiVArc expressions in multilingual context

You use multilingual and monolingual SiVArc object properties in SiVArc expressions. The reference describes which SiVArc object properties are multilingual. You use SiVArc object properties in multilingual and monolingual SiVArc properties. The SiVArc expressions are evaluated as follows in this context:

	Monolingual SiVArc object properties	Multilingual SiVArc object properties
Monolingual property	The same character string is generated for each Runtime language.	The tag is evaluated in the default generation language. You specify the default generation language in the Runtime settings of the HMI device.
Multilingual property	The same character string is generated for each Runtime language.	The tag is evaluated for all configured Runtime languages. The configured character string is generated for each Runtime language.

See also

Title (Page 209)

SymbolComment (Page 208)

Comment (Page 203)

DB (Page 193)

NetworkTitle (Page 207)

NetworkComment (Page 206)

SiVArc texts (Page 82)

5.2.11 Storage strategies for generated objects

Overview

SiVArc offers for screens and tags the option to control the storage structure of generated objects by means of SiVArc expressions in the screen rules or in the generation template.

There are various storage strategies for this:

- Mapping of the storage structure in the project tree in STEP 7
- Mapping of the storage structure in the project library
- Individual storage structure

The SiVArc storage strategies are based on the generated HMI objects in the project tree below the HMI devices in the areas of screens and tags.

The structured storage of SiVArc rules provides you with the functions of SiVArc editors.

Application example

The blocks are stored in the project tree, for example, according to function. This storage form can be created automatically for the associated screens. In the generation templates of screens, configure a SiVArc expression that references the path of the blocks in the project tree.

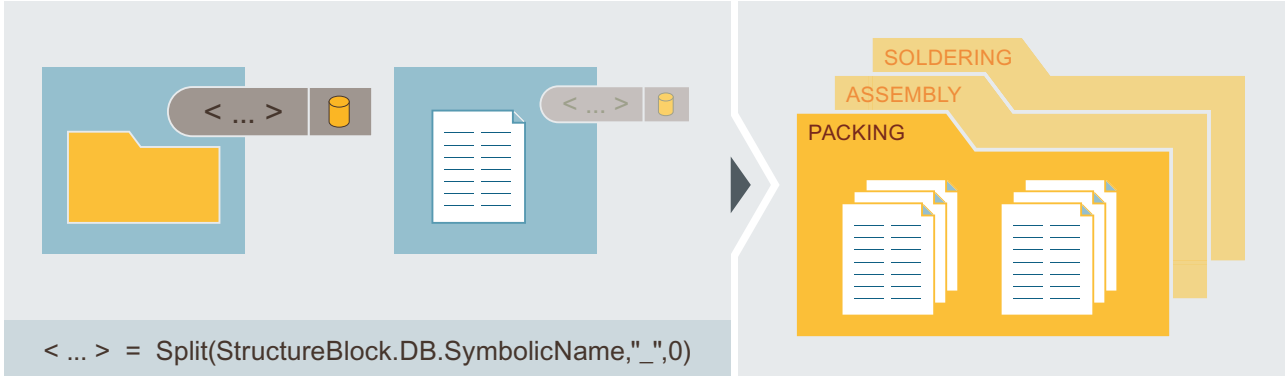
Advantage

The SiVArc storage strategies increase the consistency and standardization of your visualization project. If another storage strategy is required, you can restructure your project with little effort.

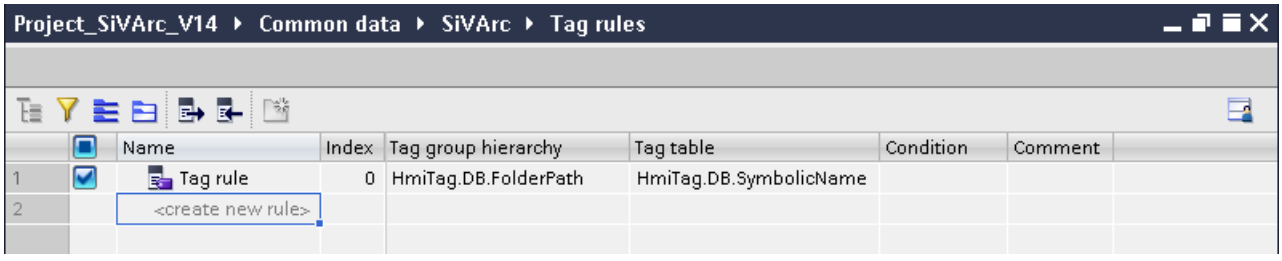
Controlling the storage of generated tags

The following strategies are possible for the storage of tags:

- Mapping of the storage structure in the project tree in STEP 7
  You can use the SiVArc expressions `HmiTag.DB.SymbolicName` and `HmiTag.DB.FolderPath` for the "Tag rules" editor to structure the tag tables based on the control program using only one tag rule.
  The project is only structured once at the controller end.
- Individual storage structure
  You define the target folder and the tag tables individually using the "Tag group" and "Tag table" columns.



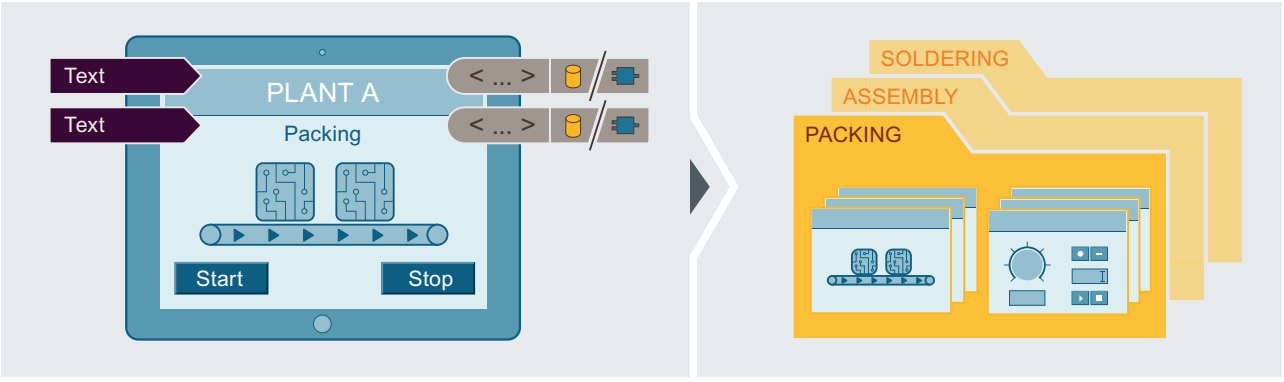
Double-click "Common data > SiVArc > tag rules" in the project tree to open the "Tag rules" editor.



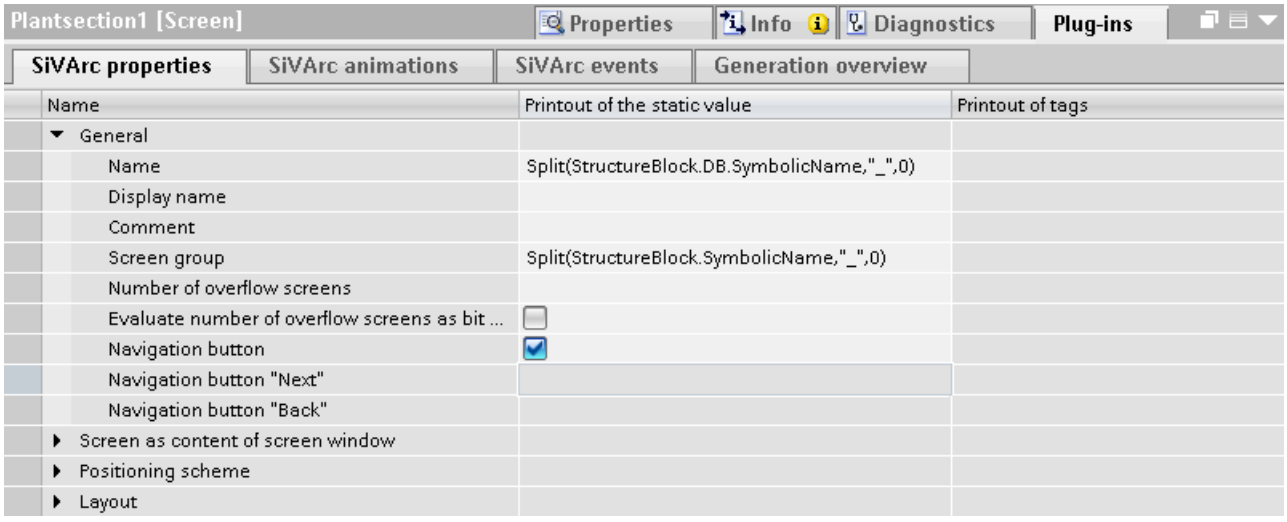
Controlling the storage of generated screens

The following strategies are possible for the storage of screens:

- Mapping of the storage structure in the project tree in STEP 7
- Mapping of the storage structure in the project library
- Individual



You control the storage of screens in the project tree with the "Name" and "Screen group" properties in the generation template of a screen. If you specify a text string under "Screen group", a group is created in the project tree with this name. The screens created based on the generation template are then stored therein.



You can synchronize the storage structure and naming of the generated objects with the library in the generation template of a screen type.

To do this, use the SiVArc expressions "`LibraryObject.FolderPath`" and "`LibraryObject.Name`"

SiVArc object property	Referenced object	SiVArc property
`LibraryObject.FolderPath`	Storage path of the screen type in the library	Screen group: The storage path from the library is generated in the project tree. Name*: The generated screen is named after the folder in which the screen type is stored.
`LibraryObject.Name`	Name of the screen type of the library	Name: The screen is named after the screen type. Screen group: The screen is stored in a folder with the name of the screen type in the project tree.

*) Use `LibraryObject.FolderPath` for the SiVArc property "Name" only if the screen type in the library is stored in one hierarchy level only. If you want to use a multi-level storage hierarchy, you can substitute the expression and `LibraryObject.FolderPath` for the backslash.

Alternatively, you can define the storage folder and screen name individually.

See also

Creating tag rules (Page 154)

Creating a generation template for a screen (Page 131)

Name (Page 205)

FolderPath (Page 203)

LibraryObject (Page 196)

HMITag (Page 195)

5.2.12 Example: Achieving high flexibility

Example scenario

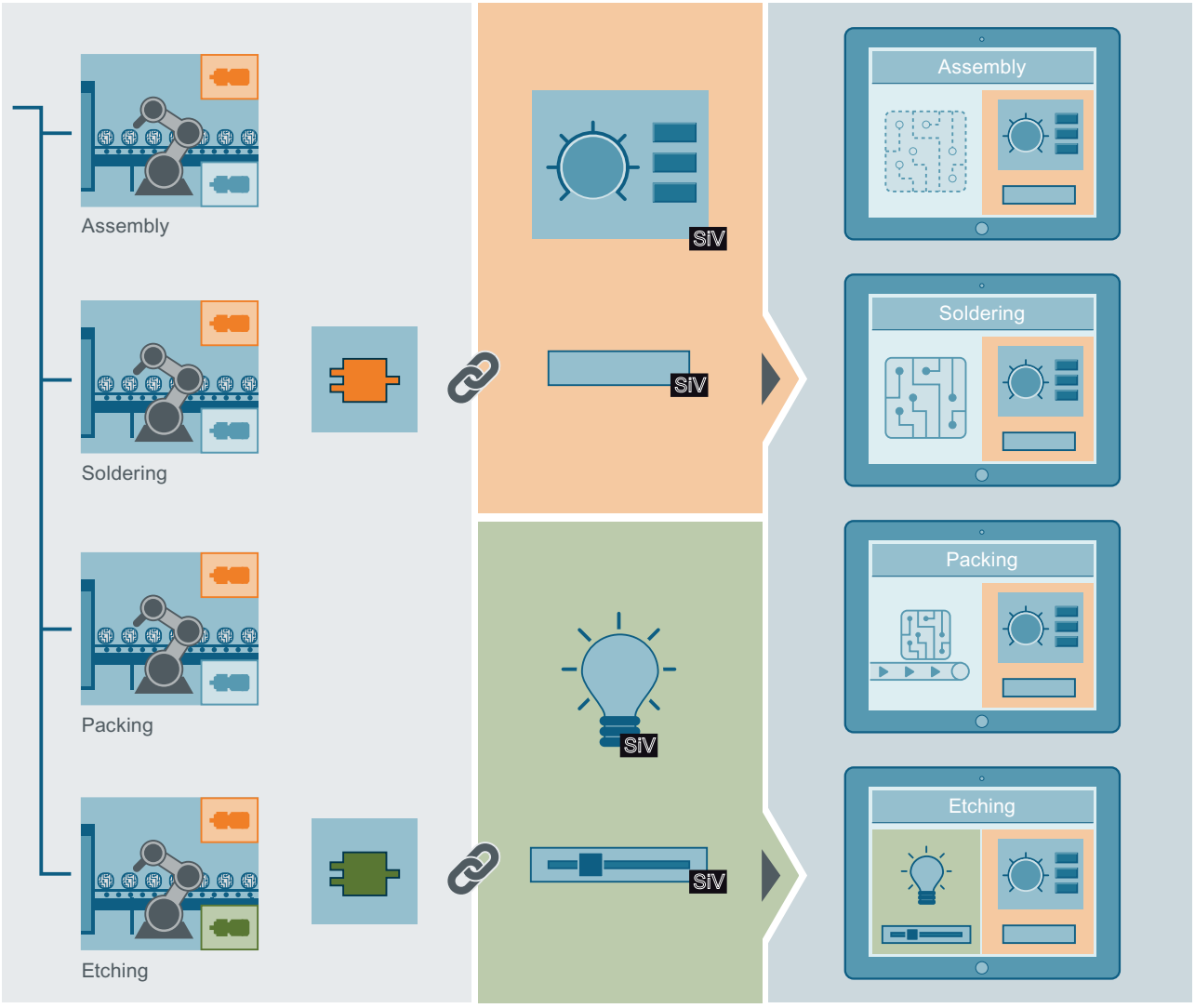
A printed circuit board factory has the plant sections "Fitting", "Soldering" and "Packing". A new type of circuit boards requires the planning and completion of an additional stage, "Etching". The plant largely consists of standard blocks.

Once the new plant section is created, the modules are tested and optimized for operation.

Implementation concept

Much of the functionality of the "Soldering" plant section is reused for the "Etching" plant section and therefore generates no additional SiVArc configurations.

Additional functions are required for the "Etching" plant section. These functions are standard functions which have already been assigned generation templates in screen rules. The engineer forms a group for the additional rules required for the "Etching" production stage and enables the relevant HMI devices.



Function block "Rotational speed"

Function block "Etching"

To test the plant, the project engineer collectively enables a screen rule group and disables modules not needed for the test. The visualization engineer tests the user interface generated from this. Based on the test result, generation templates and rules are optimized by using conditions of or modifying the SiVArc properties.

5.2.13 Example: Achieving high reusability

Example scenario

An engineering firm receives a contract from a new customer to configure a standard plant for the manufacture of printed circuit boards.

The firm already has an optimized SiVArc project for PCB production and wants to reuse it for the new customer.

The plan is to ensure the greatest possible consistency for the operation and visualization of the same functions including, for example, the following functions:

- ON/OFF
- Travel to basic position
- Display status

Implementation solution

The standardized user program contains many function blocks and standard functions. System blocks are created as library types. The engineering firm can therefore set up a full, existing set of generation templates for each standard function.

The existing generation templates for standard functions access the text sources directly on the standard block via SiVArc expressions. The call hierarchy is not taken into consideration. The trigger tags are uniquely referenced through the name of the data block of the system block. Each reuse of the type produces the associated operating elements in the visualization based on the same set of generation templates. Adaptations are therefore not necessary.

Color and forms of the generation templates are adapted to the operating concept and made available from a project-specific library.

The new corporate design for the operating screens is connected to the generation templates via positioning schemes.

5.2.14 Example: Create generation template for screen windows

Example scenario

For training purposes, several existing HMI devices are to be duplicated at a redundantly designed workplace.

Objective

Various screen windows with the corresponding start screens are to be generated on the redundant operator station. The name of the screen window indicates the program block that is visualized in it.

Screen windows in SiVArc

Screen windows are not generated directly as screen objects. A screen window is implicitly created when a screen is specified as screen object.

If there is a "DefaultScreenWindowControl" generation template in the project, SiVArc generates screen windows based on this template. If this template is not available, SiVArc creates a copy of the screen window from the toolbox.

Requirement

- A blank operator screen of the redundant workplace is stored as a generation template for the screen window named "Screen_Training".
  The SiVArc property "Name" of the generation template is configured with the SiVArc expression"`Block.DB.SymbolicName&"_SWC`".
- The "Plantsection_Soldering" program block is contained in the user program and is called repeatedly in OB1.

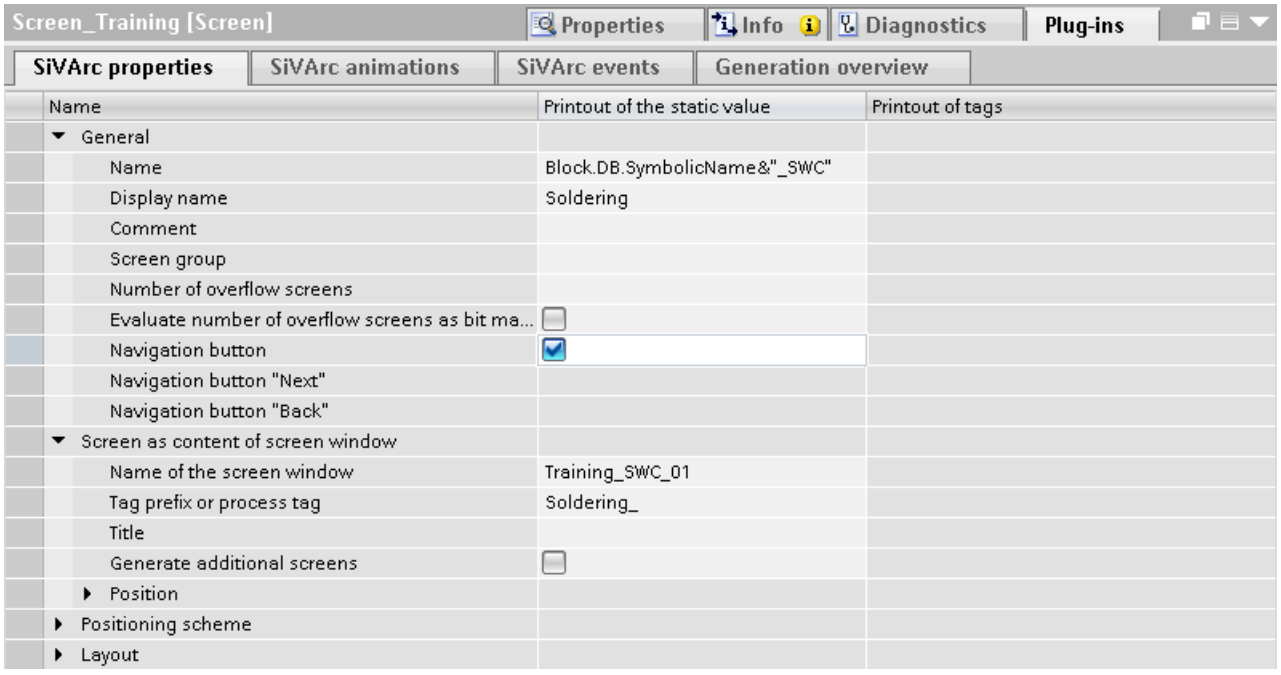
Procedure

To create a generation template copy for a screen window, follow these steps:

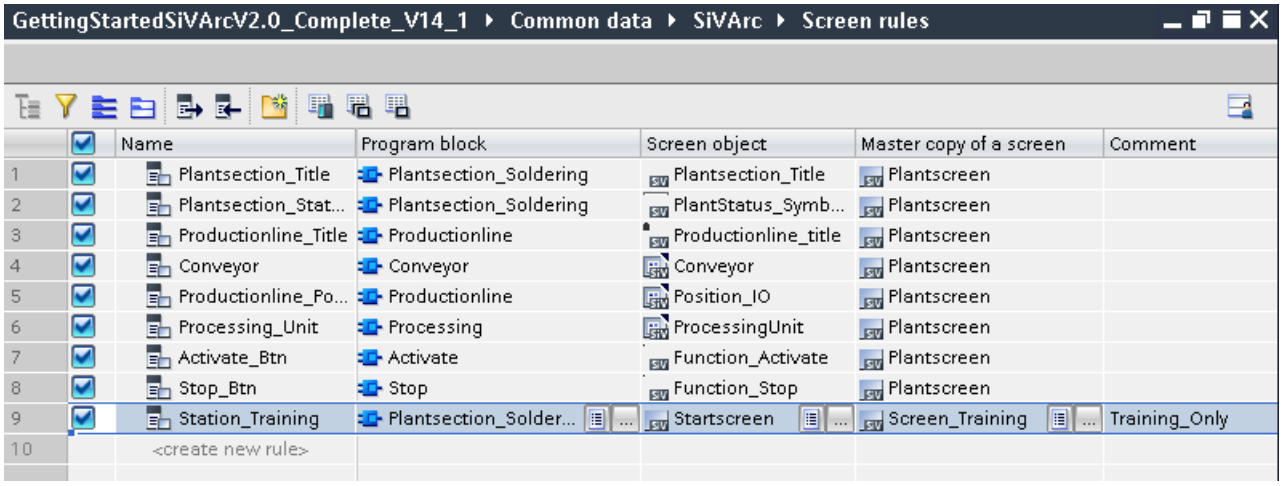
1. Open the generation template "Screen_Training" from the library.
2. Under "Plug-ins > SiVArc properties", configure the name of the screen window with the SiVArc expression "`Block.DB.SymbolicName&"_SWC`".
   The `Block.DB.SymbolicName` part references one of the following names depending on the type of block call:
   - Global: Symbolic name of the Instance DB
   - Local: Name of the block instance

   The `&"-SWC"` part adds the suffix to the name for "Screen Window Control".

3. Under "Plug-ins > SiVArc properties > Screen as content of screen window" in the Inspector window, configure desired properties:
   - Under "Name of the screen window", enter a unique name or a SiVArc expression for the screen window to be generated on the target screen.
   - Under "Tag prefix", enter the name of the tag that uses a user data type. If necessary, use a SiVArc expression.



4. Enter a new screen rule with the name "Station_Training".
5. Enter a "For training purposes only" comment.
6. Select the central program block "Plantsection_Soldering".
7. Under "Screen object", select the "Startscreen" generation template.
8. Under "Screen", select the "Screen_Training" generation template.



Result

A "Screen_Training" screen" is generated with a screen window for each call of the "Plantsection_Soldering" program block. The start screen of the "Soldering" plant is included in the screen window.

The name of the screen window contains a reference to the program block visualized in it and the suffix -SWC, for example, "Plantsection_Soldering_Instance01_SWC".

Screen windows for multiple screens

To display other screens in a generated screen window, for example, diagnostic screens, place the desired generation templates in the same library folder, for example, "Training_Screens". Also configure the following SiVArc properties:

- In the screen you have selected as the "Screen object", configure the SiVArc property "Generate additional screens".
- In the screen you have selected as the "Screen object", configure the SiVArc property "Screen in screen window".

See also

Example: Generating template screen using copy rule (Page 111)

5.2.15 Example: Generating template screen using copy rule

Example: Generation of template screen using copy rule

For training purpose, consider programming the assembling of parts in an automotive manufacturing plant, where PLC contains the main block as "Assemble_body", "Production_time" and "Production_order" as function blocks.

Objective

In automotive manufacturing plant, the engineering office receives a contract to apply single template screen to all HMI devices during SiVArc generation. The copy rule feature copies, creates multiple copies of template screen, and applies to all the blocks configured in the PLCs using user defined SiVArc expressions. Thus, copy rule optimizes, and increases the productivity of a manufacturing plant through template screens.

To generate template screens of name "Production_order" using copy rules, follow the below mentioned procedure.

Requirement

- HMI devices configured with PLC blocks.
- Template screen configured.

Procedure

To create template screens using copy rules, follow these steps:

1. Create a new template under "Templates > Add new template".
2. In the "Properties" window of the template screen, configure the template with name "Production_order".
3. Under "SiVArc > Copy rules", place the template screen in "Master copies" folder.
4. Click "create new rule".
5. Enter the name of the copy rule.
6. Under "Library object", browse the template from the "Master copies" folder. Library objects support intellisense.
7. Generate SiVArc visualization on the HMI device.

Result

A template screen is generated with name "Production_order" under the "Templates" folder. The existing template screen name is suffixed with "_Renamed".

See also

Generation templates in SiVArc (Page 72)

5.2.16 Example: Generating HMI screen using template property

Example: Generation of HMI screen using template property

For training purpose, consider programming the assembling of parts in an automotive manufacturing plant, where PLC contains the main block as "Assemble_body" function blocks as "Production_time", "Production_order", and template screen as "Production_order".

Objective

In automotive manufacturing plant, when multiple HMI screens must be generated using templates, the template property is used. The templates are added using the SiVArc property, and are applied to all the blocks configured in the PLCs using user defined SiVArc expressions. Thus, resulting in increase productivity and scalability across multiple HMI devices.

To generate HMI screens through template property "Production_order", follow the below mentioned procedure.

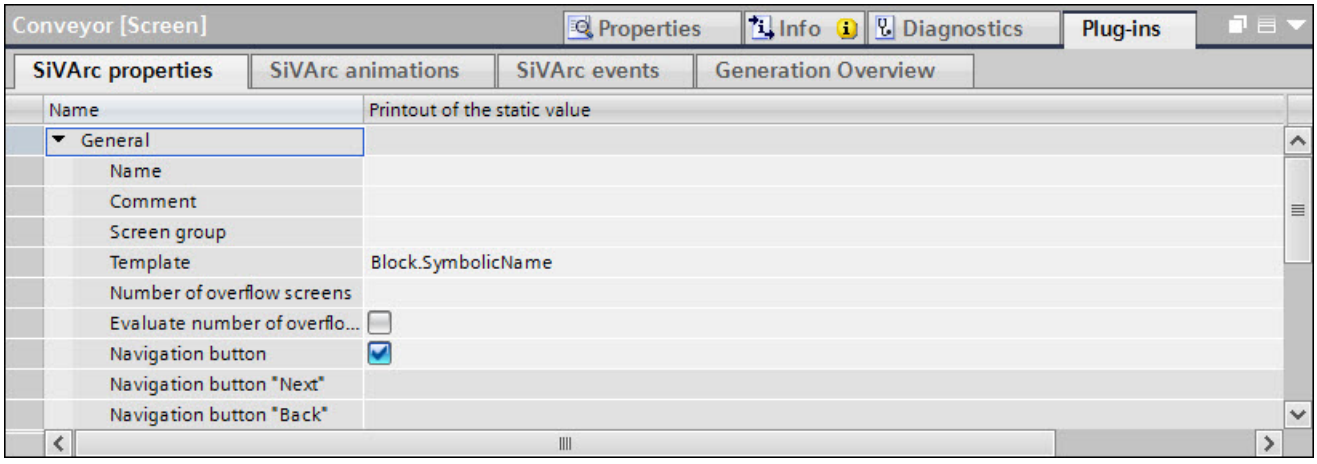
Requirement

- HMI devices configured with PLC blocks.
- Template screen configured.

Procedure

To generate HMI screens using template property, follow these steps:

1. Under "Screens > Add new screen > Conveyor".
2. Under "Plug-ins > SiVArc properties", configure the name of the screen with the SiVArc expression "`Block.SymbolicName`".
3. The `Block.SymbolicName` part references one of the following names depending on the type of block call:
   - Global: Symbolic name of the block.
   - Local: Name of the block instance
4. In the inspector window, under "Plug-ins > SiVArc properties > Template", you can either enter the template name or configure a SiVArc expresssion such that the expression resolves to an existing template name in the project library. The resolving expression name will be displayed in the target screen.

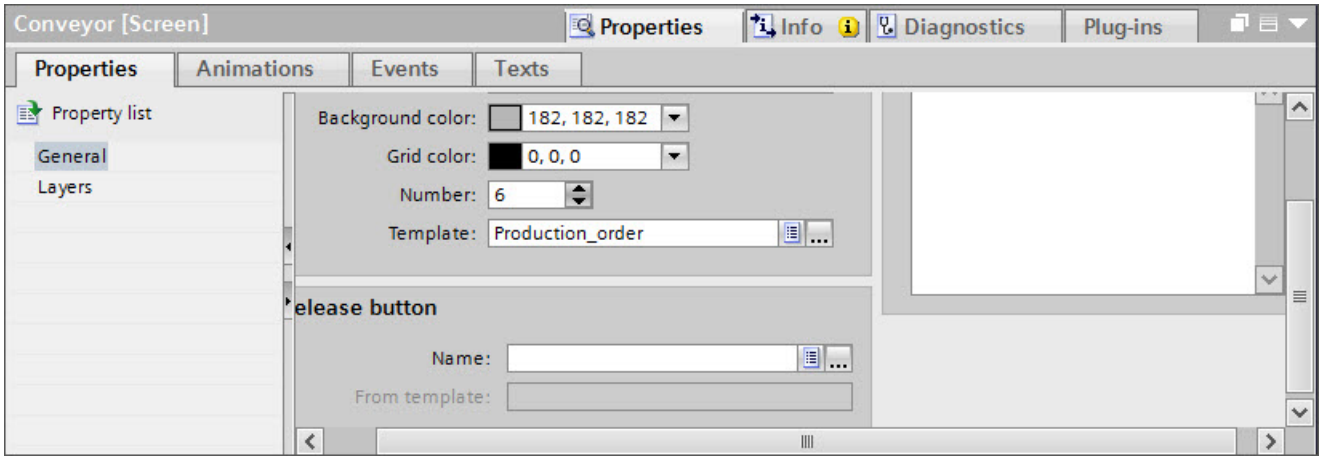


5. Place the configured HMI screen in the "Master copies" folder.
6. Under "SiVArc > Screen rules", click "Create new rule". Enter the name of the screen rule as "Conveyor rule".
7. Under "Library object", browse for a PLC block , and the screen template from the "Master copies" folder. Library object and program block supports intellisense.
8. Generate SiVArc visualization on the HMI device.

Result

A HMI screen "Conveyor" is generated with name "Production order" as the template name. The old template screen name is suffixed with "_Renamed".

The following screenshot shows the template name, which is automatically applied after SiVArc generation.



See also

Generation templates in SiVArc (Page 72)

5.2.17 Example: Create generation template with animation

Example scenario

If a robot moves to the basic position, the robot should always flash with changing colors in the screen.

Objective

The generation template for the robot is a graphic I/O field that is configured with a design animation. The status specifications follow a SiVArc expression.

SiVArc animations

SiVArc supports the following types of animation:

- Animation with tag connection (only available in WinCC Runtime Professional for S7-GRAPH overview)
- Animations of the "Display" category

For these animations, you use a SiVArc expression to define the process tags which trigger the animation in Runtime.

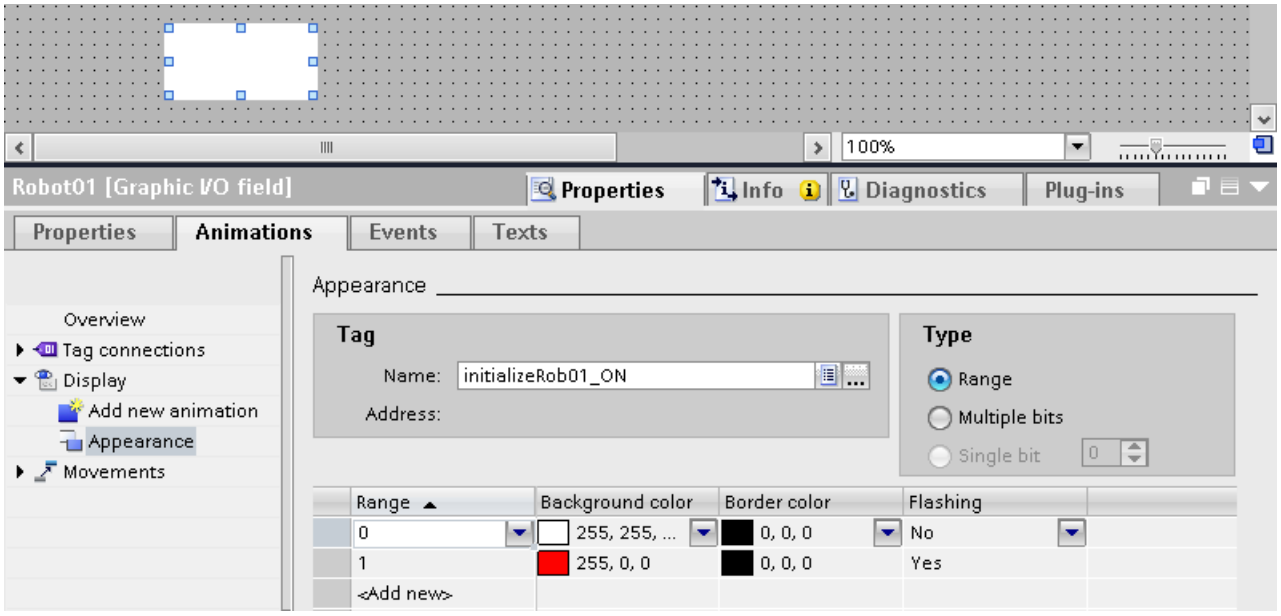
Requirement

- A "Robot01" graphic I/O field is configured as generation template to display the robot.
- The PLC tags are synchronized with the HMI tags.

- The "initializeRob01_ON" tag contains the status information of the travel to the basic position and is connected to the external tag "`Soldering_Instance_01_initializePosRob01`".
- The "Rob01" program block is contained in the user program.
- A screen rule is created that links the graphic I/O field "Robot01" to the program block "Rob01".

Procedure

1. Open the generation template of the graphic I/O field.
2. Configure a design animation.



   – Select the "Area" type.
   – For the "1" area, select red as the background color and enable the "Flashing" option.
3. Open the "SiVArc animations" tab under "Properties > Plug-ins".
4. For the "Appearance" animation under the "Tag expression", configure the SiVArc expression "`StructureBlock.DB.SymbolicName&"_initialPosRob01""`
5. Overwrite the existing generation template in the library.

Result

The generation creates the graphic I/O field "Robot01" for each instance of the "Rob01" program block. The animation was configured for each graphic I/O field. When the robot moves to the basic position in runtime, the robot flashes red in the screen.

5.2.18 Example: Create generation template with event configuration

Example scenario

The "Activate" button should trigger the drive to the basic position of the milling/soldering or positioning robot.

Objective

In the generation template of the "Activate" button, the "Click" event is configured with the SetBit" system function.

The unique "Tag" parameter for the system function is composed of the text sources from STEP 7 during generation.

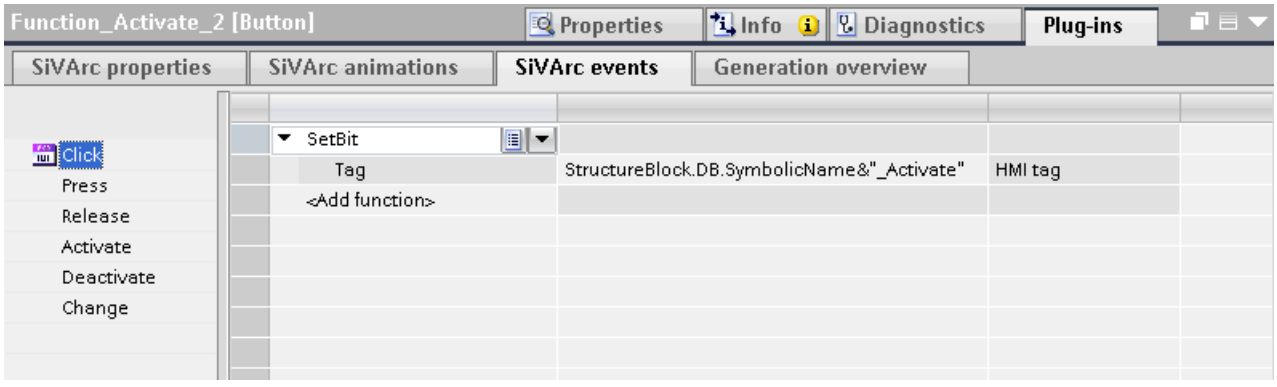
Requirement

- The generation template of the "Activate" button is configured with the "SetBit" system function.
- An screen rule is created in which the "Activate" generation template is linked to the relevant function block.
  The function block in our example is located on the second level of the call hierarchy and is addressed with the SiVArc object "`StructureBlock`".

Procedure

To create a generation template with event configuration, follow these steps:

1. Open the generation template "Activate" button in WinCC.
2. In the Inspector window under "Plug-ins > SiVArc events > Click", configure the SiVArc expression "`StructureBlock.DB.SymbolicName&"_Activate`" as a tag.



3. Overwrite the existing generation template in the library.

Result

A button that can trigger and exit the function is generated for each call of the relevant function block. The tags are already interconnected for all instances.

5.2.19 Example: Create generation template with script configuration

Example scenario

In a SiVArc sample project, temperature readings should always be output in degrees Celsius in addition to the values in Fahrenheit.

Implementation concept

To switch an additional display object, a button with the appropriate script is generated in each project.

If there is no need for conversion in a project, the SiVArc configuration engineer disables the respective screen rule in the next generation or limits the screen rule to a selection of HMI devices.

Availability of system functions and scripts

When you connect scripts to events, these scripts must exist on each target device. If the configured script does not exist in the "Scripts" editor on the target device, the display and operating object is generated without this script connection.

SiVArc supports the configuration of system functions and scripts with SiVArc expressions at all events of screens and screen objects. SiVArc supports system functions from the following categories:

- Calculation
- Bit processing
- Screens

SiVArc supports a limited selection of SiVArc events and system functions for faceplates. You can find an overview of the supported system functions in the section "Reference".

Note

Device dependency

The number and type of events in a display and operating object depends on the configured HMI device.

Additional information on device dependency of events is available in the online help of the TIA Portal in the section "Working with system functions and Runtime scripting" in the reference.

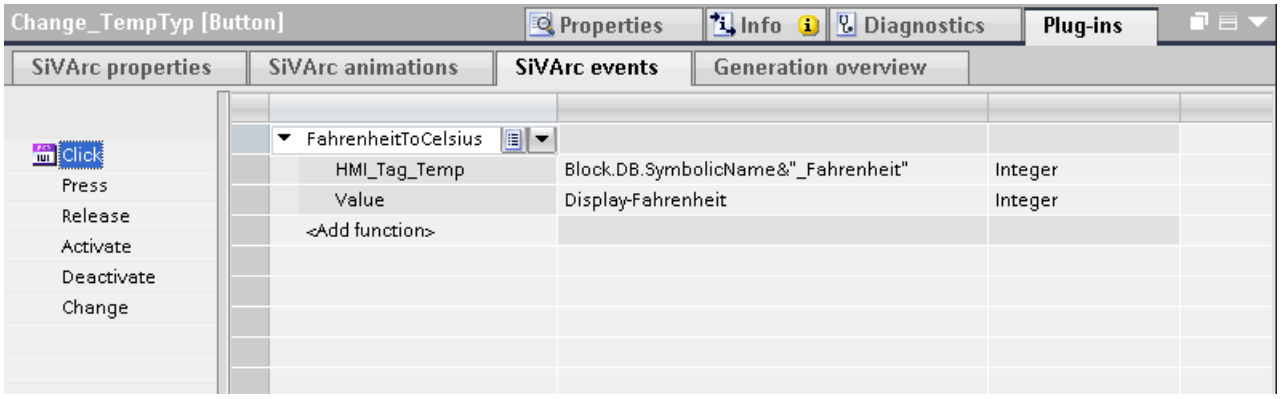
Requirement

- A "FahrenheitToCelsius" script is programmed, which converts degrees Fahrenheit to degrees Celsius and outputs the result in an I/O field.
- The script has the parameters "HMI_Tag_Temp" and "Value".

- The script is created on all target devices.
- A button is created as a "Change_TempTyp" generation template and linked to the relevant function for temperature measurement with a screen rule.

Procedure

1. Open the generation template of the "Change_TempTyp" button in WinCC.
2. Configure the "FahrenheitToCelsius" script for the "Click" event in the Inspector window under "Plug-ins > SiVArc events" .
3. Configure the "HMI_Tag_Temp" parameter with the SiVArc expression "`Block.DB.SymbolicName&"_Fahrenheit`".
4. Configure the "Value" parameter with the name of the output field "Display_Fahrenheit".



5. Overwrite the existing "Change_TempTyp" generation template in the library.
6. Create a screen rule for the "Display_Fahrenheit" I/O field.

Result

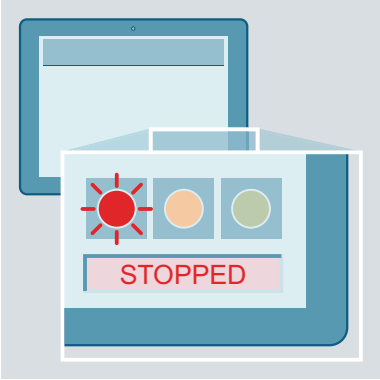
When the visualization is generated, the "Change_TempTyp" button and the "Display_Fahrenheit" I/O field are created for each call of the relevant function block.

The "FahrenheitToCelsius" script is linked to the "Change_TempTyp" button. The converted value from the respective "Fahrenheit" tag of each instance of the function is displayed in the "Display_Fahrenheit" I/O field in runtime.

5.2.20 Example: Creating generation templates for text lists

Example scenario

A traffic light indicates the plant status. Each color is assigned a status text that is displayed in a symbolic I/O field beside the traffic light.



Objective

A generation template for a text list is provided from the library. The required text definitions are to be maintained in the user program on the network.

The generation template for the text list is assigned the dynamic trigger tag. The data block for the relevant function is called "Plantsection1_DB". The name of the text list should refer to the first part of the symbolic name of the block: "Plantsection1_Textlist". With the "Split" function, the "_DB" is shortened in a SiVArc expression for the text list name.

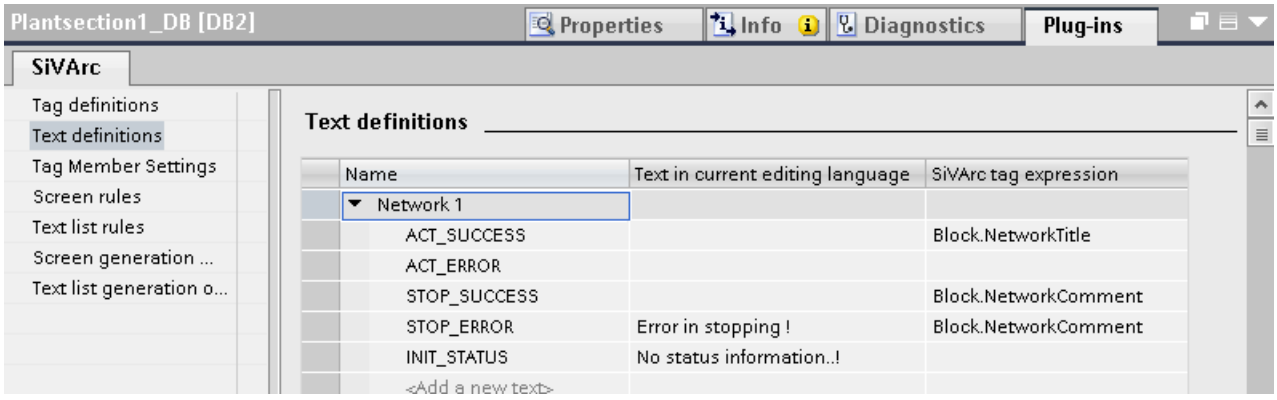
Requirement

- The "Textlist_State" generation template has been stored in the library.
- A text list rule is created linking the "Textlist_State" generation template to the "Plantsection" function block.

Define text list entries

To create text definitions, follow these steps:

1. Select the desired network in the user program for which the text list entries should be defined.
2. Select the "Text definitions" category under "Plug-ins > SiVArc".
   The following text definitions are defined in the user program on the network of the first instance of "Plantsection":



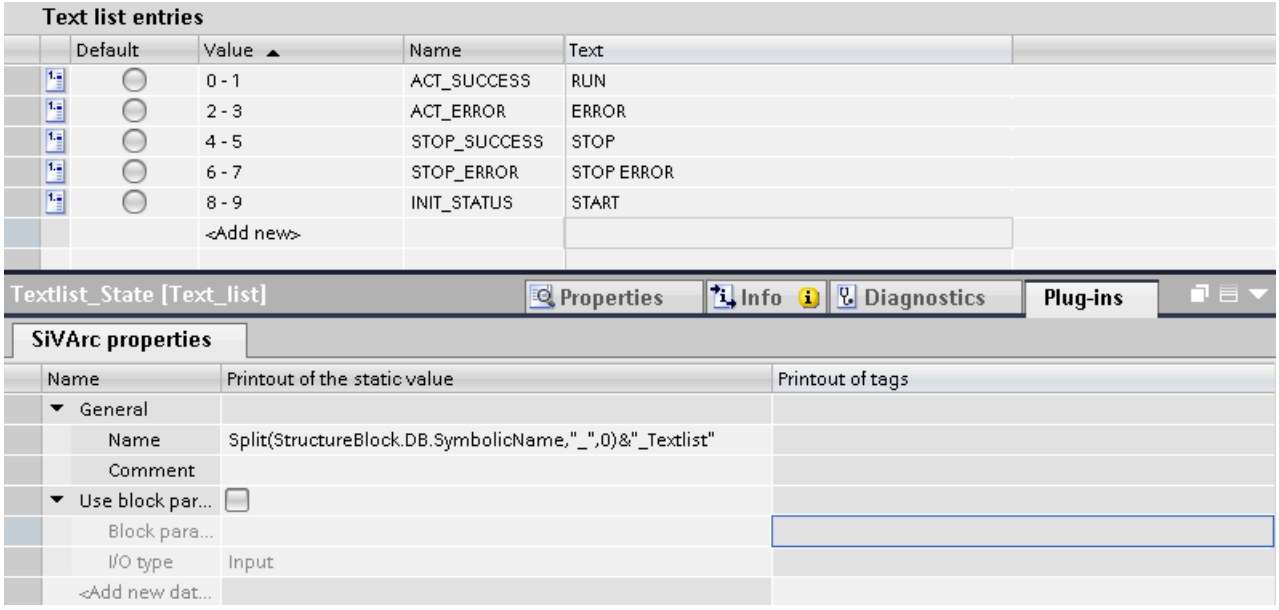
3. Enter the name for the text list entries under "Name > Network".
4. Enter a static text list entry under "Text in current editing language".
   If no dynamic text is specified, SiVArc generates the static text.
5. Under "Expression of the SiVArc tag", enter a SiVArc expression to assign a text list entry dynamically. During the generation in the example,
   SiVArc uses the network title (`Block.NetworkTitle`) and the network comment (`Block.NetworkComment`) of the function block linked in the text list rule.

Procedure

To create a generation template for a text lists, follow these steps:

1. Open the "Textlist_State" generation template from the library.
2. Select the "Area" text list type.
3. Open the text list entries for the text list.
4. Copy the name of the text definitions from the user program to the sequential values in the "Name" column.
5. Enter default text list entries.

6. Under "Plug-ins > SiVArc properties" in the Inspector window, configure the name of the text list with the SiVArc expression "Split(StructureBlock.DB.SymbolicName,"_", 0)&"_Textlist".



7. Overwrite the existing generation template in the library.

Result

During generation, the text list for the first instance of the "Plantsection" function block created. The "Plantsection1_Textlist" text list name is generated using the "Split" function and the name of the data block.

To also generate the text list for all other uses of the block, enter the text definitions at all points of use of the block in the user program.

If the entries cannot be evaluated, a text list is created based on the SiVArc master copy.

If several identical names are detected for SiVArc texts during the generation, SiVArc uses the most recently created SiVArc text.

5.2.21 Example: Create generation template for a text list for block parameters

Example scenario

The plant status of a conveyor belt is to be continuously output in the operator screen.

Objective

SiVArc is to generate an I/O field which is interconnected with a text list that takes its entries from the "State_A" block output of the "Conveyor" function block.

Requirement

- The "Textlist_Parameter" generation template has been stored in the library.
- A text list rule is created which interconnects the "Textlist_Parameter" generation template with the "Conveyor" function block.
- The conveyor belt status is contained in the comments of the tags at the "State_A" block output:
  - OFF
  - ERROR
  - STOP
  - RUN

Procedure

To create a generation template for a text list for a block parameter, follow these steps:

1. Open the "Textlist_Parameter" generation template from the library.
2. Activate "Use block parameters and relevant PLC tags" in the SiVArc properties.
3. Enter the parameter name "State_A" and I/O type "Output".
   To select several parameters, use a regular expression with asterisks. The system then evaluates all parameters with names that include the string as specified.
4. Select the "BOOL" data type and "4" as the number of tags that are to be used for the text list generation.
   If, for example, you select the number "17", the first 17 tags are processed. If there are only 15, only the first 15 are processed.
5. Overwrite the existing generation template in the library.

Result

The tags of the configured data type are recorded and evaluated by the generation. A text list entry is created in each case for four of these tags as shown below:

- The text list entries correspond to the respective comment of the tag.
- The names of the text list entries are composed of the parameter name, the data type of the parameter and a sequential number, for example, State_A_Bit _1, State_A_Bit _2, etc.

If the tag name is not contained in the symbol table, the configured number of text list entries is created with value assignment and name. The names of the text entries are then derived from the parameter. In this case, you can supplement the desired text list entries in the comments of tags and generate a second time. If you enter the text entries manually, the texts are only retained until the next generation.

5.2.22 Example: Generating pop-up screens and their use

Example scenario

On an operator screen there is just enough space for all display and operating elements required for the process control. Therefore, the dialog for language switching is swapped out to a pop-up screen.

Implementation concept

The call of a pop-up screen is configured on a button which contains the settings for the language switch.

Using pop-up screens in SiVArc

You use pop-up screens with SiVArc as you would other WinCC screens. To apply a separate positioning scheme to a pop-up screen, use positioning schemes that are created based on a pop-up screen.

To generate display and operating objects in a pop-up screen, use the pop-up screen as "master copy of a screen" in a screen rule.

Requirement

- The following generation templates have been stored in the library:
  - Pop-up screen "PopUp_ChangeLang"
  - Button "Button_PopUp_ChangeLang"
  - Start screen "StartScreen"
- A pop-up screen has been created in the library as the "PopUp_Pos_ChangeLang" positioning scheme.
- Screen rules are created for the following HMI objects:
  - Button "Button_PopUp_ChangeLang"
  - Pop-up screen "Button_PopUp_ChangeLang"

Create generation template for the pop-up screen

To create a generation template for calling a pop-up screen, follow these steps:

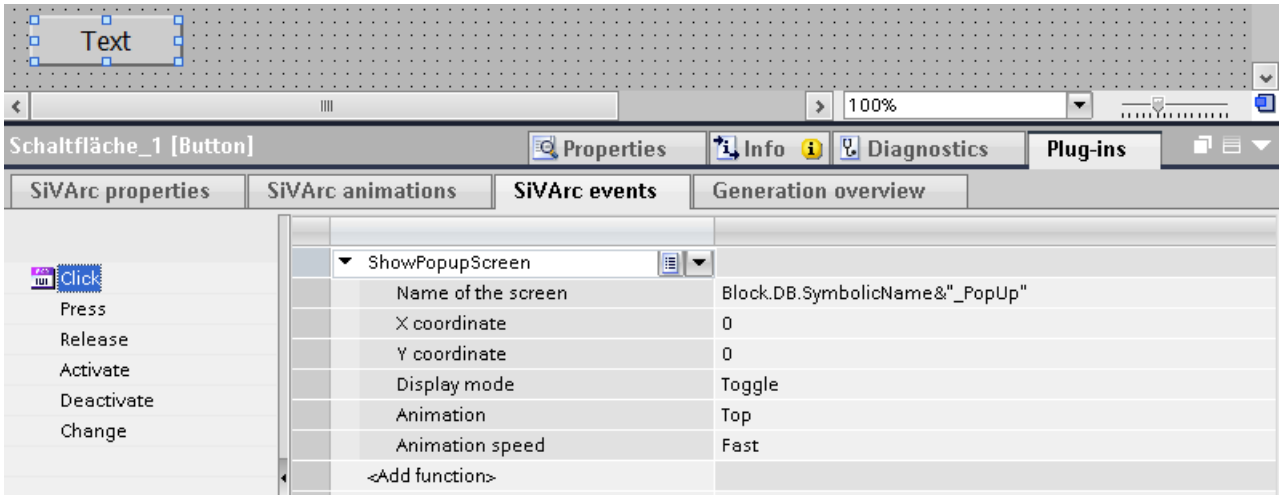
1. Open the generation template of the pop-up screen "PopUp_ChangeLang" from the library.
2. Under "Plug-ins > SiVArc properties > General" in the Inspector window, configure the following SiVArc properties:
   - To generate a unique screen name, enter a SiVArc expression or a string under "Name". Integrate the name of the called program block with "`Block.DB.SymbolicName&"_PopUp`", for example, as the name of the pop-up screen.
   - If the generated screen should be stored in a group or in the plant structure, enter a SiVArc expression under "Screen group".
   - Select "Fixed" as mode of the positioning scheme and the "PopUp_Pos_ChangeLang" positioning scheme.
3. Overwrite the existing generation template in the library.

Create generation template for the calling button

To create a generation template for calling a pop-up screen, follow these steps:

1. Open the generation template of the calling button "Button_PopUp_ChangeLang" from the library.
2. Configure the required SiVArc properties in the Inspector window under "Plug-ins > SiVArc properties". Integrate the name of the called program block with "`Block.SymbolicName&"_ButtonPopUp`", for example, as the name of the button.

3. Under "Plug-ins >SiVArc events" in the Inspector window, configure the system function "ShowPopupScreen" for the "Click" event, for example.
   - For the "Name of the screen" parameter, assign the SiVArc expression you have configured in the generation template of the pop-up screen under "Plug-ins > SiVArc Properties > General > Name": "`Block.DB.SymbolicName&"_PopUp`"
   - Configure the coordination for the display position of the pop-up screen with an integer value.
   - Select the display values.



4. Overwrite the existing generation template in the library.

Result

The start screen of the plant, the button for the language switch, the pop-up screen and the central function module are linked in the screen rules. After generating, an additional button is generated in the start screen of the plant section which calls a pop-up window for the language switch.

5.2.23 Example: Generating faceplates with animations

Example scenario

An assembly line of a manufacturing plant is designed for heavy-duty loads and is used only for special packaging formats. The speed control of the two axes should therefore be displayed on the plant screen only when the production line is in operation.

Implementation concept

The visualization of the speed control is prepared in a faceplate. The faceplate is used in the project as a generation template for controlling all speed-controlled axes. The visualization engineer creates a new generation template with a visibility animation based on the faceplate.

In the screen rules, it uses conditions to control when a faceplate with animation is generated.

Animated faceplates in SiVArc

SiVArc supports the following animations for faceplates:

- Visibility
- Allow operator control
- Appearance

To generate animations for faceplates with SiVArc, configure dynamic properties for the animation in the faceplate type that serves as generation template.

Requirement

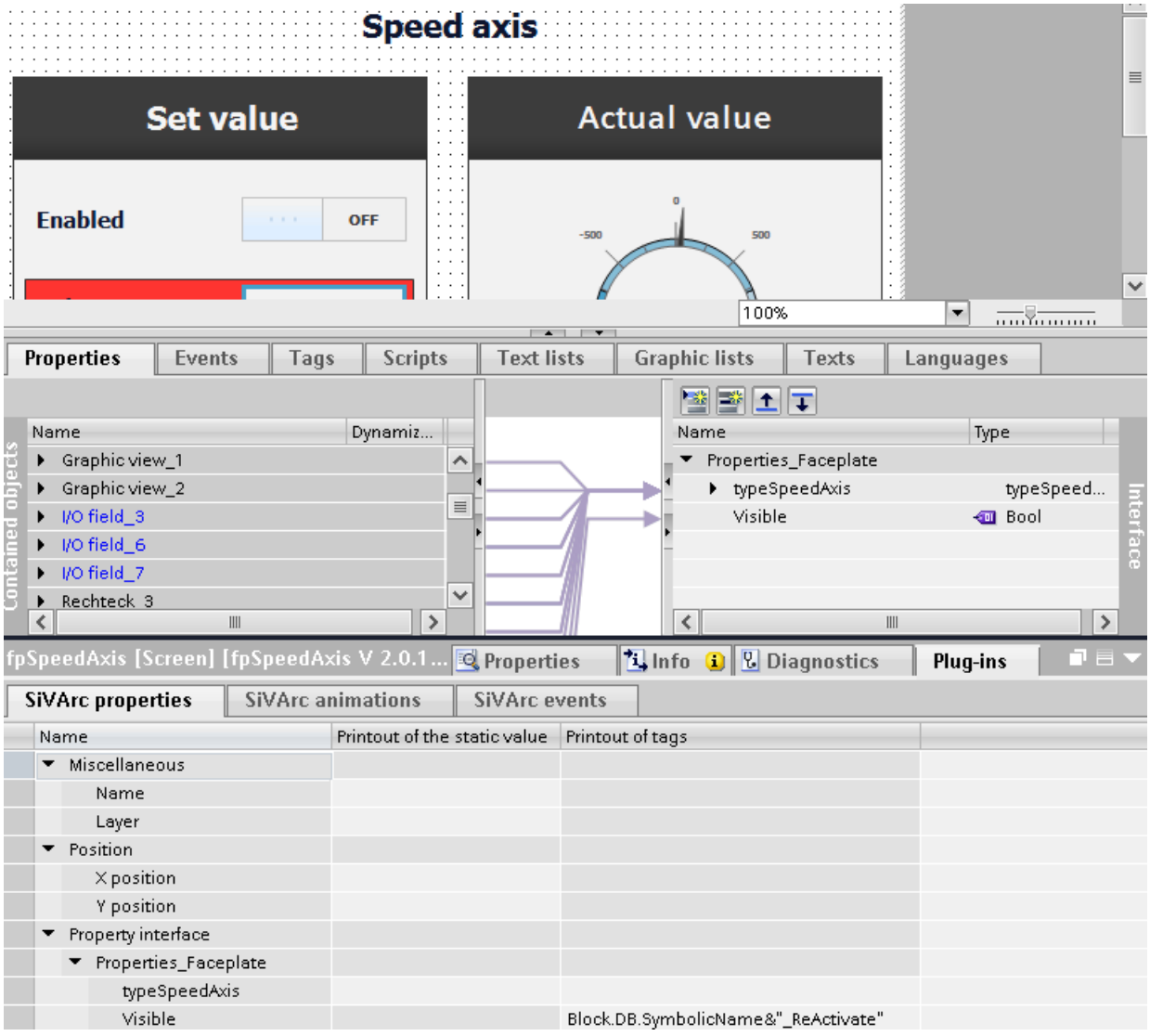
- The "fpSpeedAxis" generation template of the faceplate type is stored in the library.
- The "Conveyor_HeavyLoad_Instance01_ReActivate" tag is contained in the block interface of the relevant function block.
- A tag with the "_ReActivate" ending is only used for heavy-duty operation.

Procedure

To generate faceplates with animations, follow these steps:

1. Open the generation template of the faceplate type "fpSpeedAxis".
2. In the "Interface" list for the faceplate type, create a property with the name "Visible" of the BOOL data type.
3. Configure the "Visible" animation in the WinCC animations of all objects contained in the faceplate type. Use the "Visible" interface property as a process tag in each case when doing this.

4. In the SiVArc properties of the faceplate type, configure the "Visible" interface property with a SiVArc expression "`Block.DB.SymbolicName&"_ReActivate"`".



5. Create a new faceplate type version as a generation template.
6. Use the faceplate type and the relevant program block in a screen rule.

Result

When you have created a screen rule with this generation template, the SiVArc expression is evaluated during generation. An external tag generated by SiVArc is assigned to the property of each generated instance of the faceplate type.

In the example, the animation is only interconnected on the faceplate for the heavy-duty conveyor belt, because a tag with the "_ReActivate" ending is only present there.

5.2.24 Example: Generating "Position" animation for faceplates

Example scenario

In a printed circuit board factory, the manufactured circuit boards are packaged in boxes in the "Packaging" plant unit and transferred to a trolley. This process is to be displayed animated on the HMI device.

Implementation concept

The packed boxes are stored as faceplate generation templates in the library. To represent the horizontal movement of the finished packaged box on a trolley, the faceplates are configured with the "Position" animation. The position values for the horizontal movement are provided to the faceplate by the controller.

"Position" animation for faceplates in SiVArc

Faceplates support the "Position" animation for RT Professional.

To generate animations for faceplates with SiVArc, configure dynamic properties for the animation in the faceplate type that serves as generation template.

Requirement

- The generation template of the "Plate_Box_Ready" faceplate type is stored in the library.
- The "Packaging" function block contains the "XPosition" input parameter of the INT data type.
- The values of the "XPosition" parameter are stored in the associated data block.
- The target HMI devices for generating the visualization of the packaging plant have the same screen resolution.

Procedure

To generate a "Position" animation for a faceplate, follow these steps:

1. Open the faceplate type "Plate_Box_Ready" from the library.
2. In the "Interface" list for the faceplate type, create a "IFace_XPosition" property of the INT data type for a horizontal animation.
3. Configure a new tag connection in the WinCC animations of all objects contained in the faceplate type. Connect the tag to the "X position" property.
4. Configure the tag connected to the "X position" property with the interface property "IFace_XPosition".
5. In the SiVArc properties of the faceplate type, configure the "IFace_XPosition" interface property with the SiVArc expression "Block.DB.SymbolicName&"_XPosition".
6. Create a new faceplate type version.
7. Use the faceplate type and the relevant program block in a screen rule.

Result

After the generation, all generated instances of the "Plate_Box_Ready" faceplate type are configured with an animation. In runtime, the position of the faceplate follows the position value of the interconnected tag, for example, "Block_1_DB_XPosition".

5.2.25 Example: Creating generation templates for trend views

Example scenario

In a manufacturing plant using hydraulic simulation, when temperature parameter is measured against time parameter, the values are displayed as trends using HMI device.

Implementation concept

You use trend views to graphically represent tag values based on certain parameters, and are displayed as trends. SiVArc automates the generation of tag values, and displays the tag values as trends in HMI device. The hydraulic simulation measuring the increase or decrease in temperature for certain time period is displayed as trends in the HMI device. The trend data displayed in a HMI device is used to monitor the status of the hydraulic components.

You can configure the trend view properties in TIA and in SiVArc as well. Refer to the TIA Portal online help for detailed description of configuration of the trend view properties.

In SiVArc, you configure the trend view properties under "SiVArc Properties > Plug-ins".

Depending on your device, different properties of the trend view can be configured. For more information on the trend view properties, refer to the TIA Portal online help.

Requirements

- A screen containing a trend view named "TrendView1"is created for an HMI device.

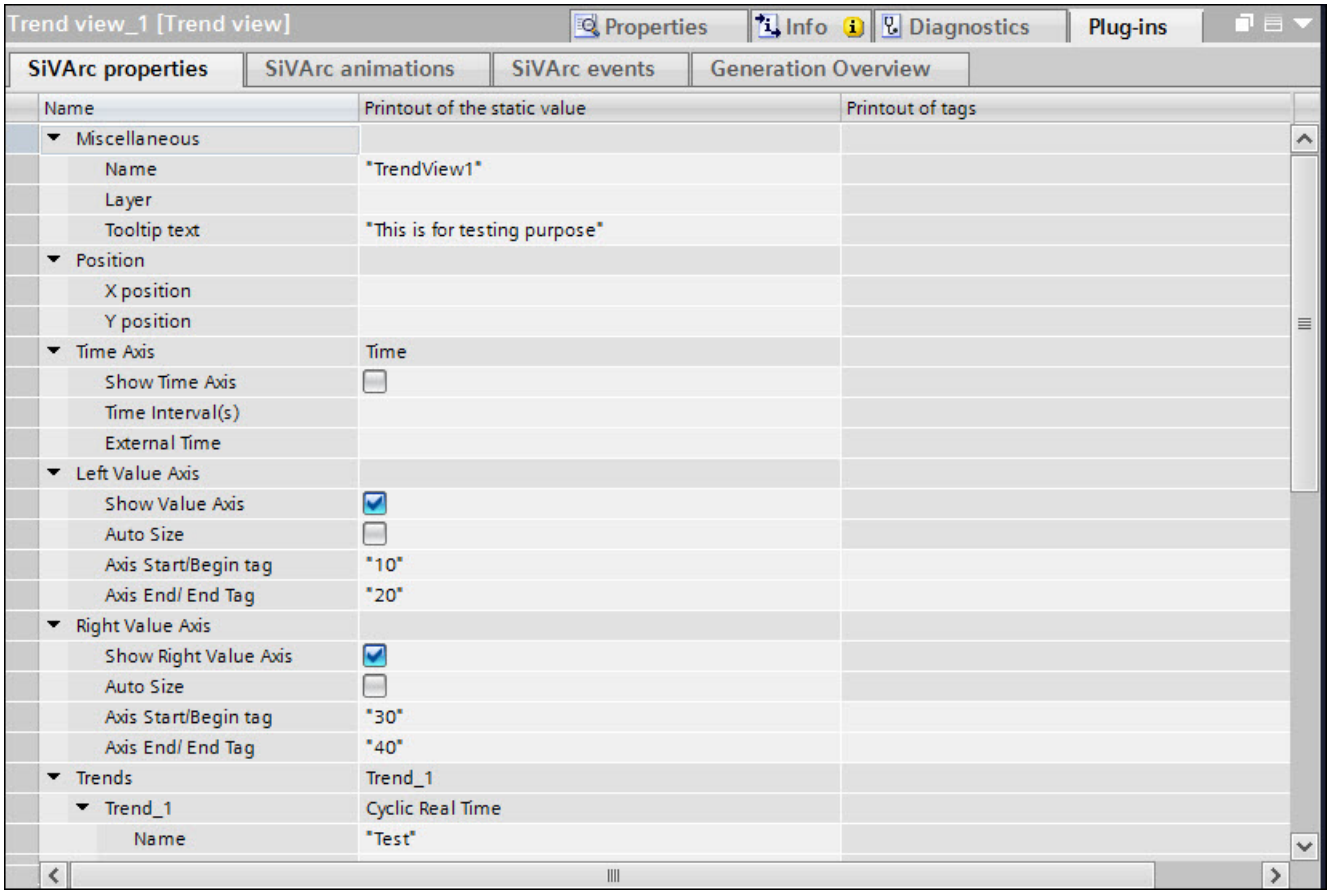
Example: Creating generation templates for trend views

The following example shows how to configure a trend view in SiVArc for RT Advanced.

1. Under “SiVArc Properties > Plug-ins", configure the properties of the TrendView1 like "Layer", and "Position".

   Note

   In RT Advanced, you can also configure the property "Tooltip text".



2. To configure time axis of the trend view, expand the "Time axis" properties under "Plug-ins"' and activate the "Display time axis" checkbox to display the time axis in the trend view control.

3. To configure the value axis, expand "Left value axis" and "Right value axis" properties.
   - Activate the checkbox "Show value axis" to be able to display the axis in the trend view.
   - To set the axis values to automatic values during object generation, activate the checkbox "Auto size".
   - Enter the tag value for "Axis start/begin tag" and "Axis end/end tag".

   Note

   In RT Advanced the following "Axis mode" are supported:
   - Points: Displays number of values
   - Tag/constant: Displays number of values
   - Time: Displays external time.

4. To configure trend values, expand "Trend" properties:
   - Under "Name", enter a valid name for the corresponding trend.
   - Under "Data source process values", enter a valid value for process values.
5. Configure screen rules for a specific screen which contains the trend. For more information on configuration of screen rule, refer section "Defining a screen rule for generating screen object".
6. Generate the visualization. For more information on generation, refer section "Generating visualization".

Result

After the generation, the tag values of temperature against time is displayed graphically as trends in the HMI.

5.2.26 Creating a generation template for a screen

Requirement

- A WinCC project is open.

Procedure

To create a generation template copy for a screen, follow these steps:

1. Create a new screen.

   Note

   Assign a meaningful name. A unique name facilitates later work because the screen name is used as the name for the generation template.

2. Configure the properties of the screen and add the required screen objects as necessary.

3. Configure the desired properties in the Inspector window under "Plug-ins > SiVArc properties > General":
   - To generate a unique screen name, enter a SiVArc expression or a string under "Name".
   - If the generated screen should be stored in a group or in the plant structure, enter a SiVArc expression under "Screen group".
   - Configure overflow screens, if required.
4. To create a master copy, store the screen in a library under "Master copies".
5. To create a screen type, store the screen in a library under "Types" and assign the type name.

Note

SiVArc properties of a screen type

Fewer SiVArc properties are available in the screen type than in the master copy of a screen.

Result

The generation template has been created for a screen.

See also

Creating a screen rule (Page 155)
Generating visualization (Page 181)
SiVArc object properties (Page 211)
Storage strategies for generated objects (Page 103)

5.3 Creating rules

5.3.1 SiVArc rules

Definition

SiVArc rules define how HMI objects are processed during generation.

The various SiVArc rules define different generation tasks:

- Screen and text list rules link generation templates and control instructions.
- Tag rules control the storage structure of the HMI tags generated by SiVArc.
- Copy rules trigger the generation of the following HMI objects based on the master copies or types:
  - Screens
  - C and VB scripts
  - Text lists
  - Tag tables

SiVArc rules are a key functionality of SiVArc and have a direct relationship to the user program. You can therefore assign SiVArc rules with know-how protection just like instructions.

Differences to the configuration without SiVArc rules

Unlike conventional WinCC configuration, the relationship between a SiVArc rule and a generated HMI object is maintained in a SiVArc project.

When you change a SiVArc rule, generated objects based on this rule are overwritten during the next generation. When you delete a rule, generated objects associated with this rule are automatically removed during the next generation.

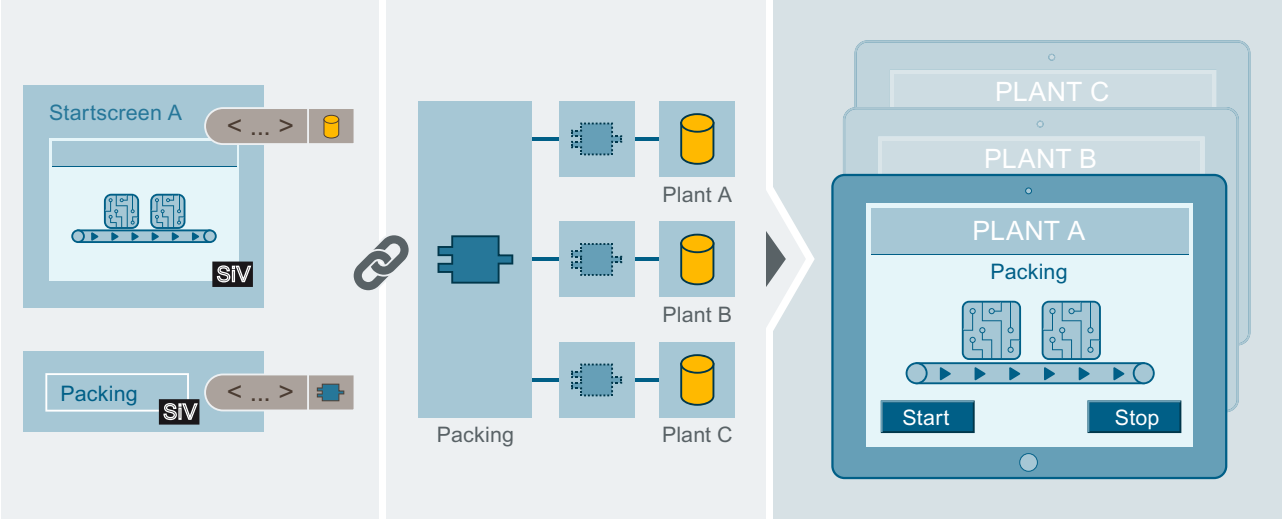
You can also create the visualization separately for individual devices using the SiVArc rules.

Purpose and benefits of SiVArc rules

You can use the SiVArc rules to centrally control the HMI objects with a direct relationship to the control program and individually for each HMI device. Changes can therefore be implemented centrally and throughout the project. Design and development of SiVArc rules provides a high degree of added value in terms of controllability and efficiency of a WinCC project.

Example: Screen rules

The following example shows in abstract form how to integrate texts from data blocks in an HMI screen using generation templates:



SiV — SiVArc generation template

< ... > — SiVArc property with referenced text source

Process instruction

Instance of an instruction in Main OB

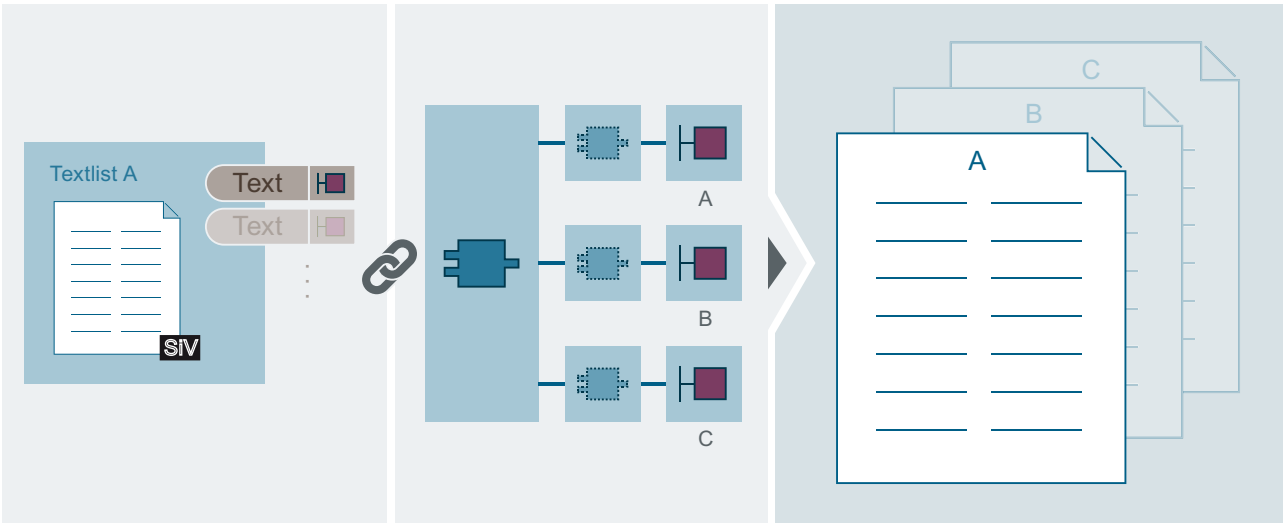
Data block

Result

A display and operating object of the referenced SiVArc master copy is created for each instance of the referenced block. The properties of the display and operating object are created according to the SiVArc rules and the SiVArc properties. SiVArc stores the generated screens according to your configuration.

Example: Text list rules

The following example shows in abstract form how to generate text lists with texts from a network:



SiV	SiVArc generation template
< ... >	SiVArc property with referenced text source
	Instruction
	Instance of an instruction in Main OB
	Network

Result

A text list of the referenced SiVArc master copy is created for each instance of the referenced block. The properties of the text list are created according to the SiVArc rules and the SiVArc properties.

SiVArc then generates the values for the text list entries configured in the user program for each called program block.

See also

Setting up know-how protection for a SiVArc project (Page 171)

Changing SiVArc rules (Page 179)

Creating SiVArc rules (Page 136)

Creating a screen rule (Page 155)

Creating text list rules (Page 157)

5.3.2 Creating SiVArc rules

Definition

You create SiVArc rules and edit them in the SiVArc editors.

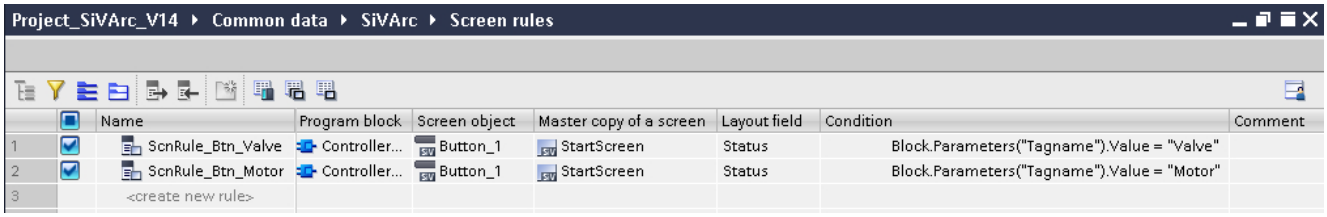
You use the SiVArc rule editors to centrally control the SiVArc generating functions. You can collectively edit and organize all rules in this way. You can collectively export or import rules, for example.

The SiVArc editors are table editors. Each type of SiVArc rule has its own editor. The editors have different objectives.

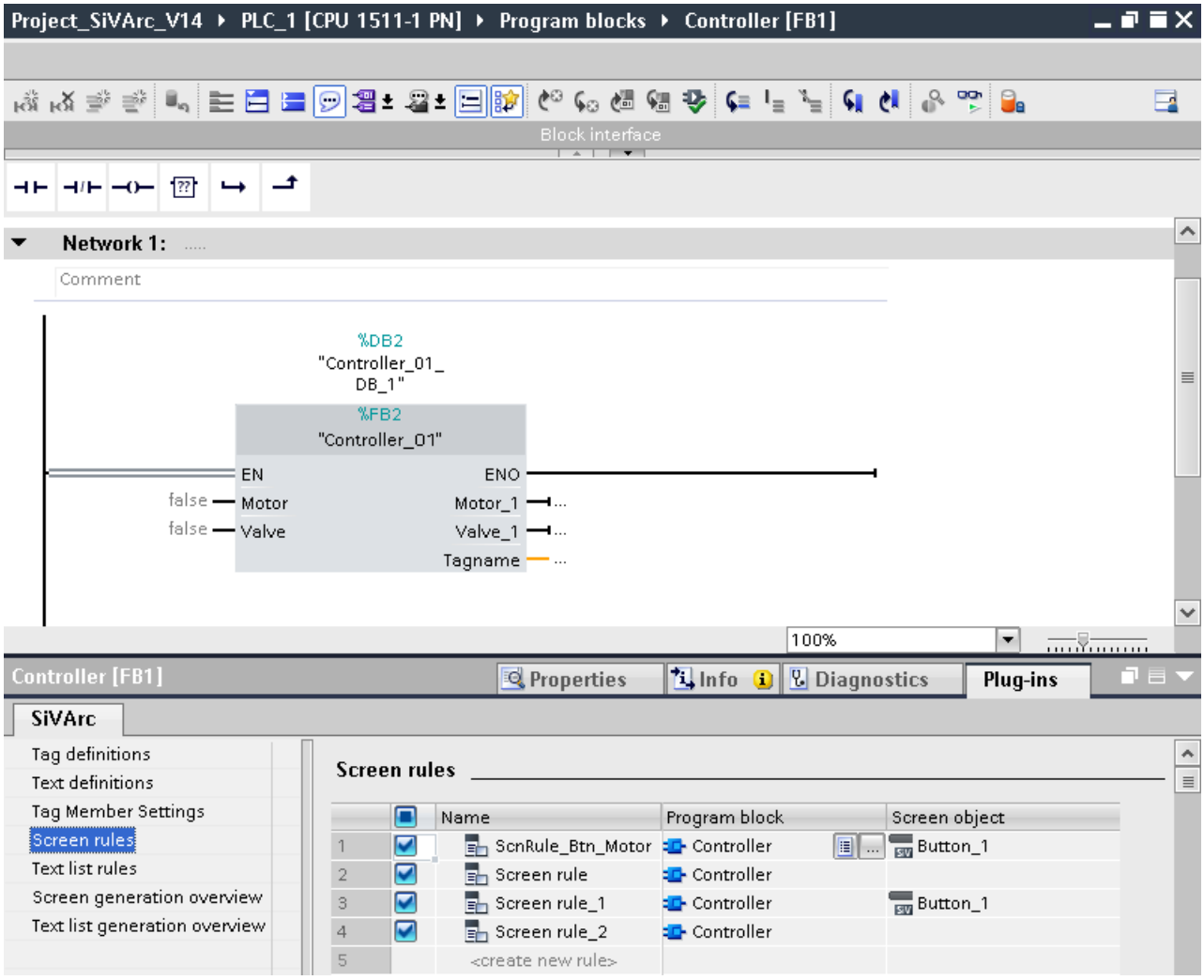
Access to the SiVArc rule editors

To open a SiVArc editor, double click the relevant entry in "Common data > SiVArc" in the project tree.

You hide or show individual columns using the icons in the toolbar, for example, the columns "PLC", "HMI device" or "HMI device type".



Screen and text list rules can also be accessed in STEP 7:



All screen and text list rules which are created for the selected program block are directly accessible at the program block. The scope of the displayed rules depends on the controller.

Except for Import/Export, you create and edit the SiVArc rules in STEP 7 like in the actual SiVArc editor. There is no toolbar in the Inspector window.

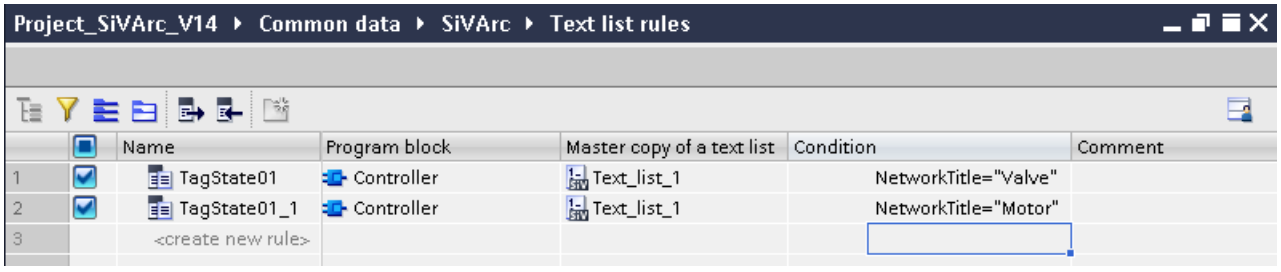
You only remove the know-how protection of SiVArc rules in STEP 7 with the commands in the shortcut menu in the project tree under "Common data > SiVArc".

The "Screen generation overview" and the "Text list generation overview" displays are additionally available in the Inspector window under "Plug-ins > SiVArc" after the first generation.

Interconnection of HMI objects with program blocks

In the "Screen rules" editor, you define the screen rules according to which SiVArc generates the HMI objects in the screens for various devices.

In the "Text list rules" editor, you define SiVArc rules according to which text lists are generated for various devices.



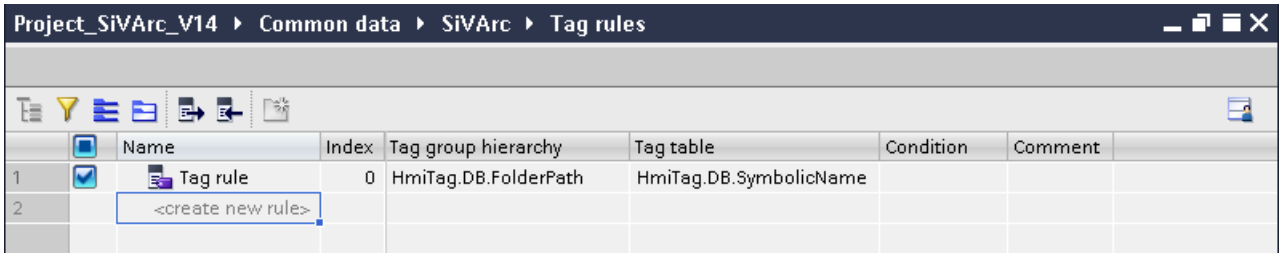
In doing so, you create the rules and specify the following for each rule:

- Linked components
  - Screen rule: Program block, screen object, screen, layout field
  - Text list rule: Program block, text list
- Conditions for executing the rule
- Comment on the rule

In the "Screen rules" editor you can show the columns "PLC", "HMI device" or "HMI device type". In the columns you can enable or disable the devices for which SiVArc applies these rules during generation. This way you generate HMI objects for selected devices.

Controlling the storage structures for tags

In the "Tag rules" editor, you define tag rules according to which the external tags generated by SiVArc are stored in structured form.

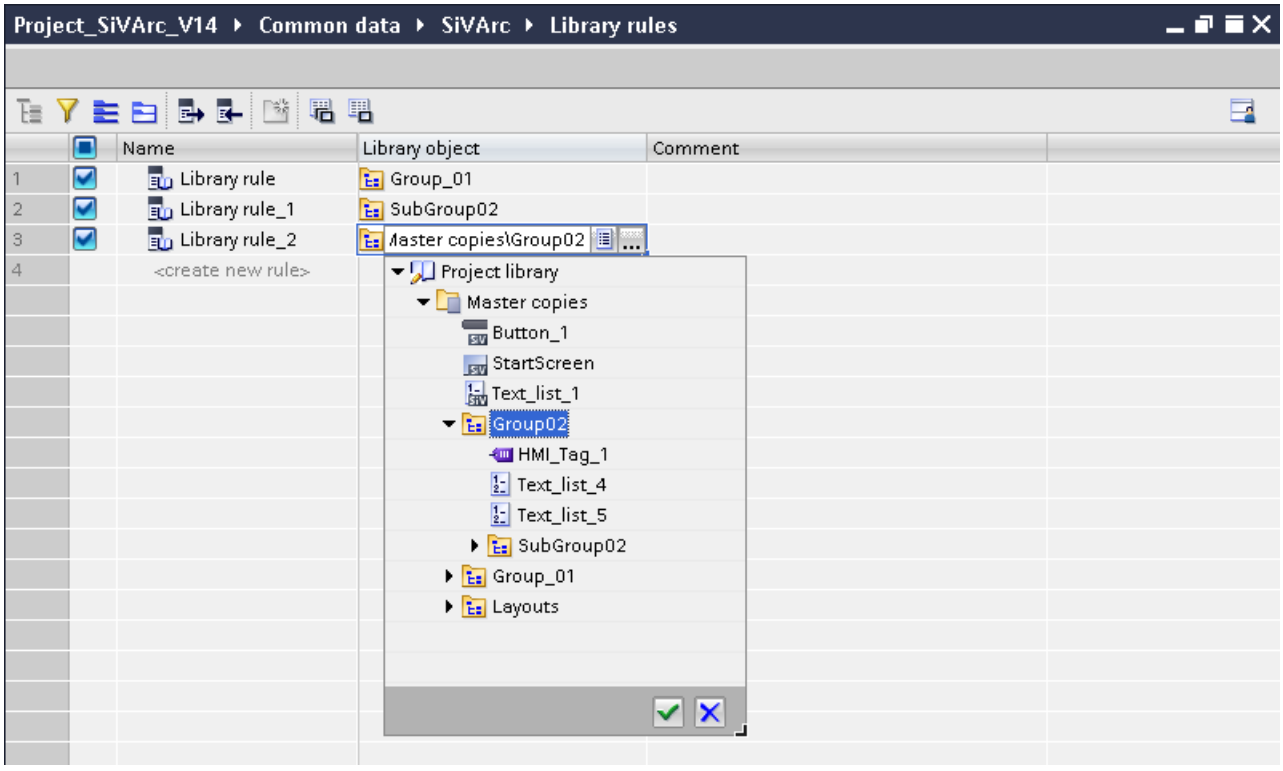


In doing so, you create the rules and define the following for the generated tags:

- Name of the tag group
- Name of the tag table
- Order in which the rule is executed
- Conditions for executing the rule
- Comment on the rule

Systematic insertion of HMI objects into a project

In the "Library rules" editor, you define the rules according to which selected objects from the library are generated for various HMI devices.



In doing so, you create the rules and specify the following:

- Library object to be created
  or
  HMI objects of a group of library objects to be created
- Comment on the rule

You can also show the columns "HMI device" and "HMI device type". In the columns you can enable or disable the devices for which SiVArc applies these rules during generation. This way you generate HMI objects for selected devices.

See also

Exporting and importing SiVArc rules (Page 168)

Setting up know-how protection for a SiVArc project (Page 171)

Editing and managing SiVArc rules (Page 166)

Creating a screen rule (Page 155)

Creating text list rules (Page 157)

Editing the view in the SiVArc editors (Page 229)

5.3.3 Using SiVArc scripting in SiVArc rules

Definition

You use SiVArc scripting in SiVArc rules in the conditions for executing a rule. In tag rules, you also use SiVArc expressions to structure external tags.

You can basically always use a SiVArc expression when the "SiVArc expressions" editor is stored in an input field.

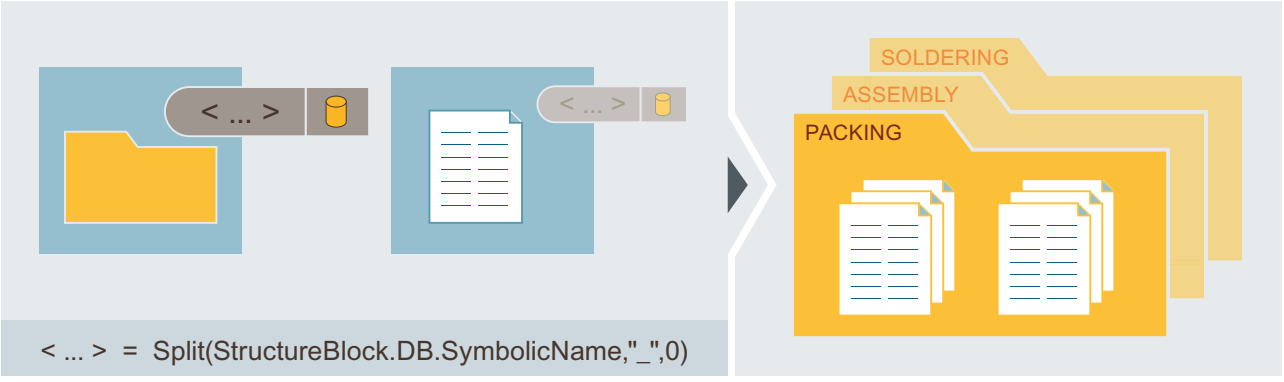
Conditions in SiVArc rules

For conditions use a SiVArc scripting function that returns the Boolean value TRUE or FALSE.

If no condition is specified, the screen rule is always executed. You can assign a condition to an entire rule group. The condition then applies to all rules contained in the group. You can refine the condition for individual rules of the rule group.

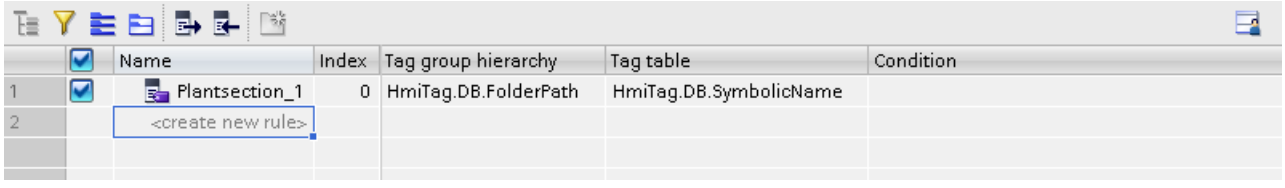
Storage structure of tags

You specify the names of the tag group and tag table in the project tree with SiVArc expressions. This way you link the storage structure with the user program. The tags are, for example, sorted according to the block instance this way.



You can use the SiVArc expressions `HmiTag.DB.SymbolicName` and `HmiTag.DB.FolderPath` for the "Tag rules" editor to structure the tag tables based on the control program using only one tag rule. Only the controller programmer structures the project. For visualization, SiVArc applies the storage structure from STEP 7.

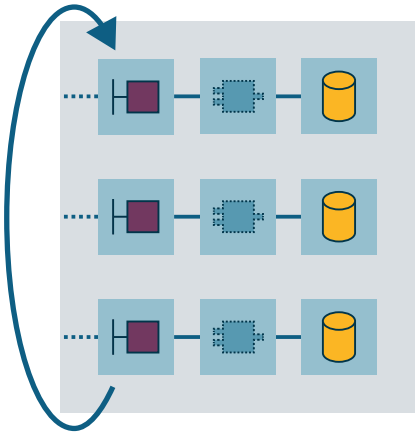
Only in the "Tag rules" editor do you use SiVArc expressions that address the SiVArc object `HMITag`.



5.3.4 Processing of rules

Operating principle

SiVArc executes the user program during generation. If a rule applies to a function block, the rule is executed. At the same time, SiVArc runs through the data blocks.



As soon as an external tag must be generated, SiVArc runs through the tag rules from top to bottom and store the tag according to the specifications. Only one rule is therefore applied for each tag.

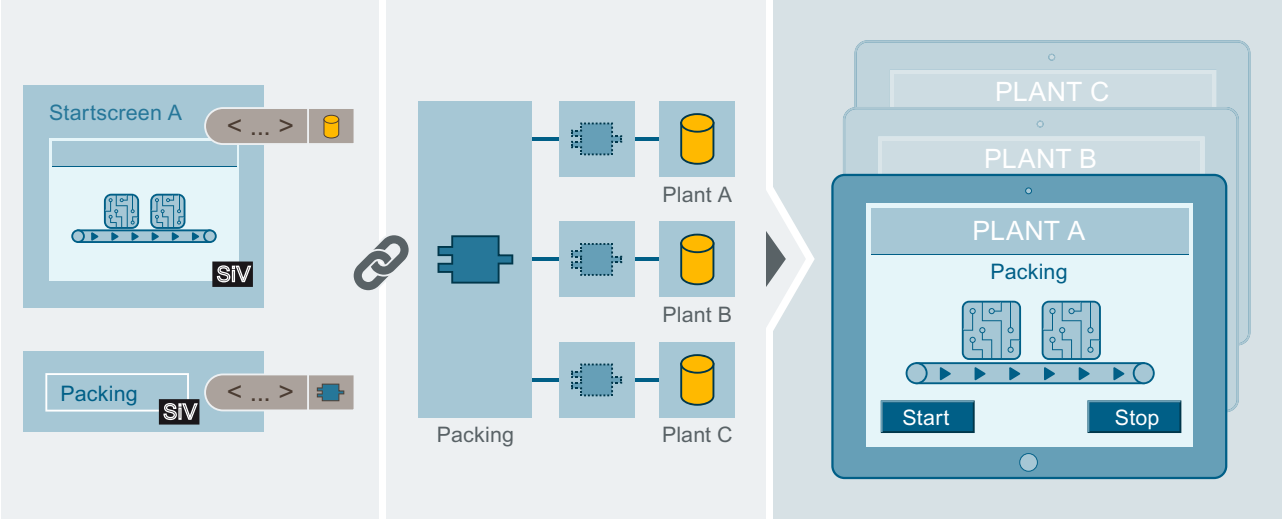
All screen and text list rules of a project that are enabled and meet a configured condition are executed.

Evaluation of the screen rules

The following principles apply to screen rules:

- You must define a screen rule for each screen object to be generated.
- If you want to generate different screen objects from a program block, you must define a screen rule with a condition for each screen object. You specify the screen object to be generated in the condition.
- If the screen for a screen object to be generated does not exist yet, the screen is created during generation.
- If a block is contained in multiple screen rules in the "Screen Rules" editor, the objects are created in the order of the screen rules.
- If you want to generate a screen without a screen object for a program block, leave the "Screen object" field blank.

SiVArc executes the user program according to the call hierarchy of all OBs of the selected PLCs. The "Main" block is executed for each PLC. SiVArc evaluates the screen rules for each called program block.



For each applicable screen rule, the corresponding display and operating object is generated in the specified screen on the basis of the generation template. The SiVArc expressions in the SiVArc properties, events and animations of the generation templates are evaluated during generation.

Evaluation of the tag rules

The order of tag rules is relevant for the storage of external HMI tags. If necessary, change the order using drag-and-drop.

Arrange plant-specific rules, for example, as the first item and rules that store tags for plant-wide functions in structured form as the last item. This ensures that all plant-specific tags are stored together.

SiVArc executes all data blocks of all PLCs that were enabled in the station selection dialog. If the "Accessible from HMI" option is selected in the data block, SiVArc generates one external tag each for the tags of the data block.

For each external tag to be generated, SiVArc runs through the tag rules from top to bottom and evaluates the associated condition. As soon as a condition is true, the rule is applied and the external tag of the rule is stored correspondingly in the project tree. The subsequent tag rules are no longer processed. Instead, SiVArc continues with the next external tag to be generated.

If none of the tag rules apply to an external tag to be generated, this external tag is stored in the default tag table.

Depending on the setting under "Options > Settings > SiVArc", SiVArc generates only external tags which are also interconnected in the generated SiVArc project.

During generation, SiVArc processes the settings for tags in the Runtime settings of the HMI device. The generated name of the external tags represent the symbolic address of the tags in the data block in accordance with the tag synchronization of WinCC.

Evaluation of copy rules

SiVArc processes the copy rules. For each copy rule, the corresponding HMI object per specified HMI device is created in the project tree.

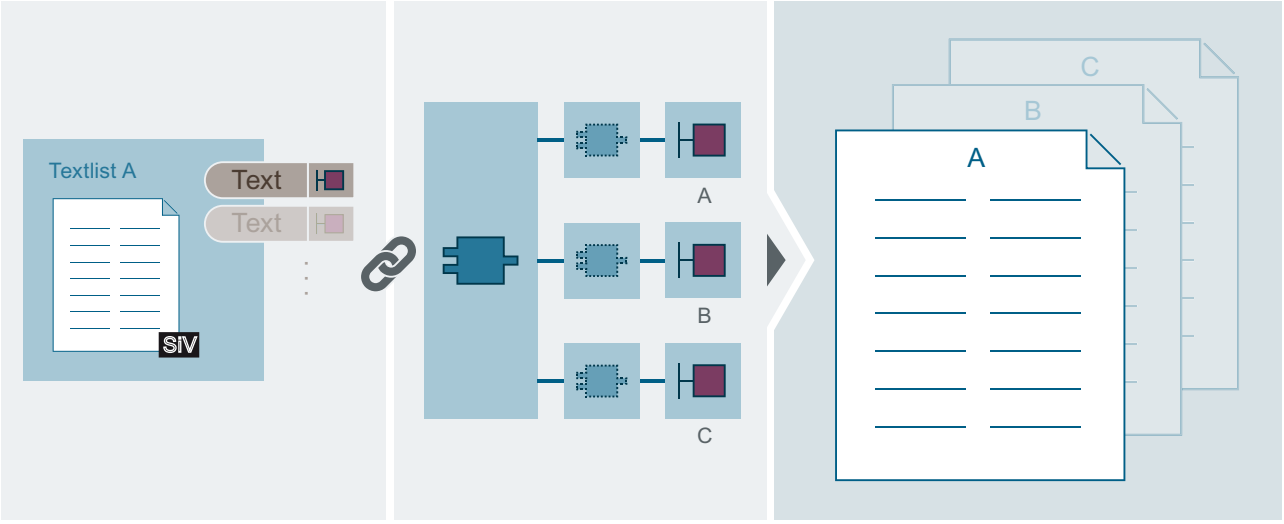
Evaluation of the text list rules

The order of the text list rules is irrelevant, because the use of the text list rules is defined by the call hierarchy of the program blocks in the user program.

SiVArc always processes all text list rules that contain the program block currently being evaluated by SiVArc.

The SiVArc properties of the text list are evaluated thereby. When the text list has been generated, the text list is expanded with the new entries and existing, identical entries are overwritten.

The text list is stored in the HMI device for which the generation was triggered.



SiVArc then generates the values for the text list entries configured in the user program for each called program block. In the process, SiVArc executes the user program according to the call hierarchy of all OBs of the selected PLCs.

Priority of the generated objects in case of naming conflicts

In case of naming conflicts, SiVArc sets the following priorities during a generation:

1. Generated objects from screen, tag and text list rules
2. Generated objects from copy rules
   Objects generated from the copy rules are treated the same as manually created objects. They are created first during the generation. If there are naming conflicts with objects that are generated later, objects are renamed according to copy rules with the extension "_renamed".
3. Manually created objects
   If the names of manually created objects and generated objects are the same, the manually created objects are renamed.

See also

Creating a screen rule (Page 155)

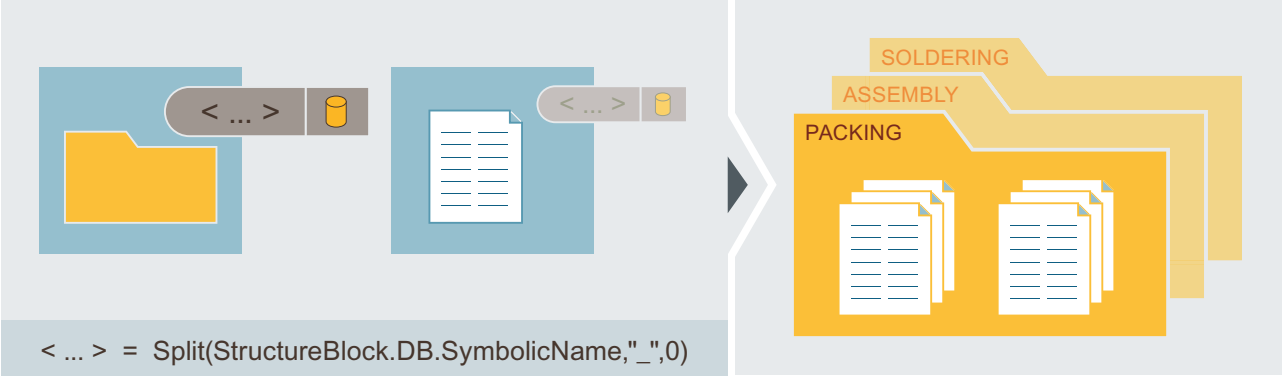
Creating text list rules (Page 157)

Picture legends (Page 231)

5.3.5 Generating tags

Definition

SiVArc generates an external HMI tag for each PLC tag with the option "Accessible from HMI". According to the mode specified project-wide under "Options > Settings > SiVArc", SiVArc generates either all these tags or only those used in the project. Starting as early as generation, external tags are stored in tag tables and groups the project tree according to your structuring specifications.



Internal HMI tags and SiVArc tags

If you want to create internal HMI tags with SiVArc, store the corresponding master copies in the library. You then use this master copy in a library rule.

External tags are always generated before internal HMI tags. In case of naming conflicts, generated internal HMI tags and manually created tags are renamed.

Unlike manually created HMI tags and SiVArc tags, generated external tags have a direct link to SiVArc and are collected again during each generation.

Advantages of the tag generation

The generation mechanism for tags enables the automatic interconnection of PLC tags in the display and operating objects. You constantly remove unnecessary tags from the project with the mode selected for tag generation. The result is an efficient and error-free configuration without using up unnecessary storage space.

Interconnections of external tags

You create automatic interconnections by configuring the process tags as SiVArc expressions in the generation templates. If the SiVArc expression is evaluated as an existing tag name during generation, the generated display object is interconnected to this tag.

Setting up the update cycle and acquisition type

If necessary, you can set the update cycle and the acquisition type of generated external HMI tags in multiple steps:

- For individual program blocks
  You define the update cycle and the acquisition type of tags for a program block with the "Use Common Configuration" option in the Inspector window of a data block under "Plug-Ins > SiVArc > HMI tag settings". This setting deactivates the settings for individual tags.
- For individual tags
  When the "Use Common Configuration" option is disabled, configure each tag individually in the data block.
- Project-wide
  In the SiVArc settings under "Common data > SiVArc > SiVArc settings > Tag generation settings", you configure all external tags of the project that are generated. This setting is only evaluated if no other setting for tag generation is defined.

User data types only support the cyclic acquisition types. If you set the "On demand" acquisition type for the entire project or for one program block, the update cycle is set to 1 s and the acquisition type to "Cyclic in operation" for user data types.

The update cycle 500 ms is automatically set for HMI devices which do not support setting the acquisition type and update cycle.

Note

Copy program block with tag configuration

You make the settings on the update cycle and acquisition type again for each program block. Even if you copy a completely configured program block, configure its settings for tag generation again.

Default settings for tag names

The following default settings are set for generated tag names in the TIA Portal:

- The separator is always "_"
- Square brackets "[" and "]" are replaced with "{" and "}"

If necessary, use the SiVArc object `TagNaming`, which processes these settings, in the SiVArc expressions. Additional information can be found in the section "Example: Adapting tag names (Page 164)".

Note

Separators in structured tags

The hierarchy levels are always separated by "." in structured tags.

Spaces in tag names

SiVArc does not take into consideration spaces when generating tags. Even if, for example, a function block that is supplied with tags via SiVArc has a space in its name.

SiVArc ignores this space. In this case errors can occur during the interconnection.

Example

A DB instance name of a function block contains a space: "TT5684 Temperature", because it is used as message text. If you do not remove the space from the name of the function block, the block is created and the interface property is highlighted in red with the non-existing tag "TT5684Temperature" (without space).

Delete the space in the tag name of the function block. In this way you adapt the SiVArc expression to the tag name as it is created from the tag rule.

See also

Principle of the tag generation (Page 149)

Creating tag rules (Page 154)

5.3.6 Use of copy rules

Definition

A copy rule copies HMI objects based on master copies or types when generating the visualization.

You create copy rules only for the following HMI objects:

- Internal tags
- Text lists
- Screens
- Scripts
- Graphic lists

Unlike screen rules, a copy rule does not support SiVArc expressions or conditions. While the HMI objects created from copy rules are independent of the user program, like other generated

objects they still have a link to SiVArc. The copied objects are therefore collected once again during the next generation and updated, if necessary. If the master copies you use were deleted from the library, the objects previously generated from them are removed from the project tree.

Purpose of copy rules

In a standardized operator control and monitoring solution, HMI objects are often created centrally and distributed as global libraries to the configuration engineers. You can use copy rules to generate these HMI objects automatically for each HMI device in your project.

You can use the library rules to systematically copy a large number of objects that can already be presorted according to specific project criteria. This way you also ensure standardization in your projects for scripts, texts, internal tags and graphics.

When you generate display and operating objects that are interconnected to scripts, you use the copy rules to ensure that these scripts also exist on the HMI device.

Plus you use copy rules to create internal tags and tag tables, for example, initially for only one HMI device. For all other HMI devices you store the tags in the project library so that SiVArc can copy them automatically to all devices during the next generation.

Operating principle

You create the HMI objects you want to copy in the library as part of a group or as individual objects. You use this group in a library rule or individual library elements as needed. You can sort the rules into groups and enable or disable them as a group, if necessary. In the copy rule you also control the HMI devices for which the rule is being executed.

For each copy rule, the corresponding HMI object per specified HMI device is created in the project tree.

Naming conflicts

Objects generated from the copy rules are treated the same as manually created objects in case of naming conflicts. They are created first during the generation. If there are naming conflicts with objects that are generated later, objects from copy rules are renamed with the extension "_renamed".

Generating internal tags

To generate internal tags, follow these steps:

1. Create a tag table.
2. Configure the internal tags in this tag table.
3. Store the tag table as master copy in the project library.
4. Create a copy rule which copies the master copy of the tag table to the specified HMI device.

See also

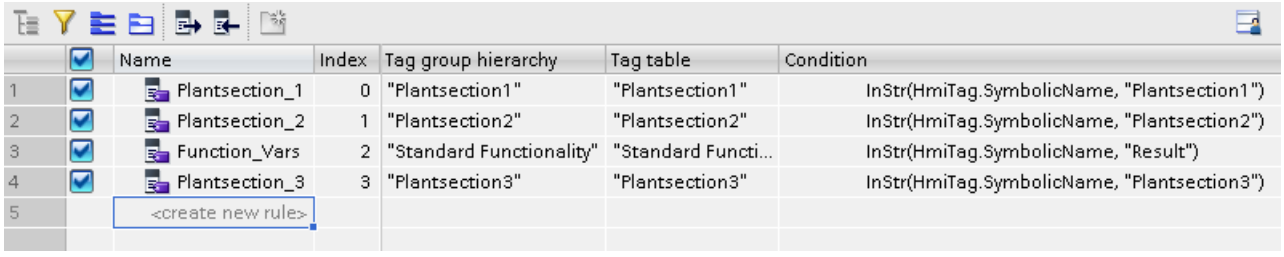
SiVArc rules (Page 132)

Creating copy rules (Page 160)

5.3.7 Correlation between SiVArc expressions and conditions

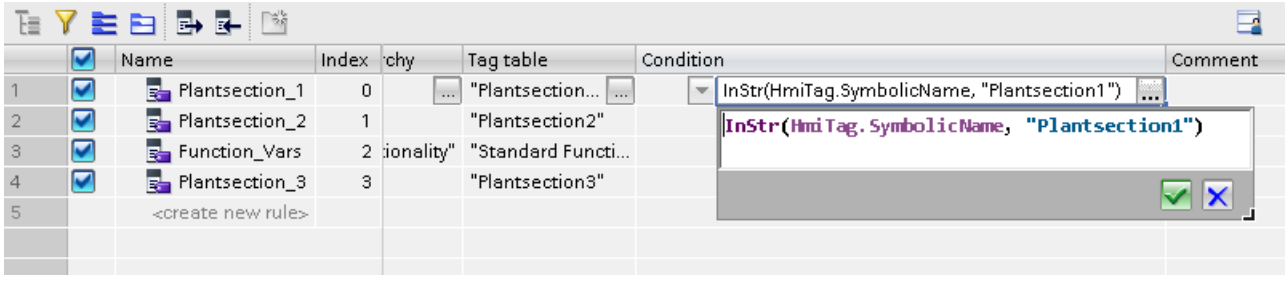
Application example

You use SiVArc expressions and conditions in tag rules, for example. You use the SiVArc expressions to specify group names and names of tag tables. You use a condition to specify whether or not the tag rule is applied to a tag.



	Name	Index	Tag group hierarchy	Tag table	Condition
1	Plantsection_1	0	"Plantsection1"	"Plantsection1"	InStr(HmiTag.SymbolicName, "Plantsection1")
2	Plantsection_2	1	"Plantsection2"	"Plantsection2"	InStr(HmiTag.SymbolicName, "Plantsection2")
3	Function_Vars	2	"Standard Functionality"	"Standard Functi...	InStr(HmiTag.SymbolicName, "Result")
4	Plantsection_3	3	"Plantsection3"	"Plantsection3"	InStr(HmiTag.SymbolicName, "Plantsection3")
5	<create new rule>				

The following condition specifies, for example, that the tag rule is only applied to tags whose names contain the text string "Plantsection1":



Description

SiVArc expressions return a text that accesses the text sources. You can also formulate conditions within SiVArc expressions. The return value is always a text.

Conditions return a Boolean value. You use conditions in SiVArc rules.

You use SiVArc expressions mainly in generation templates. The SiVArc expressions are evaluated during generation. The SiVArc properties of the generated HMI objects are configured with the generated text. This means the generated display and operating objects are named, labeled and saved in folders, for example, using SiVArc expressions and interconnected to the correct tags.

You use conditions for screen, tag and text list rules. The condition is evaluated during generation and the rules are either executed or ignored. This way you limit rules and allow for exceptions.

See also

SiVArc scripting (Page 84)

If conditions (Page 226)

SiVArc expression (Page 87)

5.3.8 Principle of the tag generation

Introduction

Tag generation with SiVArc makes use of the tag synchronization in WinCC and processes the associated Runtime settings during generation.

In addition, SiVArc has functions which control the scope of the tag generation and storage of the generated tags. Prior to generation with SiVArc, you define the update cycle and the acquisition type of tags.

Task	Implementation	TIA Portal
Scope of the generation	Identification of the tags to be generated in STEP 7	"Accessible from HMI" in the data block
	Mode for tag generation with SiVArc	"Options > Settings > SiVArc"
Tag names	Runtime settings for tags in WinCC	Runtime settings of an HMI device
Storage structure	Tag rules of SiVArc	"Tag rules" editor
Update cycle and acquisition type	At the data block in STEP 7	"Plug-ins > SiVArc > HMI tags
	Project-wide	"Common data > SiVArc > SiVArc settings > Tag generation settings"

Runtime settings for tags

During generation, SiVArc takes into account the settings for tags in the runtime settings of the HMI device. SiVArc names the generated external tags according to the naming conventions set there.

If you change the settings for tags after the first SiVArc generation, SiVArc generates all tags in accordance with the new settings. Existing SiVArc tags are renamed.

Configure the settings for tags once before the first SiVArc generation. If different settings are required for HMI device tags of the project, you can access the Runtime settings with the SiVArc object `TagNaming` using the SiVArc expressions.

Specifying the scope of the tag generation with SiVArc

You select the mode for tag generation under "Options > Settings > SiVArc" the required option for your project.

If you have already started the generation, check the setting for these options in the dialog for generating the visualization.

If you only select this setting after the first generation, are the existing external tags processed according to the rules for generated objects:

- Unused external tags in the SiVArc project are deleted. This frees up memory space.
- Manually edited tags are retained and, if necessary, renamed.

Controlling storage structures with SiVArc

If you are using SiVArc expressions with the SiVArc object `HMITag` in the tag rules, changes in the user program are implemented in the storage structure during the next generation. If functional areas of the plant are newly distributed, for example, the interconnected tags are also sorted according to the new distribution in the project tree.

SiVArc only stores the tag after the first tag rule that applies to the tag. If necessary, change the order using drag-and-drop. If another applicable tag rule is positioned further at the top afterward, the tag is stored according to this rule during the next generation.

Specifying the update cycle and acquisition type with SiVArc

You can set the update cycle and the acquisition type for the generation in three stages:

- Project-wide
- For individual program blocks
- For individual tags

These settings are newly assigned during each generation. Despite central control you can still configure flexibly and thus optimize the performance of the project in runtime.

Purpose and benefits

Tag generation enables an efficient and error-free configuration of tags. The generation enables uniform naming of tags and automatic interconnection of the generated display and operating elements.

Tag generation with SiVArc makes for easier handling of changes and optimizes an existing project.

See also

Example: Adapting tag names (Page 164)

Generating tags (Page 144)

Creating tag rules (Page 154)

Example: Using SiVArc to generate tags (Page 18)

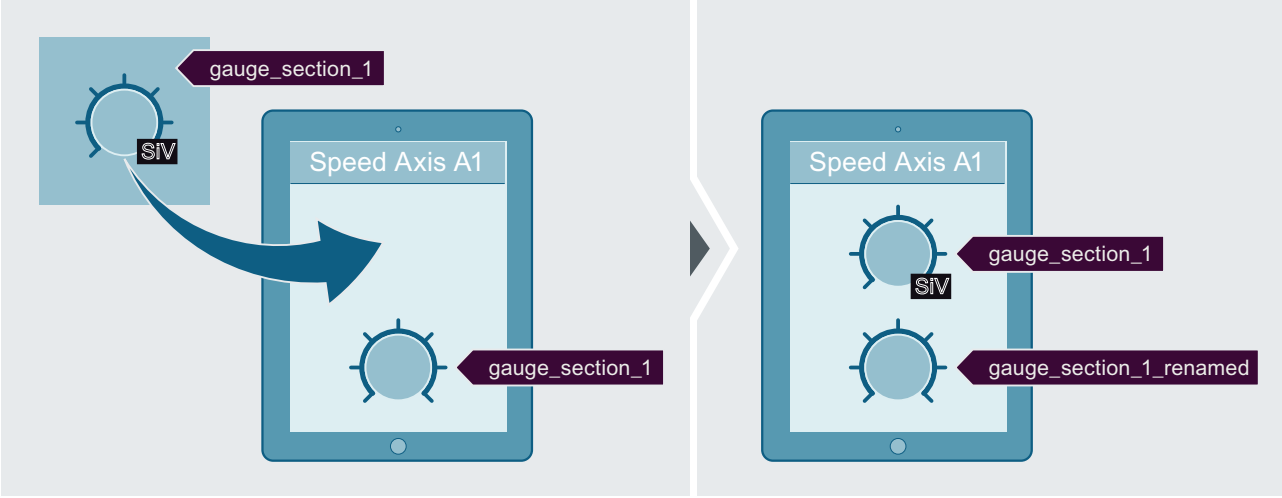
5.3.9 Avoiding conflicts during generation

Naming conflicts between existing and generated HMI objects

If generated HMI objects and manually created HMI objects are used side-by-side in a SiVArc project, naming conflicts can result. An evaluated SiVArc expression can result in the name of an existing object.

Except for screens and text lists, SiVArc behaves as follows during naming conflicts:

If a manually created HMI object with a name to be generated by SiVArc already exists, the existing object is given the suffix "_renamed". If this name is already taken as well, the name is automatically incremented.



Example: A symbolic I/O field was interconnected manually to a text list in the project. A text list of the same name is then created with SiVArc using the text list rules. The existing, manually interconnected text list is renamed. The interconnection remains unchanged.

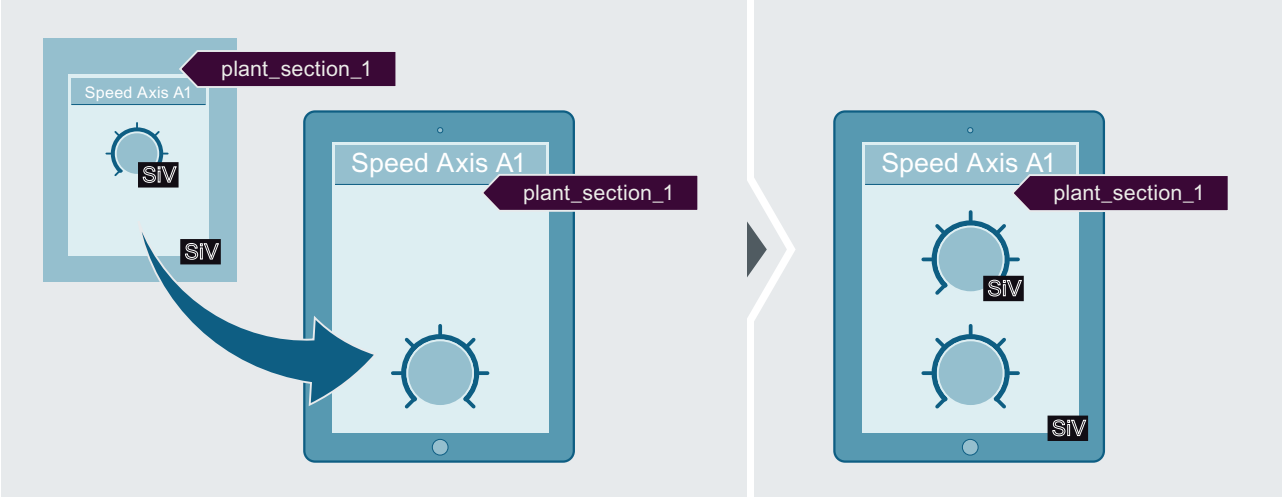
Make sure from the very beginning that the naming concepts for the generated HMI objects and for the manually created HMI objects are not the same. This prevents any corrections you may have to make later.

Identical names for screens and text lists

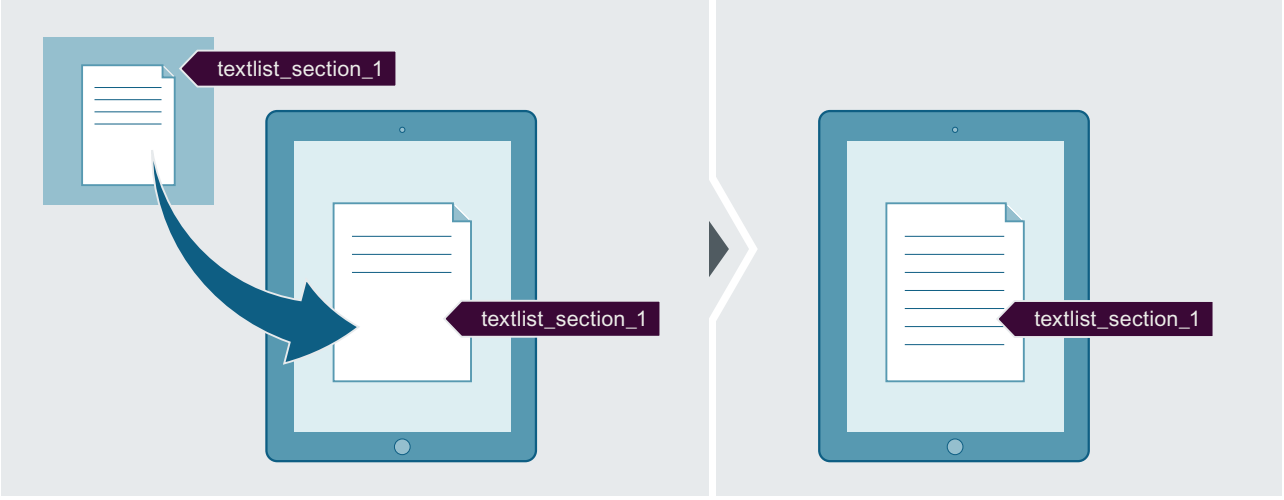
For screens and text lists SiVArc processes identical names differently:

If generated screens or text lists with the same name already exist, SiVArc captures these screens or text lists despite the naming conflict during the generation. Keep in mind that an error message is output for text lists but not for screens.

The figure below shows the generation of screens with identical names for a manually created screen. The existing screen is processed by the generation:



The figure below shows the generation of text lists with identical names for a manually created text list. The text list entries of the two text lists are combined in the existing text list:



Priority of the generated objects in case of naming conflicts

A generated object has a higher priority than a manually created HMI object in a SiVArc object. The generated object has a fixed link to SiVArc and is updated during each generation. Manual changes made to the generated object are reset.

In case of naming conflicts, SiVArc sets the following priorities during a generation:

1. Generated objects from screen, tag and text list rules
2. Generated objects from copy rules
   Objects generated from the copy rules are treated the same as manually created objects. They are created first during the generation. If there are naming conflicts with objects that are generated later, objects are renamed according to copy rules with the extension "_renamed".
3. Manually created objects
   If the names of manually created objects and generated objects are the same, the manually created objects are renamed.

Naming conflicts caused by multiple PLCs in the project

If naming conflicts occur during generation due to multiple connected PLCs, SiVArc only generates the first HMI object captured by the generation and outputs an error message.

For the unique assignment of tags in a project with multiple PLCs, use the option "Use PLC name as prefix in the HMI tag names" in the Runtime settings for tags.

When generating text list entries for multiple PLCs, naming conflicts can occur because a program block may be used in several PLCs. Depending on which text source from STEP 7 you use, the generation of text list entries reacts differently in these conflict situations:

- Block parameter
  Additional text list entries are generated with a suffix.
- Network
  Text list entries are created only for the first PLC evaluated. Text list entries to be generated for all subsequent PLCs are ignored. The error appears in an alarm and in the log.

Naming conflicts when importing rules

The following options are available for importing SiVArc rules.

- Overwriting existing rules through importing
  Rules and rule groups with the same name are updated. All other rules are retained.
- Renaming rules to be imported if rule name already exists
  In case of naming conflicts, the names of the imported rules and rule groups are given a consecutive number.
- Delete all existing rules prior to the import.
  After the import, the rule editor only contains the rules from the import file.

Purpose and benefits

If you avoid naming conflicts during configuration with SiVArc before you start, the result is a consistent project without errors. Based on a successful concept for unique differentiation of the naming concepts, you can derive additional automation projects with very little effort.

See also

Exporting and importing SiVArc rules (Page 168)

5.3.10 Creating tag rules

Introduction

Depending on your settings, SiVArc generates all external tags or only those tags that are relevant to the SiVArc project. SiVArc generates external tags for instance data blocks and global data blocks.

You specify the following in a tag rule:

- Name of the folder in which a generated tag is stored
- Name of the tag table in which a generated tag is created

Requirement

- You have created a function block with data block.
- The option "Accessible from HMI" is set at the block interface for the relevant PLC tags.
- The "Tag rules" editor is open.

Procedure

To create a tag rule, follow these steps:

1. Create a tag rule.
2. Assign a unique name to the rule.
3. Open the "SiVArc expressions" editor under "Tag group".
   - To define the name of the tag group as text, enter a character string in quotation marks.
   - To derive the name from the user program, enter a SiVArc expression.
   - To map the structure of the user program in the project tree, enter the SiVArc expression `HmiTag.DB.FolderPath .`
4. Open the "SiVArc expressions" editor under "Tag table".
   - Specify a text or a SiVArc expression as name of the tag table.
   - To write all tags of a data block to a tag table, enter the SiVArc expression `HmiTag.DB.SymbolicName`.
5. To enter a condition, if necessary, use SiVArc scripting.

Result

A tag rule has been created.

Additional settings

Optimize the following settings to control the tag generation:

- Tag generation mode under "Options > Settings > SiVArc"
- Runtime settings for tags
- Update cycle and acquisition type

Changing the arrangement of tag rules

You arrange the tag rules using drag-and-drop or via the shortcut menu commands. This functionality is only available when the columns of the "Tag rules" editor are neither sorted nor filtered. Use the shortcut menu to also re-sort "Tag rules" in the filtered editor.

To change the arrangement of the tag rules using drag-and-drop, follow these steps:

1. Select the first cell of the rule.
2. Drag the rule to the required position in the editor.

See also

Generating tags (Page 144)

Principle of the tag generation (Page 149)

Example: Adapting tag names (Page 164)

Changing SiVArc rules (Page 179)

Storage strategies for generated objects (Page 103)

Editing the view in the SiVArc editors (Page 229)

5.3.11 Creating a screen rule

Requirement

- The user program has been created.
- The generation template of a display and operating object has been created.
- The generation template of a screen has been created.
- The "Screen rules" editor is open.
- The columns "PLC" and "HMI device" are visible.

Procedure

To define a screen rule for generating a display and operating object, follow these steps:

1. Create a screen rule.
2. Assign a unique name to the rule.

3. Under "PLC", select the controllers for which the screen rule is to apply.
   If you select no controller, the screen rule applies to all controllers in the project.
4. Select the program block for which the HMI object is generated.
5. Under "Screen object", select the generation template of the display and operating object.
6. Under "Screen", select the generation template of the screen in which the object is generated.
   If a positioning scheme is stored for the generation template, select the positioning area under "Layout field". If you do not specify a positioning area, the generated HMI object is positioned in the screen according to the SiVArc positioning scheme.
7. Under "HMI device", select the HMI devices for which the screen rule is to apply. Use the toolbar icons to show the device types of the HMI devices.
   If you select no HMI device, the screen rule applies to all HMI devices that are connected to the selected controller.
   If the rule is only to be executed for objects or program blocks that meet a specific condition, program the corresponding expression under "Condition" with SiVArc scripting.

You can also add the program blocks and templates from the library using a drag-and-drop operation.

Result

When you generate the visualization, the object is generated in the specified screen.

If you have selected a positioning area in the screen rule, the HMI object is positioned within this area instead of a layout field. The layout field that is used depends on the order of generation of the screen rules and the index of the layout field.

See also

Generating visualization (Page 181)

Supported objects in the user program (Page 84)

Example: Creating screen rule with condition (Page 161)

Example: Organizing screen and text list rules (Page 162)

SiVArc rules (Page 132)

Creating SiVArc rules (Page 136)

Processing of rules (Page 141)

Changing SiVArc rules (Page 179)

Editing the view in the SiVArc editors (Page 229)

5.3.12 Creating text list rules

Introduction

A text list rule specifies which text list is generated for a program block.

Requirement

- The user program has been created.
- A generation template of the text list is stored in a library in the appropriate folder.
- The "Text list rules" editor is open.

Procedure

To define a text list rule, follow these steps:

1. Create a text list rule.
2. Assign a unique name to the rule.
3. Select the desired program block.
4. Select the required generation template of a text list.
5. To enter a condition, if necessary, use SiVArc scripting.

You can add the program blocks or master copies using drag-and-drop.

Result

When you generate the visualization, the text list is created in the "Text and Graphic Lists" editor.

See also

Example: Organizing screen and text list rules (Page 162)

Example: Creating generation templates for text lists (Page 119)

Example: Create generation template for a text list for block parameters (Page 121)

SiVArc rules (Page 132)

Creating SiVArc rules (Page 136)

Processing of rules (Page 141)

Changing SiVArc rules (Page 179)

Editing the view in the SiVArc editors (Page 229)

Creating alarm rules (Page 158)

5.3.13 Creating alarm rules

Introduction

Alarm rules form an integral part of SiVArc generation. While creating alarm rules, you program the PLC blocks using STEP 7 for generating objects in the HMI devices. So, alarm rules function as an abstract type in connecting STEP 7 with HMI devices.

Condition

- Global library and project library with master copies and types.
- The HMI objects generated are stored in library.

Procedure

To create alarm rules in SiVArc, perform the following steps:

1. Click "Alarm rules" under SiVArc folder.
2. To add a new rule, in the alarm rule editor, click "Create new rule". By default, the first rule name is displayed as alarm rule. Consecutive rule names are displayed with an incremental number suffixed as <Alarm rule_1>.
3. In "Program block" column, browse for a respective program block or program type.
4. In the "Master copy of Alarms/Classes/Groups" column, browse for a respective master copy.
5. In the "Condition" column, set a condition for the rule to be executed. This is optional in SiVArc.

Note

- No similar rules that are configured will be processed.
- In the program block, launch the shortcut menu command "Go to referenced object" . The selected program block is highlighted in the "Program blocks" area. In SiVArc "Plug-in" editor, choose alarm rules, which displays filtered list of alarm rules for a block.

Result

During SiVArc generation, the alarm objects are created only if the condition is satisfied, and the rule is successfully processed. You can view the generated objects in the "Generation Overview". For more details on generation overview, refer to the topic Generate visualization (Page 172)

General notes

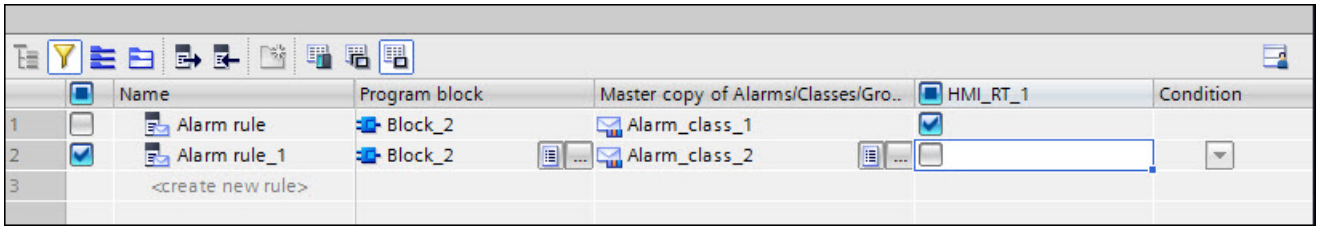
- Rules created in the "Alarm rules" editor will remain saved even after closing the editor and re-opening.
- You can perform copy, paste, drag, drop, and undo actions in the "Alarm rules"editor.

- While editing the "Alarm rules" editor options, if you modify or delete the library objects, the referenced object in the "Alarm rules" editor displays invalid reference by highlighting the area in pink color.
- You can group multiple rules by selecting the rules in the alarm rule editor, and right-click on the alarms, select "Add a new rule group".
- You can add new alarm rules within the "Rule group".
- While grouping alarm rules, the AND condition will be automatically displayed. You can choose different conditional operator from the operator drop-down list. Else, you can manually define a condition for a group.
- During the selection of the program block and master copies, SiVArc supports intellisense.

Device specific rules

Device specific alarm rules subject to alarm rules that are specific to a HMI device. The "Alarm rules" editor provides you with the show or hide HMI device option in the tool bar. Perform the following steps to configure device specific rules:

1. Click the show of hide HMI device option. The available HMI devices are displayed with a check box option.
2. You can select or de-select the HMI device for a specific alarm. By doing this, the "Alarm rules" will only be applicable for the selected HMI devices during SiVArc generation.



Note

- By default, the first available HMI device is selected.
- You can also select or de-select the PLC's.

Device type selection using properties option

To ease the selection of PLC/HMI devices among a list of PLC/HMI devices, the "Properties" tab in the alarm rule editor is used. In the "Alarm rules" editor, the "Properties" tab provides options to select or de-select the PLC/HMI devices for a specific rule. By doing this, the alarm rules are applicable to only the selected PLC/HMI devices.

See also

Generate visualization (Page 172)

5.3.14 Creating copy rules

Introduction

In a standardized operator control and monitoring solution, HMI objects are often created centrally and distributed as global libraries to the configuration engineers.

Requirement

- Global library with types and master copies.
- The HMI objects to be generated are stored in the library.

Procedure

To create HMI objects with SiVArc using copy rules, follow these steps:

1. Open the global library with the master copies and types.
2. Synchronize the content of the opened global library with the project library.
3. Create a copy rule for each HMI object to be generated.
   or
   Create a copy rule for a library folder.
4. Assign a unique name to the rules.

Result

The HMI objects are created in the respective folder of the project tree during generation. The HMI objects are created for each HMI device specified in a rule.

General notes

- During SiVArc generation, if the name of default alarm object in the HMI device and the object generated by Copy rules are identical, the application displays the following error message:
  “Object <<Alarm_name>> was modified by other SiVArc editor. Copy_rule generator can’t modify this object.”
- The copy rule editor allows users to select and de-select the rule that needs to be generated for HMI devices.
- You can drag and drop the objects from the master copy into the "Library object" column.
- The copy rule editor supports intellisense.

Note

Copy rules for alarms

Rules configured using copy rule editor are processed and generates HMI objects during SiVArc generation

See also

Use of copy rules (Page 146)

Changing SiVArc rules (Page 179)

Editing the view in the SiVArc editors (Page 229)

5.3.15 Example: Creating screen rule with condition

Example scenario

The same program block is used in a standardized user program to control a valve or a motor.

Requirement

A button labeled "Open Valve" or "Start Motor" is to be generated in the user interface depending on the use of the program block. The button for the valve is always going to be placed at the top of the screen; the button for the motor at the bottom of the screen.

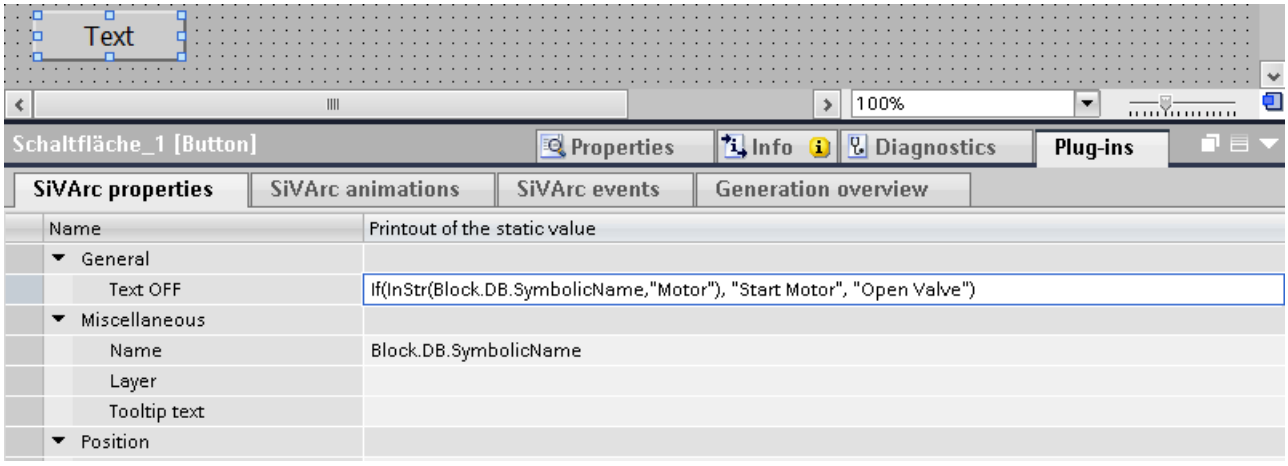
The screen rules are to be controlled so that a different button is generated for each use case of the program block.

Implementation concept

The configuration engineer uses conditions to control the labeling of the buttons and the screen rules. SiVArc generates a different button for each use case of the program block. To do so, the configuration engineer uses the conditions to access the symbolic names of the data blocks of the block instances:

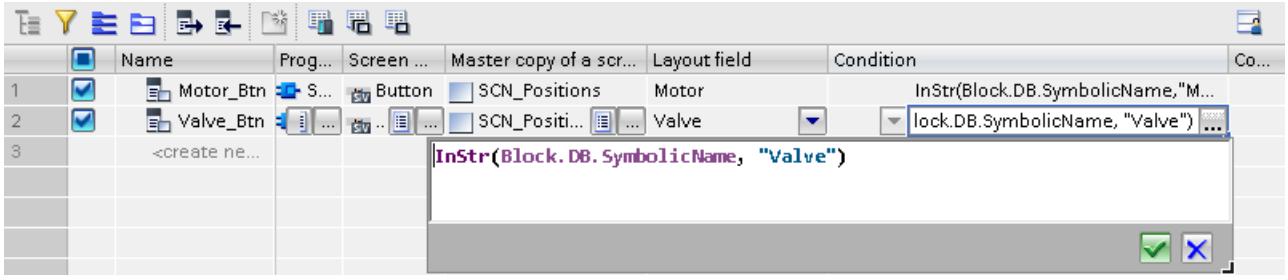
- If the program block is used for valve control, the name of the data block is "Valve_DB_Instance_<n>".
- If the program block is used for motor control, the name of the data block is "Motor_DB_Instance_<n>".

To control the labeling of the button, the configuration engineer programs a condition in the SiVArc property "Text OFF":



To control the positioning of the buttons, the configuration engineer creates a positioning scheme with the layout fields "Valve 1/1" and "Motor 1/1". The configuration engineer stores the scheme in the generation template for the process picture and creates the following two screen rules.

Ton control the execution of the screen rules, the configuration engineer programs a condition. The condition specifies that the rule is only executed when the symbolic name of the instance data block contains the character string "Valve" or "Motor".



See also

Creating a screen rule (Page 155)

5.3.16 Example: Organizing screen and text list rules

Example scenario

Multiple configuration engineers work on a large project in an engineering firm. Each engineer has a separate area of responsibility and field of expertise.

Requirement

The screen rules are to be processed separately by functionality. One configuration engineer is assigned the operation and representation of recipes. Another colleague sets up all diagnostic functions.

These functions are to be collectively enabled or disabled during generation as needed.

A single colleague is responsible for the layout of the HMI devices. She should be able to get a quick overview of the HMI devices used.

Implementation concept

A function-specific folder structure is created in the rule editors, for example:

- Start screens
  - ComfortPanel 19"
  - ComfortPanel 19" Portrait
  - MobilePanel 277 8"
  - PC station
- Diagnostic screens
  - ...
  - ...
- Recipe screens
  - ...
  - ...

To check the screen and text list rules, the controller programmer uses the toolbar to make only the relevant columns in the rule editor visible. Within the folders, the programmer sorts the rules according to the column "Program block" and filters by the relevant function block.

The main groups are collectively enabled or disabled to generate the project for each plant.

The respective groups for HMI device types are enabled or disabled to generate the project for each device.

The configuration engineers are assigned groups using the comment column; these groups contain the rules for their field of expertise. Each engineer copies his or her group to a test project. The engineer edits existing rules and creates new rules. Once released by management, the rules are once again imported in groups and reused in other projects.

The configuration engineer responsible for the layout distributes the templates for the corresponding HMI devices per screen type.

See also

Creating a screen rule (Page 155)

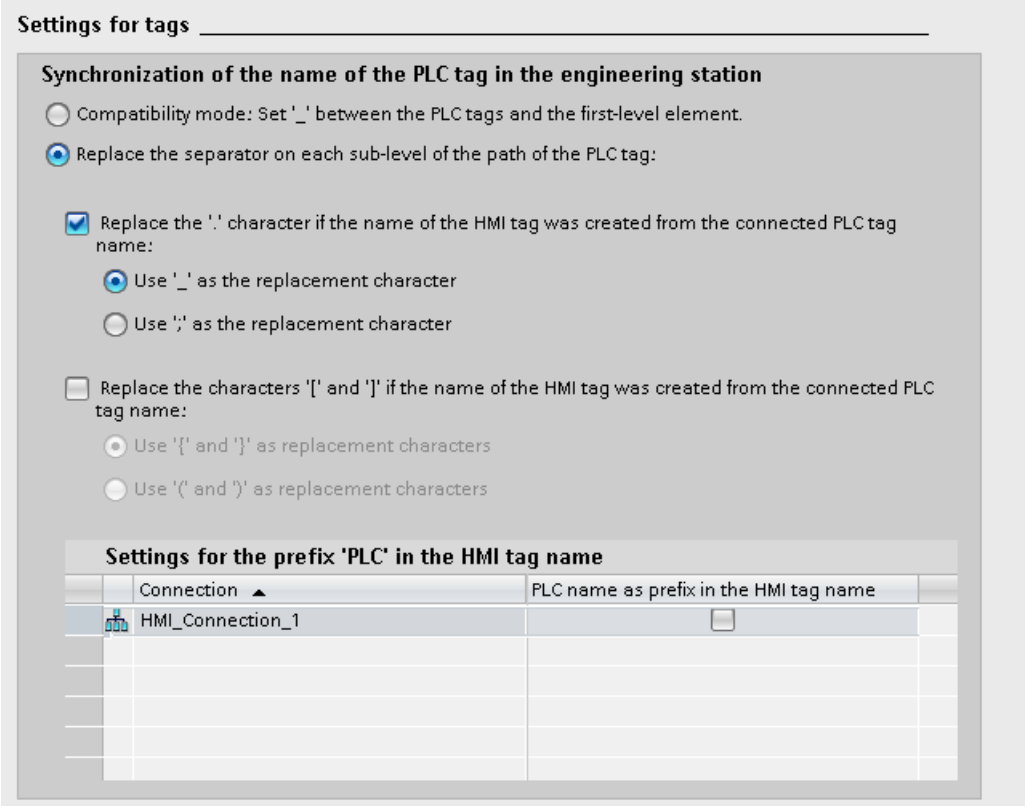
Creating text list rules (Page 157)

5.3.17 Example: Adapting tag names

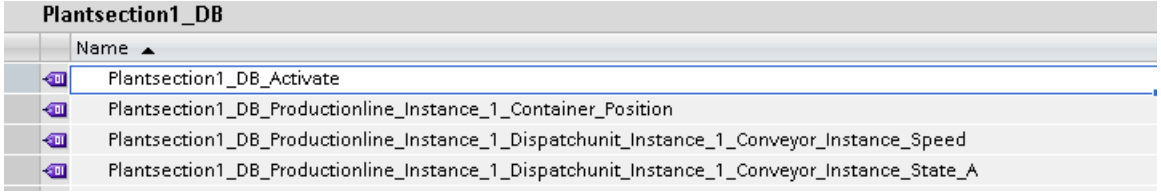
Example scenario

A new, structured PLC data type is used in a control program. To ensure synchronization of the PLC tags with the external HMI tags, the configuration engineer changes the tag settings of one HMI device. The Runtime settings for tags therefore differ for the HMI devices within a project:

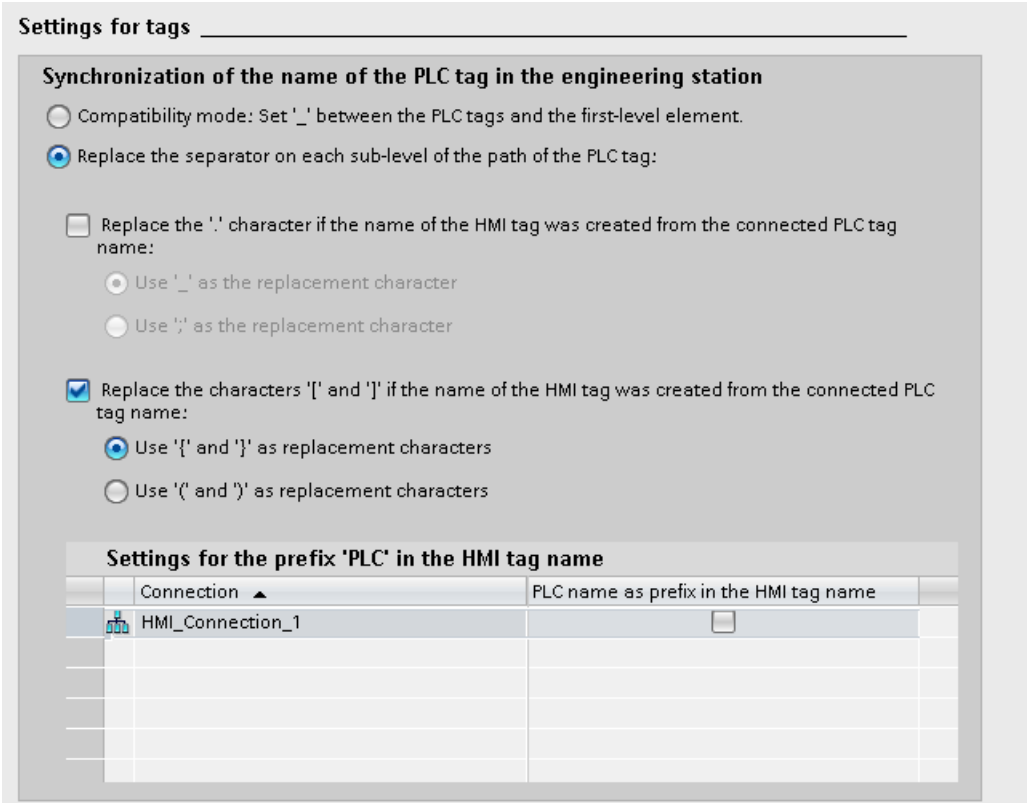
Runtime settings for HMI device 1:



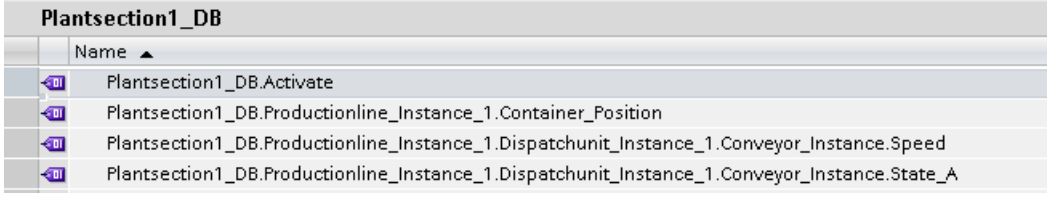
Synchronized HMI tags:



Runtime settings for HMI device 2:



Synchronized HMI tags



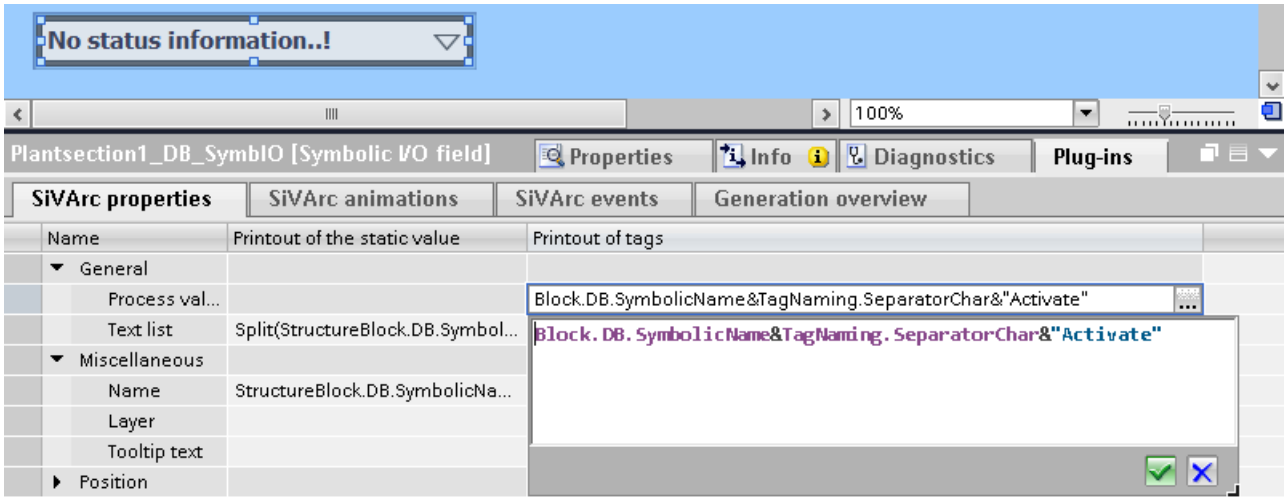
Requirement

The interconnections of the generated display and operating objects shall function even if different synchronization modes were used within a project.

Implementation concept

In the SiVArc expression that defines the trigger tag, the selected tag synchronization is addressed with the SiVArc object `TagNaming`.

The following SiVArc expression results in the tag name after generating the visualization: "Plantsection1_DB.Activate" for HMI device 1 and "Plantsection1_DB_Activate" for HMI device 2.



See also

TagNaming (Page 201)

Principle of the tag generation (Page 149)

SiVArc expression (Page 87)

5.3.18 Editing and managing SiVArc rules

Introduction

In complex SiVArc projects, there is a large number of SiVArc rules. You should therefore sort and structure your SiVArc rules clearly and make the rules available in the library.

Several functions are available to display the rules clearly organized:

- Filter function
- Grouping and sorting function
- Shortcut menus
- Drag-and-drop

To analyze the rules, navigate between the SiVArc editors, the user program and the generation templates via the "Go to..." commands of the shortcut menu.

Creating a SiVArc rule

1. Click "Add rule".
   A new row is created in the table editor.
2. Assign a unique name to the rule.
3. Insert the program blocks and generation templates from the library with drag-and-drop.

Alternatively, enter the first letters of the object that you want to reference. SiVArc shows a list of objects that can be referenced and that contain this sequence of letters in the referenced path.

When you insert a program block under "Name" with drag-and-drop, a new rule is created with the selected program block.

Grouping SiVArc rules

If you group SiVArc rules according to your own criteria, you obtain a better overview of your SiVArc project:

- You activate and deactivate rules contained in rule groups together.
- Conditions for one rule group apply to all rules within the group. You set up special cases via operands.
- You can move and arrange individual rules as you wish within and outside groups.
- When moving rules from group to group, the options that are set for the current group are applied.

Proceed as follows to create a rule group:

1. Select the rules for which a group is required.
2. Select "Add new rule group" in the shortcut menu.
   The selected rules are moved to a new group.
3. Name the rule group.

To create a subgroup, edit the required rules in exactly the same way within a group.

To open or close all rule groups at once, click the "Expand all" or "Collapse all" button.

Note

Filtering of rule groups

A group is only displayed during filtering if all rules of the group meet the filter condition.

Nested grouping of SiVArc rules

Conditions can be set for a rule group. You use rule groups to sort your SiVArc rules according to your requirements, for example, according to the plant structure, screen structure or by WinCC topics.

Application example for rule groups

All screen rules are sorted in groups in a SiVArc project according to the following screen types:

- Start screens
- Diagnostic screens
- Recipe screens

This makes it possible for you to assign the SiVArc configuration, for example, to the configuration engineers of a department according to specific topics.

You use group conditions to specify, for example, which tags must be included in a program block so that the respective rule group is included in the generation.

You generate a large variety of screens depending on the parameters included by using the condition operands of a rule within a group. In this way, you visualize many plant areas with a SiVArc project and a few rules.

Using SiVArc rules in a library

To update SiVArc rules centrally and consistently across projects, store SiVArc rules or rule groups as a master copy in a library. If a SiVArc rule with the same name already exists in the project, you can overwrite the rule or create a new rule.

If you overwrite a rule with a rule from the library, the following applies:

- SiVArc detects the generated HMI objects from a previous generation process and includes these objects in the generation.
- Manual changes to the generated objects are overwritten.

See also

Editing the view in the SiVArc editors (Page 229)

5.3.19 Exporting and importing SiVArc rules

Introduction

SiVArc rules and rule groups can be exported to MS Excel and imported from MS Excel.

Export and import are possible for each SiVArc editor or for the entire project.

You can also copy individual rules outside groups directly from the MS Excel worksheet into a SiVArc editor and vice versa.

Note

Exporting and copying rules

When you copy and paste rules, only the visible columns are inserted.

Exporting SiVArc rules of a SiVArc editor

1. Open the required SiVArc editor.
2. Click the "Export" button in the toolbar of the editor.
   A dialog opens.
3. Select the required storage location and name of the export file.
4. Click "OK".

The export file is created.

Exporting SiVArc rules of a project

1. Select "Common data > SiVArc" in the project tree.
2. In the shortcut menu, select "Export all rules".
   A dialog opens.
3. Select the required storage location and name of the export file.
4. Click "OK".

The export file is created.

Export file structure

A spreadsheet with the exported SiVArc rules is created in the workbook for each SiVArc editor. The spreadsheets have the following titles:

- ScreenRules
- TagRules
- TextlistRules
- CopyRules

Rules on importing

Note the following when you import the SiVArc rules into one individual SiVArc editor:

- The import file must have the "*.xlsx" format.
- If an import file has only one spreadsheet, this spreadsheet is imported regardless of its name.
- Only when spreadsheets of an import file have been renamed or deleted, select the required spreadsheets using a dialog.
  - To import a renamed spreadsheet, confirm the import separately in a dialog.
  - To exclude a spreadsheet from the import, skip the spreadsheet in the dialog. If you deleted it prior to the import, you still have to skip an empty view in the dialog.

Note

During import, make sure that the set configuration language of your project and the language used in the import file are the same.

Import options

The following options are available for importing SiVArc rules.

- Overwriting existing rules through importing
  Rules and rule groups with the same name are updated. All other rules are retained.
- Renaming rules to be imported if rule name already exists
  In case of naming conflicts, the imported rules and rule groups are given a consecutive number.
- Deleting all existing rules prior to the import
  After the import, the rule editor only includes the rules from the import file.

Importing rule groups

When a rule group cannot be specifically assigned, it is added in the first hierarchy level of the editor, for example, when the import file includes a circular reference or when the higher-level group is missing in the import file.

If existing rules are not renamed during the import, a rule group that is included in the import file multiple times is overwritten by the rule group listed at the bottom of the import file in each case.

Importing SiVArc rules to a SiVArc editor

1. Open the required SiVArc editor.
2. Click the "Import" button in the toolbar of the editor.
   A dialog opens.
3. Select the required import file and import option.
   A dialog opens if the import file contains multiple spreadsheets.
4. Select the required spreadsheet.
5. Click "OK".

Importing SiVArc rules into a project

1. Select "Common data > SiVArc" in the project tree.
2. In the shortcut menu, select "Import all rules".
   A dialog opens.
3. Select the required import file and import option.
4. Click "OK".

Result

The SiVArc rules are created in the SiVArc editors. The completion message includes a link to the log file. Alternatively, the import log is available under "Common data > Logs".

5.3.20 Setting up know-how protection for a SiVArc project

Introduction

Your SiVArc project includes SiVArc generation specifications individually created with the SiVArc scripting functionality. To protect SiVArc expressions in the entire project, activate the know-how protection for your project.

Know-how protection only covers the SiVArc editors, not the settings of SiVArc. The library and the SiVArc tabs in the Inspector window, as well as generated objects, are not affected.

Password

Assign a password for know-how protection. The password must be at least 8 characters long and include the following character types:

- Upper- and lower-case letters
- Special characters
- Numbers

Setting up know-how protection

1. Select "Common data > SiVArc" in the project tree.
2. Select "Know-how protection > Activate" in the shortcut menu.
   A dialog opens.
3. Specify the password.
4. Save the project.

You also use the shortcut menu to edit your password and to remove know-how protection.

Result

Know-how protection is activated for all SiVArc editors. If you want to open a SiVArc editor in the project tree, in STEP 7 or by jumping to it from the other editors, you will be prompted for a password. Know-how protection is also activated for the import and export of SiVArc rules.

5.4 Generate visualization

5.4.1 Basics on generation

Definition

When generating the visualization, you generate HMI objects depending on the texts and structures of the user program. In addition you also copy HMI objects without reference to the user program during the generation.

Unlike copying HMI objects based on library elements, products of a generation with SiVArc are once again captured and adapted by subsequent generations.

Generation phases

You generate the visualization in several phases with SiVArc. This way you improve and amend the project from one generation to the next. The existing generations are constantly being cleaned and updated in the process.

SiVArc distinguishes between the first and each subsequent generation. The subsequent generations are based on the first generation. Manual changes, for example, repositioning of the generated display and operating objects, are retained for subsequent generations.

You can trigger a completely new generation or change the selection of devices to be generated for the next generations.

Generating the visualization across devices

In case of individual changes in projects with many PLCs and multiple operator panels, it is better to generate the visualization for individual devices; for example, when replacing devices or during troubleshooting. The generation and download times are reduced accordingly. HMI objects can be generated with a screen rule or a screen rule group for multiple HMI devices.

In this way, you update and optimize the process pictures of your plant in a large SiVArc project for all devices or device types, also individually. The following functions will help you during configuration:

- Hiding and showing device-specific columns in the SiVArc editors using the toolbar
- Distributing individual rules to connected devices and controllers
- Display of device types in the screen rules for easier assignment of the matching positioning schemes
- Display of the device types in the Inspector window of the screen rules depending on the PLC

Restoring the missing assignment of the HMI device

When you copy, paste and rename a device within a project, the device retains the Runtime name. The textual cross-reference to this Runtime name cannot be removed after the renaming.

1. You can expand the entry for the HMI device in question under "Devices and networks > Network view > Network overview > Device".
2. Adapt the Runtime device name accordingly.

Your assignment will be in place again after the subsequent generation.

Scope of the generation

SiVArc offers several options to control the scope of the generation:

- Tag generation mode under "Options > Settings > SiVArc"
  If necessary, you limit the generation of tags to the tags in use.
- Exempt rules from generation in the rule editors
- Exempt devices from generation in the generation dialog

If you do not create any SiVArc rules, SiVArc only generates external tags.

First device-dependent generation

If your project contains several HMI devices or connected PLCs, SiVArc generates the visualization for the HMI devices and PLCs you have selected.

A dialog for station selection is displayed the first time generation is started in a project. In the dialog for station selection you select the devices for which SiVArc is to generate the visualization.

SiVArc generates the visualization device by device.

- If generation is not possible for a device, SiVArc continues with the next device.
- If you cancel the generation, a visualization completely generated for a device remains.

The available selection of stations is frozen after the first generation. Every following generation is based on this selection.

Identical names

If the names of manually created HMI objects are identical, the manually created HMI objects are renamed. The objects newly generated by SiVArc are always created under the name to be generated.

If a manually created HMI object with a name to be generated by SiVArc already exists, the existing object is given the suffix "_renamed". If this name is already taken as well, the name is automatically incremented.

If naming conflicts occur during generation with multiple connected PLCs, SiVArc only generates the HMI object captured by the generation and outputs an error message.

Note

HMI device runtime settings

When tags with multiple PLCs are generated, the "PLC prefix" option from the runtime settings of the HMI device is evaluated.

Ensure that the "PLC prefix" option is enabled in the runtime settings for each PLC. Otherwise, SiVArc generation will be cancelled.

Note

Exception

For screens and text lists, the behavior in the case of identical names differs as follows:

If generated screens or text lists with the same name already exist, SiVArc generates these screens or text lists again despite the naming conflict. Keep in mind that an error message is output for text lists but not for screens.

Result of the first generation

SiVArc generates the HMI objects based on the SiVArc rules and saves them according to the configuration.

Renewed device-dependent generation

If you do not enable a device for the next generation in the dialog for station selection again, the generated objects and the manual changes are retained in the project.

Note

To remove generated objects of a PLC that is no longer enabled during the next generation, delete the connection between the PLC and HMI device.

If an existing connection between the HMI device and controller was deleted, a warning is issued in the dialog for the station selection. When you delete a connection between PLC and HMI device, all associated generated objects are removed during the next generation.

Generation with new station selection

The dialog for station selection is always displayed with the initial generation in a project.

The station selection dialog does not appear again the next time generation is started. SiVArc then generates the same HMIs and PLCs as during the previous generation. To change the settings, follow these steps:

1. Select the project or the device in the project navigation.
2. Click "Generation of visualization > Generate with station selection" in the shortcut menu of Runtime in the project tree.
   Alternatively, press the shortcut <ALT+Shift+G>.

Changes to the controller

If you delete a PLC with which you have already generated a visualization, all objects generated with this PLC are deleted during the next generation.

If you delete a block call in the user program and generate it once again, the objects generated for this block call are deleted.

Suppressing check of PLC compilation

You suppress the check of the PLC compilation using the "SivarcDisableCompileClean" file. In this case, the generation is run even if the PLC compilation is not error free.

To do so, create an empty file with the name "SivarcDisableCompileClean" in the SiVArc installation directory that contains the "Siemens.Simatic.Sivarc.dll" file.

Note

If a file with the name "SivarcDisableCompileClean" is not contained in the SiVArc installation directory and the PLC compilation contains errors, the SiVArc generation is canceled.

If the "SivarcDisableCompileClean" file is contained in the SiVArc installation directory and the PLC compilation is free of errors, the SiVArc generation is run.

See also

Generating visualization (Page 181)

Avoiding conflicts during generation (Page 151)

Subsequent changes (Page 176)

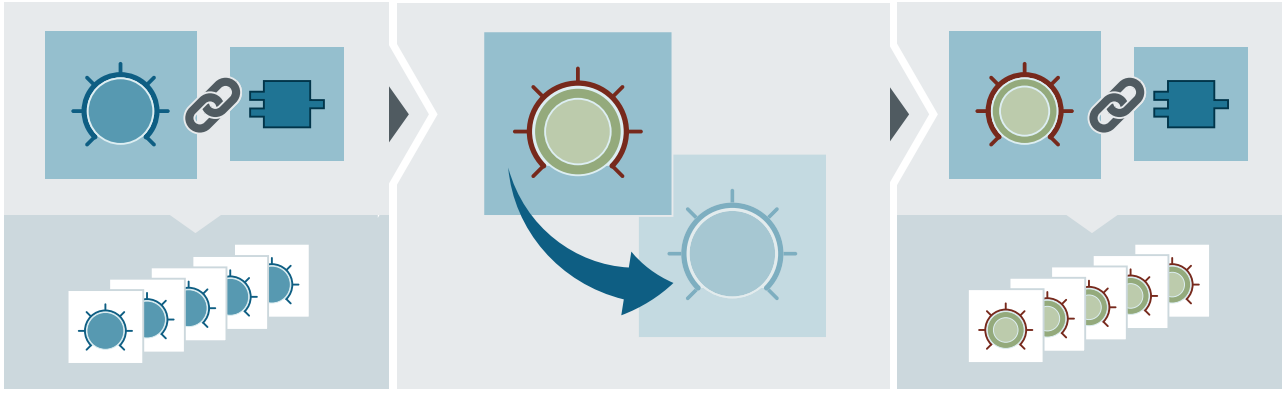
5.4.2 Subsequent changes

5.4.2.1 Changing generated objects

Application reference

You change generated display and operating elements centrally with SiVArc by using the generation templates or in the user program. Manual changes to the generated objects will be lost during the next generation.

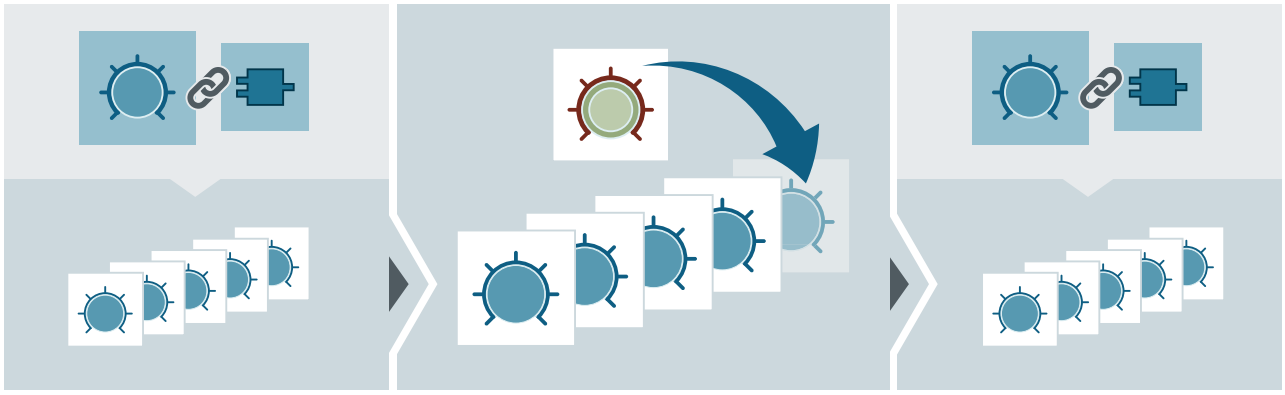
If the display and operating elements are to be adapted graphically at a later time, the visualization engineer merely changes the associated generation template. The configuration engineer stores the template in the library under the same name as the previous one. During the next generation the display and operating elements are graphically adapted without the need for additional configuring.



SiVArc reference to generated objects

Generated HMI objects have a permanent reference to the SiVArc rules from which they were derived. This reference has the following results with each new generation:

- Objects that no longer have a reference to the SiVArc configuration are removed. The reference is lost, for example, when a rule is deleted.
- Objects whose specifications for generation were changed are updated.
- Except for SiVArc-compliant manual changes, all manual changes to the generated objects are undone.



SiVArc-compliant manual changes

The following changes to generated display and operating elements remain in effect for all upcoming generations:

- The first new positioning of generated objects
  New positioning remains in effect even if the position has been defined with its own positioning scheme. Even if you change the positioning scheme, the manually configured position is retained after the next generation.
  Only for screen objects with fixed positioning is a manual change in the position reset to the fixed positioning saved upon the next generation process.
- Changes to size and rotation angle remain in effect except for faceplates and screen windows.
- You can change the displayed screen in a generated screen window.
- Manually changed text list entries are retained after a subsequent generation.

Generation matrix

Changes made using the generation matrix are retained with each subsequent generation:

- You use the generation matrix to generate objects to other screens.
- You use the generation matrix to generate screens to other devices.

Manually created HMI objects

Manually created HMI objects are not included in the SiVArc generation, except in case of naming conflicts.

If you reuse generated display and operating elements with copy and paste in your project, these are considered manually created objects and lose their SiVArc reference. Just like all other manually created screen objects, the copied objects are never deleted by SiVArc.

Note

Generated display and operating elements from the generation template of a screen

If you want to reuse screen objects by copying and pasting them, only use screen objects outside a generation from a master copy of a screen.

Name changes of generated display and operating elements

If the name of a generated HMI object has been changed, the object is created and interconnected again at the next SiVArc generation. The object with the changed name is also included in the project.

Change the name of generated display and operating elements only by using the SiVArc expression in the generation template or in the text sources in the user program. The name of the generated object is updated accordingly during the next generation.

Manually overwritten text list entries

When you overwrite generated text list entries, the changed text list entry is retained during the next generation only for the default text of the master copy.

If the text for the text list is generated from the network text definition in STEP 7 or the symbol tables and you change this text, the changes are overwritten by the next generation.

The example below illustrates how SiVArc processes changed text list entries:

The text list contains two entries: "Entry_1" and "Entry_2". "Eintry_1" contains a text generated by SiVArc. "Entry_2" contains a text which has been copied from the master copy of the text list.

- Change "Entry_2" and start the SiVArc generation. After generation, your changes are in the "Entry_2".
- Change "Entry_1" and start the SiVArc generation. After generation, your changes are overwritten at the "Entry_1" by the text generated by SiVArc.
- Change "Entry_1" and "Entry_2" and start the SiVArc generation. After generation, your changes are overwritten at the "Entry_1" by the text generated by SiVArc. Your changes to "Entry_2" are overwritten by the text from the master copy of the text list.

Advantages of the SiVArc change mechanisms

The SiVArc functionality for changing generated display and operating elements enables an efficient and consistent adaptation for the configuration engineer involving very little work.

SiVArc makes it possible for the company to distribute standardized display and operating elements throughout the company and apply them even in ongoing projects.

See also

Example: Using the generation matrix (Page 189)

Identifications in the SiVArc project (Page 180)

Picture legends (Page 231)

5.4.2.2 Changing SiVArc rules

Effects

When you change SiVArc rules, you interfere centrally in an existing project.

SiVArc rules must be edited, for example, when a function is always going to be visualized on another screen or if an operating object is to be removed because the plant has changed.

Editing SiVArc rules later

You can change rules already created by selecting the rule and using commands from the shortcut menu. If you change the name and storage paths of objects in the project, the affected rules are updated accordingly.

Change the name and storage paths of objects only in the project or in the project library. Changes to global libraries or the path information for referenced objects are not supported by SiVArc.

Using SiVArc rules in a library

To update SiVArc rules centrally and consistently across projects, store SiVArc rules or rule groups as a master copy in a library. If a SiVArc rule with the same name already exists in the project, you can overwrite the rule or create a new rule.

If you overwrite a rule with a rule from the library, SiVArc responds as if you were changing the rule manually:

- SiVArc detects the relevant HMI objects from a previous generation process and includes these HMI objects in the generation.
- Manual changes to the relevant HMI objects are overwritten.

Changing the names of SiVArc rule master copies

Proceed as follows to create a link between a renamed screen rule in the library and the screen rule based on it in the project:

1. Change the screen rules in the project manually in accordance with the new names of the master copies in the library.
2. Now copy the renamed master copies to your project. Overwrite the existing, newly named screen rules in the project.

Editing references of a SiVArc rule

If you edit referenced HMI objects or program blocks in the project or project library, the SiVArc rule is automatically adjusted.

If you change referenced objects in the global library, the corresponding SiVArc rules become invalid.

Advantages of editing SiVArc rules

Because SiVArc rules consist of dynamic links in the project, they can be easily adapted without causing inconsistencies in the project. You can work with fixed rule sets, for example, that you adapt individually or continue to develop further throughout the company.

See also

SiVArc rules (Page 132)

Creating tag rules (Page 154)

Creating a screen rule (Page 155)

Creating text list rules (Page 157)

Creating copy rules (Page 160)

5.4.2.3 Identifications in the SiVArc project

Generated objects and SiVArc configurations in the project

The following objects are identified in a SiVArc project:

- Generated objects which are detected by the next generation
- Objects that contain SiVArc configurations

The following table shows in which form the objects are identified in SiVArc:

Location	Icon/ Identification	Object
Project tree		Relevant object (HMI screen)
Project library or Global library		Master copy with configured SiVArc properties, events or animations
		Type with configured SiVArc properties, events or animations
		Type version with configured SiVArc properties, events or animations

You specify the identification for generated screen objects in the "Screens" editor under "Options > Settings > SiVArc".

Generated HMI object

A generated HMI object is generated again and overwritten in the next generation. Before the next generation, the objects with a matching name from the previous generation are recorded.

If you change the name of a generated object, it will no longer be captured by the generation.

Note

Copying generated objects to other projects

If you copy a generated object to other projects with or without SiVArc, the identification is retained.

Identification in the "Screens" editor

The identification in the "Screens" editor is optional. You enable the identifications and specify the required colors for border and background in the TIA Portal settings under "Options > Settings > SiVArc".

5.4.3 Generating visualization

Requirements

- User program and hardware were compiled without errors.
- Screen rules have been defined.
- The master copies and faceplate types used in the screen rules are stored in the project library or global library.
- Tags have been defined.
- Text list rules are defined.
- All used instances of types are updated to the latest version.

Note

Changes to the controller require a compilation

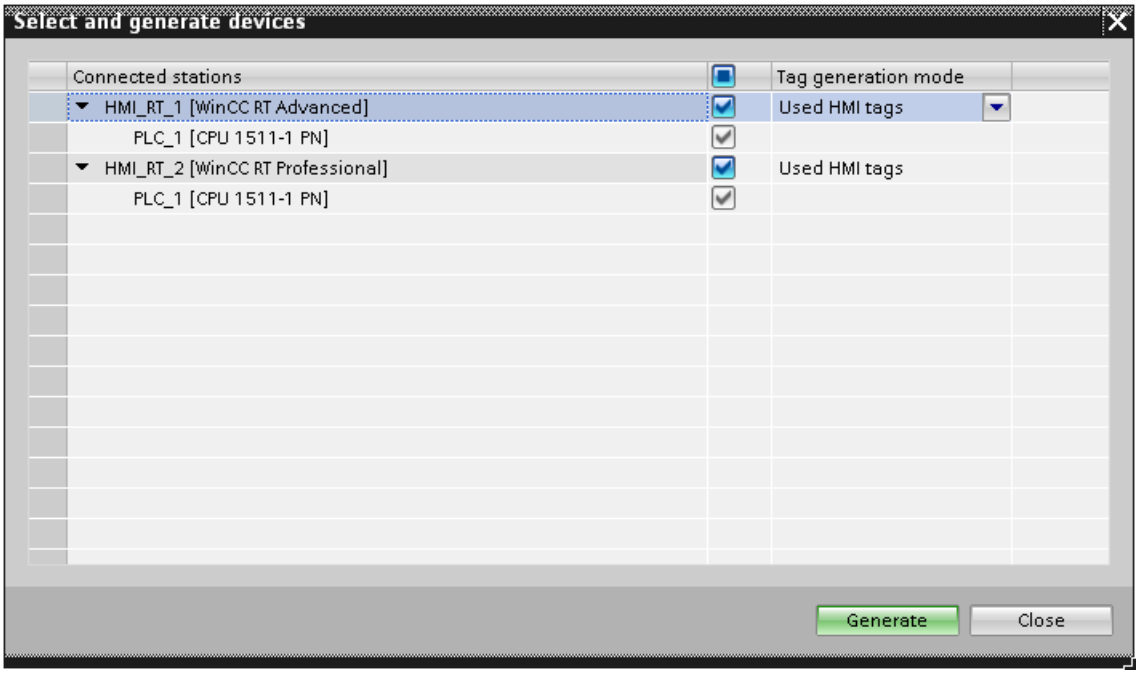
Changes in the user program or in the hardware configuration must be compiled before you generate the visualization.

Generation without station selection

- Click "Generation of visualization > Generate" in the shortcut menu of Runtime or the HMI device in the project tree.

Generation with station selection

1. Ensure that the "PLC prefix" option is enabled in the runtime settings of the HMI device for all PLCs.
2. Click "Generation of visualization > Generate with station selection" in the shortcut menu of the project in the project tree.
   The dialog "Select and generate devices" is opened.



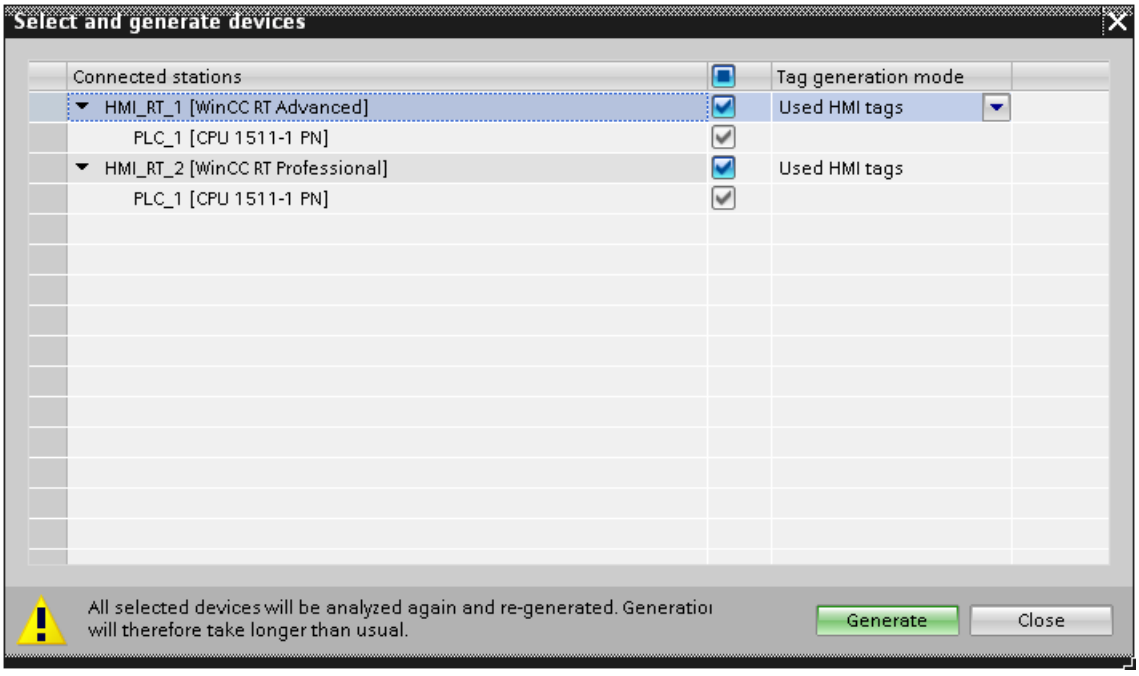
3. Activate the HMI devices and PLCs for which a visualization is generated.
   To generate the visualization for all devices, activate the option in the header.
4. Click "Generate".

Restart overall generation

If you have made changes to the user program, it may be necessary to restart an overall generation. Even if nothing was changed in the user program, the restarted overall generation runs through the entire user program.

The selection of the connected stations remains the same as in the first generation and cannot be modified.

You start the overall generation by selecting the project or the device and using the shortcut <Alt>+<Shift>+<F>.



Device display

Devices that exist in the project but are not connected to a controller are not shown in the SiVArc editors.

Note

IPI devices

Controllers and devices that are connected via IPI with the project are not displayed in the selection window.

5.5 Checking result

5.5.1 Check the result

Introduction

Comprehensive SiVArc projects require additional analysis and optimization after the first generation.

This document provides an overview of the options for analysis and post-processing of a SiVArc project.

Check mechanisms

SiVArc provides different functions and editors to check the generation. It includes the following points:

- Which objects were actually generated?
  You can find an overview in the "Generation overview" editor under "Common data > SiVArc". The generation overview is created during the first generation and updated in future generations.
- Which objects were not generated or generated with errors?
  You can find errors during generation in the Inspector window under "Info" with navigation to the error location.
  In addition, a log of the generation is displayed in the project tree under "Common data > Logs".
  The organization of the rules in the editors is helpful during troubleshooting. This way you can generate your projects in sections to locate any potential errors more easily.

To get a project that will run, compile and download the project to test it. Testing will ensure that the project is complete and runs properly in runtime.

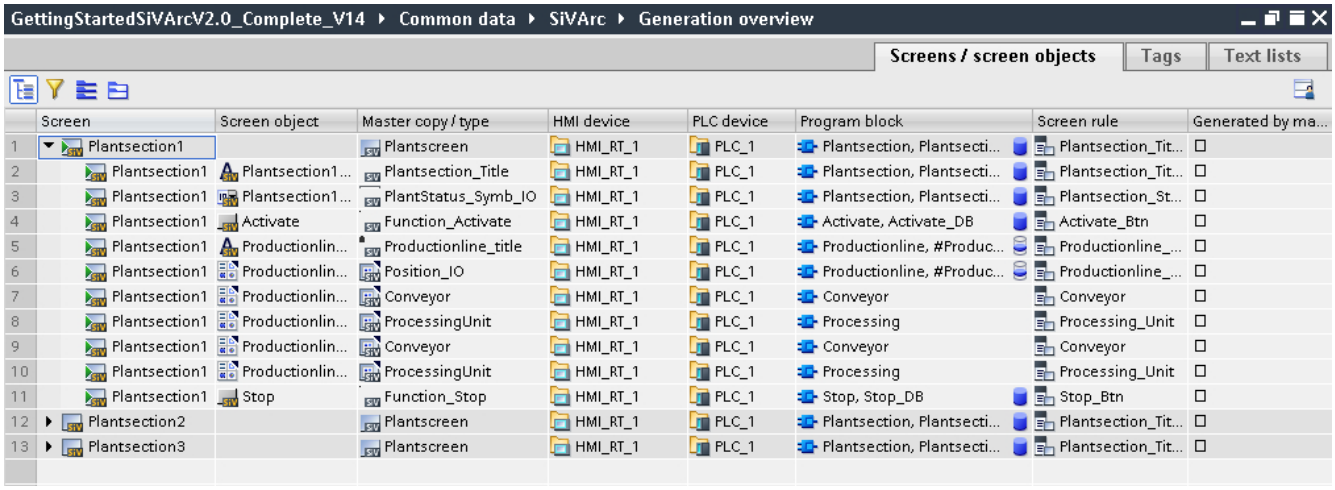
Checking for completeness of the generation

To open the generation overview, double click "Common data > SiVArc > Generation overview" in the project tree. You can also open the generation overview from the completion message to generate the visualization in the Inspector window.

To identify blocks, screen rules or generated display and operating objects in the project listed in the generation overview, select the shortcut menu command "Go to referenced object".

Use the filter and sorting functions of the editor to show different views in the generation overview. The relationships between screen rules, generated display and operating objects and devices can be read this way.

With the help of the generation overview, you plan and configure subsequent changes for an additional generation.



	Screen	Screen object	Master copy / type	HMI device	PLC device	Program block	Screen rule	Generated by ma...
1	Plantsection1		Plantscreen	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	☐
2	Plantsection1	Plantsection1...	Plantsection_Title	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	☐
3	Plantsection1	Plantsection1...	PlantStatus_Symb_IO	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_St...	☐
4	Plantsection1	Activate	Function_Activate	HMI_RT_1	PLC_1	Activate, Activate_DB	Activate_Btn	☐
5	Plantsection1	Productionlin...	Productionline_title	HMI_RT_1	PLC_1	Productionline, #Produc...	Productionline_...	☐
6	Plantsection1	Productionlin...	Position_IO	HMI_RT_1	PLC_1	Productionline, #Produc...	Productionline_...	☐
7	Plantsection1	Productionlin...	Conveyor	HMI_RT_1	PLC_1	Conveyor	Conveyor	☐
8	Plantsection1	Productionlin...	ProcessingUnit	HMI_RT_1	PLC_1	Processing	Processing_Unit	☐
9	Plantsection1	Productionlin...	Conveyor	HMI_RT_1	PLC_1	Conveyor	Conveyor	☐
10	Plantsection1	Productionlin...	ProcessingUnit	HMI_RT_1	PLC_1	Processing	Processing_Unit	☐
11	Plantsection1	Stop	Function_Stop	HMI_RT_1	PLC_1	Stop, Stop_DB	Stop_Btn	☐
12	Plantsection2		Plantscreen	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	☐
13	Plantsection3		Plantscreen	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	☐

The generation overview is also available at several places in the SiVArc project:

- WinCC
  Inspector window of a generated screen
  All generated display and screen objects of the selected screen are displayed in the "Generation overview" tab.
- STEP 7
  Inspector window of a block
  The "Screen generation overview" and the "Text list generation overview" displays show all screens generated from the selected program block, the associated screen objects and text lists.

Many functions for filtering and sorting the "Generation overview" editor make it easier to get an overview from different perspectives.

Troubleshooting using targeted individual generation

You can also check your project section by section. You generate individual sections of the project with the rule editors.

- You can switch rules on and off as a group.
- When you disable a rule after the generation, all associated generated objects are removed from the generation.
- Enabling and disabling rules overwrites the condition of a rule. When a rule has the "TRUE" condition, for example, it is only applied when the rule is enabled. When a rule has the condition "FALSE", it is not included in the generation even if the rule is enabled.
- When you enable the rules again for the next generation, the associated objects are generated once again.

Last-minute changes

You use the generation matrix to implement final changes without having to analyze and change the SiVArc rules.

See also

Using the generation matrix (Page 186)

Example: Using the generation matrix (Page 189)

Editing and managing SiVArc rules (Page 166)

Editing the view in the SiVArc editors (Page 229)

5.5.2 Using the generation matrix

Application of the generation matrix

The generation matrix can be used for subsequent changes to the assignment of generated objects; it is mainly intended for commissioning engineers who have to make last-minute adjustments in the project.

You get the most out of your generation matrix when you only use screen rules to generate screens and screen objects during the configuration.

Description

The generated screens and screen objects for an HMI device or an HMI device type are displayed in the "Generation matrix" editor after each generation.

In addition, you can adjust the assignment as follows:

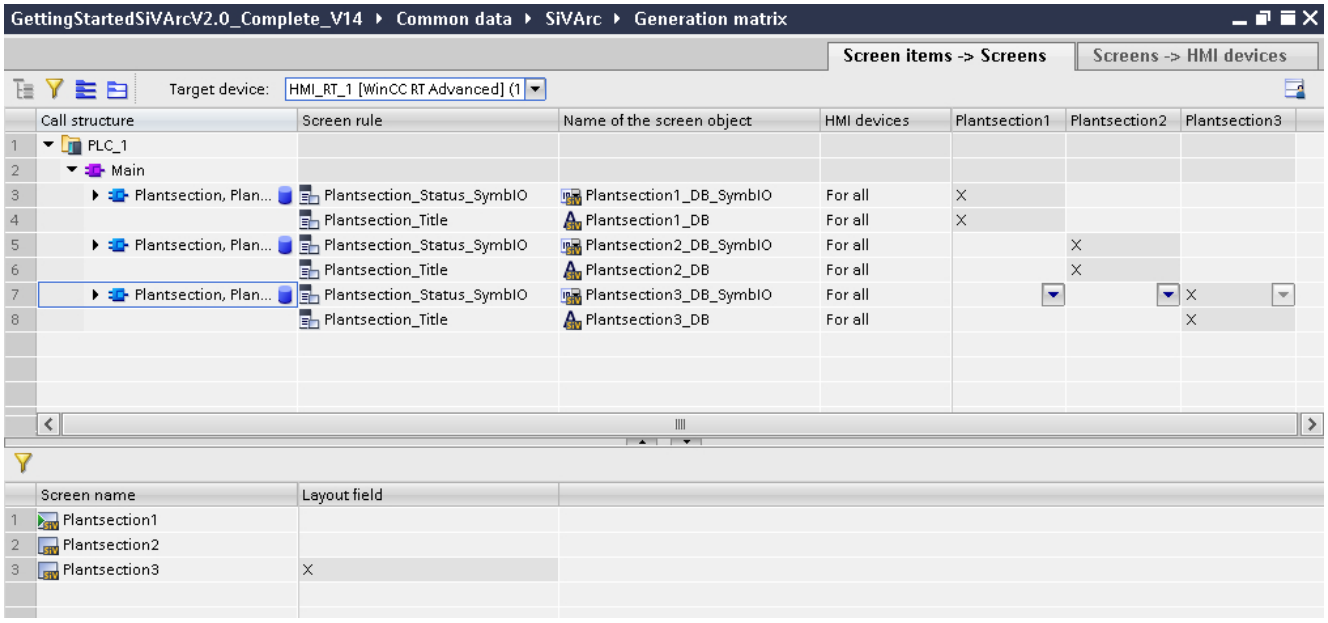
- Generate screen object in another screen
- Generate screen in another HMI device

Changed assignments become effective at the next generation. Depending on your settings, the screen navigation is adjusted at the same time.

Tab "Screen objects -> Screens"

In the toolbar of the editor, you select the HMI device for which the matrix is to be displayed under "Target device". SiVArc also displays the device type for all devices.

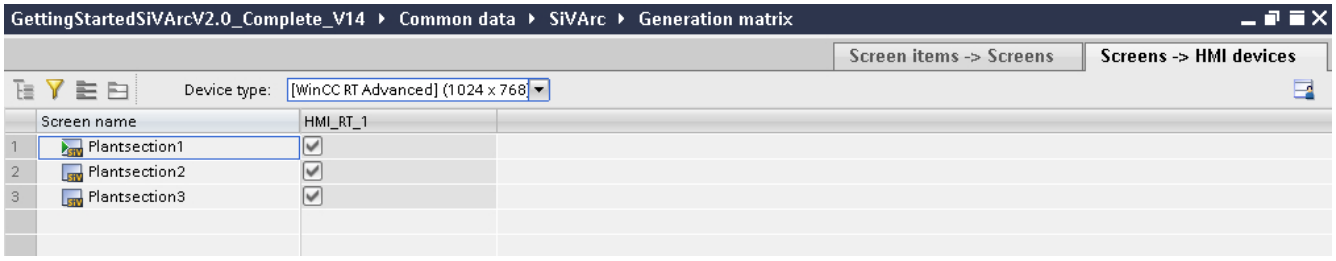
In this tab, assign a generated screen object to another screen.



"Screens -> HMI devices" tab

In the toolbar of the editor, you select the HMI device type for which the matrix is to be displayed under "Device type". The editor then displays the screens of all HMI devices of this type.

On this tab, assign a generated screen to another HMI device.



Adjust assignment of generated screen objects and screens

1. To change the assignment of a screen object, select the layout field or "X" in the corresponding cell in the "Screen objects -> Screens" tab.
2. To change the assignment of a screen, select the check box in the corresponding cell in the "Screens -> HMI devices" tab.
3. Generate the visualization.

Adapting navigation buttons for screens

Navigation buttons leading to a screen that is newly generated with the matrix are generated again according to the screen hierarchy.

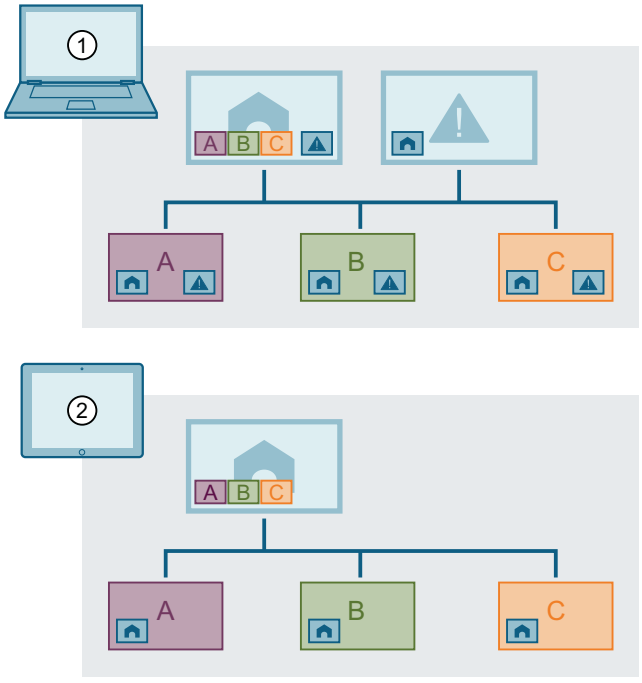
1. Activate the "SiVArc > SiVArc settings > Matrix settings > Generate navigation objects" option.
2. Reassign the screens.
3. Generate the visualization.

The screens and navigation buttons leading to this screen are generated again.

Example: Moving screens with navigation to other devices with the generation matrix

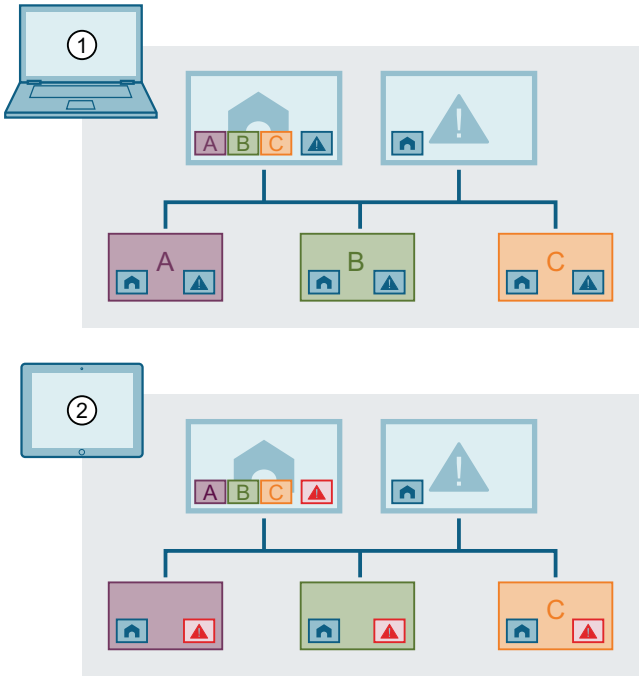
You have generated a start screen, a diagnostic screen and lower-level screens on HMI device 1. The start screen and the diagnostic screen can be displayed from each lower-level screen with the help of navigation buttons.

A diagnostic screen was not generated on HMI device 2.



① HMI device 1
② HMI device 2

When you move the diagnostic screen to HMI device 2 with the generation matrix, the navigation buttons are adapted accordingly.



① HMI device 1
② HMI device 2

See also

Example: Using the generation matrix (Page 189)

Editing the view in the SiVArc editors (Page 229)

5.5.3 Example: Using the generation matrix

Example scenario

An HMI device is still missing when a plant is commissioned. The vendor does not deliver, and the project may be delayed. All contents of the HMI device are therefore integrated into another HMI device until the vendor delivers the missing HMI device.

Requirement

To ensure smooth commissioning without delays, the commissioning engineer is supposed to change the plant structure in the WinCC project last minute.

Implementation concept

The commissioning engineer uses the generation matrix to move generated display and operating objects as well as screens to meet the new requirements.

The change is evaluated during the next generation. The navigation is automatically adjusted by hierarchy.

The original project is retained and can be generated again as soon as the missing HMI device has been delivered.

See also

Using the generation matrix (Page 186)

Reference

6

6.1 SiVArc objects

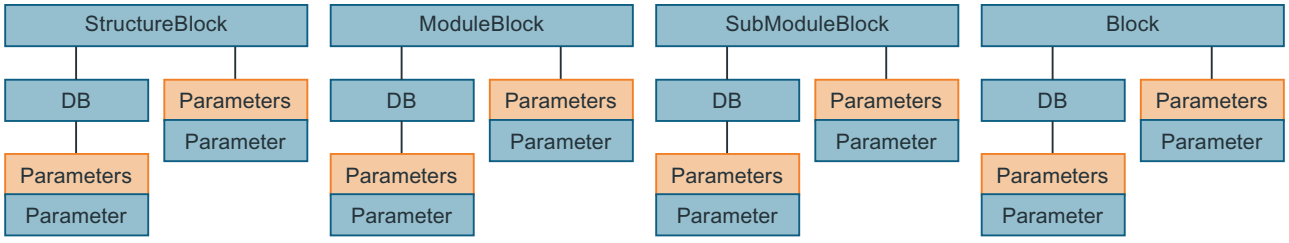
6.1.1 Object hierarchy

Introduction

You can use SiVArc expressions to directly address data from different areas of the TIA Portal.

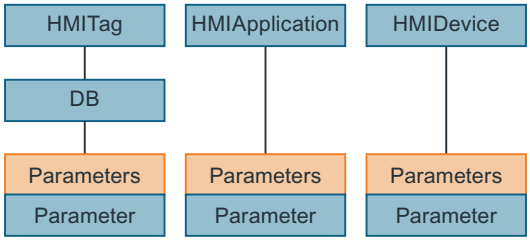
Program call in STEP 7

You can use keywords to access the blocks in the user program, the associated data blocks and their parameters.



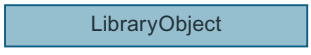
WinCC data

You can use the following key words to access external tags, devices and applications of the visualization.



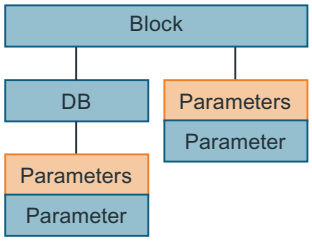
Library data

You can use the keyword `LibraryObject` to access the storage location of a generation template in the library.



6.1.2 Block

Description



Represents the program block that is currently being executed by SiVArc regardless of its position within the call hierarchy.

Use

Use the "Block" object as follows:

- "FolderPath" object property
  `Block.FolderPath`
  Accesses the path of the block in the project tree within the "Program blocks" folder, e.g. "Plant\Plantsection\Productionline"
- "Name" object property
  `Block.Name`
  Accesses the internal name of the block, e.g. "FB1".
- "SymbolicName" object property
  `Block.SymbolicName`
  Accesses the user-defined name of the block.
- "NetworkComment" object property
  `Block.NetworkComment`
  Accesses the comment entered in the network of the block.
- "NetworkTitle" object property
  `Block.NetworkTitle`
  Accesses the title of the network in which the block is instanced.
- "Number" object property
  `ModuleBlock.DB.Number`
  Accesses the block number in the block properties.
- "Parameters" list
  `ModuleBlock.Parameters("Activate").Value`
  Accesses a block parameter.
- "SymbolComment" object property
  `Block.SymbolComment`
  Accesses the user-defined comment in the block properties.
- "Title" object property
  `Block.Title`
  Accesses the header of the block in the block properties.

- "Version" object property
  `Block.Version`
  If the block is an instance of a block type, this expression accesses the type version of the block type in the library.
- "Parameters" list
  `Block.Parameters(<Name Parameter>).AssignedTag.Comment`
  Accesses the comment of a tag that is assigned to the block parameter.

6.1.3 DB

Description

Represents the data block of a block. The DB object is a SiVArc object of the second hierarchy level. A block from the call hierarchy or `HMITag` object always precedes the DB object.

Use

Use the "DB" object as follows:

- "Comment" object property
  `ModuleBlock.DB.Comment`
  Accesses the comment in the block properties.
- "FolderPath" object property
  `HMITag.DB.FolderPath`
  Accesses the path of the block in the project tree within the "Program blocks" folder, e.g. "DBs\Plant"
- "Number" object property
  `SubModuleBlock.DB.Number`
  Accesses the block number in the block properties.
- "SymbolicAddress" object property
  `StructureBlock.DB.SymbolicAddress`
  Accesses the user-defined name of the data block.
  If the data block is a multi-instance, the symbolic address of the block is returned.
- "TagPrefix" object property
  `StructureBlock.DB.TagPrefix`
  Accesses the user-defined name of the data block.
  If the data block is a multi-instance, the symbolic address in HMI format is returned. Instead of ".", "_" is used as the delimiter between the name of the data block and the name of the tag.

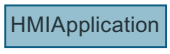
- "SymbolicName" object property
  `HMITag.DB.SymbolicName`
  Accesses the user-defined name of the data block.
- "Type" object property
  `ModuleBlock.DB.Type`
  Accesses the type of data block: Single instance (IDB) or multi-instance (MDB).

See also

Influence of multilingualism on a generation template (Page 101)

6.1.4 HMIApplication

Description



Represents the Runtime software on an HMI device.

Use

You can use the HMIApplication object to access a Runtime application of an HMI device.

Use the "HMIApplication" object as follows:

- "Name" object property
  `HMIApplication.Name`
  Accesses the user-defined name of the Runtime software for an HMI device, e.g. RT_HMI_1.
- "Type" object property
  `HMIApplication.Type`
  Accesses the type of Runtime software, e.g. WinCC RT Advanced.

Note

If your HMI device is a panel, the HMIDevice and HMIApplication objects are the same.

6.1.5 HMIDevice

Description

Represents the HMI device in the project.

Use

You can use the HMIDevice object to access an HMI device in the project.

Use the "HMIDevice" object as follows:

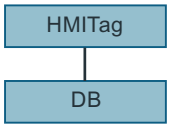
- "Name" object property
  `HMIDevice.Name`
  Accesses the user-defined name of an HMI device, e.g. HMI_1.
- "Type" object property
  `HMIDevice.Type`
  Accesses the type of HMI device, e.g. KTP400.

Note

If your HMI device is a panel, the HMIDevice and HMIApplication objects are the same.

6.1.6 HMITag

Description



Represents the external tag.

Use

You can use the HMITag object to store generated external tags in the project tree in structured form.

Note

Possible applications

You use the HMITag object exclusively in the "Tag rules" editor.

Use the "HMITag" object as follows:

- "FolderPath" object property
  `HMITag.DB.FolderPath`
  Accesses the path of the block in the project tree within the "Program blocks" folder, e.g. "Plant\Plantsection\Productionline"
- "SymbolicName" object property
  `HMITag.DB.SymbolicName`
  Accesses the user-defined name of the data block.

See also

Storage strategies for generated objects (Page 103)

6.1.7 LibraryObject

Description

Represents the screen type in the project library.

Use

You use the LibraryObject object exclusively in the SiVArc properties "Name" and "Screen group" of a generation template for a screen.

- "FolderPath" object property
  `LibraryObject.FolderPath`
  References the path of the screen type in the library If you use the SiVArc expression in the SiVArc property "Screen group", the storage path is created from the library in the project tree. If you use the SiVArc expression in the "Name" property, the generated screen is named after the folder in which the screen type is stored.

  Note

  You can only use this expression under "Name" in reference to a one-level hierarchy in the library. If you would like to you use a multi-level storage hierarchy, you can use the expression `LibraryObject.FolderPath` as substitute for the backslash.

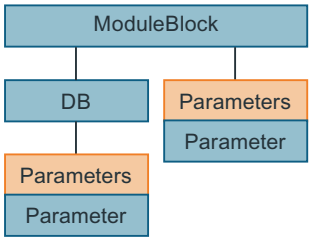
- "Name" object property
  `LibraryObject.Name`
  References the name of the screen type of the library If you use the SiVArc property "Screen group" in the SiVArc expression, the screen is stored in a folder with the name of the screen type in the project tree. If you use the SiVArc expression in the SiVArc property "Name", the screen is named after the screen type.

See also

Storage strategies for generated objects (Page 103)

6.1.8 ModuleBlock

Description



Represents the program block of the second level of the call hierarchy. You can use the ModuleBlock object for absolute addressing of the block of the second level.

Use

You can use the ModuleBlock object to access various properties of the block and the associated data block.

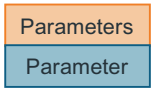
Use the "ModuleBlock" object as follows:

- "FolderPath" object property
  `ModuleBlock.FolderPath`
  Accesses the path of the block in the project tree within the "Program blocks" folder, e.g. "Plant\Plantsection\Productionline"
- "Name" object property
  `ModuleBlock.Name`
  Accesses the internal name of the block, e.g. "FB1".
- "NetworkComment" object property
  `ModuleBlock.NetworkComment`
  Accesses the comment entered in the network of the block.
- "NetworkTitle" object property
  `ModuleBlock.NetworkTitle`
  Accesses the title of the network in which the block is instanced.
- "Number" object property
  `ModuleBlock.DB.Number`
  Accesses the block number in the block properties.
- "Parameters" list
  `ModuleBlock.Parameters("Activate").Value`
  Accesses a block parameter.

- "SymbolComment" object property
  `ModuleBlock.SymbolComment`
  Accesses the user-defined comment in the block properties.
- "SymbolicName" object property
  `ModuleBlock.SymbolicName`
  Accesses the user-defined name of the block.
- "Title" object property
  `ModuleBlock.Title`
  Accesses the header of the block in the block properties.
- "Version" object property
  `ModuleBlock.Version`
  If the block is an instance of a block type, this expression accesses the type version of the block type in the library.

6.1.9 Parameters

Description



The Parameters object is a list of all parameters at the block. The Parameter-Objekt represents a parameter in the specified data block or block.

Use

You can use the Parameters object to access a specific parameter value in the block.

Use the "Parameters" object as follows:

- "Assigned" object property
  `StructureBlock.Parameters("<Name Parameter>").Value`
  Returns TRUE if the parameter is assigned.
- "Comment" object property
  `Parameters("<Name Parameter>").Comment`
  Accesses the comment of the parameter.
- "InitialValue" object property
  `Parameters("<Name Parameter>").InitialValue`
  Accesses the default value of the parameter.
- "Value" object property
  `Parameters("<Name Parameter>").Value`
  Accesses the value of the parameter.

6.1.10 S7Control

Description

Represents the PLC in the project.

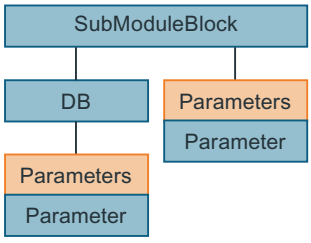
Use

You use the S7Control object to access the name of a PLC:

- "Name" object property
  `S7Control.Name`

6.1.11 SubModuleBlock

Description



Represents the program block of the third level of the call hierarchy. You can use the SubModuleBlock object for absolute addressing of the block of the third level.

Use

You can use the SubModuleBlock object to access various properties of the block and its data block.

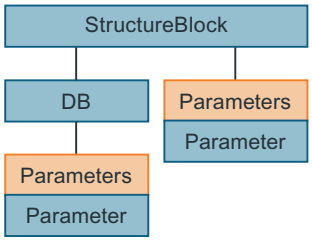
Use the "SubModuleBlock" object as follows:

- "FolderPath" object property
  `SubModuleBlock.FolderPath`
  Accesses the path of the block in the project tree within the "Program blocks" folder, e.g. "Plant\Plantsection\Productionline"
- "Name" object property
  `SubModuleBlock.Name`
  Accesses the internal name of the block, e.g. "FB1".
- "NetworkComment" object property
  `SubModuleBlock.NetworkComment`
  Accesses the comment entered in the network of the block.

- "NetworkTitle" object property
  `SubModuleBlock.NetworkTitle`
  Accesses the title of the network in which the block is instanced.
- "Number" object property
  `SubModuleBlock.DB.Number`
  Accesses the block number in the block properties.
- "Parameters" list
  `SubModuleBlock.Parameters("Activate").Value`
  Accesses a block parameter.
- "SymbolComment" object property
  `SubModuleBlock.SymbolComment`
  Accesses the user-defined comment in the block properties.
- "SymbolicName" object property
  `SubModuleBlock.SymbolicName`
  Accesses the user-defined name of the block.
- "Title" object property
  `SubModuleBlock.Title`
  Accesses the header of the block in the block properties.
- "Version" object property
  `SubModuleBlock.Version`
  If the block is an instance of a block type, this expression accesses the type version of the block type in the library.

6.1.12 StructureBlock

Description



Represents the program block of the first level of the call hierarchy. You can use the StructureBlock object for absolute addressing of the block of the first level.

Use

You can use the StructureBlock object to access various properties of the block and its data block.

Use the "StructureBlock" object as follows:

- "FolderPath" object property
  `SubModuleBlock.FolderPath`
  Accesses the path of the block in the project tree within the "Program blocks" folder, e.g. "Plant\Plantsection\Productionline"
- "Name" object property
  `SubModuleBlock.Name`
  Accesses the internal name of the block, e.g. "FB1".
- "NetworkComment" object property
  `SubModuleBlock.NetworkComment`
  Accesses the comment entered in the network of the block.
- "NetworkTitle" object property
  `SubModuleBlock.NetworkTitle`
  Accesses the title of the network in which the block is instanced.
- "Number" object property
  `SubModuleBlock.DB.Number`
  Accesses the block number in the block properties.
- "Parameters" list
  `SubModuleBlock.Parameters("Activate").Value`
  Accesses a block parameter.
- "SymbolComment" object property
  `SubModuleBlock.SymbolComment`
  Accesses the user-defined comment in the block properties.
- "SymbolicName" object property
  `SubModuleBlock.SymbolicName`
  Accesses the user-defined name of the block.
- "Title" object property
  `SubModuleBlock.Title`
  Accesses the header of the block in the block properties.
- "Version" object property
  `SubModuleBlock.Version`
  If the block is an instance of a block type, this expression accesses the type version of the block type in the library.

6.1.13 TagNaming

Description

Represents the Runtime settings for tags.

Use

You can use the TagNaming object to access the selected replacement delimiter in the Runtime settings for tags for the lower levels of the path of the PLC tag.

Use the "TagNaming" object as follows:

- "SeparatorChar" object property
  `TagNaming.SeparatorChar`
- "IndexStartChar" object property
  `TagNaming.IndexStartChar`
- "IndexEndChar" object property
  `TagNaming.IndexEndChar`

Return values

The "PLC1" controller contains the structured data block "DB1". The "Db1.a[1].b.c[3]" data block element is used in a picture. Depending on your settings, the TagNaming object returns the following values:

Return values	WinCC tag name	Selected Runtime setting
`TagNaming.SeparatorChar` = "." `TagNaming.IndexStartChar` = "[" `TagNaming.IndexEndChar` = "]"	Db1_a[1].b.c[3]	Compatibility mode
	Plc1.Db1.a[1].b.c[3]	PLC prefix
	Db1.a[1].b.c[3]	Delimiter replaced without character selection The tag name is enclosed in quotation marks at the point of use in the screen: "Db1.a[1].b.c[3]"
`TagNaming.SeparatorChar` = ; `TagNaming.IndexStartChar` = "(" `TagNaming.IndexEndChar` = ")"	Db1;a(1);b;c(3)	Replace the period and bracket with ; ( )
`TagNaming.SeparatorChar` = "_" `TagNaming.IndexStartChar` = "{" `TagNaming.IndexEndChar` = "}"	Plc1_Db1_a{10}_b_c{3}	Replace the period and bracket with _ { } PLC prefix

6.2 SiVArc object properties

6.2.1 Assigned

Description

Returns TRUE if there is an assignment at the specified block parameter.

Syntax

<`Object`>`.Assigned`

Object

- Parameter

6.2.2 Comment

Description

Returns the entered comments.

Syntax

<Object>.Comment

Object

- Parameter
- DB

Comment

If you query the comment of a data block, the comment from the block properties is returned.

If you query the comment of a parameter, the comment from the symbol table is returned.

Multiple languages

The SiVArc expression "DB.Comment" can be configured in multiple languages.

See also

Influence of multilingualism on a generation template (Page 101)

6.2.3 FolderPath

Description

Returns the path.

Syntax

<Object>.FolderPath

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block

- DB
- LibraryObject

Comment

If you query the storage path of a program block, the storage path within the "Program blocks" folder is returned.

If you query the storage path of a library object, the storage path within the "Master copies" or "Types" folder is returned.

A "\" is returned as a separator between the folder hierarchy.

See also

Storage strategies for generated objects (Page 103)

6.2.4 HMITagPrefix

Description

Returns the value of the "TagPrefix" property for a screen window.

The "TagPrefix" property, for example, is the name of the associated data block of the program block that SiVArc is currently evaluating.

Syntax

<Object>.HMITagPrefix

Object

- DB

6.2.5 IndexEndChar

Description

Returns the closing bracket set in the Runtime settings when structuring external tags.

Syntax

<Object>.IndexEndChar

Object

- TagNaming

6.2.6 IndexStartChar

Description

Returns the opening bracket set in the Runtime settings when structuring external tags.

Syntax

<Object**>**.IndexStartChar

Object

- TagNaming

6.2.7 InitialValue

Description

Returns the default value of a parameter.

Syntax

<`Object`**>**`.InitialValue`

Object

- Parameter

6.2.8 Name

Description

Returns the internal name, e.g. "FB1"

Syntax

<Object**>**.Name

Object

- S7Control
- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block
- HMIApplication
- HMIDevice

See also

Storage strategies for generated objects (Page 103)

6.2.9 NetworkComment

Description

Returns the network comment.

Syntax

<`Object`**>**`.NetworkComment`

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block

Multiple languages

The "NetworkComment" object property can be configured in multiple languages.

See also

Influence of multilingualism on a generation template (Page 101)

6.2.10 NetworkTitle

Description

Returns the network title.

Syntax

<Object>.NetworkTitle

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block

Multiple languages

The "NetworkTitle" object property can be configured in multiple languages.

See also

Influence of multilingualism on a generation template (Page 101)

6.2.11 Number

Description

Returns the block number.

Syntax

<Object>.Number

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block
- DB

6.2.12 SeparatorChar

Description

Returns the separator character specified in the Runtime settings.

The separator is placed between the lower levels of the path of the PLC tag that are included in the synchronized name of the external tag.

Syntax

<Object>.SeparatorChar

Object

- TagNaming

6.2.13 SymbolComment

Description

Returns the user-defined comment in the block properties.

Syntax

<Object>.SymbolComment

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block
- DB

Multiple languages

The "SymbolComment" object property can be configured in multiple languages.

See also

Influence of multilingualism on a generation template (Page 101)

6.2.14 SymbolicName

Description

Returns the user-defined name of a block or tag.

Syntax

<Object>.SymbolicName

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block
- DB
- HMITag

Comments

If you query the user-defined name of a data block that is called as a multi-instance (MDB), the name of the block stored in the block interface is called. The block name for MDBs is stored under the static local data.

6.2.15 Title

Description

Returns the block title.

Syntax

<Object>.Title

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block

Multiple languages

The "Title" object property can be configured in multiple languages.

See also

Influence of multilingualism on a generation template (Page 101)

6.2.16 Type

Description

Returns the type.

Syntax

`<Object>.Type`

Object

- DB
- HMIApplication
- HMIDevice

Comment

If you query the type of a data block, the type "MDB" (multiple-instance block) or "IDB" (instance block) is returned as a string.

If you query the type of HMI device, the device type is returned as a string, for example, "KTP400".

If you query the type of Runtime software, the type of software is returned as a string, for example, "WinCC RT Advanced".

6.2.17 Value

Description

Returns the value.

Syntax

`<Object>.Value`

Object

- Parameter

6.2.18 Version

Description

Returns the version of a block type.

Syntax

<Object>.Version

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block

Comment

The property is only evaluated when the block SiVArc is currently evaluating is an instance of a block type in the library.

6.3 SiVArc object properties

Access to HMI devices

Using the following tags, you access HMI devices within the project tree.

SiVArc object property	Addressed property
HmiDevice.Name	Name of the HMI device in the project tree For example, "HMI_1", "PC_system_1"
HmiDevice.Type	Type of HMI device in the project tree For example, "KTP700 Mobile", "SIMATIC PC station"
HmiApplication.Name	Name of application For example, "HMI_1", "HMI_RT_40
HmiApplication.Type	Type of application For example, "WinCC RT Advanced", "WinCC RT Professional"
LayoutFieldIndex	Index value of the layout field used during the generation To be used in SiVArc expressions

If the HMI device is a panel, HmiDevice and HmiApplication are identical.

SiVArc object properties for the name of the controller and external tags

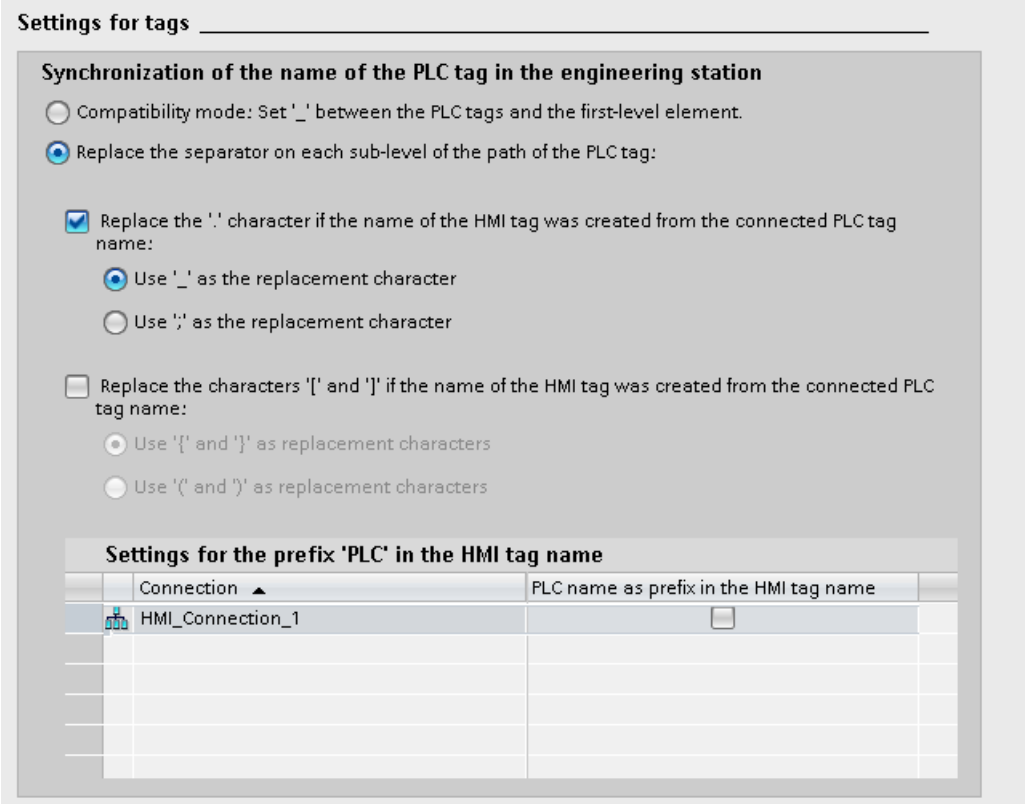
You use the SiVArc object properties `Name` and `SymbolicName` to reference the name of the S7 controller or to generate external tags:

You can only use the SiVArc expressions `HmiTag.SymbolicName` and `HmiTag.DB.SymbolicName` in the "Tag rules" editor.

SiVArc object property	Referenced object	Formulation in the SiVArc expression
`Name`	Name of the S7 PLC	`S7Control.Name`
`SymbolicName`	Name of the external tag (tag name)	`HmiTag.SymbolicName`
`DB.SymbolicName`	Name of the DB	`HmiTag.DB.SymbolicName`
`DB.FolderPath`	Path of the DBs	`HmiTag.DB.FolderPath`

SiVArc object properties for name synchronization of external tags

You define how the names of PLC tags and external tags are to be synchronized in the Runtime settings of the HMI device:



To synchronize the names of external tags according to the settings for tags in the TIA Portal with SiVArc, use `TagNaming` tags.

SiVArc object property	Referenced object	Formulation in the SiVArc expression
`SeparatorChar`	The following separators on each sublevel of the PLC tag path: "." "_" ";"	`TagNaming.SeparatorChar`
`IndexStartChar`	The following separators on each sublevel of the PLC tag path: "[" "(" "{"	`TagNaming.IndexStartChar`
`IndexEndChar`	The following separators on each sublevel of the PLC tag path: "]" ")" "}"	`TagNaming.IndexEndChar`

See also

"Tag rules" editor (Page 26)

6.4 Functions

6.4.1 Functions in SiVArc

In SiVArc the functions listed in the following section are defined.

You can use functions in SiVArc expressions. You cannot change the function names.

6.4.2 "Contains" function

`Contains` function

The `Contains` function determines whether character string is contained in another string. The function is case sensitive and space sensitive.

Function	Result
`Contains("ButtonText", "Text")`	True
`Contains("ButtonText", "ttonT")`	True
`Contains("ButtonText", "butt")`	False
`Contains("ButtonText", "txeT")`	False
`Contains("ButtonText", "Text")`	False
`Contains("ButtonText", "Text ")`	False
`Contains("ButtonText", "Te xt")`	False
`Contains("ButtonText", "on")`	False
`Contains("ButtonText 1", "ButtonText 2")`	False

6.4.3 "EndsWith" function

`EndsWith` function

The `EndsWith` function determines whether the end of a character string matches a specified character string. The function is case sensitive and space sensitive.

Function	Result
`EndsWith("ButtonText", "Text")`	True
`EndsWith("ButtonText", "ButtonText")`	True
`EndsWith("ButtonText", "butt")`	False
`EndsWith("ButtonText", "Butt")`	False
`EndsWith("ButtonText", "Text")`	False
`EndsWith("ButtonText", "Text ")`	False
`EndsWith("ButtonText", "Te xt")`	False
`EndsWith("ButtonText", "t")`	True
`EndsWith("ButtonText", "T")`	False
`EndsWith("ButtonText ", "Text")`	False
`EndsWith("ButtonText 1", "ButtonText 2")`	False

6.4.4 "Format" function

`Format` function

The `Format` function returns a formatted string. Statements within a format string specify the form in which the string is returned.

The function has two function parameters:

- String that is returned formatted.
- Format string that specifies the formatting of the string.
  Use the format string "b" to display the result as binary code. If the result of an expression is a floating-point number, the result is displayed rounded in binary format.

Function	Result
`Format(5,"0.00")`	5.00
`Format((VAR_1 Or 2#11100), "b")`	2#11101

You can find more information on the format string by searching for "Strings.Format method" in the Microsoft Developer Network.

6.4.5 "FormatNumber" function

`FormatNumber` function

The `FormatNumber` function returns a string formatted as a number.

The function has five function parameters:

Position	Parameters	Description	Notes
1	Expression	String that is returned formatted as a number	If the string cannot be formatted as a number (e.g. "hello world"), an error is displayed.
2	NumberofDigitsAfterDecimalPoint	Number that specifies how many decimal places are displayed to the right of the decimal separator. `FormatNumber("12,4", 3,-2,-2,-2) ("12 comma 4")`= 12,400	The default value -1 specifies that the country settings of the computer are used.
3	ApplyLeadingNumber	Number that specifies whether or not a leading 0 is displayed for fractions. `FormatNumber("0,4", 3,-1,-2,-1)` = 0,400	The possible settings are listed under "List of constants".
4	UseHigherLevelAsNegativeNumbers	Number that specifies whether or not negative values are displayed in brackets. `FormatNumber("-12",1,-2,-1,0)` = (12,0)	If the number is displayed in brackets, the minus sign is not shown. The possible settings are listed under "List of constants".
5	GroupNumbers	Number that specifies whether or not high numbers are grouped with the thousands separator. `FormatNumber("1288,4", 3,-2,-2,0)`= 1288,400	The form of the thousands separator (e.g. point, comma or space) is defined in the country settings of the computer. The possible settings are listed under "List of constants".

List of constants

Table 6-1 ApplyLeadingNumber

ApplyLeadingNumber	Value	
TRUE	-1	Display leading 0.
FALSE	0	Do not display leading 0.
UseDefault	-2	Use country settings of the computer.

Table 6-2 UseHigherLevelAsNegativeNumbers

ApplyLeadingNumber	Value	
TRUE	-1	Display negative values in brackets. The minus sign is not shown.
FALSE	0	Output negative values without brackets. The minus sign is shown.
UseDefault	-2	Use country settings of the computer.

Table 6-3 UseHigherLevelAsNegativeNumbers

ApplyLeadingNumber	Value	
TRUE	-1	Group numbers with thousands separator.
FALSE	0	Do not group numbers with thousands separator.
UseDefault	-2	Use country settings of the computer.

Examples

The table below applies to settings for Germany. The thousands separator for Germany is the point and the decimal separator is the comma.

Function	Result
`FormatNumber("12,4",3,-2,-2,-2) ("12 comma 4")`	12.400
`FormatNumber("12.4",3,-2,-2,-2) ("12 point 4")`	124.000
`FormatNumber("1288,4",3,-2,-2,-1)`	1.288,400
`FormatNumber("1288,4",3,-2,-2,0)`	1288.400
`FormatNumber("-12",1,-2,-2,0)`	-12.0
`FormatNumber("-12",1,-2,-1.0)`	(12.0)

6.4.6 Function "InStr"

Function `InStr`

The `InStr` function checks whether a string is completely contained in another string. This is case sensitive. The function returns a Boolean value ("True" or "False").

The function has two function parameters:

- String in which the check is performed.
- String that contains the compared text.

The following examples show the values that the `InStr` function produces:

Function	Result
`InStr("Hello","Hello")`	True
`InStr("Hello","hello")`	False

Function	Result
`InStr("Hello","el")`	True
`InStr("12345",3)`	True
`InStr("12345","6")`	False

6.4.7 Function "IsDefined"

`IsDefined` function

Using a string as a parameter, the `IsDefined` function checks whether there is a tag with a name matching the specified string.

You can use this function for the following syntax elements:

- SiVArc tags
- SiVArc object property
- Arrays of "String" data type

Example: You have created the following user-defined tag:

`ButtonText "Cycle_time"`

Function	Result
`IsDefined("ButtonText")`	True
`IsDefined("ButtonText[0]")`	True
`IsDefined("ButtonText[1]")`	True
`IsDefined("ButtonText[2]")`	False

6.4.8 Function "LBound"

`LBound` function

The `LBound` functions expects an array as a parameter and returns the smallest possible index.

Function	Result
`LBound(Split("SG19_FG97_ST090", "_"))`	0
`LBound(Split("SG19_FG97", "_"))`	0

6.4.9 Function "Left"

`Left` function

The `Left` function returns a string containing a specified number of characters from the leftmost characters of a string.

The function has two function parameters:

- String from which a substring is returned.
- Number indicating the character length of the substring
  If the number is 0, an empty string is returned.
  If the number is greater than the number of characters in the specified string, an error is displayed.

Function	Result
`Left("ButtonText", 6)`	"Button"
`Left("ButtonText", 0)`	"" (Empty string)
`Left("ButtonText", "10")`	"ButtonText"
`Left("ButtonText", 11)`	Error (Number is greater than the number of characters in the string)

6.4.10 Function "Len"

`Len` function

The `Len` function returns the number of characters in a string. The function expects a string as a function parameter.

Function	Result
`Len("ButtonText")`	10
`Len("")`	0
`Left("ButtonText", Len("ButtonText"))`	"ButtonText"

6.4.11 Function "LTrim""

`LTrim` function

The `LTrim` function removes leading spaces from a string. The function expects a string as a function parameter.

Function	Result
`LTrim ("  ButtonText")`	"ButtonText"
`LTrim ("ButtonText")`	"ButtonText"

6.4.12 Function "Max""

`Max` function

The `Max` function expects two numbers as a parameter and returns the higher of the two.

Function	Result
`Max(12, 3)`	12
`Max(3, 123)`	123

6.4.13 Function "Mid"

`Mid` function

The `Mid` function returns a substring within a string from a specified position.

The function has three function parameters:

- String from which the substring is copied.
- Number indicating the starting position in the string.
  If the starting position is greater than the number of characters in the string, an error is displayed.
- Number indicating the length of the substring from the starting position.
  If the specified length is greater than the longest possible substring length from the starting position in the string, an error is displayed.

Function	Result
`Mid("ButtonText", 5,3)`	"nTe"
`Mid("ButtonText", 0, 10)`	"ButtonText"

Function	Result
`Mid("ButtonText", 10, 3)`	Error (Starting position is greater than the number of characters in the string)
`Mid("ButtonText", 7, 10)`	Error (Specified length is greater than the longest possible substring from position 7)

6.4.14 Function "Min"

`Min` function

The `Min` function expects two numbers as a parameter and returns the smaller of the two.

Function	Result
`Min(12, 3)`	3
`Min(3, 123)`	3

6.4.15 Function "Replace"

`Replace` function

The `Replace` function searches a string from left to right for a substring and replaces the substring with another substring. The search function is case sensitive. The changed string is returned.

The function has three function parameters:

- String in which a substring is found and replaced.
- String indicating the substring to be found.
  If the substring to be found is an empty string, the string first transferred is returned unchanged.
- String inserted in place of the substring found.

The find and replace function continues after the new substring.

Function	Result
`Replace("ButtonText", "Text", "Button")`	"ButtonButton"
`Replace("ButtonText", "ButtonText", "Hello World")`	"Hello World"
`Replace("aaa", "aa", "bb")`	"bba"
`Replace("a", "a", "a")`	"a"

Function	Result
`Replace("a", "", "b")`	"a"
`Replace("aA", "a", "b")`	"bA"

6.4.16 "Right" function

`Right` function

The `Right` function outputs a substring from the rightmost character of a string. The number of characters is specified when the function is called.

The function has two function parameters:

- String from which a substring is generated and returned.
- Number specifying the number of rightmost characters that is returned.
  If the number is 0, an empty string is returned.
  If the number is greater than the number of characters in the string, an error is displayed.

Function	Result
`Right("ButtonText", 4)`	"Text"
`Right("ButtonText", 0)`	"" (Empty string)
`Right("ButtonText", 10)`	"ButtonText"
`Right("ButtonText", 11)`	Error (Number is greater than the number of characters in the string)

6.4.17 Function "RTrim"

`RTrim` function

The `RTrim` function removes spaces from the end of a string. The resulting string is returned.

If there are no spaces at the end of the string, the string is returned unchanged.

Function	Result
`RTrim("ButtonText  ")`	"ButtonText"
`RTrim("ButtonText  ")`	"ButtonText"

6.4.18 "Split" function

`Split` function

The `Split` function splits a string. The delimiter required for this is freely definable.

The function has two function parameters:

- String
- Delimiters

Depending on the syntax, a substring is returned or the number of contained substrings:

- Substring as a return value
  `Split("<String>","<Separator>")(<Index>)`
  You reference the substring through an index that starts with zero.
- Number of contained substrings as the return value
  `Split("<String>","<Separator>").Length`

The following examples show the numerical values that the `Split` function produces:

Function	Result
`Split("SG19_FG97_ST090","_")(0)`	SG19
`Split("SG19.FG97.ST090",".")(1)`	FG97
`Split("42",".")(0)`	42
`Split(".",".")(0)`	"" (Empty string)

The following examples show the number of substrings that the `Split` function produces:

Function	Result
`Split("SG19_FG97_ST090", "_").Length`	3
`Split("SG19.FG97.ST090", ".").Length`	3

6.4.19 "StartsWith" function

`StartsWith` function

The `StartsWith` function determines whether the start of a character string matches a specified character string. The function is case sensitive and space sensitive.

Function	Result
`StartsWith("ButtonText", "Butt")`	True
`StartsWith("ButtonText", "butt")`	False
`StartsWith("ButtonText", "Text")`	False
`StartsWith("ButtonText", "ButtonText")`	True
`StartsWith("ButtonText", " Butt")`	False
`StartsWith("ButtonText", "Butt ")`	False

Function	Result
StartsWith("ButtonText", "Bu tt")	False
StartsWith("ButtonText", "B")	True
StartsWith("ButtonText", "b")	False
StartsWith(" ButtonText", "Butt")	False
StartsWith("B uttonText", "Butt")	False
StartsWith("ButtonText 1", "ButtonText 2")	False

6.4.20 "StrComp" function

StrComp function

The StrComp function compares two strings. The function sorts the string alphanumerically starting with the first character, and is case-sensitive. A number is returned on the basis of the sorting of the strings.

The following cases are possible:

- The first string is placed before the second string. The return value is -1.
  StrComp("ABCD", "Abcd") = -1
  StrComp("A", "a")= -1 ("A" comes before "a" in the alphabet)
- The second string is placed before the first string. The return value is 1.
  StrComp("ABCD", "AAcd")= 1
- The two strings are identical. The return value is 0.
  StrComp("Abcd", "Abcd")= 0

6.4.21 "TrailNum" function

TrailNum function

The TrailNum function returns the last positive numerical value from a string, for example, the number in the name of a program block.

The following examples show the numerical values that the TrailNum function produces:

Function	Result
TrailNum("42")	42
TrailNum("Anzahl42")	42
TrailNum("Anzahl0042")	42
TrailNum("Anzahl-42")	42
TrailNum("Minimum42_Maximum84")	84

The following examples show the use of the TrailNum function in a SiVArc expression.

A function block with the symbolic name "SG19_FG97_ST090+IR001_FB" is programmed in the TIA Portal.

SiVArc expression	Result
`"MyBlock_"&TrailNum(ModuleBlock.SymbolicName)`	"MyBlock_1"
`"MyBlock_"&TrailNum(ModuleBlock.SymbolicName[0])`	"MyBlock_19"

If you do not specify string indexing, the last number in the string value is output.

6.4.22 "Trim" function

`Trim` function

The `Trim` function removes all spaces from the start and end of a string. The resulting string is returned.

If there are no spaces either at the start or at the end of the string, the string is returned unchanged.

Function	Result
`Trim("ButtonText")`	"ButtonText"
`Trim("ButtonText")`	"ButtonText"
`Trim("ButtonText")`	"ButtonText"
`Trim("ButtonText")`	"ButtonText"

6.4.23 "UBound" function

`UBound` function

The `UBound` functions expects an array as a parameter and returns the largest possible index.

Function	Result
`UBound(Split("SG19_FG97_ST090", "_"))`	2
`UBound(Split("SG19_FG97, "_"))`	1
`UBound(Split("", "."))`	0

6.5 Operators

You can use the following operators in SiVArc expressions.

Note that operators are case-sensitive. On the one hand, this relates to the operators themselves with logical and bitwise operators. On the other hand, you must consider the case of strings set in the relation with comparison operators, for example, when you compare two strings to check for identical names.

Arithmetic operators

Arithmetic operator	Example	Result
+	4+2	6
-	4-2 -4+2	2 -2
*	4*2	8
/	4/2	2

Relational operators

Relational operators	Example	Result
=	4=4 4=2	True False
<> ("different than")	4<>4 4<>2	False True
>	4>2 2>4	True False
>=	4>=2 4>=4	True True
<	4<2 2<4	False True
<=	4<=2 4<=4	False True

Logic operators

Logic operators	Example	Result
And	True And True True And False False And False	True False False
Or	True Or True True Or False False Or False	True True False
Not	Not True Not False	False True

Bit-by-bit operators

Bit-by-bit operators	Example	Result
And	16 And 16	16
Or	8 Or 4	12
Xor	3 Xor 1	2

Operators for string sequences

Concatenation operator	Example	Result
`&`	`"Tool"&"Bar"`	ToolBar

Priority of the operators

The following table indicates the priority with which operators are processed when you use multiple operators in a SiVArc expression. 1 has the highest priority.

Operator	Not - (unary)	*, /	+, -	&	=, <> >, >= <, <=	And	Or	Xor
Priority	1	2	3	4	5	6	7	8

Use parentheses to change the processing order.

6.6 String indexing

Use

Substrings with a string are separated by the _ character. To access a substring, use the indexing operator `[]`.

The counting of the substring starts at `0`. You can access the substring via the number in the indexing operator.

Example

The "FB_Name" tag is defined with the value "SG19_FG97_ST090+IR001_FB" in the TIA Portal.

String indexing in the SiVArc expression	Result
`FB_Name[0]`	SG19
`FB_Name[1]`	FG97
`FB_Name[2]`	ST090+IR001
`FB_Name[3]`	FB

6.7 If conditions

You formulate logical conditions in SiVArc expressions with the If operator.

If operator

The If operator has the following syntax:

```
If(<condition>, <thenExpression>, <elseExpression>)
```

`<condition>`	Boolean or integer
`<thenExpression>`	is produced when `<condition>` is either `True` or an integer value other than `0`
`<elseExpression>`	is produced when `<condition>` is either `False` or `0`

You can also nest the conditions and use an If condition in another If condition.

Examples

If condition	Result
`If(True, "On", "Off")`	On
`If(0, "On", "Off")`	Off
`If(42, "On", "Off")`	On
`If(4>2, If(False, 4, 2), 42)`	2

6.8 Supported data types for PLC tags

SiVArc supports all basic data types that can be displayed on the HMI device by the PLC in WinCC V13.1.

SiVArc also supports the structured data types ARRAY, STRUCT and UDT.

Basic data types

Name	Data type
BOOL	Boolean value
BYTE	Binary and hexadecimal numbers with 8 bits
CHAR	ASCII character
DINT	Double integer, integer with sign
DTL	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)
DWORD	Binary and hexadecimal numbers with 32 bits
DATE	IEC date in increments of 1 day
DATE_AND_TIME	Date and time (Year-Month-Day-Hour:Minute:Second; Fixed point number)
INT	Integer, integer with sign
LDT	Date and time (Year-Month-Day-Hour:Minute:Second)
LINT	
LREAL	
LTIME	
LTIME_OF_DAY	

Name	Data type
LWORD	
REAL	Real numbers (IEEE floating-point number)
S5TIME	Time period in S5T# format, Step7 time in increments of 10 ms
SINT	
STRING	Character string
TIME	Time period in IEC format, IEC time in increments of 1 s, integer with sign
TIME_OF_DAY	Time of day in increments of 1 ms
UDINT	
UINT	
ULINT	
USINT	
WORD	Binary and hexadecimal numbers with 16 bits
WString	Unicode character string with variable length
WChar	Unicode characters with 16 bits

Structured data types

SiVArc supports structured PLC tags and all associated elements that have been released for WinCC. During the generation, SiVArc creates structured external tags and elements according to the PLC tag. Tags and elements are automatically connected to the PLC tags and their elements.

Name	Data type
ARRAY	Array
ARRAY DBs	
ARRAY DB STRUCT	
STRUCT	Structure
UDT	User Defined Data Type (PLC data type)

Note

Condition for PLC data types (UDTs)

If a PLC data type is an array of a structured data type (STRUCT or UDT), SiVArc breaks down the array into individual tags of this data type in WinCC. If a PLC data type contains arrays of structured data types as elements, these are shown as structured elements in the "HMI tags" editor.

6.9 Supported system functions for faceplates

System functions

Depending on the HMI device for which it is generated, use the following system functions at SiVArc events:

System function	RT Advanced	RT Professional
ActivateScreen	x	x
DecreaseTag	x	x
IncreaseTag	x	x
InvertBit	x	x
InvertBitInTag	x	x
SetBit	x	x
SetBitInTag	x	x
SetTag	x	x
ResetBit	x	x
ResetBitInTag	x	x
ActivateScreenInScreenWindow	---	x
ActivatePreviousScreen	x	---
ShiftAndMask	x	---

6.10 Editing the view in the SiVArc editors

Introduction

You can filter and sort SiVArc rules in the editor or in the generation overview without affecting the order of generation. If necessary, store the new layout until the next start of the TIA Portal. You can also group the view by columns in all SiVArc editors. The filter functions are deactivated in this case.

While the list is being filtered or sorted, you can continue editing the SiVArc rules or create new rules. The active filter criteria are applied to new and edited rules.

Note

New rules in the filtered editor

If you create a new rule in the filtered editor, the new rule is a copy of the rule displayed at the lowest position. If the list is filtered by the name of the SiVArc rules, the new SiVArc rule is not displayed.

Filtering contents of editors for the view

When the "Group" button is deactivated, you can filter the contents of the editors.

To filter SiVArc rules in the editor, follow these steps:

1. Click the "Filter" button in the toolbar of the editor.
   A filter line is displayed below the header of the editor.
2. Open the selection dialog in the filter cell of the required column.
3. In the selection dialog, select the objects that you want to display in the editor.
   The rules are filtered according to your selection.

To hide the filter line, click the "Filter" button again.

Sorting contents of the editors for the view

When the "Group" button is deactivated, you can sort the contents of the editors.

You can also re-sort SiVArc rules while the list is displayed filtered and vice versa.

To sort SiVArc rules in the editor, follow these steps:

- Click the column header according to which you want to sort the display.
  The display is sorted by the selected column in descending alphabetical order. When the rule editor contains groups, the rules within the groups are also sorted according to this column.

Saving sorting and filter

To retain the filter or the sorting of the rules until the next start of the TIA Portal, follow these steps:

- Click the "Save window settings" button in the toolbar of the editor.
  When the TIA Portal is opened the next time, the SiVArc rules are arranged and filtered as they were the last time.

Regrouping the display

When the display is opened for the first time, the contents are shown grouped according to the first column.

To regroup the contents in the editor, follow these steps:

1. To activate the grouping function, click the "Group" button.
   The "Group" button is displayed pressed.
2. Click the column heading for whose content you want to group the display.
   All SiVArc rules or SiVArc objects are grouped according to the content of the selected column in the display.

6.11 Picture legends

Overview

Symbols and abstracted displays are used in the illustrations of the SiVArc documentation as follows:

Symbol/Display	Description
	Function block
	Instance in OB1 (call in OB1)
	Network in OB1
	Type of a function block (Library type)
	Instance of a function block type (Library type)
	Data block
	SiVArc configuration
	SiVArc expression that accesses a data block
	SiVArc expression that accesses a function block
	SiVArc expression that accesses a network entry

SiVArc error messages

7

7.1 Reference to alarms

7.1.1 Critical errors

7.1.1.1 CriticalError_ObsoleteFbTypeVersionFound

ID	CriticalError_ObsoleteFbTypeVersionFound
Cause	The version of the program block in the rule does not correspond to that of the block in the STEP 7 program.
Solution	Check the version number of the called program block.

7.1.1.2 CriticalError_ScreenMastercopyUsedAsScreenTypeAndObject

ID	CriticalError_ScreenMastercopyUsedAsScreenTypeAndObject
Cause	The utilized master copy of the screen is used as screen type as well as screen object in the screen rules.
Solution	Make sure that a master copy of a screen which is to be used as screen object is not used as screen type in any rule.

7.1.1.3 CriticalError_VersionforTiaTypeLibraryTypeInWork

ID	CriticalError_VersionforTiaTypeLibraryTypeInWork
Cause	The library type used by SiVArc is being edited.
Solution	Open the respective library type and release the current version or discard it.

7.1.2 Error

7.1.2.1 Error_AlarmAsMasterCopyGroupNotSupported

ID	Error_AlarmAsMasterCopyGroupNotSupported
Cause	The alarm object is part of master copy group, and due to this rule is not executed.
Solution	Alarm objects present in the master copy group are not supported by SiVArc for generation. Store the alarm object as a single master copy.

7.1.2.2 Error_AlarmCanNotBeCreated

ID	Error_AlarmCanNotBeCreated
Cause	The alarm object cannot be created. SiVArc cannot rename the system created default alarm classes and groups.
Solution	SiVArc cannot rename the system created default alarm classes and groups, therefore use the existing system created alarm object or set the SiVArc "Name" property of the HMI object in the library.

7.1.2.3 Error_AlarmTypeNotFound

ID	Error_AlarmTypeNotFound
Cause	The "Copy template of an alarm object" in the SiVArc alarm rule does not exist in the project library or a global library.
Solution	1. Open the SiVArc editor "Alarm rules". 2. Enter an alarm object that was created in one of the libraries in the affected line in the "Master copy of alarms/classes/groups" column. 3. Double-click on the relevant line in the "Master copy of alarms/classes/groups" column to make the available objects appear. 4. The available selectable objects are displayed and can be selected.

7.1.2.4 Error_AlarmTypeNotSupported

ID	Error_AlarmTypeNotSupported
Cause	This error is displayed if the master copy configured in the alarm rules is not of type "HmiDiscreteAlarm/HmiAnalogAlarm/HmiAlarmClass/HmiAlarmGroup".
Solution	The master copy used in the alarm rules need to be a "HmiDiscreteAlarm/HmiAnalogAlarm/HmiAlarmClass/HmiAlarmGroup" type.

7.1.2.5 Error_CanNotParseOverflowScreenCount

ID	Error_CanNotParseOverflowScreenCount
Cause	The value of the expression in the SiVArc property "Number of overflow screens" is not valid. For example, "one" instead of "1".
Solution	1. Correct the expression in the SiVArc property "Number of overflow screens" for the relevant screen template. 2. The expression must return a positive integer.

7.1.2.6 Error_CanNotResolveOverflowScreenCount

ID	Error_CanNotResolveOverflowScreenCount
Cause	The value of the expression in the SiVArc property "Number of overflow screens" is not valid. For example, "-1" instead of "1".
Solution	1. Correct the expression in the SiVArc property "Number of overflow screens" for the relevant screen template. 2. The expression must return a positive integer.

7.1.2.7 Error_ConflictAlarmRule

ID	Error_ConflictAlarmRule
Cause	Similar alarm object cannot be created from "Copy rules" editor and "Alarm rules" editor.
Solution	Alarm object generated from one editor should be deleted before generating from other editor.

7.1.2.8 Error_ConflictCopyRule

ID	Error_ConflictCopyRule
Cause	This object has been modified by another editor in SiVArc, for example, by the "Screen rules", "Tag rules" or "Text list rules" editor. That is why this object cannot be modified by the "Library rules" editor.
Solution	Disable or remove the respective rule that is causing the conflict from the affected editor.

7.1.2.9 Error_ConflictsBetweenFaceplatesInLibraries

ID	Error_ConflictsBetweenFaceplatesInLibraries
Cause	In the SiVArc screen rule editor, a faceplate type from a global library is referenced in a SiVArc screen rule which is also contained in a directory in the project library.
Solution	If a faceplate type is present in both the global and the project library, it must be stored with same path in both libraries. Drag the faceplate type either into the project library or the global library so that both have the same path.

7.1.2.10 Error_DuplicateAlarmObjectCanNotBeCreated

ID	Error_DuplicateAlarmObjectCanNotBeCreated
Cause	The duplicate alarm class/group cannot be created because alarm name should be unique for alarm classes and groups.
Solution	SiVArc cannot create duplicate alarm classes/groups because name should be unique for alarm classes and groups. Set the SiVArc "Name" property with different name of the HMI object in the library.

7.1.2.11 Error_DuplicateAlarmRule

ID	Error_DuplicateAlarmRule
Cause	The values defined in the current alarm column is identical to one of the other rules. The "Comment" column is not considered for duplicate rules.
Solution	Open the SiVArc "Alarm rules" editor, and delete the duplicate rule, since the rule is not considered for the generation.

7.1.2.12 Error_ContentScreenCannotGenerate

ID	Error_ContentScreenCannotGenerate
Cause	One of the master copies is used for a screen, the other is used as a reference for a screen window. In this case, the screen referenced by the screen window is not generated.
Solution	In the "Screen objects" column, do not use a screen the name of which is identical to another screen name during the generation.

7.1.2.13 Error_DifferencScriptSignature

ID	Error_DifferencScriptSignature
Cause	The referenced script of the HMI object has invalid parameters.
Solution	1. Check the signature of the script called. 2. For the correct script signature, see the script definition in the WinCC project under "Scripts".

7.1.2.14 Error_DuplicatedScreenItemFoundFromMultiPlc

ID	Error_DuplicatedScreenItemFoundFromMultiPlc
Cause	A screen object with the same name already exists due to the generation of another controller in the same screen. The screen object cannot be created again in the same screen.
Solution	1. Check your screen rules. 2. Make sure that different controllers do not generate the same screen objects on the same screens. 3. Correct either the screen rules or the SiVArc property "Name" of the screen objects. Alternative procedure 1. Disable the controller prior to generation.

7.1.2.15 Error_DuplicateCopyRule

ID	Error_DuplicateCopyRule
Cause	The values that have been set in the respective columns for copy rules are identical to the values of another rule. The duplications in the "Comment" column are ignored by this check.
Solution	Open the "Copy rules" editor in SiVArc and delete the rules assigned twice because these rules are otherwise ignored for the generation.

7.1.2.16 Error_DuplicatedTextListEntryFoundFromMultiPLC

ID	Error_DuplicatedTextListEntryFoundFromMultiPLC
Cause	A text list entry with the same name already exists due to the generation of another controller in the text list. The text list entry cannot be created again in the text list.
Solution	1. Check the text definitions for the faulty text list entries in the STEP 7 program. 2. Make sure that different controllers do not generate the same text list entries in the same text lists.

7.1.2.17 Error_DuplicateScreenRule

ID	Error_DuplicateScreenRule
Cause	The values that have been set in the respective columns for screen rules are identical to the values of another rule. The duplications in the "Comment" column are ignored by this check.
Solution	Open the "Screen rules" editor in SiVArc and delete the rules assigned twice because these rules are otherwise ignored for the generation.

7.1.2.18 Error_DuplicateTextlistRule

ID	Error_DuplicateTextlistRule
Cause	The values that have been set in the respective columns for text list rules are identical to the values of another rule. The duplications in the "Comment" column are ignored by this check.
Solution	Open the "Text list rules" editor in SiVArc and delete the rules assigned twice because these rules are otherwise ignored for the generation.

7.1.2.19 Error_EventCreationFailedDueToErrorInExpression

ID	Error_EventCreationFailedDueToErrorInExpression
Cause	The master copy of a screen object has an error in the expression of the configured parameters. The function is ignored during the SiVArc generation.
Solution	Ensure that the expressions for the event parameters are correct.

7.1.2.20 Error_EventCreationFailedDueToVariableNotDef

ID	Error_EventCreationFailedDueToVariableNotDef
Cause	The master copy of the screen object was configured with an undefined SiVArc tag as an expression for the system function or user script parameters. The function is ignored during the SiVArc generation.
Solution	Ensure that the configured SiVArc tags have been defined in the SiVArc STEP 7 plug-in editor before you use them as function parameters for SiVArc events.

7.1.2.21 Error_EventExceedsMaxFunctionCalls

ID	Error_EventExceedsMaxFunctionCalls
Cause	An event is assigned with multiple function calls in the master copy of a HMI object. The number of function calls exceeds the maximum permitted number of generated functions in the higher-level device.
Solution	The number of function calls in an event is not allowed to exceed the number of supported function calls in the higher-level device. The same number or fewer function calls are permitted on the higher-level device. All panels support 16 function calls in an HMI object.

7.1.2.22 Error_EventNotSupported

ID	Error_EventNotSupported
Cause	The configured event for the master copy of the HMI object was configured on a device of another device series. The configured event is not supported for the HMI object on the device on which it was generated. For example, the "Press left mouse button" event for the "Button" HMI object is only available in Professional RT, but not in RT Advanced. If the master copy of the button in RT Professional is assigned the "Press left mouse button" event and this master copy was used in RT Advanced, this error message is output by SiVArc.
Solution	If you are planning cross-device generation, configure only the events for the master copies of the HMI objects that are supported by all devices.

7.1.2.23 Error_ExceptionMessage_Debug

ID	Error_ExceptionMessage_Debug
Cause	A SiVArc internal error has occurred.
Solution	Please contact Support.

7.1.2.24 Error_FaceplateCanNotCreate

ID	Error_FaceplateCanNotCreate
Cause	A general error occurred during creation of the faceplate type. Possible causes: • Error updating the project library with a faceplate type from the global library. • Error during instantiation of the faceplate type
Solution	1. Delete the faceplate type in the respective library. 2. Create a new faceplate type.

7.1.2.25 Error_FailedToExportHmiOmToCoreBlob

ID	Error_FailedToExportHmiOmToCoreBlob
Cause	An internal SiVArc error occurred during serialization/export of the HMI object model.
Solution	1. Close the project without saving. 2. Open the project again. Notice: All unsaved changes will be lost and must be made again.

7.1.2.26 Error_FbLibraryTypeNotFound

ID	Error_FbLibraryTypeNotFound
Cause	In the rule editor, a SiVArc rule references a PLC program block type from a library which does not exist.
Solution	In the rule editor, select an existing PLC program block type from the project library or global library for the SiVArc rule.

7.1.2.27 Error_FolderPathTooLong

ID	Error_FolderPathTooLong
Cause	The overall length of the path of the specified group exceeds 128 characters. The requested screen is created in the main folder.
Solution	Reduce the path length in the SiVArc "Screen group" property of the screen to a maximum of 128 characters to create the screen in the required group.

7.1.2.28 Error_FolderPathTooLong_Tag

ID	Error_FolderPathTooLong_Tag
Cause	The total length of the path of the specified group exceeds 128 characters. The desired HMI tag is therefore created in the main folder.
Solution	Decrease the entry in the "Tag group hierarchy" column in the tag rules so that the resulting path length has max. 128 characters.

7.1.2.29 Error_FunctionFailed

ID	Error_FunctionFailed
Cause	The function called in the event does not exist or there are errors in its definition. For example, user-defined script
Solution	Define a new function. Correct the faulty definition.

7.1.2.30 Error_FunctionIsNotAllowed

ID	Error_FunctionIsNotAllowed
Cause	The function called is not valid for the HMI object in question.
Solution	Change the called function in the "SiVArc events" editor.

7.1.2.31 Error_FunctionIsNotAllowedSystemFunction

ID	Error_FunctionIsNotAllowedSystemFunction
Cause	The called system function is not supported for the respective HMI object in the utilized HMI device. The "ActivateScreenInScreenWindow" system function, for example, is not supported in HMI devices of the type WinCC RT Advanced.
Solution	Correct the function or enter a valid function for the HMI object.

7.1.2.32 Error_FunctionNameInvalid

ID	Error_FunctionNameInvalid
Cause	• The function name is not supported. • The name is empty.
Solution	Check the function name and enter a new name.

7.1.2.33 Error_GroupGenerationFailed

ID	Error_GroupGenerationFailed
Cause	The desired tag group was not generated during generation of tags. This can be caused by internal problems in the TIA Portal. In this case, the TIA Portal cannot provide the desired objects or the required service to SiVArc.
Solution	Try to generate it again, or add a tag group manually.

7.1.2.34 Error_HierarchicalLayoutScreen_EmptyValue

ID	Error_HierarchicalLayoutScreen_EmptyValue
Cause	The positioning scheme is configured with reference to another positioning scheme, but one of the positioning scheme properties is not set or empty. Therefore, the rule is not applied.
Solution	1. Open the editor of the affected master copy of the positioning scheme in SiVArc. 2. Configure valid properties for the positioning scheme: – If "Static" mode is set under "Layout selection", select the "Layout screen or folder" option by selecting the master copy that is available in one of the libraries. – If "Dynamic" mode is set under "Layout selection", select the "Layout screen or folder" option by selecting the library folder that contains the master copies of the positioning scheme and is available in one of the libraries. Assign to the SiVArc expression the property "Expression for layout screen name" which can lead to one of the names of the master copies of the layout screen contained in the folder. All available screens and folders are displayed when you click the object selection under "Layout screen or folder".

7.1.2.35 Error_HmiDeviceTypeToChangeNotSupported

ID	Error_HmiDeviceTypeToChangeNotSupported
Cause	In its current version, SiVArc does not support switching the HMI device type.
Solution	Reset the original HMI device type.

7.1.2.36 Error_InconsistentAlarmRuleNoAlarmType

Quintessence

ID	Error_InconsistentAlarmRuleNoAlarmType
Cause	The "Copy template of an alarm object" in the SiVArc alarm rule does not exist in the project library or global library.
Solution	1. Open the SiVArc editor "Alarm rules". 2. Enter an alarm object that was created in one of the libraries in the affected line of the "Master copy of alarms/classes/groups" column. 3. Double-click the corresponding line in the "Master copy of alarms/classes/groups" column for the available images to appear. The available images that can be selected are displayed.

7.1.2.37 Error_InconsistentAlarmRuleNoFbType

ID	Error_InconsistentAlarmRuleNoFbType
Cause	The "Program block" in the SiVArc alarm rule does not exist in the STEP 7 program.
Solution	1. Open the SiVArc editor "Alarm rules" 2. Enter an alarm object that was created in one of the libraries in the affected line in the "Master copy of alarms/classes/groups" column. 3. Double-click the corresponding line in the "Master copy of alarms/classes/groups" column to make the available images appear. The available images that can be selected are displayed.

7.1.2.38 Error_InconsistentCopyRuleNoLibraryItem

ID	Error_InconsistentCopyRuleNoLibraryItem
Cause	The definition of the copy rule is invalid. The column with the library entries is empty.
Solution	Correct the invalid entry. Define an entry for the generation of the copy rule.

7.1.2.39 Error_InconsistentScreenruleNoFbType

ID	Error_InconsistentScreenruleNoFbType
Cause	The program block in the SiVArc rule does not exist in the STEP7 program.
Solution	1. Enter a block that was created in the project in the "Program block" column. 2. Double-click the "Program block" column to display the available blocks. All available blocks are displayed.

7.1.2.40 Error_InconsistentScreenRuleNoScreenType

ID	Error_InconsistentScreenRuleNoScreenType
Cause	The "Master copy of a screen" in the SiVArc screen rule does not exist in the project library or a global library.
Solution	1. Double-click the row of the "Master copy of a screen" column. All available screen templates are displayed. Proceed as follows to create a new screen in the project library: 1. Create a new screen. 2. Copy the created screen. 3. Paste the copied screen into the "Master copies" folder in the project library.

7.1.2.41 Error_InconsistentTagManagementRule

ID	Error_InconsistentTagManagementRule
Cause	The expression in the "Tag table" column in the SiVARc "Tag rules" editor is invalid.
Solution	Check the expression in the "Tag table" column in the "Tag rules" editor. Ensure that there is a valid expression.

7.1.2.42 Error_InconsistentTextListRuleNoFbType

ID	Error_InconsistentTextListRuleNoFbType
Cause	The "Program block" in the SiVArc text list rule does not exist in the STEP 7 program.
Solution	1. Open the "Text list rules" SiVArc editor. 2. In the relevant row of the "Master copy of a text list" column, select a text list that was created in one of the libraries. 3. Double-click the "Master copy of a text list" column. All available text lists are displayed.

7.1.2.43 Error_InconsistentTextListRuleNoTextListType

ID	Error_InconsistentTextListRuleNoTextListType
Cause	The "Master copy of a text list" in the SiVArc text list rule does not exist in the project library or a global library.
Solution	1. Open the "Text list rules" SiVArc editor. 2. In the relevant row of the "Master copy of a screen" column, select a screen object that was created in one of the libraries. 3. Double-click the "Master copy of a screen" column. All available screens are displayed.

7.1.2.44 Error_IncorrectRuntimeSingleObjectCulture

ID	Error_IncorrectRuntimeSingleObjectCulture
Cause	In the HMI Runtime settings the Runtime language for non-multilingual objects is not available in the Runtime languages activated below these objects.
Solution	1. Open the "Runtime settings" in the HMI device. 2. Select the "Language & Font" settings. 3. Select a valid Runtime language in "Runtime language for single-language objects". This language must exist in the enabled Runtime languages.

7.1.2.45 Error_InitialCoordOutsideOfScreen

ID	Error_InitialCoordOutsideOfScreen
Cause	The value of the SiVArc property "Positioning scheme" is outside the screen. Example: The "Y position" value of a screen object is greater than the value of the corresponding screen.
Solution	1. Check the defined value in the "Positioning scheme" property. The value entered must not exceed the coordinates of the screen.

7.1.2.46 Error_InProjectLibrary

ID	Error_InProjectLibrary
Cause	The described error occurred in the project library.
Solution	Check the project library and remove the described error or contact Support.

7.1.2.47 Error_InvalidLayerValue

ID	Error_InvalidLayerValue
Cause	The configured value or the result of SiVArc expression leads to an invalid value of the layer property for the generated screen object.
Solution	Configure the appropriate value or the result of the SiVArc expression in the range between 0 - 31.

7.1.2.48 Error_InvalidOverflowScreenGeneration

ID	Error_InvalidOverflowScreenGeneration
Cause	The number of screen objects to be generated with the respective program block exceeds the number of available layout fields. Overflow screens cannot be generated for pop-up screens. The screen objects that exceed the number of available layout fields are not generated.
Solution	SiVArc generates the screen objects in the layout fields of pop-up screens only if the number of the screen objects to be generated is lower than or equal to the number of available layout fields. Ensure that all screen objects to be generated can be placed in the layout field.

7.1.2.49 Error_InvalidScreenItemName

ID	Error_InvalidScreenItemName
Cause	In RT Professional, a screen object and a screen in which the screen object is used are not permitted to have identical names. If a screen object and a screen that contains the screen object have identical names, the screen object is not generated in RT Professional.
Solution	Ensure that the name of the screen object and the screen are not the same.

7.1.2.50 Error_ItemAddedToScreenType

ID	Error_ItemAddedToScreenType
Cause	No objects can be created on a screen type.
Solution	Correct the screen rule which creates the screen object accordingly. Either use a master copy as the screen or delete the screen object from the rule.

7.1.2.51 Error_ItemHasNoName

ID	Error_ItemHasNoName
Cause	The screen object generated does not have a set name. This error occurs due to an expression that does not return a value.
Solution	1. Check the master copy of the screen object in the library. 2. Modify the SiVArc "Name" property: • No entry • At least one character

7.1.2.52 Error_Layout_ScreenItemTooBig

ID	Error_Layout_ScreenItemTooBig
Cause	The width or height of the screen object is greater than the associated screen template.
Solution	1. Check the width and height of the screen object. The value may not be greater than the value of the target object or screen.

7.1.2.53 Error_LayoutField_DoesNotExist

ID	Error_LayoutField_DoesNotExist
Cause	The layout field group used in the respective rule is not available in the positioning scheme that is referenced by the rule of the screen master copy. Therefore, the rule is not applied.
Solution	Ensure that the layout field group is configured in the correct positioning scheme and referenced by the correct screen master copy of the rule. As an alternative, use one of the existing layout field groups that are contained in the rule.

7.1.2.54 Error_LayoutField_DoesNotExistOnScreenMasterCopy

ID	Error_LayoutField_DoesNotExistOnScreenMasterCopy
Cause	The layout field group used in the screen rule is not usually contained in the screen master copy. Therefore, the rule is not applied.
Solution	Ensure that the layout field group is configured in the screen master copy of the rule. As an alternative, use one of the existing layout field groups that are contained in the rule.

7.1.2.55 Error_LayoutFieldDifferentScreenMasterCopies

ID	Error_LayoutFieldDifferentScreenMasterCopies
Cause	Two screen master copies have been configured with a SiVArc expression that contains the same screen name. These screen master copies are configured in different positioning schemes. Conflicts therefore arise when selecting layout fields. The screen object is not generated.
Solution	1. Open the "Screen rules" editor in SiVArc. 2. Open the screen master copy of the rule involved and go to "Properties". 3. Compare the properties of the two screen master copies and configure the same positioning scheme for both screen master copies.

7.1.2.56 Error_LayoutScreen_EmptyValue

ID	Error_LayoutScreen_EmptyValue
Cause	The master copy of the screen used in the screen rule is configured with reference to another positioning scheme, but one of the positioning scheme properties is not set or empty. Therefore, the rule is not applied.
Solution	1. Open the "Screen rules" SiVArc editor. 2. Open the master copy for the affected rule. 3. Configure valid properties for the positioning scheme: – If "Static" mode is set under "Layout selection", select the "Layout screen or folder" option by selecting the master copy that is available in one of the libraries. – When "Dynamic" mode is set under "Layout selection": Select the "Layout screen or folder" option by selecting the library folder that contains the master copies of the positioning scheme and is available in one of the libraries. Assign to the SiVArc expression the property "Expression for layout screen name" which can lead to one of the names of the master copies of the layout screen contained in the folder. All available screens and folders are displayed when you click the object selection under "Layout screen or folder".

7.1.2.57 Error_LayoutScreenAsMasterCopyGroupNotSupported

ID	Error_LayoutScreenAsMasterCopyGroupNotSupported
Cause	The master copy of the positioning scheme is available in a group of master copies in the library, but this is not permitted. Therefore, the rule is not applied.
Solution	Objects that are in a group of master copies in the library are not supported by SiVArc. Remove the positioning scheme from the master copy group and save it as a single master copy. Note: You create groups of master copies in the library by copying multiple objects and paste them as a single master copy into a library folder.

7.1.2.58 Error_LayoutScreenNotFound

ID	Error_LayoutScreenNotFound
Cause	The positioning scheme referenced by a master copy does not exist in the project library or a global library.
Solution	1. Open the "Screen rules" SiVArc editor. 2. Open the screen master copy of the rule involved and go to "Properties". 3. Configure valid properties for the positioning scheme: – If "Static" mode is set under "Layout selection", select the "Layout screen or folder" option by selecting the master copy that is available in one of the libraries. – When "Dynamic" mode is set under "Layout selection": Select the "Layout screen or folder" option by selecting the library folder that contains the master copies of the positioning scheme and is available in one of the libraries. Assign to the SiVArc expression the property "Expression for layout screen name" which can lead to one of the names of the master copies of the layout screen contained in the folder. All available screens and folders are displayed when you click the object selection under "Layout screen or folder".

7.1.2.59 Error_LibObjAsMasterCopyGroupNotSupported

ID	Error_LibObjAsMasterCopyGroupNotSupported
Cause	The library object is included in a group of several master copies; the rule is therefore not executed.
Solution	Objects that are in a group of master copies in the library are not supported by SiVArc. Save the library object as a single master copy.

7.1.2.60 Error_LibObjTypeNotSupported

ID	Error_LibObjTypeNotSupported
Cause	The copy rule contains an object that is not supported by the generation.
Solution	The object is not supported by the generation. Delete the rule or select another object that supports this rule.

7.1.2.61 Error_LibraryObjectExists

ID	Error_LibraryObjectExists
Cause	This project already contains an object of this type.
Solution	Perform one of the following actions: • Delete the object from the project. • Rename the object. • Delete the rule.

7.1.2.62 Error_MasterCopyOfInstanceScreenTypeNotSupported

ID	Error_MasterCopyOfInstanceScreenTypeNotSupported
Cause	The master copy is an instance of a screen type.
Solution	Ensure that a master copy of a screen is not an instance of a screen type.

7.1.2.63 Error_MasterCopyOfScreenCanNotBeFound

ID	Error_MasterCopyOfScreenCanNotBeFound
Cause	The "Master copy of a screen" in the SiVArc screen rule does not exist in the project library or a global library.
Solution	1. Open the "Screen rules" SiVArc editor. 2. In the relevant row of the "Master copy of a screen" column, select a screen object that was created in one of the libraries. 3. Double-click the "Master copy of a screen" column. All available screens are displayed.

7.1.2.64 Error_MasterCopyOfScreenCanNotBeMoved

ID	Error_MasterCopyOfScreenCanNotBeMoved
Cause	An error occurred during generation of a screen instance in the HMI device.
Solution	Check if the screen in the master copies is faulty or if it cannot be used in the HMI device type due to other restrictions. If necessary, replace the faulty screen with a correct screen or a newly created screen.

7.1.2.65 Error_Matrix_InvalidLayoutFieldGroup

ID	Error_Matrix_InvalidLayoutFieldGroup
Cause	The layout field group of the navigation buttons is used to generate a screen object, but this is not permitted. The screen object is not generated.
Solution	1. Open the "Generation matrix" SiVArc editor. 2. Navigate to the assignment of the corresponding screen object and select a different layout field group as the group that is used for the navigation buttons.

7.1.2.66 Error_Matrix_InvalidScreenItemMasterCopy

ID	Error_Matrix_InvalidScreenItemMasterCopy
Cause	The screen object master copy used in the generation of the screen object is inconsistent. For example, the screen object is not supported or it does not exist in the library.
Solution	1. Open the "Generation matrix" SiVArc editor. 2. Navigate to the assignment of the corresponding screen object. 3. Navigate to the screen rule and correct the problem with the screen object that is referenced in the "Screen object" column.

7.1.2.67 Error_Matrix_InvalidScreenMasterCopy

ID	Error_Matrix_InvalidScreenMasterCopy
Cause	The screen master copy used for generating the screen is inconsistent. For example, the screen is not supported or it does not exist in the library.
Solution	1. Open the "Generation matrix" SiVArc editor. 2. Navigate to the assignment of the corresponding screen. 3. Navigate to the screen rule and correct the problem with the screen master copy that is referenced in the "Master copy of a screen" column.

7.1.2.68 Error_Matrix_LayoutFieldGroupDoesnotExist

ID	Error_Matrix_LayoutFieldGroupDoesnotExist
Cause	The layout field group selected for the assignment of the screen object in the generation matrix does not exist in the screen master copy that is used to generate the target screen.
Solution	1. Open the "Generation matrix" SiVArc editor. 2. Navigate to the assignment of the corresponding screen object. 3. Select one of the two options: – Select one of the layout fields from the list box for assigning the screen object. – Create the required layout fields in the screen master copy of the rule from which the assigned screen object is generated.

7.1.2.69 Error_MaxTagCountReached

ID	Error_MaxTagCountReached
Cause	More than 500,000 HMI tags are to be created in the current generation. This number exceeds the maximum permitted number of tags to be generated.
Solution	Reduce the number of tags to be generated. You do this by either resetting a station selection or by resetting the "Visible for HMI" flag of a tag in the PLC program.

7.1.2.70 Error_MergeAlarms

ID	Error_MergeAlarms
Cause	Two or more SiVArc alarm rules result in the generation of alarm objects with the same name. In this case, the alarm objects are merged into an alarm object if the alarm object templates are identical. However, the alarm objects templates are not identical here.
Solution	• Option 1: In the SiVArc editor "Alarm rules", set the same template for the rules that result in the generation of alarm objects with the same name. • Option 2 : Enter expressions for the corresponding copy templates in the SiVArc properties editor for "Name" that lead to different values.

7.1.2.71 Error_MergeTextLists

ID	Error_MergeTextLists
Cause	Two or more SiVArc text list rules result in the generation of text lists with the same name. In this case, the text lists are merged into one list if the text list master copies are identical. However, the test list master copies are not identical here.
Solution	• Set the same master copy in the SiVArc "Text list rules" editor for rules that result in the generation of text lists with the same name. • In the SiVArc Properties editor, enter expressions for "Name" that will result in different values for the respective master copies.

7.1.2.72 Error_MissingScript

ID	Error_MissingScript
Cause	The script called in the HMI object does not exist.
Solution	Select an available script in the "SiVArc events" editor of the HMI object.

7.1.2.73 Error_MultilingualPropertyLengthTooLong

ID	Error_DuplicateAlarmObjectCanNotBeCreated
Cause	The resolved value for multilingual property of an HMI object is longer than supported characters.
Solution	Multilingual properties for HMI objects may have a maximum length of 128 characters. Check the SiVArc multilingual property "Alarm text" in the corresponding alarm object within the copy template library.

7.1.2.74 Error_NameTooLong

ID	Error_NameTooLong
Cause	The name of the generated object cannot be generated with the original name due to the naming conventions of WinCC. The name was abbreviated.
Solution	Open the relevant master copy of the library object and adapt the expression for the "Name" property in order to abbreviate the "Name" string generated by SiVArc.

7.1.2.75 Error_NotSupportedLayoutScreen

ID	Error_NotSupportedLayoutScreen
Cause	The referenced master copy of the positioning scheme is not supported. Example: "Button" has been referenced instead of "Screen".
Solution	1. Open the "Screen rules" SiVArc editor. 2. Open the screen master copy for the affected rule. 3. Configure valid properties for the positioning scheme: – If "Static" mode is set under "Layout selection", select the "Layout screen or folder" option by selecting the master copy that is available in one of the libraries. – When "Dynamic" mode is set under "Layout selection": Select the "Layout screen or folder" option by selecting the library folder that contains the master copies of the positioning scheme and is available in one of the libraries. Assign to the SiVArc expression the property "Expression for layout screen name" which can lead to one of the names of the master copies of the layout screen contained in the folder. All available screens and folders are displayed when you click the object selection under "Layout screen or folder".

7.1.2.76 Error_NotSupportedPopupScreenType

ID	Error_NotSupportedPopupScreenType
Cause	In RT Professional or on a Basic Panel, a pop-up screen is generated as a screen master copy in a screen rule.
Solution	Pop-up screens are not supported on RT Professional and Basic Panels. Delete the rule or select another object that supports this rule.

7.1.2.77 Error_NotSupportedScreenObject

ID	Error_NotSupportedScreenObject
Cause	The screen object called in the screen rule is not supported. Example: The screen object is a "Screen window"
Solution	1. In the "Screen object" column, select a screen object that is available in the project library. 2. Double-click the "Screen object" column. 3. The available screen objects are displayed. Proceed as follows to create a new screen in the project library: 1. Create a new screen. Copy the created screen. 2. Paste the copied screen into the "Master copies" folder in the project library.

7.1.2.78 Error_NotSupportedScreenType

ID	Error_NotSupportedScreenType
Cause	The called screen template is not supported. Example: A "Button" has been called instead of a "Screen".
Solution	1. Under "Master copy of a screen", select a screen from the project library. 2. Double-click a "Screen object". 3. All available screen templates are displayed. Alternative procedure: 1. Create a new screen. 2. Save the newly created screen in the project library.

7.1.2.79 Error_NoValidLicense

ID	Error_NoValidLicense
Cause	No valid SiVArc license available.
Solution	Install a valid SiVArc license on the computer.

7.1.2.80 Error_ObjectCreationFailedDueToErrorInExpression

ID	Error_ObjectCreationFailedDueToErrorInExpression
Cause	An error occurred when triggering the expression. Possible causes: Syntax error in the expression or missing tags.
Solution	Correct the expression in the corresponding "SiVArc property".

7.1.2.81 Error_ObjectCreationFailedDueToErrorInExpressionInMultilingualContext

ID	Error_ObjectCreationFailedDueToErrorInExpressionInMultilingualContext
Cause	An error occurred when resolving an expression. This alarm is usually caused by a syntax error in the expression or by undefined or faulty tags. Keep in mind that this is a property which supports multiple languages. This means that the expression is resolved for each SiVArc-relevant language, and the result can be different for each language. The error may only occur in one language. The error occurred in the displayed language.
Solution	1. Open the "SiVArc properties" editor. 2. Check the expression of the corresponding SiVArc property and the definitions of the tags used in it. These tags may also support multilingualism and must be checked for the faulty language. 3. Adapt the expression of the corresponding SiVArc property or the definitions of the utilized tags correspondingly. The "Alarms for expressions" help offers additional information on the displayed type of error.

7.1.2.82 Error_ObjectCreationFailedDueToVariableNotDef

ID	Error_ObjectCreationFailedDueToVariableNotDef
Cause	The expression entered for the specified property of the object causing the error includes an undefined tag.
Solution	Check the expression in the property of the object causing the error. Either correct the expression or define the tag.

7.1.2.83 Error_ObjectCreationFailedDueToVariableNotDefInMultilingualContext

ID	Error_ObjectCreationFailedDueToVariableNotDefInMultilingualContext
Cause	An undefined tag is used in an expression. Note that this is a multilingual expression. It is resolved for each SiVArc-relevant language. If this expression includes a multilingual, predefined tag, the result can be different for each language. If the predefined, multilingual tags contain different values, the result may be different error messages.
Solution	You must check for each SiVArc-relevant language whether the expression is faulty. This is necessary because predefined, multilingual tags included in the expression can assume different values for each language. The result may be different error messages for each language.

7.1.2.84 Error_ObjectGenerationFailed_InvalidName

ID	Error_ObjectGenerationFailed_InvalidName
Cause	The name of the generated object is invalid for the current HMI device.
Solution	1. Open the "SiVArc properties" editor. 2. Correct the "Name" property so that the expression results in a valid name.

7.1.2.85 Error_ObjectGenerationFailed_IsInvalidOnCurrentDevice_Screen

ID	Error_ObjectGenerationFailed_IsInvalidOnCurrentDevice_Screen
Cause	The requested screen cannot be generated in the current HMI device. Either the type is not supported or the master copy was created in an HMI device that is not supported.
Solution	1. Open the "Screen rules" SiVArc editor. 2. In the relevant row of the "Master copy of a screen" column, select a screen that is available in the project library and is supported by the target HMI device. The available screens are shown when you double-click the corresponding row in the "Master copy of a screen" column. Because the "Screen rules" SiVArc editor includes the rules for all HMI devices, you may see some screens that are not supported by a specific HMI device.

7.1.2.86 Error_ObjectGenerationFailed_IsInvalidOnCurrentDevice_ScreenItem

ID	Error_ObjectGenerationFailed_IsInvalidOnCurrentDevice_ScreenItem
Cause	The screen object you want to generate is not supported by the utilized HMI device. The called screen object may be a "Screen Window Control", for example, which is not supported by WinCC Advanced devices.
Solution	1. Open the "Screen rules" SiVArc editor. 2. In the relevant row of the "Screen object" column, select a screen object that is available in the project library and is supported by the target HMI device. The available screen objects are shown when you double-click the corresponding row in the "Screen object" column. Because the "Screen rules" SiVArc editor includes the rules for all HMI devices, you may see some screen objects that are not supported by a specific HMI device.

7.1.2.87 Error_ObjectGenerationFailedBecauseInvalid

ID	Error_ObjectGenerationFailedBecauseInvalid
Cause	An error occurred during generation of an HMI tag.
Solution	Check whether the PLC tag is valid.

7.1.2.88 Error_ObjectGenerationFailedBecauseInvalidTable

ID	Error_ObjectGenerationFailedBecauseInvalidTable
Cause	The respective HMI tag table is unavailable.
Solution	Correct the name of the tag table or specify a valid tag table.

7.1.2.89 Error_ObjectGenerationFailedBecauseLibraryIdInvalid

ID	Error_ObjectGenerationFailedBecauseLibraryIdInvalid
Cause	The corresponding object is not available in the library.
Solution	Correct the object in the library or select a suitable screen object in the "Screen rules" SiVArc editor.

7.1.2.90 Error_OverflowScreenCount_VarNotDef

ID	Error_OverflowScreenCount_VarNotDef
Cause	The expression entered for the number of overflow screens includes an undefined tag.
Solution	1. Open the SiVArc properties editor for the master copy that is causing the error. 2. Check the "Number of overflow screens" property. 3. Either correct the expression or define a suitable tag so that the resolved expression results in a positive integer smaller than 33.

7.1.2.91 Error_OverflowScreenCountWrongValue

ID	Error_OverflowScreenCountWrongValue
Cause	The value entered or resulting from the resolved expression for the number of overflow screens is either less than 0 or greater than 32, which means it is located outside the permitted range.
Solution	1. Open the SiVArc properties editor for the master copy that is causing the error. 2. Check the "Number of overflow screens" property. 3. Either correct the expression or define a suitable tag so that the resolved expression results in a positive integer smaller than 33.

7.1.2.92 Error_ParentScreenCanNotBeFound

ID	Error_ParentScreenCanNotBeFound
Cause	It is not possible to generate the screen object from the screen rules or at a screen that does not exist in the HMI device. Each screen object that is generated in an HMI device must be assigned to a screen.
Solution	Before generating the screen object, make sure that the parent screen exists in the HMI device. The screen on which the screen objects are located must be created using screen rules and before the application of the rules for the screen object.

7.1.2.93 Error_PlcDeviceIsInvalidIpiProxy

ID	Error_PlcDeviceIsInvalidIpiProxy
Cause	The corresponding proxy was not initialized or an error occurred during initialization, or the proxy was initialized without data blocks.
Solution	Update the corresponding proxy or disable the generation of this proxy before the SiVArc generation is started again.

7.1.2.94 Error_PlcDeviceNeedsCompile

ID	Error_PlcDeviceNeedsCompile
Cause	The STEP7 program includes changes that have not yet been compiled.
Solution	Compile the STEP7 program.

7.1.2.95 Error_PlcPrefixNotSet

ID	Error_PlcPrefixNotSet
Cause	Conflict of tag names.
Solution	1. Open the "Runtime settings" of the HMI device. 2. Click "Settings for tags". 3. Enable the option "PLC name as prefix in the HMI tag name"

7.1.2.96 Error_ReadUICulture

ID	Error_ReadUICulture
Cause	Could not determine the set language of the user interface.
Solution	Please contact Support.

7.1.2.97 Error_ReleasedVersionforFbLibraryTypeNotFound

ID	Error_ReleasedVersionforFbLibraryTypeNotFound
Cause	A rule references a PLC program block type in the "Screen rules" or "Text list rules" SiVArc editor that is still in progress and does not contain a released version.
Solution	Release the version for the corresponding PLC program block type or discard the version in progress.

7.1.2.98 Error_RuleImport_Workbook

ID	Error_RuleImport_Workbook
Cause	The XLS file to be imported does not contain a worksheet which matches the specification. The import file is empty or the included worksheets do not contain the correct column names.
Solution	Check the content of the import file. There must be a worksheet with correct column names that correspond to the rule properties.

7.1.2.99 Error_ScreenAsMasterCopyGroupNotSupported

ID	Error_ScreenAsMasterCopyGroupNotSupported
Cause	An invalid library element is referenced in the 'Master copy of a screen' column in the screen rule. It is not a master copy for an individual screen, but a group of several master copies/screens . Such groups are not supported as a 'master copy of a screen'.
Solution	Copy the screen that you want to select in the screen rule as a single element into the master copies folder in the library.

7.1.2.100 Error_ScreenItemCanNotCreatedOnScreenInstance

ID	Error_ScreenItemCanNotCreatedOnScreenInstance
Cause	The screen is an instance of a screen type.
Solution	Ensure that a master copy of a screen is not an instance of a screen type.

7.1.2.101 Error_ScreenItemGenerationFailedBecauseLibraryIdInvalid

ID	Error_ScreenItemGenerationFailedBecauseLibraryIdInvalid
Cause	The master copy of a screen object is not available in the library.
Solution	Open the "Screen rules" SiVArc editor and select a suitable object in the "Screen object" column.

7.1.2.102 Error_ScreenItemNameIsEmpty

ID	Error_ScreenItemNameIsEmpty
Cause	The resolved expression of the SiVArc property "Name" results in an empty value.
Solution	Correct the SiVArc name property of the utilized screen object so that the name is resolved into a valid value.

7.1.2.103 Error_ScreenModuleReleasedVersionNotFound

ID	Error_ScreenModuleReleasedVersionNotFound
Cause	There is no released version of the faceplate called.
Solution	Release the called faceplate in the project library: 1. Select "Release version" from the shortcut menu of the faceplate.

7.1.2.104 Error_ScreenNameInvalid

ID	Error_ScreenNameInvalid
Cause	The resolved expression of the SiVArc property "Name" results in an invalid value. Different HMI device types have different restrictions for the valid screen names.
Solution	Correct the "Name" property of the utilized screen in the "SiVArc properties" editor so that the name is resolved into a valid value.

7.1.2.105 Error_ScreenNameIsEmpty

ID	Error_ScreenNameIsEmpty
Cause	The resolved expression of the SiVArc property "Name" results in an empty value.
Solution	Correct the "Name" property of the utilized screen in the "SiVArc properties" editor so that the name is resolved into a valid value.

7.1.2.106 Error_ScreenObjectAsMasterCopyGroupNotSupported

ID	Error_ScreenObjectAsMasterCopyGroupNotSupported
Cause	An invalid library element is referenced in the 'Screen object' column in the screen rule. It is a group of several master copies/screens rather than a master copy for an individual screen object. Such groups are not supported as a 'screen elements'.
Solution	Copy the screen object that you want to select in the screen rule as a single element into the master copies folder in the library.

7.1.2.107 Error_ScreenObjectNotFound

ID	Error_ScreenObjectNotFound
Cause	The name of the screen object has been changed in the project library.
Solution	1. Select a screen element in the "Screen object" column. 2. Double-click the "Screen object" column. All available screen objects are displayed. Proceed as follows to create a new picture element in the project library: 1. Open an available screen. 2. Move a screen object to the screen using drag-and-drop. 3. Assign parameters to the screen object. 4. Copy the screen object. 5. Paste the screen object into the "Master copies" folder of the project library.

7.1.2.108 Error_ScreenRuleNoScreenInstanceAsScreenType

ID	Error_ScreenRuleNoScreenInstanceAsScreenType
Cause	The master copy is an instance of a screen type.
Solution	Ensure that a master copy of a screen is not an instance of a screen type.

7.1.2.109 Error_ScreenTypeNotFound

ID	Error_ScreenTypeNotFound
Cause	The name of the screen template has been changed in the project library.
Solution	Make sure that the name of the screen template has not been unintentionally changed in the screen rules editor.

7.1.2.110 Error_SivarcRuleConditionError

ID	Error_SivarcRuleConditionError
Cause	The expression in the screen rule condition contains a syntax error. A required tag could not be found.
Solution	1. Check the expression in the condition.

7.1.2.111 Error_SivarcRuleConditionError2

ID	Error_SivarcRuleConditionError2
Cause	The tag which that was defined in the expression does not exist.
Solution	1. Check the names of the tags in the expression. 2. Navigate to the network of the specific program block. 3. Enter a valid tag definition under "Tag definition".

7.1.2.112 Error_SivarcRuleConditionWrongType

ID	Error_SivarcRuleConditionWrongType
Cause	The expression in the screen rule under "Condition" has not been resolved into True or False.
Solution	1. Resolve the expression under "Condition" into True or False.

7.1.2.113 Error_TagExists

ID	Error_TagExists
Cause	An HMI tag with the same name has already been generated.
Solution	Check the STEP 7 programs used and make sure that no HMI tags will be created with the same name.

7.1.2.114 Error_TagGen_UnsupportedDataType

ID	Error_TagGen_UnsupportedDataType
Cause	The data type in the block is not supported.
Solution	Disable the "Accessible from HMI" option.

7.1.2.115 Error_TagRuleError

ID	Error_TagRuleError
Cause	An expression in the tag editor contains a syntax error.
Solution	Check the expression in the column specified.

7.1.2.116 Error_TagRuleError_VarNotDef

ID	Error_TagRuleError_VarNotDef
Cause	Could not execute the tag rule because it uses an expression with a user-defined tag.
Solution	Open the "Tag rules" editor and correct the faulty expression in the rule. In tag rules, only predefined tags may be used in expressions.

7.1.2.117 Error_TagTableCanNotCreate

ID	Error_TagTableCanNotCreate
Cause	The resolved name of the tag table is invalid. Different HMI device types can have different restrictions for the valid names.
Solution	Open the "Tag rules" editor and adapt the entry in the "Tag table" column.

7.1.2.118 Error_TextEntryAlreadyExists

ID	Error_TextEntryAlreadyExists
Cause	The configuration of text lists in the master copy results in the generation of multiple entries with the same name. The names of the text list entries must be unique.
Solution	Ensure there are no multiple text list entries with the same name in the call structure of the PLC.

7.1.2.119 Error_TextListAsMasterCopyGroupNotSupported

ID	Error_TextListAsMasterCopyGroupNotSupported
Cause	An invalid text list element is referenced in the 'Master copy of a text list' column in the screen rule. It is not a master copy of an individual text list, but a group of several master copies/text lists. Such groups are not supported as a 'master copy of a text list'.
Solution	Copy the text lists that you want to select in the text list rule as a single element into the master copies folder in the library.

7.1.2.120 Error_TextListCreationFailedDueToErrorInExpressionInMultilingualContext

ID	Error_TextListCreationFailedDueToErrorInExpressionInMultilingualContext
Cause	An error occurred when resolving an expression. This alarm is usually caused by a syntax error in the expression or by undefined or faulty tags. Keep in mind that this is a property which supports multiple languages. This means that the expression is resolved for each SiVArc-relevant language, and the result can be different for each language. The error may only occur in one language. The error occurred in the displayed language.
Solution	1. In the STEP 7 program, open the program block that includes the function call initiating the text list generation. 2. Click the block comment, and check the SiVArc text definitions for the faulty text list entry in the "Plug-Ins". Check the entries in the "SiVArc tag expression" column in particular. Keep in mind that predefined tags can also support multiple languages and must be checked for the faulty language. 3. Correct the faulty expression or the definitions of the utilized tags accordingly. 4. The "Alarms for expressions" help offers additional information on the displayed type of error.

7.1.2.121 Error_TextlistCreationFailedDueToNoGenerationlevelTagsMatched

ID	Error_TextlistCreationFailedDueToNoGenerationlevelTagsMatched
Cause	The tags that are synchronized with a regular expression in the master copy do not match the generation levels of the master copy.
Solution	The operand type of the synchronized tags must match at least one generation level configured in the master copy. If the generation levels match, the tag address is used as a starting point for the text list generation.

7.1.2.122 Error_TextlistCreationFailedDueToNoMatchingProgramblockVariables

ID	Error_TextlistCreationFailedDueToNoMatchingProgramblockVariables
Cause	The regular expression in the text list master copy cannot be resolved for any program block tag of the function block.
Solution	Check whether the regular expression configured in the master copy can be resolved for a valid program block tag of the function block. The regular expression can be a simple string or an expression with an asterisk. • The error occurs when the regular expression has been configured as "Text*" with an asterisk and the program block tags are named "FirstVar1", "SecondVar1". In this case, configure the tags in the following format: "TextFirstVar1", "TextSecondVar1". • If the regular expression has been configured without an asterisk, for example "FirstVar1", the function block tag must contain the exact name ("FirstVar1"). Other names will be ignored for the generation.

7.1.2.123 Error_TextListCreationFailedDuetoNonMatchingDataBlockCallers

ID	Error_TextListCreationFailedDuetoNonMatchingDataBlockCallers
Cause	You have configured a regular expression in the text list master copy. However, the expression cannot be resolved for any valid program block tags which are used for generating the tags of the PLC tag table.
Solution	The values of the synchronized program block tags based on a regular expression of a master copy must be valid tags from the PLC symbol table. These tags cannot contain any default values or data types unknown to the PLC.

7.1.2.124 Error_TextlistCreationFailedDueToNoRegularExpression

ID	Error_TextlistCreationFailedDueToNoRegularExpression
Cause	The "Block parameter name" property in the text list master copy is empty.
Solution	The "Block parameter name" property in the text list master copy must contain either a simple string or a regular expression. The regular expression must be based on the program block tags of the text list control that are used for the text list generation.

7.1.2.125 Error_TextListCreationFailedDueToVariableNotDefInMultilingualContext

ID	Error_TextListCreationFailedDueToVariableNotDefInMultilingualContext
Cause	An undefined tag is used in an expression. As a result, an error occurred when resolving the expression. Keep in mind that this is a multilingual expression. It is resolved for each SiVArc-relevant language. If this expression includes a multilingual, predefined tag, the result can be different for each language. If the predefined, multilingual tags contain different values, the result may be different error messages.
Solution	1. In the STEP 7 program, open the program block that includes the function call initiating the text list generation. 2. Click the block comment, and check the SiVArc text definitions for the faulty text list entry in the "Plug-Ins". In particular, check the entries in the "SiVArc tag expression" column and make sure that an undefined tag is not used. Keep in mind that predefined tags can also support multiple languages and must be checked for the faulty language.

7.1.2.126 Error_TextListTypeNotFound

ID	Error_TextListTypeNotFound
Cause	A text list was entered in the "Master copy of a text list" column in a SiVArc text list rule which does not exist in the project.
Solution	1. Open the "Text list rules" SiVArc editor. 2. Select a master copy of a text list which exists in a library for the corresponding, faulty SiVArc text list rule.

7.1.2.127 Error_TextListTypeNotSupported

ID	Error_TextListTypeNotSupported
Cause	The master copy that you have configured in the text list rules is not the "HmiTextList" type. To generate a text list in an HMI device, the master copy must have the text list type.
Solution	Verify that the master copy used in the text list rules has the text list type.

7.1.2.128 Error_UICultureNotSupported

ID	Error_UICultureNotSupported
Cause	The user interface of the TIA Portal is not set to English.
Solution	Set the user interface of the TIA Portal to English.

7.1.2.129 Error_WriteableLibraryAlarmType

ID	Error_WriteableLibraryAlarmType
Cause	The alarm object referenced in alarm rule is from global library; where the global library is opened without write protection.
Solution	Close the global library, and re-open it with write protection.

7.1.2.130 Error_WriteableLibraryLayoutScreen

ID	Error_WriteableLibraryLayoutScreen
Cause	The positioning scheme referenced by the screen master copy of the rule is located in a global library opened for writing.
Solution	Close the library and open it again with write protection.

7.1.2.131 Error_WriteableLibraryLibObjType

ID	Error_WriteableLibraryLibObjType
Cause	The library object referenced in the rule is located in a global library opened for writing.
Solution	Close the library and open it again with write protection.

7.1.2.132 Error_WriteableLibraryScreenObject

ID	Error_WriteableLibraryScreenObject
Cause	A screen object is referenced in a SiVArc screen rule and this object is located in a global library opened for writing.
Solution	Close the global library opened for writing. Open the library as read-only.

7.1.2.133 Error_WriteableLibraryScreenType

ID	Error_WriteableLibraryScreenType
Cause	A required global library is open without write protection.
Solution	Close the library and open it again with write protection.

7.1.2.134 Error_WriteableLibraryTextListType

ID	Error_WriteableLibraryTextListType
Cause	A text list is referenced in a rule in the "Text list rules" SiVArc editor and this list is located in a global library opened for writing.
Solution	Close the global library opened for writing. Open the library as read-only.

7.1.3 Warnings

7.1.3.1 LogWarning_TextEntryCouldNotBeResolved

ID	LogWarning_TextEntryCouldNotBeResolved
Cause	No corresponding text definition could be found in the PLC program for the name of the text list entry in the master copy.
Solution	The text definitions of all entries in the master copy must also be defined in the PLC program. Multilingual entries are generated in the newly created text lists for all matching text definitions.

7.1.3.2 Warning_AdditionalContentScreeninMasterCopryGroup

ID	Warning_AdditionalContentScreeninMasterCopryGroup
Cause	An attempt was made to generate additional screens in a library folder and this folder contains a group of master copies which also include master copies of screens. These could not be generated as additional screens because SiVArc master copy groups are not supported.
Solution	Copy the master copies of the screens that are contained in the master copy group as single elements into the library folder from which additional screens should be generated.

7.1.3.3 Warning_AlarmNameTooLong

ID	Warning_AlarmNameTooLong
Cause	The name of an HMI object is longer than supported characters.
Solution	Names of alarm objects may have maximum length of 128 characters. Check the SiVArc name property in the corresponding alarm object in the copy template library.

7.1.3.4 Warning_AnimationHasInvalidTag

ID	Warning_AnimationHasInvalidTag
Cause	The specified HMI tag to be connected does not exist or is not accessible for the HMI device.
Solution	1. Enable the "Accessible in HMI" option at the PLC tag.

7.1.3.5 Warning_BaseScreenInOtherFolder

ID	Warning_BaseScreenInOtherFolder
Cause	A generated screen with the same name already exists in another folder.
Solution	1. Adapt the "Master copy" screen object in the library. 2. Correct either the SiVArc property "Name" or the SiVArc property "Folder".

7.1.3.6 Warning_DeleteObjectInUse

ID	Warning_DeleteObjectInUse
Cause	• To prevent undesirable loss of data, the screen is not deleted. • The screen is no longer generated by SiVArc and is to be deleted, even though it still contains objects.
Solution	Delete the screen manually.

7.1.3.7 Warning_DeleteObjectInUseTagFolder

ID	Warning_DeleteObjectInUseTagFolder
Cause	A tag group is no longer generated by SiVArc and is to be deleted, although it still contains objects that were generated by the user.
Solution	Delete the tag group manually.

7.1.3.8 Warning_DeleteTagtInUse

ID	Warning_DeleteTagtInUse
Cause	A tag is no longer generated by SiVArc and is to be deleted, although objects created by the user still access this tag.
Solution	Delete the tag manually if you no longer need it.

7.1.3.9 Warning_EndlessCallLoopDetected

ID	Warning_EndlessCallLoopDetected
Cause	An infinite loop was detected in the call hierarchy of the specified PLC block.
Solution	Check the Step 7 program and remove the infinite loop.

7.1.3.10 Warning_EventHasInvalidPropertyName

ID	Warning_EventHasInvalidPropertyName
Cause	The required HMI object property for the system function parameter is not available during SiVArc generation. Example: The "SetPropertyByTag" system function has the "Property name" parameter. This warning is issued if the specified property name for the HMI object that you specified in the "HMI object" parameter is not available.
Solution	Assign the correct language-specific property name in the "Property name" parameter.

7.1.3.11 Warning_EventHasInvalidScreen

ID	Warning_EventHasInvalidScreen
Cause	The screen expected by a system function parameter is not available during SiVArc generation. Example: The "SetPropertyByTag" system function has the "Screen name" parameter. If the value specified for a screen is resolved but the screen does not exist on the HMI device, this warning is issued.
Solution	Ensure that the screen with the specified name is available on the HMI device.

7.1.3.12 Warning_EventHasInvalidScreenItem

ID	Warning_EventHasInvalidScreenItem
Cause	The expected HMI object is not available for assignment to a system function parameter during SiVArc generation. Example: The "SetPropertyByTag" system function has the "HMI object" parameter. If the value specified is resolved into an HMI object that does not exist in the specified screen, this warning is issued.
Solution	Ensure that the corresponding screen and HMI object are available with the specified name.

7.1.3.13 Warning_EventHasInvalidTagType

ID	Warning_EventHasInvalidTagType
Cause	The tag specified in the parameter has a different type than the one expected for this parameter. Example: The tag in the "SetBit" function must be a BOOL type. Another type, e.g. INT, is not allowed.
Solution	Ensure that the tag for the parameter has the correct type.

7.1.3.14 Warning_FunctionHasInvalidTag

ID	Warning_FunctionHasInvalidTag
Cause	The HMI tag to be connected does not exist.
Solution	Check the name of the HMI tag which creates the tag name in the HMI object in the master copy of the library.

7.1.3.15 Warning_FunctionListCanNotAdd

ID	Warning_FunctionListCanNotAdd
Cause	An error occurred when adding the function to the HMI object.
Solution	1. Check the settings under "SiVArc events" of the HMI object in the master copy of the library. 2. If the function is a user-defined script, check that the script exists in the project. 3. Check that the functions of the parameters are valid.

7.1.3.16 Warning_FunctionParameterInvalidValueSetDefault

ID	Warning_FunctionParameterInvalidValueSetDefault
Cause	An expression for an Enum parameter of a system function is resolved to an unexpected value. Example: The "Layout" parameter of the "ShowPopupScreen" system function expects the values "Switch", "On" or "Off". Any other value is ignored and the parameter is automatically set to the default value.
Solution	For the parameters, enter only values or SiVArc expressions that are resolved to valid values.

7.1.3.17 Warning_FunctionParameterValueIsInvalid

ID	Error_FunctionParameterValueIsInvalid
Cause	• One or more parameters of the called function have an invalid data type. • The resolved value of an expression results in an invalid data type.
Solution	• Make sure that the defined data types are valid for the function called. • The resolved expression must create a valid value. Valid data types: – hmitag – string – double – int32.

7.1.3.18 Warning_FunctionParameterValueLengthIsInvalid

ID	Warning_FunctionParameterValueLengthIsInvalid
Cause	The values for the integer function parameters are outside the permitted range. Example: An integer value must be entered in the "Object number" parameter of the "ActivateScreen" system function. If the specified value is not within the permitted range, the warning is issued.
Solution	For the parameters, enter only values or SiVArc expressions that are resolved within the permitted range.

7.1.3.19 Warning_InstanceOfScreenTypeInTest

ID	Warning_InstanceOfScreenTypeInTest
Cause	It is not possible to change a write-protected screen instance of a screen type.
Solution	Enable the relevant screen type.

7.1.3.20 Warning_InvalidProperty

ID	Warning_InvalidProperty
Cause	The configured or resolved value of the expression results in a text or graphic list that does not exist.
Solution	Enter only the names of the text or graphic lists that already exist when configuring the text or graphic list properties of screen objects in the SiVArc plug-in.

7.1.3.21 Warning_InvalidTRefProperty

ID	Warning_InvalidTRefProperty
Cause	The SiVArc expression configured for a faceplate property was resolved with an invalid value. The value specified is either not supported on the referenced HMI device or does not exist. Example: A graphic or text list that is referenced by a faceplate property does not exist on the HMI device.
Solution	Open the corresponding faceplate type and enter the valid expression in the property that can be resolved to a valid value.

7.1.3.22 Warning_LayoutFieldForNavButtonNotFound

ID	Warning_LayoutFieldForNavButtonNotFound
Cause	The problem can have two causes: 1. The layout field group for navigation that was configured in the screen master copy does not exist. 2. The layout field group for navigation only contains one layout field. In this case, the second navigation button is created without positioning.
Solution	1. Open the "Screen rules" editor in SiVArc. 2. Open the screen master copy of the rule involved and go to "Properties". 3. Set the "Layout field for navigation" property by selecting a layout field from the list box. As an alternative, you can create a new layout field with the same name as the name you have set in the "Layout field for navigation" property. Ensure that the layout field group for generating the navigation buttons has two layout fields.

7.1.3.23 Warning_Matrix_NavigationItemHasInvalidActivateScreenReference

ID	Warning_Matrix_NavigationItemHasInvalidActivateScreenReference
Cause	The "ActivateScreen" system function references a screen that is not available on the HMI device. The system function is not generated for the navigation object based on the screen assignments of the matrix.
Solution	1. Open the "Generation matrix" editor in SiVArc and open the corresponding assignment. 2. Go to the row with the screen name that is referenced in the deleted "ActivateScreen" system function. 3. If the screen is also used for the target device, select the check box of the target device and generate again. If the screen for the target device is not relevant, navigation to a non-existent screen through the deleted function is not possible.

7.1.3.24 Warning_Matrix_ScreenDoesNotExist

ID	Warning_Matrix_ScreenDoesNotExist
Cause	The screen does not exist on the HMI device and the navigation object based on the screen assignments of the matrix cannot be generated.
Solution	The screen assigned to the navigation object that exists in the source device in the generation matrix must be available all target devices. Ensure that the rule that generates screens for navigation objects for the source devices is also executed for the target devices.

7.1.3.25 Warning_NameTooLong_Tag

ID	Warning_NameTooLong_Tag
Cause	The name of an HMI tag is longer than 128 characters. A tag name has a direct effect on the depth of a structure that is used in a PLC program.
Solution	Names for HMI tags should not exceed 128 characters. Reprogram the structure in such a way that it avoids deep nesting. Disable the "Accessible from HMI" option.

7.1.3.26 Warning_NameTooLong_TagTable

ID	Warning_NameTooLong_TagTable
Cause	The name of an "HMI tag table" is longer than 128 characters.
Solution	The name of an "HMI tag table" must not exceed 128 characters. 1. Check the text or expression in the "Tag table" in the tag rule column. 2. Correct the name.

7.1.3.27 Warning_NavigationItemNotFound

ID	Warning_NavigationItemNotFound
Cause	User-defined buttons do not exist for the overflow screens. Standard buttons are used instead.
Solution	In the library, save user-defined buttons for the overflow screens: • Button for previous screen: Master copy\PreviousButton • Button for next screen: Master copy\NextButton

7.1.3.28 Warning_NavigationItemNotSupported

ID	Warning_NavigationItemNotSupported
Cause	The project library contains user-defined screen objects which are to serve as navigation buttons for the overflow screen navigation. However, these are not buttons. A text field was named NextButton or PrevButton, for example, and copied to the master copies of the project library.
Solution	Save buttons as user-defined buttons in the library for use in the overflow screen navigation: • Storage location of the utilized navigation buttons for previous screen: "MasterCopy\PrevButton". • Storage location of the utilized navigation buttons for next screen: "MasterCopy\NextButton"

7.1.3.29 Warning_NoDeviceSelectedInAllScreenRules

ID	Warning_NoDeviceSelectedInAllScreenRules
Cause	No device (PLC, HMI, HMI device type) is selected for any rule in the screen rule editor.
Solution	Check the information about the devices (PLC, HMI, HMI device type) in the screen rule editor.

7.1.3.30 Warning_NoHmiDevicesSelectedForGeneration

ID	Warning_NoHmiDevicesSelectedForGeneration
Cause	No HMI device is selected in the SiVArc device selection dialog.
Solution	To perform a SiVArc generation, you must select at least one HMI device in the device selection dialog.

7.1.3.31 Warning_NoSelectedPlcDevices

ID	warning_NoSelectedPlcDevices
Cause	A PLC was not selected for generation in the SiVArc PLC station selection dialog.
Solution	Start SiVArc generation again and select at least one PLC for the generation in the station selection dialog.

7.1.3.32 Warning_NoTextEntriesCouldBeResolved

ID	Warning_NoTextEntriesCouldBeResolved
Cause	SiVArc has generated a text list for which not even a single text could be generated for a text entry. This occurs if no associated text definitions were found in the PLC program.
Solution	Navigate to the location where the program block was called that has initiated the corresponding text list generation. Now take a look at all networks above up to the block title or the program block that initiated the last text list generation. Navigate to the SiVArc Plugin editor of each network and check in the "Text definitions" category if there is a text definition with an ID (= entry in "Name" column) that matches at least one name in the text list entries of the corresponding text list.

7.1.3.33 Warning_NotSupportedAnimation

ID	Warning_NotSupportedAnimation
Cause	An unsupported animation is used at a screen object. For example, the "Control enable" animation can be configured at the "GRAPH overview" object for Runtime Professional. Runtime Advanced does not support this animation for this object.
Solution	Make sure to only use animations that are supported by the desired device.

7.1.3.34 Warning_NotSupportedTemplateScreen

ID	Warning_NotSupportedTemplateScreen
Cause	Template screens are not supported in WinCC RT Professional.
Solution	Check the SiVArc "Template screen" property in the corresponding screen in the copy template library with WinCC RT Advanced. Remove the expression if it is not required to avoid additional warning message.

7.1.3.35 Warning_OverflowScreenCountMismatch

ID	Warning_OverflowScreenCountMismatch
Cause	There are several SiVArc screen master copies that define the same screen. The SiVArc screen types have defined different values for the SiVArc property "Number of overflow screens".
Solution	In the library, modify the SiVArc property "Number of overflow screens" in the various master copies so that they all request the same number of overflow screens.

7.1.3.36 Warning_PropertyCanNotSet

ID	Warning_PropertyCanNotSet
Cause	An error occurred when setting a property of a faceplate or an HMI object: The value at which the property is to be set is invalid for the property. This is the case, for example, when the value has an incorrect data type.
Solution	Check the value or expression in the SiVArc faceplate properties/SiVArc properties of the HMI object in the library. Ensure that the resulting value matches the faceplate interface data type (for example, a string is used for an interface object with the "String" type).

7.1.3.37 Warning_PropertyCanNotSetReadOnly

ID	Warning_PropertyCanNotSetReadOnly
Cause	An error occurred when setting a property of a faceplate or a screen object: The property that is to be set is write-protected in the utilized HMI device or is not supported by it. The restrictions depend on the type of HMI device, which means they are different for WinCC Professional and WinCC Advanced HMI devices.
Solution	Avoid the use of screen objects that are not supported by the HMI device.

7.1.3.38 Warning_PropertyCanNotSetReadOnlyDynamicValue

ID	Warning_PropertyCanNotSetReadOnlyDynamicValue
Cause	An error occurred when setting a dynamization of a property of a faceplate or a screen object: The property that is to be set is write-protected in the utilized HMI device or is not supported by it. The restrictions depend on the type of HMI device, which means they are different for WinCC Professional and WinCC Advanced HMI devices, for example.
Solution	Avoid the use of screen objects that are not supported by the HMI device.

7.1.3.39 Warning_PropertyCanNotSetReadOnlyStaticValue

ID	Warning_PropertyCanNotSetReadOnlyStaticValue
Cause	An error occurred when setting a static value of a property of a faceplate or a HMI object: The property that is to be set is write-protected in the utilized HMI device or is not supported by it. The restrictions depend on the type of HMI device, which means they are different for WinCC Professional and WinCC Advanced HMI devices, for example.
Solution	Avoid the use of HMI objects that are not supported by the HMI device.

7.1.3.40 Warning_PropertyHasInvalidAlarmLog

ID	Warning_PropertyHasInvalidAlarmLog
Cause	The specified HMI alarm log to be interconnected with the property of an HMI object does not exist.
Solution	1. Check the SiVArc properties of the faulty HMI object in the library. 2. Check the expression with which SiVArc generates the name of the HMI alarm log to be linked. Ensure that the alarm log entered must be available in the device. Example: In the case of a faulty alarm class, for the copy template used in the library, check the expression in the SiVArc "Alarm log" property > "Printout of tags" column.

7.1.3.41 Warning_PropertyHasInvalidTag

ID	Warning_PropertyHasInvalidTag
Cause	The specified HMI tag to be interconnected with the property of a HMI object does not exist, has the wrong version or has the wrong type.
Solution	Check the SiVArc properties of the incorrect HMI object in the library: Check the expression used by SiVArc to create the name of the HMI tag to be linked. Example: In case of a bad I/O field: Check the expression in the SiVArc property "Process value" in the "Printout of tags" column for the master copy used in the library.

7.1.3.42 Warning_PropertyNotSupported

ID	Warning_PropertyNotSupported
Cause	The configured property of HMI object is not supported for the selected device.
Solution	1. Check the SiVArc properties of the HMI object in the library with other device where the given property is supported. 2. Remove the expression if not required. Example: In case of a discrete alarm of WinCC RT Professional, for the copy template used in the library, check the expression in the SiVArc "Status tag" property > "Printout of tags" column.

7.1.3.43 Warning_Renamed

ID	Warning_Renamed
Cause	Naming conflict between a tag created by the user and a tag generated by SiVArc. Name conflict between a user-created screen and a screen generated by SiVArc. To avoid the loss of data, the object created by the user has been renamed.
Solution	Resolve the naming conflict either by deleting the renamed object or by giving it a different name.

7.1.3.44 Warning_RenamedInstanceOfScreenType

ID	Warning_RenamedInstanceOfScreenType
Cause	There was a name conflict between a user-created instance of a screen type and a SiVArc-generated instance of the screen type. To avoid the loss of data, the user-generated object has been renamed.
Solution	Resolve the naming conflict either by deleting the renamed object or by giving it a different name.

7.1.3.45 Warning_RenamedScreenItem

ID	Warning_RenamedScreenItem
Cause	Naming conflict between a screen object that was created by the user and a screen object generated by SiVArc. To avoid the loss of data, the object created by the user has been renamed.
Solution	Resolve the naming conflict either by deleting the renamed object or by giving it a different name.

7.1.3.46 Warning_RuleImport_CyclicReferenceFoundForGroup

ID	Warning_RuleImport_CyclicReferenceFoundForGroup
Cause	The parent-child relationship between rule groups is not correct in the Excel worksheet.
Solution	Check and correct the relationships of the rule groups in Excel or move the rules and rule groups in the rule editor.

7.1.3.47 Warning_RuleImport_InvalidDeviceTypeValue

ID	Warning_RuleImport_InvalidDeviceTypeValue
Cause	The "Device type" column of the Excel worksheet contains values other than TRUE or FALSE.
Solution	Check and correct the information in the "Device type" column in Excel or change the data in the rule editor.

7.1.3.48 Warning_RuleImport_InvalidDeviceValue

ID	Warning_RuleImport_InvalidDeviceValue
Cause	The "Devices" column of the Excel worksheet contains values other than TRUE or FALSE.
Solution	Check and correct the information in the "Devices" column in Excel or change the data in the rule editor.

7.1.3.49 Warning_RuleImport_NoValidWorksheetFound

ID	Warning_RuleImport_NoValidWorksheetFound
Cause	The format of the Excel worksheets is invalid or the worksheets are not available.
Solution	Verify that the Excel worksheets have the correct names and conform to the correct format.

7.1.3.50 Warning_RuleImport_ObsoleteColumnsFound

ID	Warning_RuleImport_ObsoleteColumnsFound
Cause	The Excel worksheet contains one or more columns that are not present in the rule editor and cannot be imported.
Solution	Create the missing columns in the rule editor before the next import or ignore the alarm

7.1.3.51 Warning_RuleImport_ParentGroupNotFoundForGroup

ID	Warning_RuleImport_ParentGroupNotFoundForGroup
Cause	It is not possible to import a rule group from Excel because the parent rule group does not exist in the rule editor.
Solution	Create the rule group and move the rule group into this group or correct the data in Excel.

7.1.3.52 Warning_RuleImport_ParentGroupNotFoundForRule

ID	Warning_RuleImport_ParentGroupNotFoundForRule
Cause	It is not possible to import a rule from Excel because the parent rule group does not exist in the rule editor.
Solution	Create the rule group and move the rule into this group or correct the data in Excel.

7.1.3.53 Warning_ScreenItemAlreadyExistsInLinkedScreen

ID	Warning_ScreenItemAlreadyExistsInLinkedScreen
Cause	Two objects generated by SiVArc have the same name. The screen object created by a positioning scheme cannot be created if there is a screen object with an identical name in the same screen or in the overflow screens.
Solution	Check the SiVArc "Name" property of the screen objects. One possible solution would be to supplement the "Name" property with an additional SiVArc expression in order to generate unique names.

7.1.3.54 Warning_ScreenItemAlreadyExistsInScreen_2

ID	Warning_ScreenItemAlreadyExistsInScreen_2
Cause	Two objects generated by SiVArc have the same name.
Solution	Check the SiVArc property "Name" of the screen objects.

7.1.3.55 Warning_ScreenItemCanNotCreatedByLib

ID	Warning_ScreenItemCanNotCreatedByLib
Cause	An assigned name for a library has been allocated to an unexpected screen object type. For example, a button has been saved in the master copy with the name "DefaultScreenWindowControl".
Solution	Rename the screen object in the library.

7.1.3.56 Warning_ScreenItemDoesNotFit

ID	Warning_ScreenItemDoesNotFit
Cause	A screen object for which fixed positioning is set in the SiVArc properties editor does not fit or fits only partially on the screen.
Solution	Check the defined values in the "Position" category for the corresponding master copy in the SiVArc properties editor. The registered coordinates plus width or height of the screen object may not exceed the area of the target window. Alternatively, the size of the target window can be increased as appropriate.

7.1.3.57 Warning_ScreenItemIsNotVisibleFromLib

ID	Warning_ScreenItemIsNotVisibleFromLib
Cause	A screen object is to be generated in an HMI device, but the screen object is not supported by the HMI device type used. For example, no screen objects of the S7 Graph Overview type may be used in HMI devices of the WinCC Advanced type.
Solution	Correct the screen object in the screen rule or enter a screen object supported by the HMI device.

7.1.3.58 Warning_ScreenItemNameTooLong

ID	Warning_ScreenItemNameTooLong
Cause	The name of the generated screen object is longer than 128 characters.
Solution	Names for screen objects may not exceed 128 characters. Check the SiVArc "Name" property in the corresponding screen object in the master copy library.

7.1.3.59 Warning_ScreenItemsCanNotMove

ID	Warning_ScreenItemsCanNotMove
Cause	A screen object generated by the user cannot be restored.
Solution	You must create the screen object again after the SiVArc generation.

7.1.3.60 Warning_ScreenSizeChangeForRtAdvanced

ID	Warning_ScreenSizeChangeForRtAdvanced
Cause	When generating a screen for an HMI device of the "WinCC RT Advanced" type, the size of the screen was adjusted automatically. The screen size in HMI devices of the "WinCC RT Advanced" type is fixed at 1024 x 768 pixels.
Solution	Use a master copy in the appropriate size of 1024 x 768 pixels for WinCC RT Advanced.

7.1.3.61 Warning_ScreenWindowControlNotFound

ID	Warning_ScreenWindowControlNotFound
Cause	No user-defined standard ScreenWindowControl was found in the library. Instead, a ScreenWindowControl from the toolbox is used.
Solution	Create a user-defined, standard ScreenWindowControl under the path "Master copy\DefaultScreenWindow-Control" in the library.

7.1.3.62 Warning_TagSettingsForProfessionalDevice

ID	Warning_TagSettingsForProfessionalDevice
Cause	SiVArc tag settings were configured in the project. These are ignored for Runtime Professional.
Solution	--

7.1.3.63 Warning_TagTableNameExists

ID	Warning_TagTableNameExists
Cause	A tag table with the same name already exists in another folder. The generated HMI tag is added to the existing tag table.
Solution	Check the "Tag table" column in the tag rules. Modify the text or the expression to create a unique name for the tag table.

7.1.3.64 Warning_TextEntryTooLong

ID	Warning_TextEntryTooLong
Cause	The text of a text list entry could not be generated for a particular language because the text to be placed has too many characters.
Solution	In the PLC program, navigate to the corresponding SiVArc plug-in editor and abbreviate the text in the corresponding text definition to the permitted length.

7.1.3.65 Warning_TextlistCreationIncompleteDueToNoMatchingTagForMatchedFunctionBlockVariables

ID	Warning_TextlistCreationIncompleteDueToNoMatchingTagForMatchedFunctionBlockVariables
Cause	The corresponding tag from the PLC symbol table could not be found for the synchronized program block tags based on a regular expression in the text list master copy.
Solution	All values of the synchronized program block tags must match the values in the PLC symbol table. If you configure tags of different types, make sure that the corresponding tags in the PLC tag table are configured as well for the program block tags.

7.1.3.66 Warning_TextlistCreationIncompleteDueToNonMatchingDataBlockCallers

ID	Warning_TextlistCreationIncompleteDueToNonMatchingDataBlockCallers
Cause	The regular expression configured in the text list master copy is resolved to at least one tag of the function block. However, the synchronized program block tags have zero or default values.
Solution	The values of the synchronized program block tags based on a text list master copy must contain valid tags of the PLC symbol table. Ensure that the values do not contain default values or values of an invalid type.

7.1.3.67 Warning_TextlistCreationIncompleteDueToNonMatchingSymbolTableTags

ID	Warning_TextlistCreationIncompleteDueToNonMatchingSymbolTableTags
Cause	There is at least one valid value that cannot be found in the PLC symbol table for the synchronized program block tags based on a regular expression in the text list master copy.
Solution	Ensure that there are no values in the program block tags for which there are no corresponding tags in the PLC symbol table.

7.1.3.68 Warning_UndefinedCycleTime

ID	Warning_UndefinedCycleTime
Cause	The data type of the program block is configured with an acquisition cycle, which is not available on the generated HMI device.
Solution	Before the SiVArc generation, make sure that the acquisition cycles used are configured on the target device.

7.1.3.69 Warning_UndefinedCycleTimeForBlock

ID	Warning_UndefinedCycleTimeForBlock
Cause	The program block is configured with an acquisition cycle, which is not available on the generated HMI device. This can occur when the "Use Common Configuration" option is selected.
Solution	Before the SiVArc generation, make sure that the acquisition cycles used are configured on the target device.

SiVArc Readme

8

8.1 Security information

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.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, visit

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

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 under

http://www.siemens.com/industrialsecurity. (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

Network drive

Ensure that network drives are protected from unauthorized access in your network infrastructure and computers.

Communication via Ethernet

In Ethernet-based communication, end users themselves are responsible for the security of their data network. Proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to overload of the device.

8.2 Notes on use

Contents

Information that could no longer be included in the online help and important information about product features.

Objects revert to default position after SiVArc upgrade

Before project upgrade from TIA Portal version 13 to version 14, and higher versions, it is recommended to perform SiVArc generation in TIA Portal version 13 with all PLCs. Failing to do the recommended step, after project upgrade, any changes performed such as choosing of additional PLCs would result in shifting of screen objects to its default position.

Index

G

H

I

K

L

M

N

O

P

R

S

T

U

V