

SIMATIC Visualization Architect

System Manual


<u>Security information</u>	1
<u>Data protection</u>	2
<u>Basics</u>	3
<u>Installation</u>	4
<u>Elements and basic settings</u>	5
<u>Working with SiVArc</u>	6
<u>SiVArc Openness</u>	7
<u>Reference</u>	8
<u>Tooltips</u>	9


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.

Table of contents

1	Security information	9
2	Data protection	11
3	Basics	13
3.1	Introduction.....	13
3.2	Applications.....	14
3.3	Example: Using SiVArc to generate the visualization	15
3.4	Example: Using SiVArc to generate tags.....	18
3.5	Configuring an HMI solution with SiVArc.....	19
4	Installation.....	23
4.1	Installing SiVArc	23
5	Elements and basic settings	25
5.1	SiVArc Settings.....	25
5.2	User Management Control (UMAC) in SiVArc	26
5.3	SiVArc editors	26
5.3.1	"Screen rules" editor.....	26
5.3.2	"Tag rules" editor.....	28
5.3.3	"Text list rules" editor	29
5.3.4	"Alarms rules" editor.....	30
5.3.5	"SiVArc expressions" editor	32
5.3.6	"Copy rules" editor	33
5.3.7	"Generation matrix" editor.....	35
5.3.8	Expression resolver	38
5.3.9	Generation overview.....	41
5.3.10	Energy Suite Generation	42
5.4	SiVArc in the HMI editors.....	44
5.4.1	Property configurator	44
5.4.2	"SiVArc properties" tab	47
5.4.3	"SiVArc events" tab.....	50
5.4.4	"SiVArc animations" tab.....	51
5.4.5	"Generation overview" tab.....	52
5.5	SiVArc in PLC editors.....	53
5.5.1	Support for software units	53
5.5.2	SiVArc support for program blocks using Structured Control Language.....	54
6	Working with SiVArc	57
6.1	Plan layout.....	57
6.1.1	Positioning schemes	57
6.1.1.1	Overview for positioning of generated objects.....	57
6.1.1.2	Positioning according to defined schemes	59

6.1.1.3	Fixed positioning of the generated object.....	68
6.1.1.4	Free positioning	69
6.1.1.5	Nesting depth	70
6.1.2	Overflow mechanisms.....	71
6.1.3	Supported devices	76
6.1.4	Designing a layout	76
6.1.5	Creating user-defined positioning scheme	78
6.1.6	Configuring overflow screens without screen objects	81
6.1.7	Example: Using a layout with free positioning.....	82
6.1.8	Example: Using overflow mechanisms	82
6.1.9	Example: Using a dynamic layout	85
6.1.10	Example: Using a combined layout	87
6.1.11	Example: Using generated screen navigation	88
6.2	Creation of generation templates	90
6.2.1	Generation templates in SiVArc	90
6.2.2	Supported HMI objects.....	96
6.2.3	Sources for texts	98
6.2.3.1	Overview of text source in SiVArc project.....	98
6.2.3.2	SiVArc texts.....	101
6.2.4	Supported objects in the user program	103
6.2.5	SiVArc scripting	103
6.2.6	SiVArc expression.....	106
6.2.6.1	Overview of SiVArc expressions	106
6.2.6.2	SiVArc tags	107
6.2.7	Requirements for a generation template.....	109
6.2.8	Parameterization concept.....	112
6.2.8.1	Example: Generating Unified faceplates.....	112
6.2.8.2	Example: Creating a parameterization concept	112
6.2.8.3	Assignment of block and generation template	113
6.2.8.4	Structure of SiVArc expressions.....	115
6.2.9	Influence of the user program on a generation template.....	117
6.2.10	Influence of multilingualism on a generation template	120
6.2.11	Storage strategies for generated objects	122
6.2.12	Example: Achieving high flexibility	125
6.2.13	Example: Achieving high reusability	127
6.2.14	Example: Create generation template for screen windows	127
6.2.15	Example: Generating template screen using copy rule	130
6.2.16	Example: Generating HMI screen using template property	131
6.2.17	Example: Create generation template with animation	133
6.2.18	Example: Create generation template with event configuration.....	135
6.2.19	Example: Create generation template with script configuration	136
6.2.20	Example: Creating generation templates for text lists.....	138
6.2.21	Example: Create generation template for a text list for block parameters.....	141
6.2.22	Example: Generating pop-up screens and their use.....	142
6.2.23	Example: Generating faceplates with animations.....	144
6.2.24	Example: Generating "Position" animation for faceplates.....	149
6.2.25	Example: Creating generation templates for trend views.....	150
6.2.26	Example: Hardware configuration for screen.....	152
6.2.27	Example: Hardware configuration for alarms	154
6.2.28	Creating a generation template for a screen	155
6.3	Creating rules	156

6.3.1	SiVArc rules.....	156
6.3.2	Creating SiVArc rules.....	160
6.3.3	Using SiVArc scripting in SiVArc rules.....	163
6.3.4	Processing of rules.....	165
6.3.5	Generating tags.....	168
6.3.6	Use of copy rules.....	171
6.3.7	Correlation between SiVArc expressions and conditions.....	174
6.3.8	Principle of the tag generation.....	175
6.3.9	Avoiding conflicts during generation.....	177
6.3.10	Creating tag rules.....	180
6.3.11	Creating a screen rule.....	181
6.3.12	Creating text list rules.....	183
6.3.13	Creating alarm rules.....	184
6.3.14	Creating copy rules.....	186
6.3.15	Example: Creating screen rule with condition.....	187
6.3.16	Example: Organizing screen and text list rules.....	188
6.3.17	Example: Adapting tag names.....	190
6.3.18	Editing and managing SiVArc rules.....	192
6.3.19	Exporting and importing SiVArc rules.....	194
6.3.20	Setting up know-how protection for a SiVArc project.....	197
6.4	Generate visualization.....	198
6.4.1	Basics on generation.....	198
6.4.2	Subsequent changes.....	202
6.4.2.1	Changing generated objects.....	202
6.4.2.2	Changing SiVArc rules.....	205
6.4.2.3	Identifications in the SiVArc project.....	206
6.4.3	Generating visualization.....	207
6.5	Checking result.....	210
6.5.1	Check the result.....	210
6.5.2	Using the generation matrix.....	213
6.5.3	Example: Using the generation matrix.....	216
7	SiVArc Openness.....	217
7.1	Introduction.....	217
7.2	SiVArc service properties.....	217
7.3	Copying rules or groups from library.....	219
7.4	Finding a screen rule and screen rule group.....	220
7.5	Deleting rules and rule groups.....	221
7.6	UMAC set up for openness.....	222
7.7	SiVArc generation.....	222
8	Reference.....	225
8.1	SiVArc objects.....	225
8.1.1	PLC Tags.....	225
8.1.2	I/O device.....	225
8.1.3	Software units.....	225
8.1.4	Object hierarchy.....	226
8.1.5	Block (Panels, Comfort Panels, RT Advanced, RT Professional).....	227

8.1.6	DB (Panels, Comfort Panels, RT Advanced, RT Professional)	228
8.1.7	HMIApplication (Panels, Comfort Panels, RT Advanced, RT Professional)	229
8.1.8	HMIDevice (Panels, Comfort Panels, RT Advanced, RT Professional)	229
8.1.9	HMITag (Panels, Comfort Panels, RT Advanced, RT Professional)	230
8.1.10	LibraryObject (Panels, Comfort Panels, RT Advanced, RT Professional)	231
8.1.11	ModuleBlock (Panels, Comfort Panels, RT Advanced, RT Professional)	232
8.1.12	Parameters (Panels, Comfort Panels, RT Advanced, RT Professional)	233
8.1.13	S7Control (Panels, Comfort Panels, RT Advanced, RT Professional)	233
8.1.14	SubModuleBlock (Panels, Comfort Panels, RT Advanced, RT Professional)	234
8.1.15	StructureBlock (Panels, Comfort Panels, RT Advanced, RT Professional)	235
8.1.16	TagNaming (Panels, Comfort Panels, RT Advanced, RT Professional)	236
8.2	SiVArc object properties	237
8.2.1	Assigned	237
8.2.2	Comment	237
8.2.3	FolderPath	238
8.2.4	HMITagPrefix	239
8.2.5	IndexEndChar	239
8.2.6	IndexStartChar	240
8.2.7	InitialValue	240
8.2.8	Name	240
8.2.9	NetworkComment	241
8.2.10	NetworkTitle	241
8.2.11	Number	242
8.2.12	SeparatorChar	242
8.2.13	SymbolComment	243
8.2.14	SymbolicName	243
8.2.15	Title	244
8.2.16	Type	245
8.2.17	Value	245
8.2.18	Version	245
8.3	SiVArc object properties	246
8.4	Functions	248
8.4.1	Functions in SiVArc	248
8.4.2	"Contains" function	248
8.4.3	"EndsWith" function	248
8.4.4	"Format" function	249
8.4.5	"FormatNumber" function	249
8.4.6	Function "InStr"	251
8.4.7	Function "IsDefined"	252
8.4.8	Function "LBound"	252
8.4.9	Function "Left"	253
8.4.10	Function "Len"	253
8.4.11	Function "LTrim"	254
8.4.12	Function "Max"	254
8.4.13	Function "Mid"	254
8.4.14	Function "Min"	255
8.4.15	Function "Replace"	255
8.4.16	"Right" function	256
8.4.17	Function "RTrim"	256
8.4.18	"Split" function	257
8.4.19	"StartsWith" function	257

8.4.20	"StrComp" function	258
8.4.21	"TrailNum" function.....	258
8.4.22	"Trim" function.....	259
8.4.23	"UBound" function	259
8.5	Operators	259
8.6	String indexing	261
8.7	If conditions.....	261
8.8	Supported data types for PLC tags	262
8.9	Supported system functions for faceplates	264
8.10	Editing the view in the SiVArc editors	264
8.11	Picture legends	266
9	Tooltips	267
9.1	UMAC	267
9.1.1	CTHMISivarc	267
	Index.....	269

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://support.automation.siemens.com>)

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.

Data protection

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). This means for this SIMATIC product, the product does not process / save any personal information.

Basics

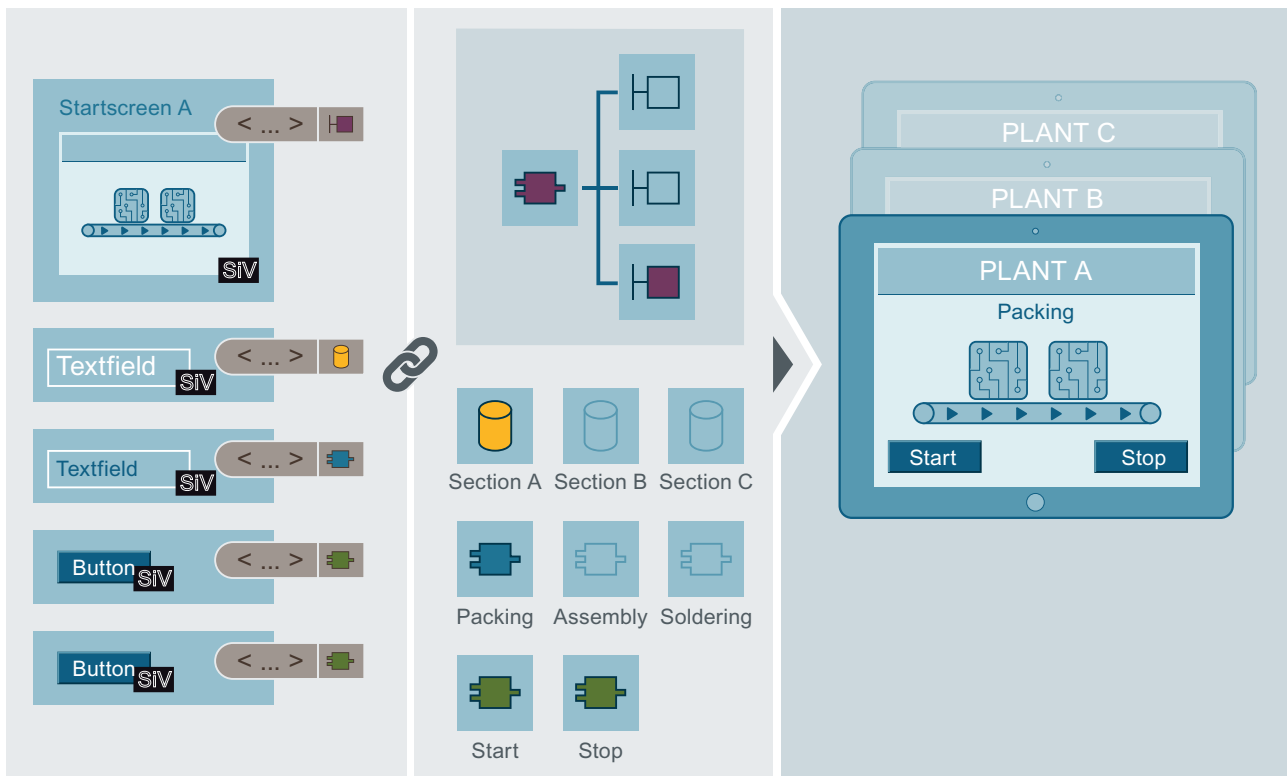
3.1 Introduction

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.

In the SiVArC editors, you can perform copy, paste, undo, redo of system or script functions. Using global search, you can search for system or script function names or its parameter values. The search results contain a "Go to" link, which navigates to the searched items.

SiVArC settings are available as part of global settings.

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

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 103)

Configuring an HMI solution with SiVArC (Page 19)

Overview of SiVArC expressions (Page 106)

Picture legends (Page 266)

Creation of generation templates (Page 90)

3.2 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:

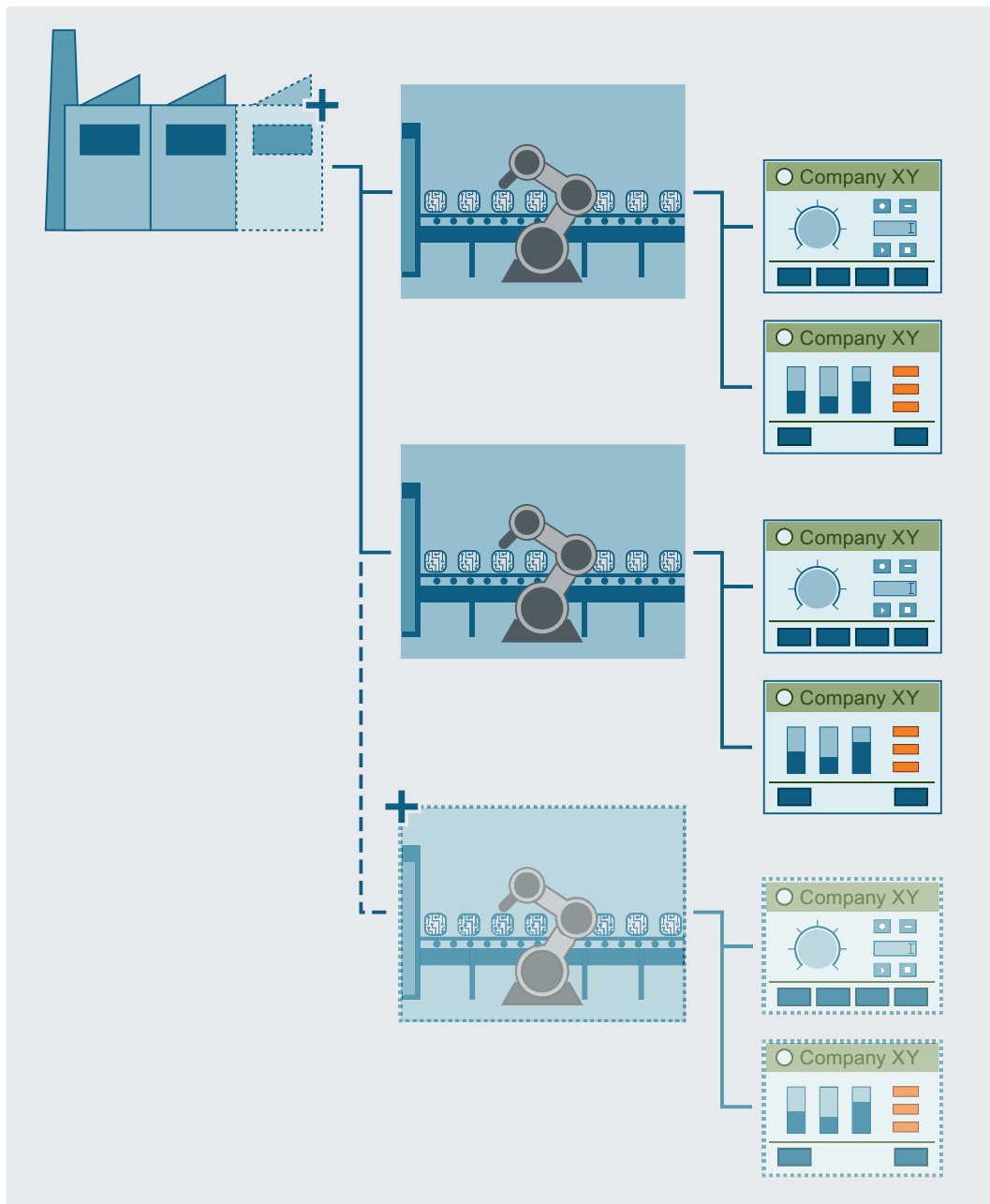
- You have configuration experience in STEP 7 and WinCC.
- You have a basic knowledge of Visual Basic Script (VBS).

3.3 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:

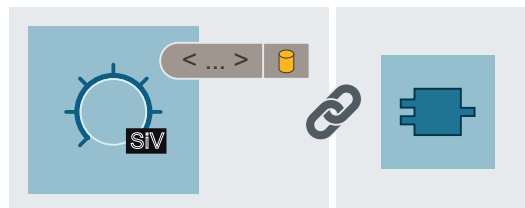
- Set PLC tags to "Accessible from HMI"
- Check the text sources in STEP 7 and make them consistent, if necessary
- Optimize the storage of the program blocks in the project tree
- Expand existing libraries

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.

3.4 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

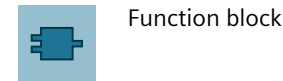
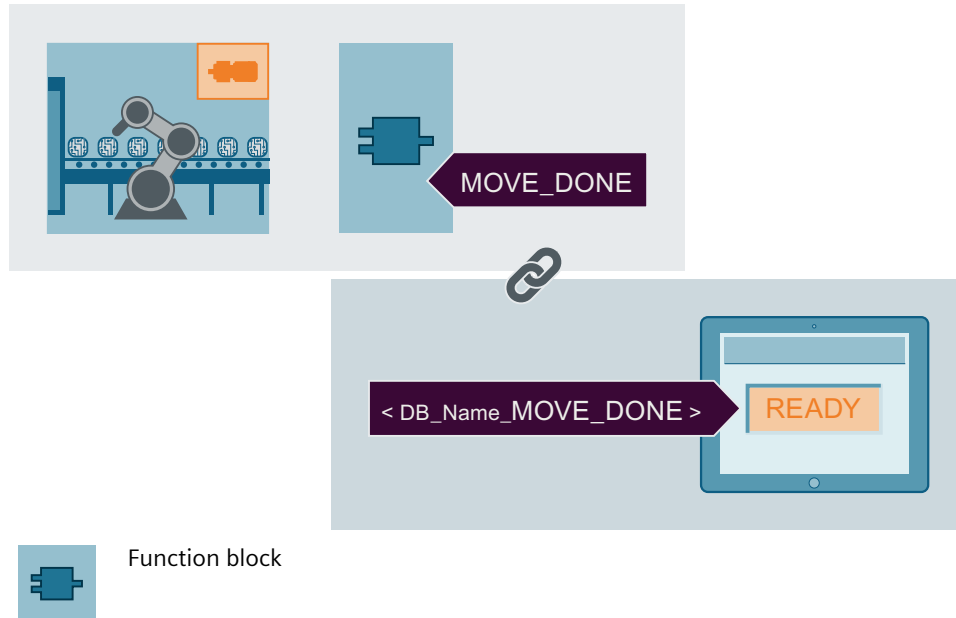
Tag names

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.



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 175)

3.5 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

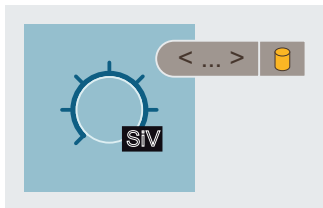
- The plant is a standard facility.
- The structured user program is created.
- A visualization and operating concept is created.
- Standard blocks in the user program are accessible via libraries.

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

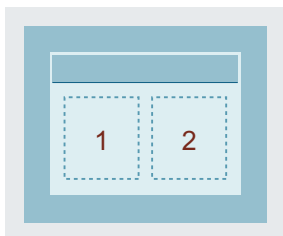
Procedure

To generate HMI solutions with SiVArc, follow these steps:

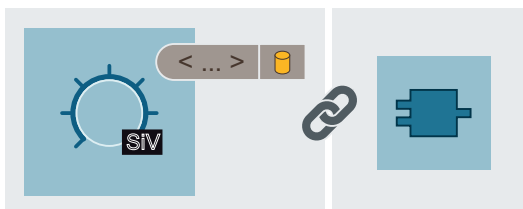
1. Design the layout.
 - Which HMI devices are in use?
 - How is the corporate design implemented in the screen?
 - How many generation templates for screens are required?
 - How many positioning schemes are required?
2. Define which external tags are going to be generated by SiVArc.
3. Create the generation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



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



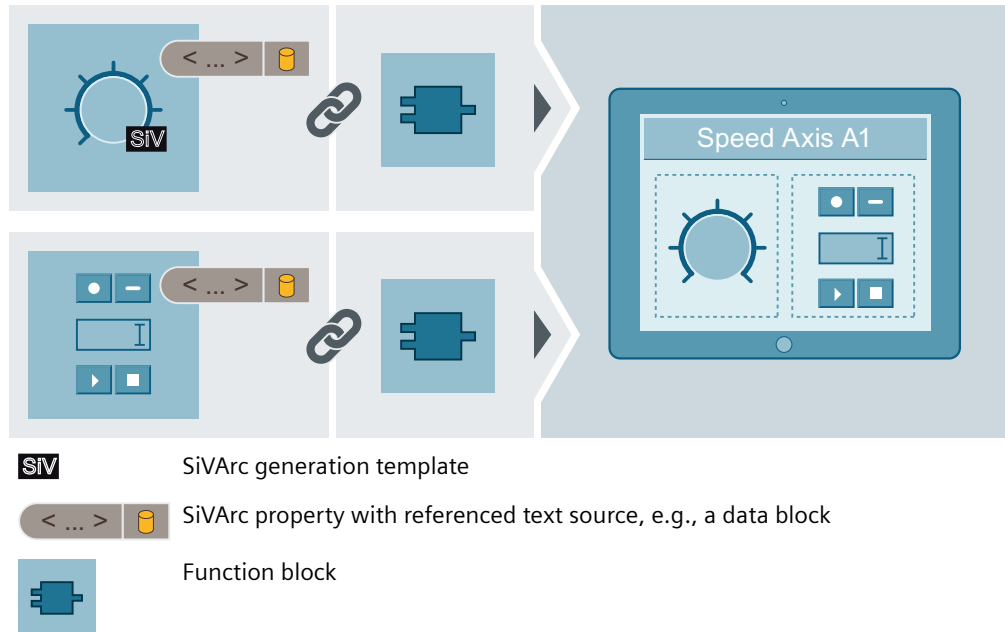
5. You create screen rules to link the generation templates with the function blocks.



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.



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.

You can search SiVArc configured expression using global search.

See also

Plan layout (Page 57)

Creation of generation templates (Page 90)

Creating rules (Page 156)

Generate visualization (Page 198)

Installation

4.1 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 or WinCC Unified.

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.

Requirement

- STEP7 Professional V17.0 is installed.
- SIMATIC WinCC Professional V17.0 or SIMATIC WinCC Advanced, SIMATIC WinCC Unified V17.0 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.

4.1 Installing SiVArc

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
- SIMATIC WinCC Unified

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

5.1 SiVArc Settings

Generation Settings

You can enable the below mentioned under Settings > SiVArc > Generation > "Warning settings":

- Hide warnings if screen object already exists - applicable if a screen object is already available in the generated screen
- Hide warning if variable not defined in multi lingual context - applicable for variables with language specific context
- Hide warning if screen size changed - applicable for cross device generation with screen size mismatch
- Hide warnings if text entries could not be resolved - applicable if no text is defined under "S7 plug-in editor > Text definitions"

Library type settings

Use default version of library type for SiVArc generation - allows you to use the default version of library type in faceplates for SiVArc generation.

Acquisition cycle and mode settings for WinCC Unified

A project consisting of Unified HMI device and Advanced device is configured with block tag member settings or project level settings; and if acquisition cycle supported by Advanced device:

- By default T1s will be configured for HMI tags in Unified devices

A project consisting of Unified HMI device and Advanced device is configured with block tag member settings or project level settings; and if acquisition cycle supported by Unified device:

- By default 1s will be configured for HMI tags in Advanced devices

Acquisition mode can be configured at project level or at a block level. At project level, the types are:

- Cyclic in operation
- On demand

At a **PLC block level** for Tag member settings, you can choose "**Use common configuration**" option to set the acquisition cycle and mode to be same across the input tags.

For more information on WinCC Unified device functionality, refer *TIA portal user guide*.

SiVArc supported objects configured for WinCC Unified devices are termed as Unified objects. Eg: Unified button is used in the master copy of a screen rule. All changes made to SiVArc objects are applicable to Unified objects also. For Unified devices, you can configure screen plug-in properties. SiVArc supported expressions like `HmiDevice.Name`, `HmiDevice.Type`, `HmiApplication.Name`, `HmiApplication.Type`,

`HmiDevice.Resolution.Width`, `HmiDevice.Resolution.Height` can be configured in the plug-in properties.

5.2 User Management Control (UMAC) in SiVArc

About role of UMAC in SiVArc

By using the User Management Control (UMAC), only authorized users will be allowed to access SiVArc related features. To be able to use User Management Control in SiVArc, you need SiVArc admin rights. This right can be assigned by the user with Engineering standard (ES) administrator role. For more information on using User Management Control, please refer to the TIA Portal online help.

Note

- It is mandatory for a user to assign the Engineering Standard role to access the Engineering properties, and features. Failing to provide this access, the user will not be able to open the project, and displays error message.
- If a user has Engineering administrator rights, but no SiVArc administration rights, tries to close the project and re-opens, an unauthorized access related error message displays.

NOTICE

Know-how protection

While converting TIA portal V15 and lower version projects to V15.1, the credentials set with "know-how protection" feature will be automatically deleted. Ensure you protect the project through UMAC.

5.3 SiVArc editors

5.3.1 "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. WinCC Unified screens support SiVArc expressions.
- **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.

Any editor that supports expressions will save the values to the corresponding property by default when you switch the editors without manually saving the expression.

You can add a new rule/rule group at any desired index within any rule editor by right clicking on the rule > insert a new rule/insert a new rule group.

1. While adding new rule/rule group at an index point, the new rule/rule group will be inserted immediately below the selected index.
2. While adding a new rule/rule group by selecting a group, the new rule/rule group will be added immediately within the selected rule group at the first index.

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". The maximum character limit for "Condition" is 1000.

See also

SiVArc tags (Page 107)

Editing the view in the SiVArc editors (Page 264)

Exporting and importing SiVArc rules (Page 194)

Editing and managing SiVArc rules (Page 192)

5.3.2 "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
- Tool bar includes device specific columns displays the available HMI devices and types within the TIA portal project. The same options are available under "SiVArc Properties" area.

	<input checked="" type="checkbox"/>	Name	Index	Tag group hierarchy	Tag table	Condition
1	<input checked="" type="checkbox"/>	Tag rule	0	HmiTag.DB.FolderPath	HmiTag.DB.SymbolicName	
2		<<create new rule>>				

See also

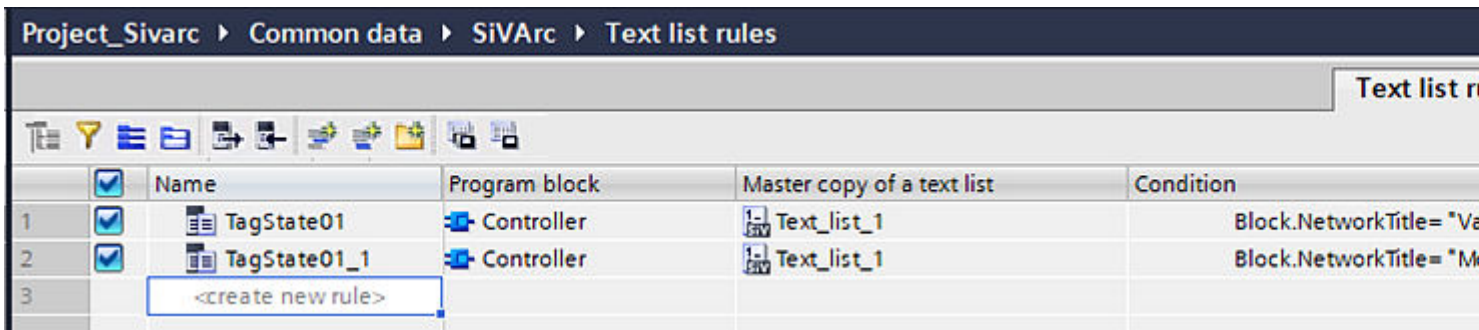
- SiVArc object properties (Page 246)
- Editing the view in the SiVArc editors (Page 264)
- Exporting and importing SiVArc rules (Page 194)
- Editing and managing SiVArc rules (Page 192)
- "Copy rules" editor (Page 33)
- Checking result (Page 210)

5.3.3 "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
- Tool bar includes device specific columns displays the available HMI devices and PLCs within the TIA portal project. The same options are available under "SiVArc Properties" area.



See also

Editing the view in the SiVArc editors (Page 264)

Exporting and importing SiVArc rules (Page 194)

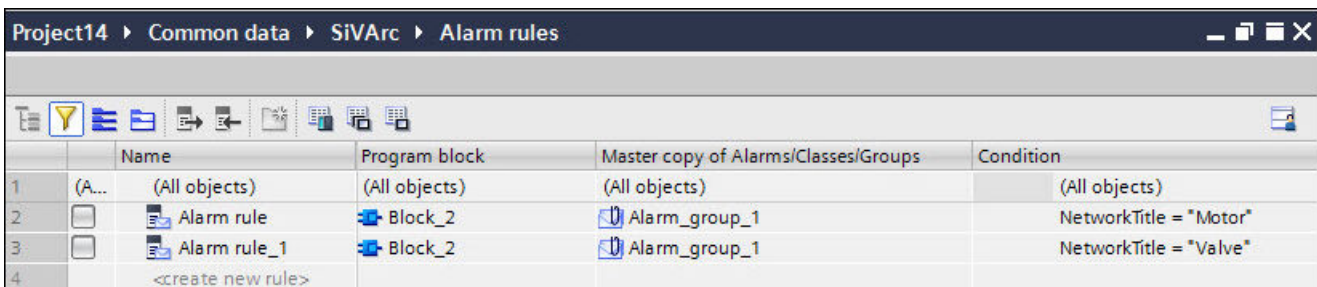
Editing and managing SiVArc rules (Page 192)

5.3.4 "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.
- Rule trigger- 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 functionalities are available in "Alarm rules" editor:

- Intellisense support
- Auto complete (drag the corners of the cell to auto complete values)

- Cut and copy/paste options
- Create single rule with identical entries when only objects differ. You can choose to select the objects under "Master copy of Alarms/Classes/Groups". In case of multiple selections of alarm objects, the editor displays the entry as "Multiple Selection", and for single selection, the object name is displayed.

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. For more details about alarm rule generation, see section "Generation Matrix". (Page 35)

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, and 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 184)

"Generation matrix" editor (Page 35)

5.3.5 "SiVArc expressions" editor

Description

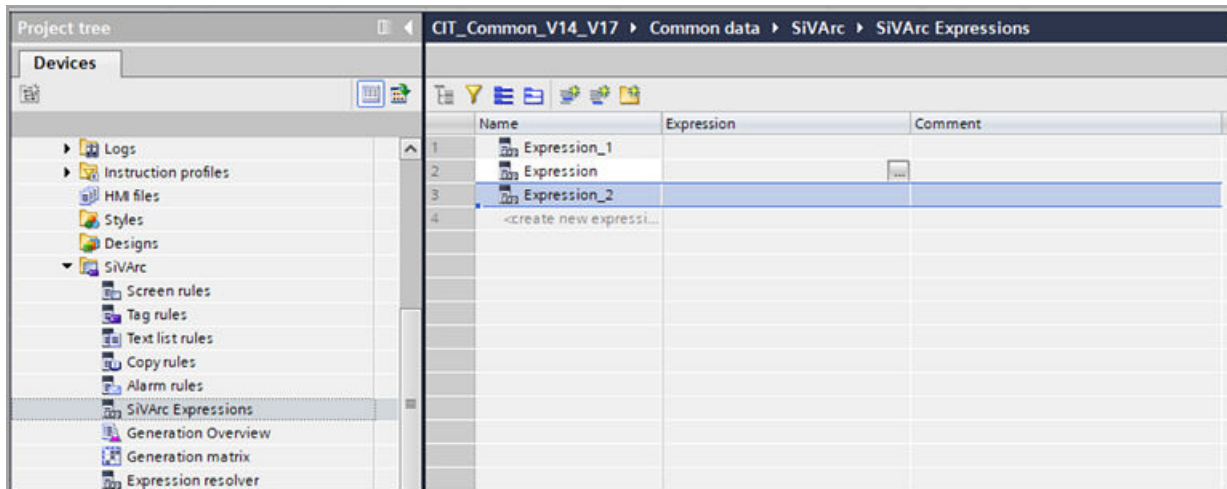
SiVArc expressions editor allows you to add, define, and group expression which can be used within the project library and the global library. The expressions can be used globally by invoking the name of the expression.

The "SiVArc expressions" is made up of:

- Name:
Unique name of the expression
- Expression:
Enter the SiVArc supported expressions. Expression editor supports `StringPos(Parameter1,Parameter2)`.
Case 1:When you configure an expression using `StringPos` function: `StringPos (Parameter1, Parameter2)`; system returns the position of first char of the Parameter2 (substring) in Parameter1 (main string).
example: `StringPos("Sivarc_Project","Project")=8` as ("Project" is a substring of "Sivarc_Project" – Returns 8, the position of 'P' in original string)
Case 2: When you configure an expression using "StringPos" in which the parameters to extract characters with case mismatch Or string unavailability; System returns 0.
example: `StringPos("Sivarc_Project","Por")=0` as ("Por" is not a substring of "Sivarc_Project" – Returns 0
Case 3: When you configure an expression using "StringPos" where both the strings or any one of the strings in the "StringPos" function as empty, then the function will return -1.
example: `StringPos("", "")=-1` as (Both the Parameters are empty)
- Comment:
Individual comment for SiVArc expression

Tool bar includes group, filter, collapse all expanded rows, expand all hierarchical rows, insert above, insert below. You can perform the following in the expression editor:

1. In the expression editor, click "Create new expression" to add a new expression.
2. You can choose to select the expressions and group them by clicking the group option in tool bar.
3. You can create a group by right clicking on the expressions row > "Add a new expression group".
4. You can create a group within the existing group by right clicking on the group > "Insert a new expression group".
5. You can insert new expressions above/below any selected row using the tool bar option.



You can drag and drop the created expressions as objects within the Project/Global library. In such cases, the object is made available within the object picker of the expression editor. The expression can be reused in multiple instances. These are termed as global expressions. You can use the global expressions in the condition column of rule grid for screens, alarms, tags, copy rule.

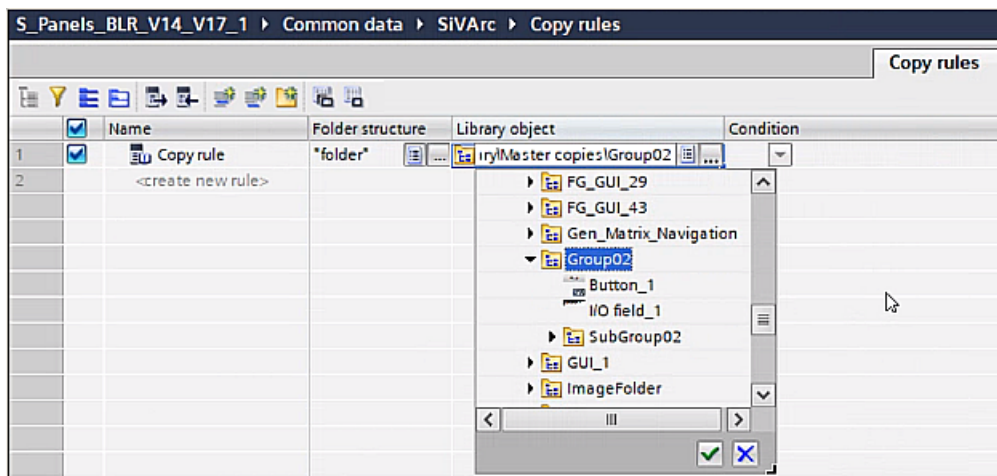
5.3.6 "Copy rules" editor

Description

A rule is made up as follows:

- Name
Unique name of copy rule
- Folder structure
User defined folder structure can be done by either manually or with expression support. This applies to library objects where creation of folders are supported.
If the value in "Folder structure" column is empty, then SiVArc generates all the objects configured from the root node till nested depth of level '4' under PNV defined default folder. If you want to replicate the objects with folder structure as configured in project library, then you can configure folder structure column through supported expression.
You can also choose to configure any desired folder structure for SiVArc generation in the folder structure column.
- Project Library
The folder structure consists of root folder placed at level "0" , and the subsequent folders within root folder as nested folders, level "1", level "2" and so on. SiVArc considers nested depth of the folders to only level "4" for generation.

- Library object
Master copy or type of an object that is generated or a library folder that contains **library objects > layout fields**.
If the layout field is in edit mode, the previous released layout field will be considered. The master copy or the library type must be contained in the project library. Unified screen is available as objects within the object picker, and upon SiVArc generation, the screen is generated on the unified device only. Graphics is supported as a library object.
- Condition (optional) - SiVArc expression that is evaluated when processing copy rule. If no condition is specified, the copy rule is always executed.
- 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. For Unified devices, if copy rule uses system text lists, during SiVArc generation an error is thrown.
- 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 264)
- Exporting and importing SiVArc rules (Page 194)
- Editing and managing SiVArc rules (Page 192)

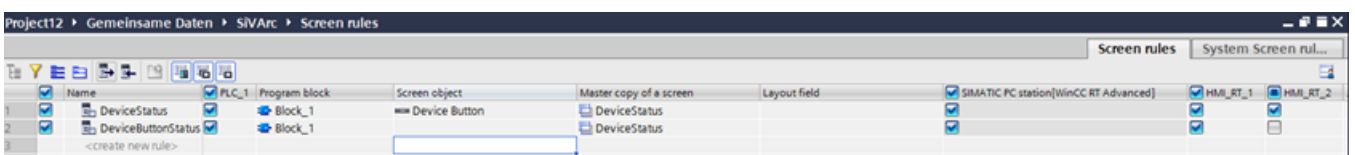
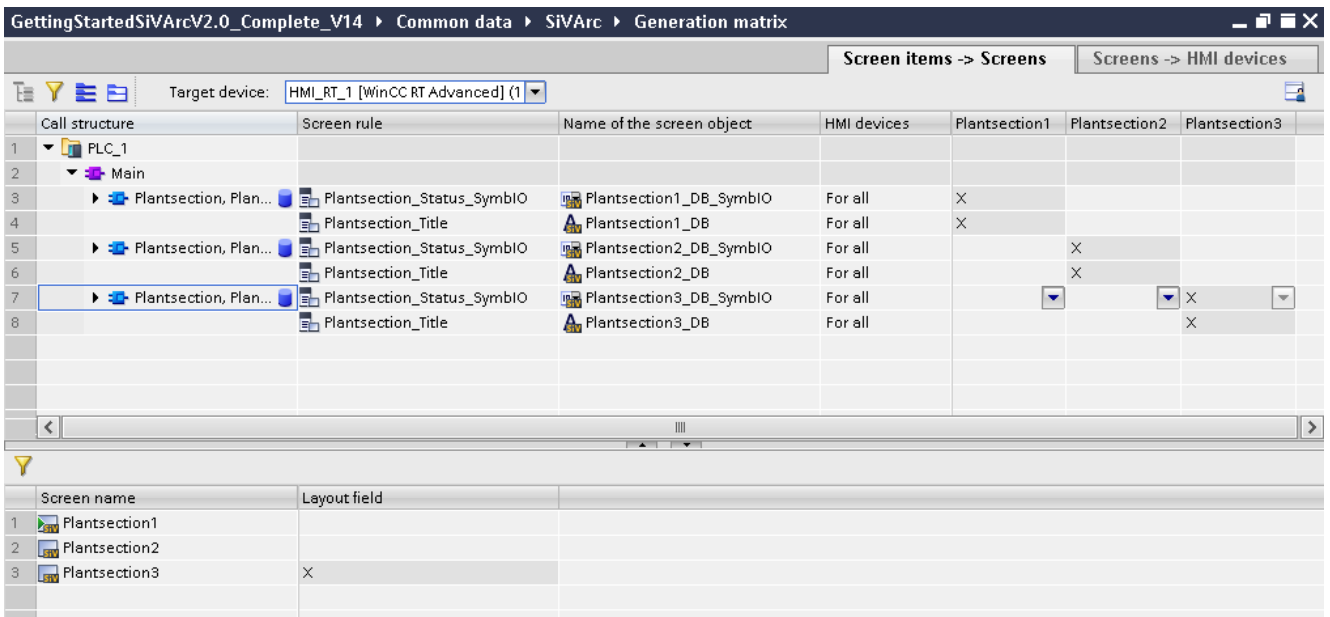
5.3.7 "Generation matrix" editor

Tab "Screen objects -> Screens"

In the toolbar of the editor, you select the HMI device (legacy/WinCC Unified) 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.

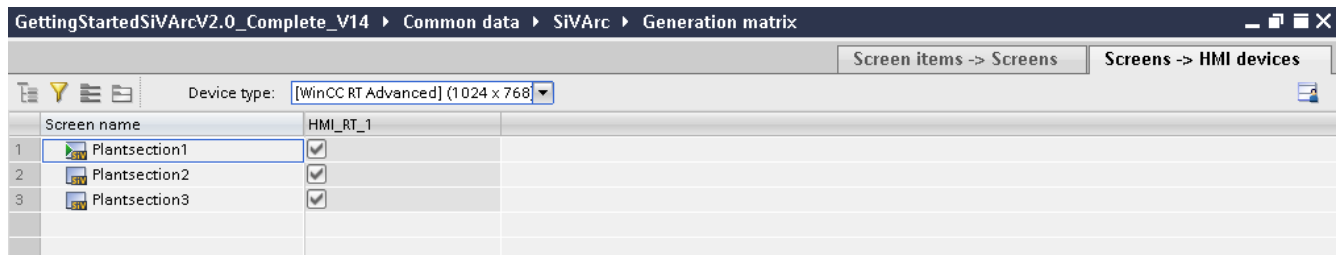


"Screens -> HMI devices" tab

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 (including WinCC Unified) A separate column is shown for each HMI device. The columns are sorted alphabetically.

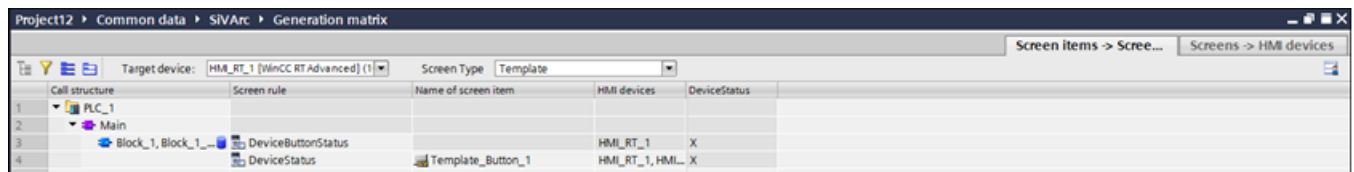


Template and Pop-up Screens

Generation matrix consists of two functions:

allows you to assign template/pop-up screens or screen objects to different target HMI devices. You can assign template/pop-up screen objects to different target HMI device similar to screens. The toolbar of the editor displays "Screen Type" drop-down option, consisting of Screen, Template, and Pop-up as shown in the screen shot below:

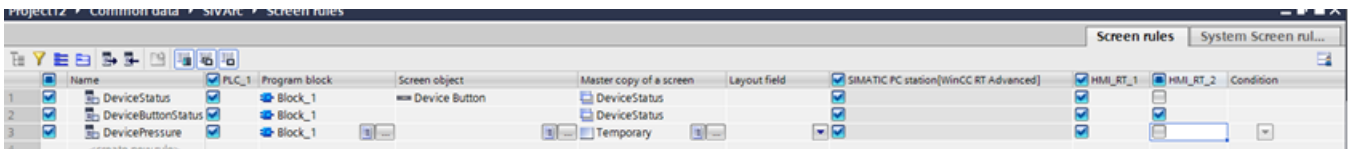
- Matrix 1 - Screen items -> Screens - Allows you to assign screen items on screens of different HMI devices
- Matrix 2 - Screens-> HMI devices - Allows you to assign screens to different HMI devices



Scenario: Navigating objects to different HMI devices without accessing screens

A commissioning engineer would choose to move screen objects to different target HMI devices using SiVArc visualization only, since he is restricted from accessing screen rules. Let us consider the below screen rules configuration:

1. Configure screen rules as mentioned below:
 - DeviceButtonStatus containing template screen only generated in HMI device 1
 - DeviceStatus containing the screen object "Template_button_1" in it, and generated in HMI device 1

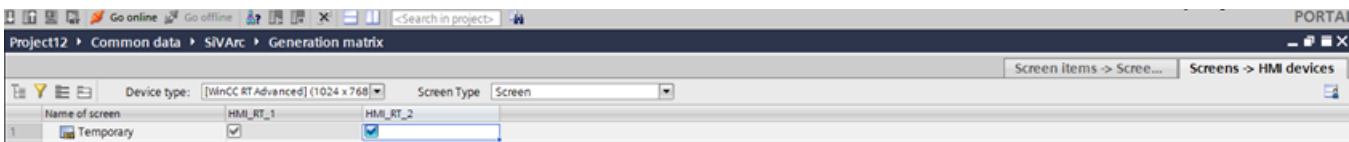


2. Upon SiVArc generation, "DeviceButtonStatus" screen with "Device Button" screen item gets generated on HMI_RT_1, and "Device Status" on HMI_RT_2.
3. Create a target screen with "Device Button" screen item placed within. Configure events with "Click" as "Activate Screen" directing to the target screen.

Note

While configuring screens, you can configure events for a screen item directly. But, with templates and pop-up screens, you must use a target screen to navigate the screen item to the template screen. Target screen is necessary, since for templates and pop-up screens, events cannot be configured directly.

4. Configure a target screen as "Temporary" In "Screen rules", and choose the target HMI device as HMI_RT_1.
5. Click SiVArc settings > "Matrix Settings" > "Generate navigation objects".
6. Goto Generation matrix > "Screens-> HMI devices", select "screen Type" as "Screen", and select the checkbox HMI_RT_2 as shown in screenshot below:



7. Perform SiVArc generation on HMI device 2. Observe the following:
 - The screen item created in step2 will be moved to the screen under folder "Screen management".
 - The screen with the screen item will be generated, and is available under folder "Screens".

The template screen item "Device Button" is moved to template screen "DeviceButtonstatus" through the target screen, i.e "Temporary" in Screens -> HMI devices".

Note

- The above procedure is applicable to pop-up screens also.
- In the case of copy rules, after modifying the screen rule, you must manually set the device for the respective modified screen in Generation matrix (including navigation objects).

5.3.8 Expression resolver

Description

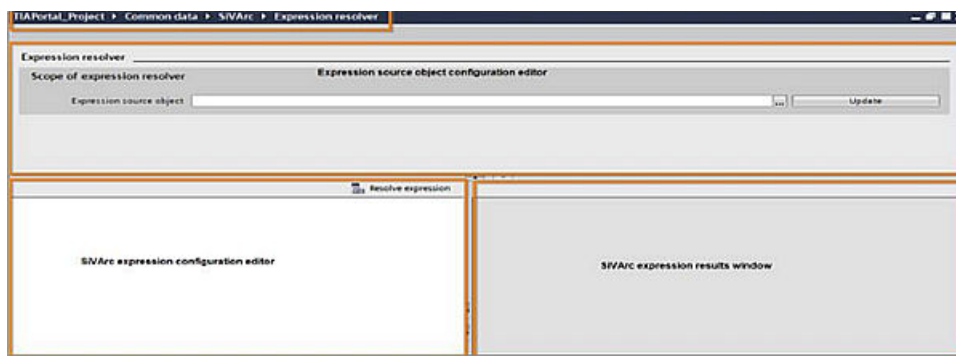
Expression resolver lets you know the resolved value of the SiVArc expression for the given expression source object. The entered expression is resolved, and the results are displayed on the fly without SiVArc generation in the editor.

"Expression source object" can be specific to a block instance, HMI device, IO device, Library object.

Expression resolver editor is categorized into 3 areas:

1. Expression source object configuration editor
2. SiVArc expression configuring editor
3. Expression results window

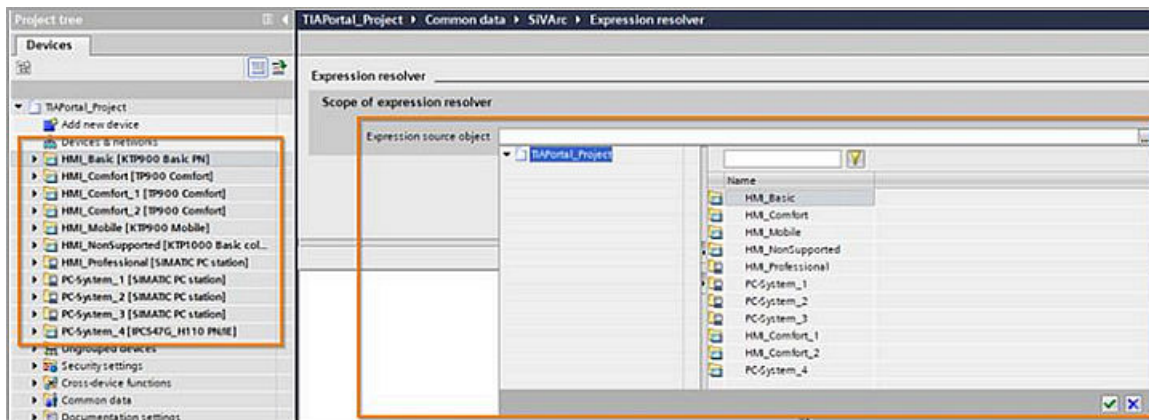
Below screenshot shows the Expression resolver editor:



Source object configuration

Expression resolver supports only block instance and HMI device as an expression source object.

HMI device as source object - A project configured with HMI devices, upon selecting the project node in the left side/hierarchical view, the object picker displays the available HMI devices in the right side as shown in the below screenshot:



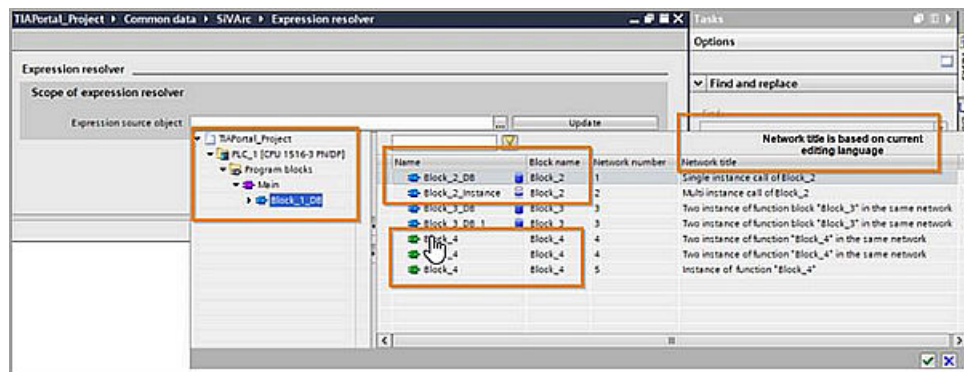
Program block call as an expression source object

Object picker shows the call structure of PLCs configured in the TIA portal project.

When an organization block or a block call is selected in the tree view (left side), the right side of the object picker shows available program block calls in the selected call in columns such as:

- Name - displays the instance name in case of function block call and block name in case of function call
- Block name - displays the name of block associated with the call
- Network number - displays the number of network where the current call is available
- Network title for the current call is shown if the configuration exists in PLC program for current editing language

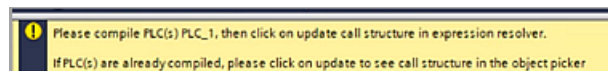
You can choose to use filters on the columns. Below screenshot shows the object picker with the above mentioned details:



The "Expression source object" picker displays call structure only for compiled PLCs in the project.

If a PLC is not compiled, object picker does not show call structure for the Organization blocks defined in the PLC.

In case of uncompiled PLCs, Expression resolver displays a banner with the information of PLCs. Consider a project configured with 3 PLCs "PLC_1", "PLC_2" and "PLC_3". If only PLC_1 is compiled, the following banner with message is displayed:



When Expression resolver editor is opened for the first time, editor loads the call structure of all compiled PLCs automatically. Editor shows the banner only if there are uncompiled PLCs, and object picker displays the same based on the loaded call structure. When you modify a PLC program, expression resolver does not show the new/modified changes.

If you wish the resolver to consider the modified/new changes, you must compile the PLC, and click "Update" button which is available in source object configuration editor. "Update" operation loads the available call structure of compiled PLCs and the same will be used to resolve the expressions.

SiVArc expression configuring editor

Configuration editor supports intellisense, syntax highlighting, copy, paste, delete operations similar to existing local expression selection editor.

Configuration editor contains a resolve button in the toolbar. After entering the expression, click the "Resolve expression" button to see the resolved value in the results window.

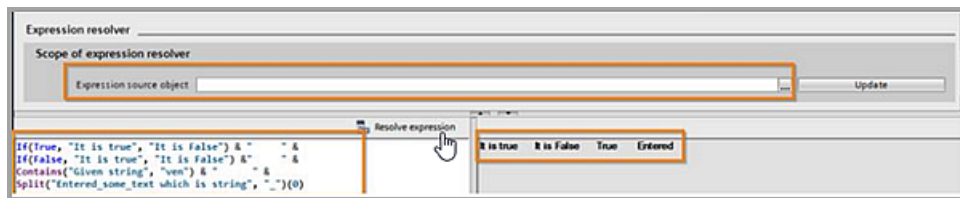
Expression results window

Displays the resolved value for the entered expression in "Expression configuration editor". This is dependent on your selection in "Expression source object".

Consider the following scenarios of how a resolver works:

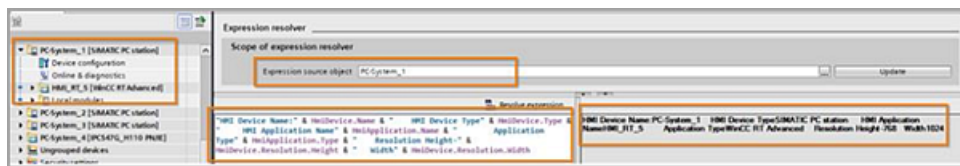
Case1: String based operations without any expression source object.

If no selection is made in the "Expression source object", but in the SiVArc expression editor expressions are entered, upon clicking the "Resolve expression" button, the expression resolves as shown in the below screenshot:



Case2: HMI device based expressions

When you choose an HMI device in the "Expression source object", and enter expressions in the SiVArc expression editor, following results are displayed as shown in the below screenshot:



Case3: Block based expressions

When you choose a block in the "Expression source object", and enter expressions in the SiVArc expression editor, following results are displayed as shown in the below screenshot:



Note

Except text definitions, all the SiVArc supported block based expressions, tag definitions are supported by the resolver.

Block.DB.HMITagPrefix- During SiVArc generation, the expression "Block.DB.HMITagPrefix" is resolved by removing invalid characters in the resolved value. Resolver does not include any HMI device validation for the resolved value.

Block.NetworkTitle and Block.NetworkComment - Expression is resolved based on the editing language. Resolver fetches the configured network title and comment for the selected editing language and displays the same in the results.

Case4: Infinite loop

In case of an infinite loop in the PLC call structure, SiVArc generation displays a message.

The Resolver supports infinite loop in the PLC call structure.

5.3.9 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 screenshot shows the 'Generation Overview' window in SiVArc. The window title is 'Projectv17inc5 > Common data > SiVArc > Generation Overview'. There are three tabs: 'Screens / Screen Items', 'Template Screens / Screen Items', and 'Popup Screens / Screen Items'. The 'Screens / Screen Items' tab is active, displaying a table with the following columns: Name of screen, Name of screen item, Master copy / library type, HMI station, S7 station, Rule Trigger, Screen rule, Generated by matrix, Layout field, and Call path. The table contains four rows of data, each representing a generated screen item.

Name of screen	Name of screen item	Master copy / library type	HMI station	S7 station	Rule Trigger	Screen rule	Generated by matrix	Layout field	Call path
Template_1	Template_1	Template_1	HMI_RT_1	PLC_1	Block_2, Block_2_DB	Screen rule_4	<input type="checkbox"/>		main/block_2
Template_2	Template_2	Template_2	HMI_RT_1	PLC_1	Block_2, Block_2_DB	Screen rule_6	<input type="checkbox"/>		main/block_2
Template_1	Template_1	Template_1	HMI_1	PLC_1	Block_2, Block_2_DB	Screen rule_4	<input type="checkbox"/>		main/block_2
Template_2	Template_2	Template_2	HMI_1	PLC_1	Block_2, Block_2_DB	Screen rule_6	<input type="checkbox"/>		main/block_2

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

"Screens/ Screen Items" 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

"Screens/ Screen Items" tab	"Tags" tab	"Text lists" tab
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
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)	---	---

Similar options are available for Pop up screen/Screen items, Text lists, Alarms.

5.3.10 Energy Suite Generation

Introduction

SiVArc extends its generation support to clients such as Energy Suite. Energy Suite generates screens and screen items based on the configured screen rules, which are referred as system screen rules. You can choose to generate based on SiVArc screen rules along with system screen rules through the station dialog box " Select station and controllers for SiVArc generation". In "Select station and controllers for SiVArc generation" station dialog box , the "Rule Set" column allows you to choose the type of rules that you would like to generate, and can be one of the following:

- User Created Rules - Relates to user created rules within SiVArc
- Energy Suite Rules - Relates to rules created by client Energy suite
- All Rules - Combination of SiVArc and Energy Suite rules.

Note

When All Rules are selected, SiVArc created rules are prioritized over Energy Suite rules while generation.

- If Energy Suite is not installed, during SiVArc generation, the rule set drop-down will list only the "User created rules" in read only mode.
- If Energy Suite rules are available without a valid Energy Suite license, and you choose to perform SiVArc generation with the rule set as "All rules", an error of missing license is displayed.

Generation scenarios of SiVArc screen rules and system screen rules

User created rules	System screen rules	Result
Rule configured in SiVArc for Screen1	Rule configured by Energy Suite for Screen1	System generates Screen1
Rule configured in SiVArc for Screen1 with Button 1	Rule configured by Energy Suite for Screen1 with Button 2	System generates Button 1 and Button 2 on Screen1
Rule configured in SiVArc for Screen 1 with Button 1	Rule configured by Energy Suite for Screen1 with Button 1	System generates Button 1 on Screen1
User created Screen 1 with Button 1	Rule configured by Energy Suite for Screen1 with Button 1	System renames Screen 1 to Screen 1_Renamed System generates Screen 1 and Button 1 on it
User created Screen 1 with Button 1	Rule configured by Energy Suite for Screen1 with Button 2	System renames Screen 1 with Button 1 to Screen 1_Renamed System generates Screen 1 created by Energy Suite rule, and places Button 2 on Screen x1
User created Screen 1 with Button 1 Rule configured in SiVArc for Screen1 with Button 2	Rule configured by Energy Suite for Screen1 with Button 3	System renames Screen 1 with Button 1 to Screen 1_Renamed System generates Button 2 and/or Button 3 on Screen 1 based on rule selection
Copy rule configured in SiVArc for Screen 1	Rule configured by Energy Suite for Screen1	System generates Screen1 and Screen1 is renamed to Screen1_renamed System generates Screen 1 through Energy Suite rule
Copy rule configured in SiVArc for Screen 1 (second generation uses this rule)	Rule configured by Energy Suite for Screen1 (first generation uses this rule)	In first generation, system generates Screen 1 Copy rule is not considered for generation, and is ignored with error

See also

Generation overview (Page 41)

"Tag rules" editor (Page 28)

5.4 SiVArc in the HMI editors

5.4.1 Property configurator

Description

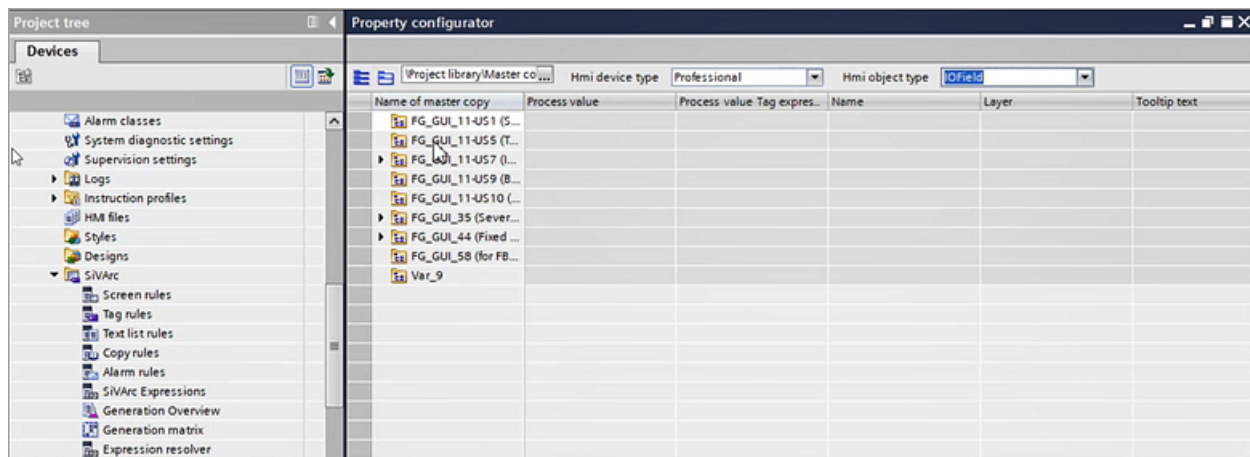
The property configurator allows you to configure the SiVArc properties of a library object placed under the **Master copies** folder. The property configurator provides an alternate way to choose library objects that you would like to view and configure the properties for a certain HMI device type. You can choose to browse different folder levels in the Master copies folder from where you would like the configurator to display the configurable properties for a selected object and HMI device type. The "Property configurator" editor enables you to perform necessary tasks that are supported by other editors.

The SiVArc plug-in editor acts as a reflection of the objects in "Property configurator". The "Property configurator" editor provides options as mentioned below:

- Select library folder - allows you to browse the folder levels of Master copies which will display the folders that contain the mastercopy objects of the selected HMI device type and the HMI object type
- HMI device type - drop down lists the supported HMI device types
- HMI object type - drop down lists the supported HMI object types

Lets us consider an example of configuring SiVArc properties of a library object such as Button.

1. In the Property configurator editor, browse for a folder within the " Project library\Master copies" from where you would like to view/configure the SiVArc properties in folders containing the library objects. You can choose the root node also if you want to view/configure the SiVArc properties for a library object. The below screenshot displays the folder selection within "Project library\Master copies":



2. Select an HMI device as Professional from the "HMI device type" drop down. WinCC Unified device types will be displayed in in the Unified environment.

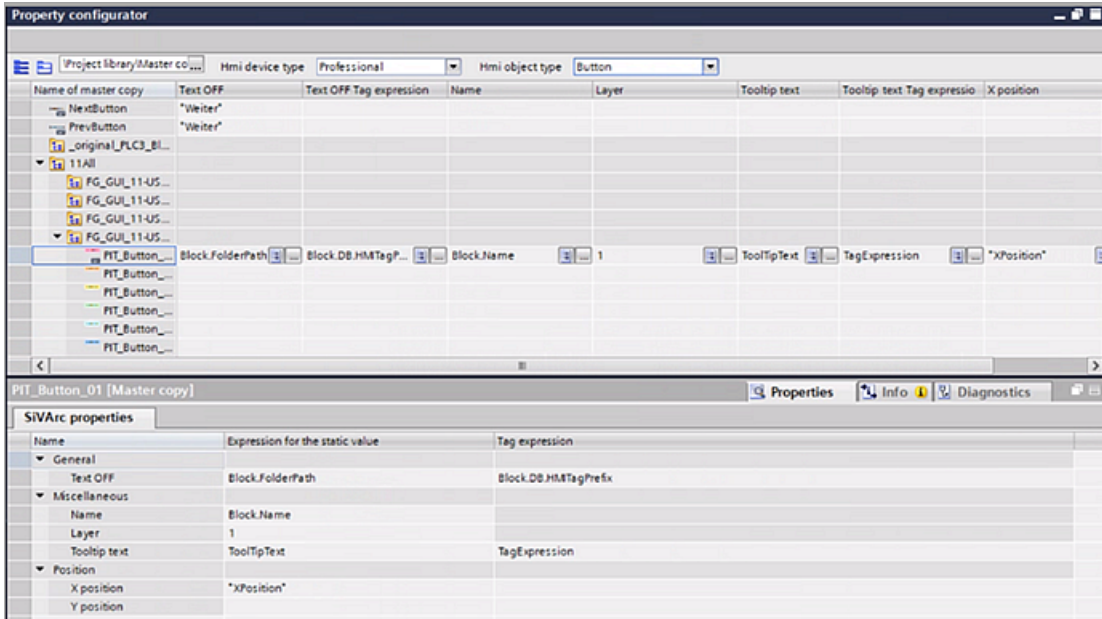
3. Select Button object from the "HMI object type" drop down as shown in the belowscreenshot. The Property configurator editor displays the Master copies folders that contain the object Button.

Note

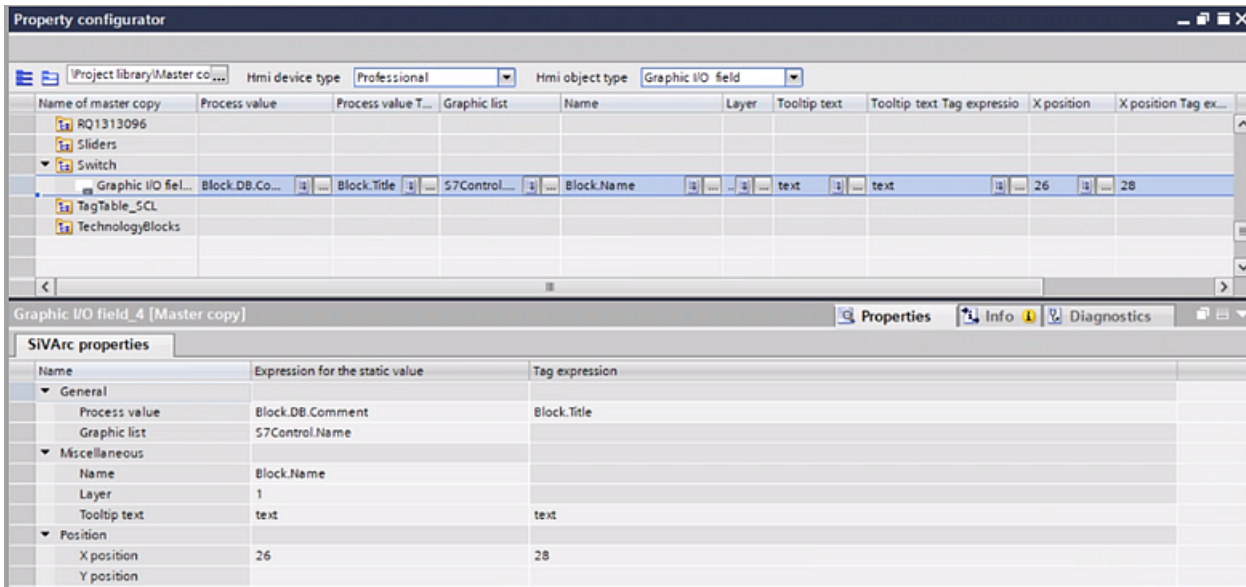
The "HMI object type" column is dependent on the type of HMI device you choose in the "HMI device type" column.

5.4 SiVArc in the HMI editors

- The columns within the "Property configurator" editor displays the SiVArc properties that can be configured with SiVArc expressions for Button as shown in the below screenshot:



- If you have chosen "HMI object type " as Graphic IO Field, the "Property configurator" editor will display columns that are supported by SiVArc. However, the SiVArc plug-in editor will display properties that are enabled on the Engineering system. Consider the properties displayed in the "Property configurator" editor, and the properties within the SiVArc plug-in editor as shown in the below screenshot. The "Property configurator" displays the properties that are supported by SiVArc, while the SiVArc plug-in editor displays the properties that are enabled on the Engineering system.



Important points:

- Any editing/deleting task performed on the library object's properties within the "Master copies" folder will reflect in the "Property configurator" editor, vice-versa is applicable.
- Renaming or relocation of a library object within the "Master copies" folder will reflect in the "Property configurator" editor.

See also

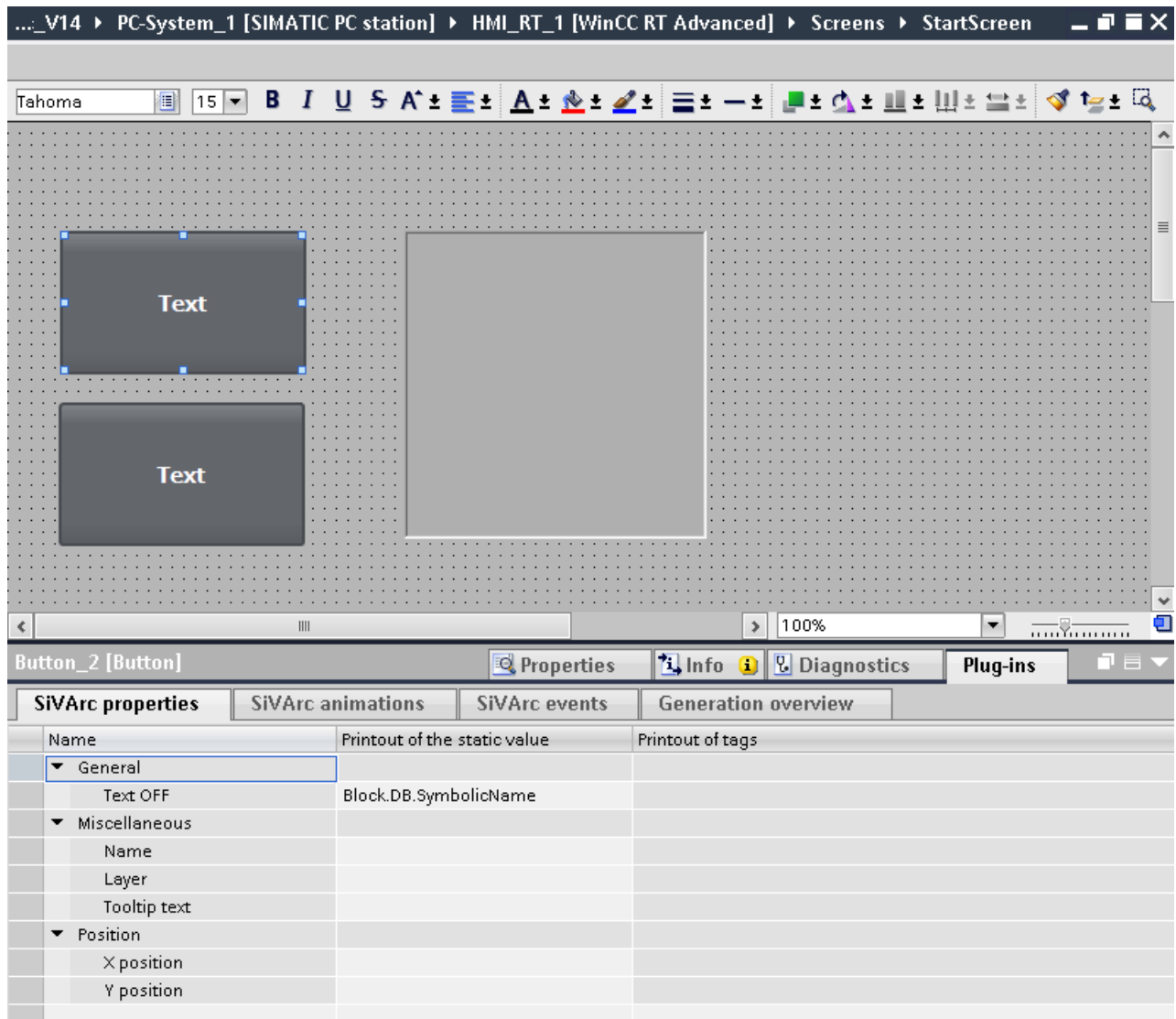
"SiVArc expressions" editor (Page 32)

5.4.2 "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. You can configure color properties for screen and screen objects in Unified devices. You can configure values for color properties in RGB format as "0-255, 0-255, 0-255" e.g. "120, 120, 120"

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



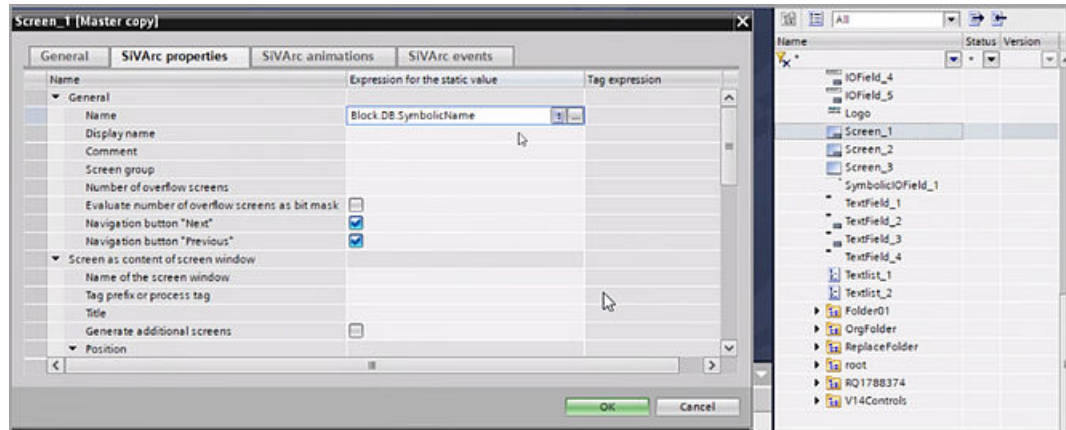
While configuring overflow screens, you can configure "Previous" and "Next" navigation buttons separately in the SiVArc Plug-ins editor. During upgradation of projects:

- if the "Generate navigation button" option was selected in the older version, the screen master copies will be updated to the latest version as follows:
 - if the "Generate navigation button" option was selected in the older version, the next and previous buttons will be selected automatically in the latest version
 - if the "Generate navigation button" option was de-selected in the older version, the next and previous buttons will be de-selected in the latest version.
- Screens existing in the HMI device in the older version will contain "Navigation button Previous" and "Navigation button Next" as selected.
- Screens existing in library type in the older version will contain "Navigation button Previous" and "Navigation button Next" as selected.

Accessing SiVArc property through Master Copy

An alternate way to view/edit an object's property can be done through the Project library >Master copies folder as compared to navigating through project folder. Perform the following:

1. Right-click on any object that you choose to edit, and select Properties option.
2. The SiVArc properties pop-up appears. By accessing the SiVArc properties through Master copy, you can configure the properties through local expression and global expression.
3. You must save the changes after updating the SiVArc properties for an object.



Note

Multi lingual support for screen objects is not available for all devices except WinCC Unified devices.

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 a 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.

5.4.3 "SiVArc events" tab

Description

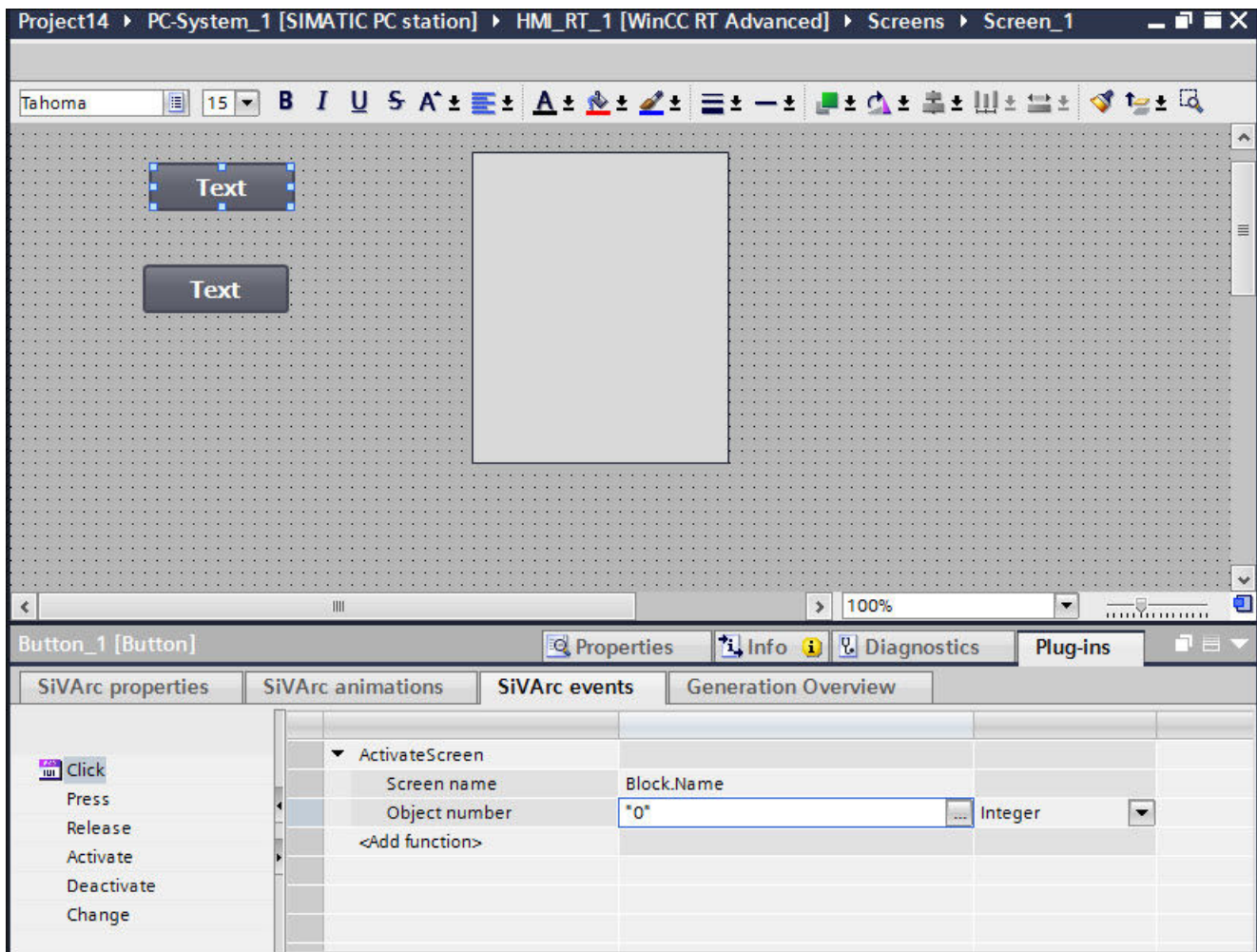
Case 1: In the "SiVArc events" tab, you can configure a function list to an event of a generation template of a screen or screen object. You can configure events for screen and screen objects. You add system functions or a script function to the function list.

You can configure the parameters of the system function or script with SiVArc expressions. Configure screen rules for SiVArc generation. Post generation, the expression that you set under SiVArc events gets resolved, and is displayed in the screen events window under SiVArc plug-ins.

Case 2: Screens with events configured can be placed under library o types, and are used as screen object in screen rules. Post SiVArc generation, the screen is generated under **Screens** in PNV node.

For WinC Comfort Unified/ Unified SCADA RT device:

- Java script based global modules are supported.
- Activate screen event is configured through change screen system function.



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.

Library objects

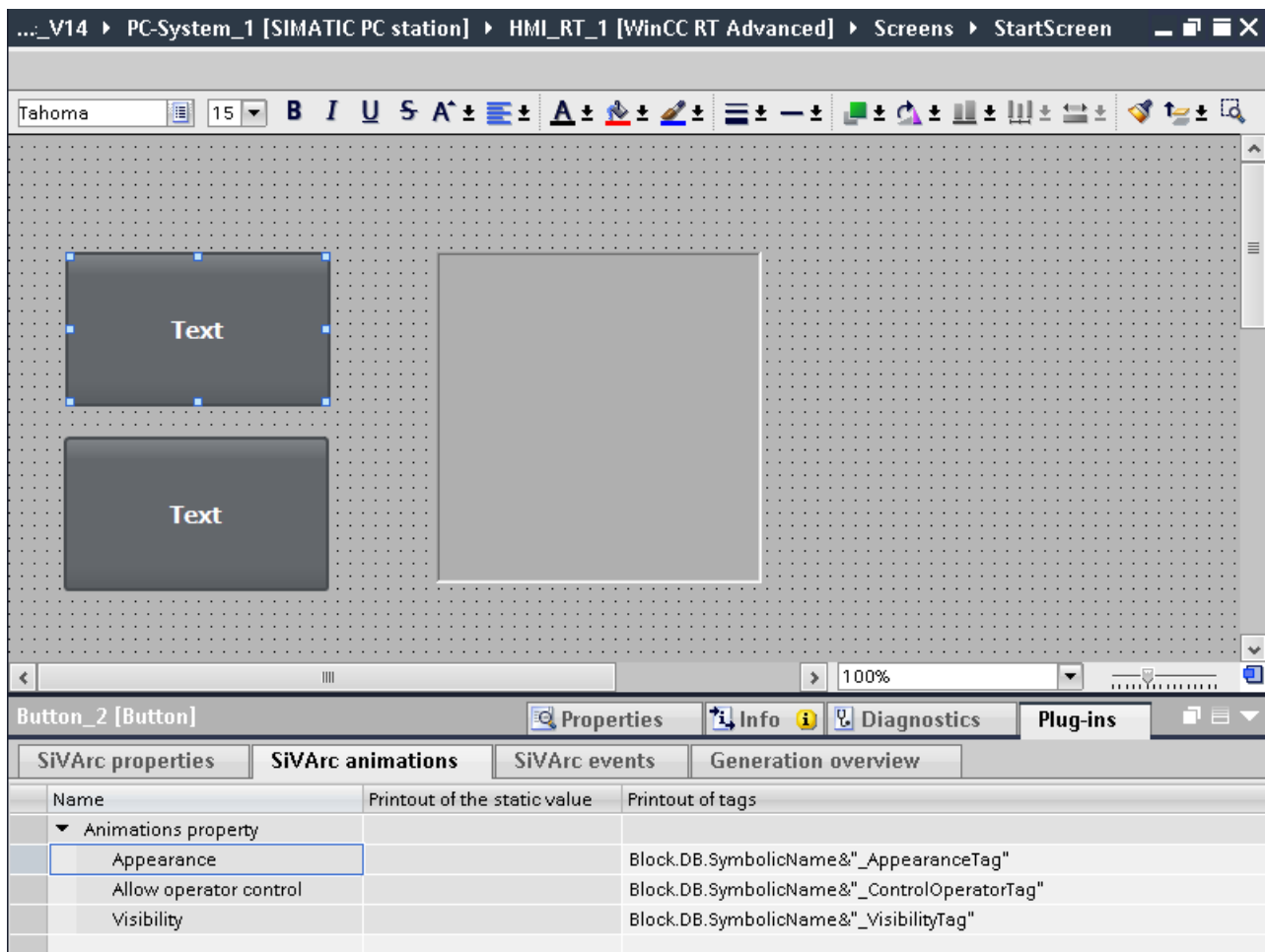
You can configure events for library elements by opening properties dialog, and the SiVArc events tab is available for view/edit options.

5.4.4 "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.



You can configure animations for supported library objects by right clicking an object, and the SiVArc plug-in editor is available for view/edit options.

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.

Library objects

You can configure animations for library elements by opening properties dialog, and the SiVArc animation tab is available for view/edit options.

5.4.5 "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

	Name of screen	Name of screen item	Master copy / library type	HMI station	S7 station	Rule Trigger	Screen rule	Generated by matrix	Layout field	Call path	Generation date
1	▶ Template_1		Template_1	HMI_RT_1	PLC_1	Block_2, Block_2_DB	Screen rule_4	<input type="checkbox"/>		main/block_2	11/25/2021
2	▶ Template_2		Template_2	HMI_RT_1	PLC_1	Block_2, Block_2_DB	Screen rule_6	<input type="checkbox"/>		main/block_2	11/25/2021
3	▶ Template_1		Template_1	HMI_1	PLC_1	Block_2, Block_2_DB	Screen rule_4	<input type="checkbox"/>		main/block_2	11/25/2021
4	▶ Template_2		Template_2	HMI_1	PLC_1	Block_2, Block_2_DB	Screen rule_6	<input type="checkbox"/>		main/block_2	11/25/2021

The "Template Screens / Screen Items and Popup Screens / Screen Items" tabs display the generated object details of template/screen items and popup screens/ screen items and rules respectively.

In case of generated screen objects, the "Plug-ins" tab will display the generation overview.

5.5 SiVArc in PLC editors

5.5.1 Support for software units

Software units in SiVArc

Software units act as a container which segregates various user defined programs. SiVArc allows you to configure expressions for various properties of HMI objects, which resolves to the name of the software unit containing the PLC block used in SiVArc rule creation. For more information on software units, see *TIA portal help*.

Requirement

- An existing TIA portal project
- PLC with firmware v2.6 and above
- PLC connected to an HMI device

Procedure

To configure SiVArc expressions for software units, perform the following steps:

1. Under Software units folder, click "Add new software unit" to add a new software unit.
2. Configure program blocks as per your requirement.
3. Add a screen, and configure the screen name with expression that resolves to the name of software unit. For example: Name = "SoftwareUnit.Name". For more information on configuring screens, refer SiVArc editors. (Page 26)
4. Configure a SiVArc screen rule consisting of a program block selected from software unit.
5. Start SiVArc generation. The generated screen name resolves to the name of software unit containing the program block that was used in SiVArc screen rule creation.

Note

Software unit support is configurable for all HMI objects using SiVArc plugin editors.

5.5.2 SiVArC support for program blocks using Structured Control Language

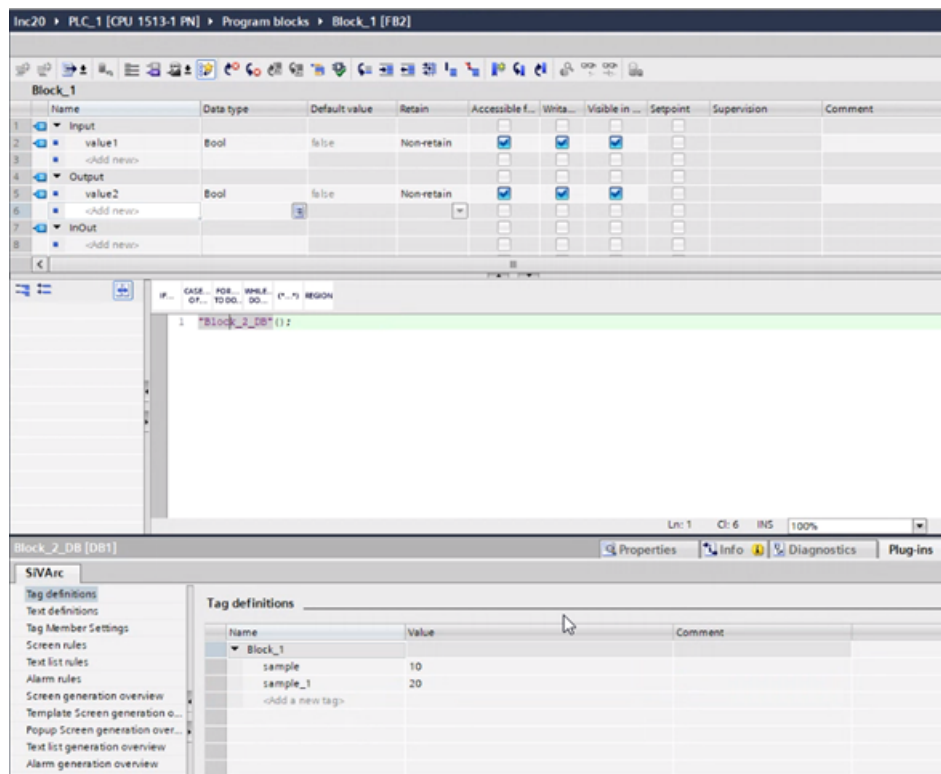
SiVArC support for program blocks using SCL

With programming languages such as STL, FBD, LAD, SiVArC supports SCL for program blocks through single instance or multi instance calls. You can configure SiVArC properties like text/tag definitions/tag member settings/ rule editors for SCL blocks in the Plug-ins editor. SiVArC supports only block level configuration for SCL. You can configure SiVArC data by block level, that applies to all instance calls within a SCL block

Example scenario of SCL usage in Tag definitions

Let us consider the configuration of Tag definitions for the SCL block. You can configure Tag definitions for the caller block (SCL). The tag definitions apply for all the instance calls in the SCL block. The Tags defined are used in SiVArC generation.

1. For example, consider "Block_2" is called in Block_1 with instance DB "Block_2_DB".
2. Select any area withing the SCL programming editor. Click SiVArC Plug-ins.
3. Define and configure tags for "Block_1" as shown in the figure:



4. The defined tag can be used in any SiVArC expression editor like SiVArC properties for screen objects, condition in rule editor. While configuring Screen objects in SiVArC Plug-ins, Enter the Screen name with Tag defined "Sample". For more details on configuring Screen objects, refer section, Screen rules editor.
5. Upon SiVArC generation, screen with name "10" gets generated. For more details on generation, refer section Generate visualization (Page 198)

Note

- Configuration of Tag/text definitions/Tag member settings for SCL block is similar to LAD block. For more details on block level configuration, refer to LAD documentation from TIA portal user guide
 - Tag Member Settings can be configured in WinCC Advanced devices only.
 - Screen/Text lists/ Alarms rules provides quick access to view and edit the configured rules.
 - Screen, Text lists, Alarm rules of caller block will be shown when a multi instance called block is selected.
 - Screen/Text lists/ Alarms generation overview provides quick access to view the SiVArc generation.
-

For Unified HMI devices, when you create PLC block variables with user defined data type/array data type and PLC tags with user defined data type; upon SiVArc **generation > AllHmiTags** results in HMI tags for PLC block variables, and PLC tags for user defined/array data type.

See also

Generate visualization (Page 198)

Working with SiVArc

6.1 Plan layout

6.1.1 Positioning schemes

6.1.1.1 Overview for positioning of generated objects

Introduction

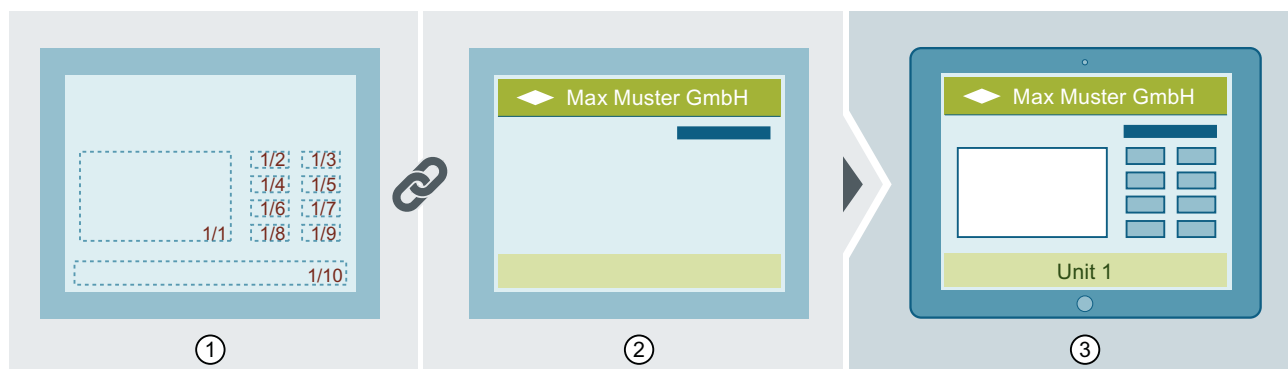
With SiVArc the layout of your process pictures is divided up into two tasks:

- 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

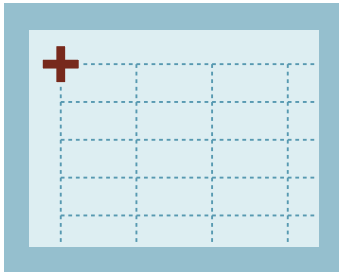
6.1 Plan layout

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.

Plantsection1 [Screen]				
SiVArc properties		SiVArc animations	SiVArc events	Generation overview
Name	Printout of the static value	Printout of tags		
▶ General				
▶ Screen as content of screen window				
▼ Positioning scheme				
X position	150			
Y position	150			
Row spacing	200			
Column spacing	600			
▶ Layout				

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:

Plantsection1_DB_SymbIO [Symbolic I/O field]				
SiVArc properties		SiVArc animations	SiVArc events	Generation overview
Name	Printout of the static value	Printout of tags		
▶ General				
▶ Miscellaneous				
▼ Position				
X position	675			
Y position	100			

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 59)

Fixed positioning of the generated object (Page 68)

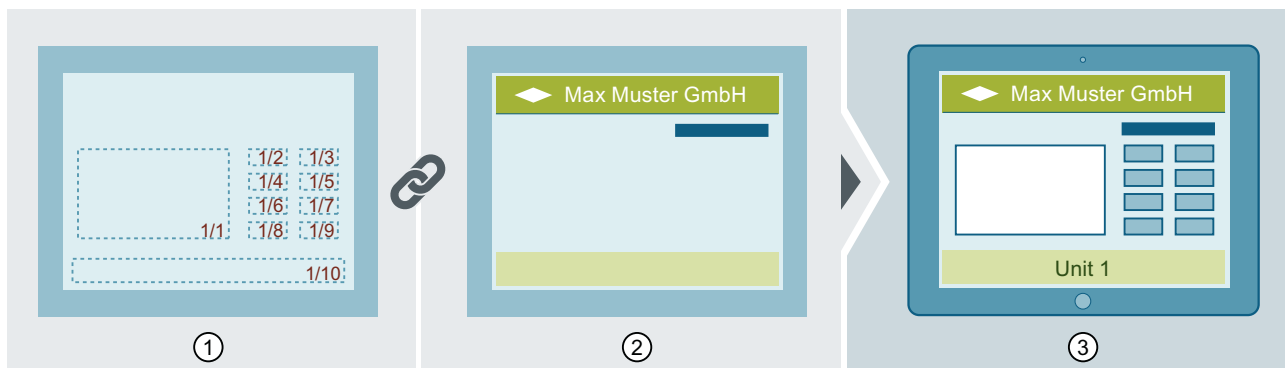
Free positioning (Page 69)

Designing a layout (Page 76)

6.1.1.2 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

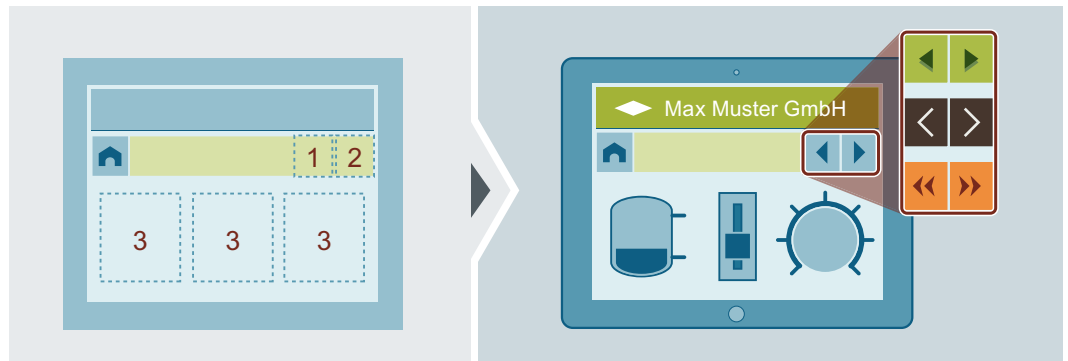
Use

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

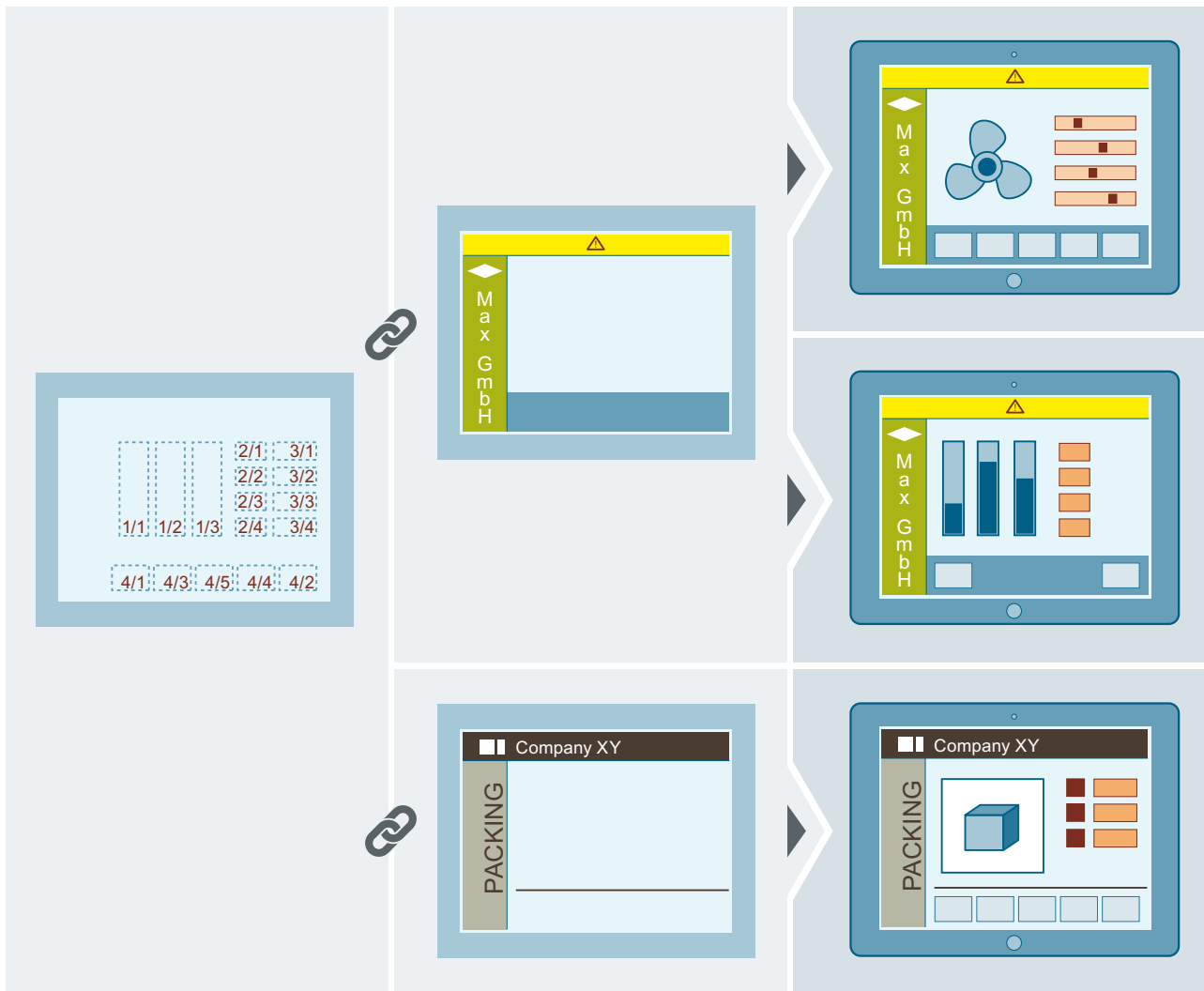
A user-defined positioning scheme offers multiple benefits:

- 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.

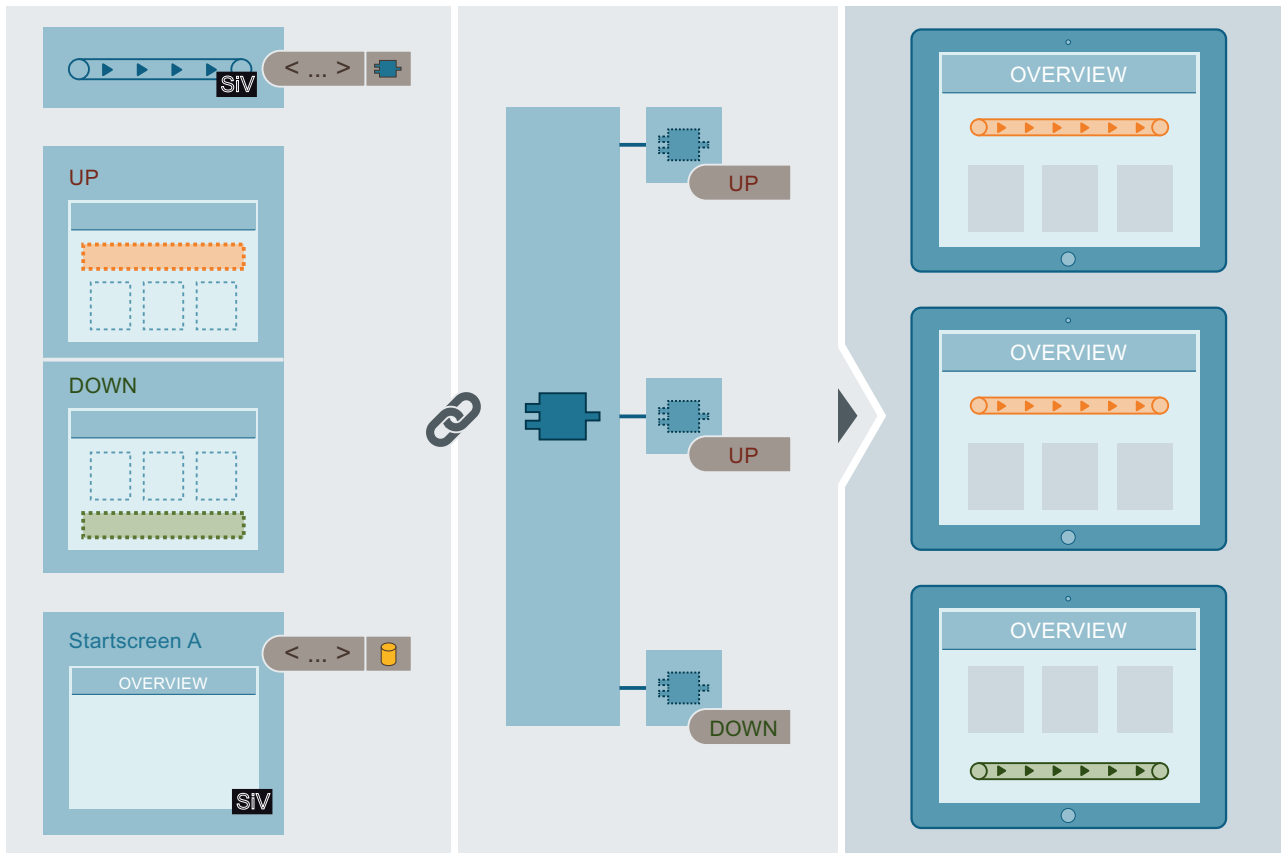


- Separation of layout and positioning
You can also use existing positioning schemes across projects due to the separation of layout and positioning.

6.1 Plan layout



- 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



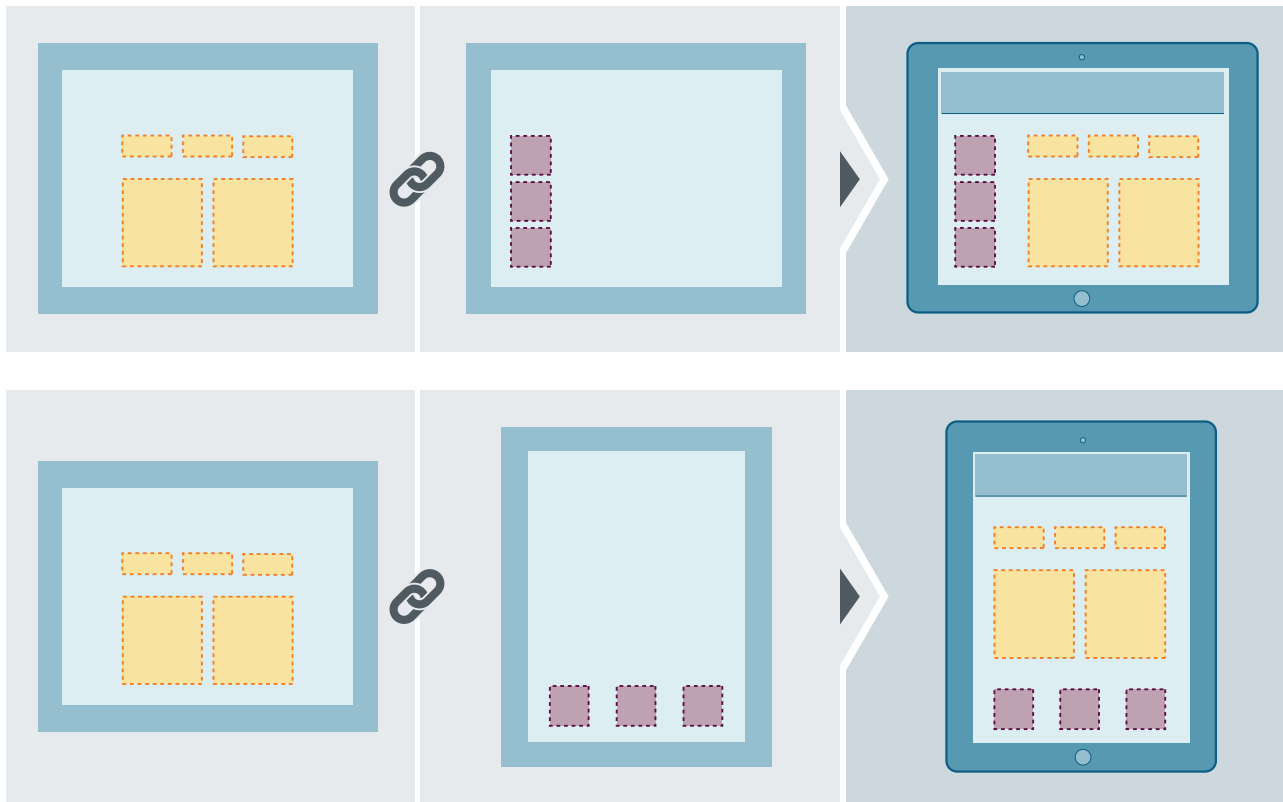
Function block



Instantiated function block

6.1 Plan layout

- 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

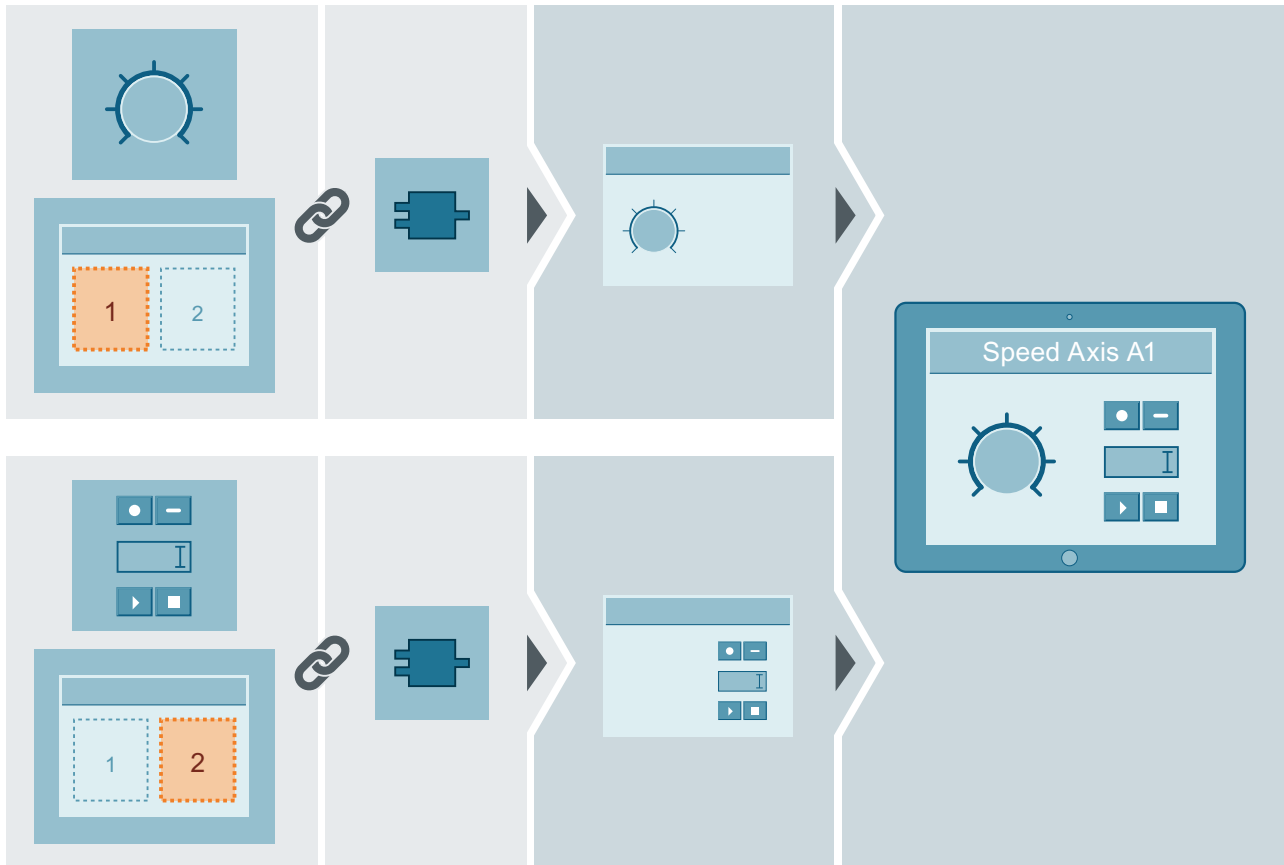
Note

Positioning schemes for pop-up screens

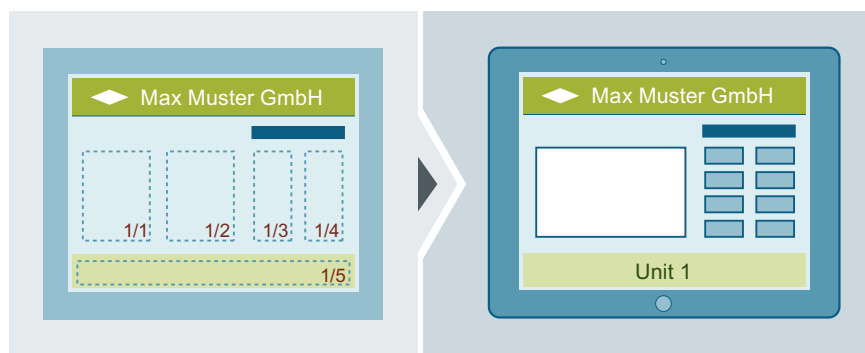
A positioning scheme for a pop-up screen cannot be used for any other display and operating object.

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.

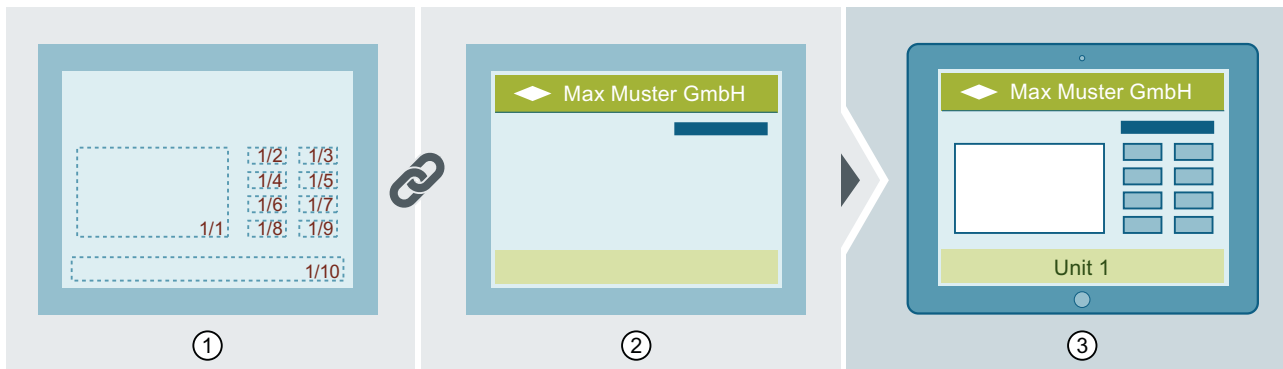


You can use a positioning scheme in a screen rule as you would a generation template for a screen. Layout and positioning are then contained in the same template:



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

6.1 Plan layout



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

Screen_1 [Screen]			
SiVArc properties			
Name	Printout of the static value	Printout of tags	
General			
Positioning scheme			
Layout			
Show layout fields	<input type="checkbox"/>		
Layout selection	Static		
Layout screen or folder	LayoutStartScreen		
Expression for layout screen name	"Recipe"		
Layout field for navigation	Monitoring		

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:

	Name	Master copy of a screen	Layout field	Condition
1	Plantsection_Title	Plantscreen		HmiApplication.Name
2	Plantsection_Stat...	Plantscreen		
3	Productionline_Title	Plantscreen		
4	Productionline_Po...	PositionSectionDriveAxis	Gauge	
5	Processing_Unit	PositionSectionPackaging	Gauge	
6	Activate_Btn	Plantscreen	Bar	
7	Stop_Btn	Plantscreen		
8	<create new rule>			

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.

SiVArc properties		SiVArc animations	SiVArc events	Generation overview
Name	Printout of the static value	Printout of tags		
▼ General				
Use as layout field	<input checked="" type="checkbox"/>			
Layoutfield name	Monitoring			
Layout field index	1/4			
Font size name	12			
▼ Alignment				
Horizontal alignment	Left			
Vertical alignment	Top			

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 57)
- Overflow mechanisms (Page 71)
- Creating user-defined positioning scheme (Page 78)
- Example: Using a dynamic layout (Page 85)
- Example: Using a combined layout (Page 87)
- Example: Using generated screen navigation (Page 88)
- Picture legends (Page 266)

6.1.1.3 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.

Plantsection1_DB_SymbIO [Symbolic I/O field]				
SiVArc properties		SiVArc animations	SiVArc events	Generation overview
Name	Printout of the static value	Printout of tags		
▶ General				
▶ Miscellaneous				
▼ Position				
X position	675			
Y position	100			

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.

Configuring expression editor for X and Y position properties

SiVArc supports configuring the position of static and dynamic screen objects through expressions that resolves to integer values or direct integer values in the SiVArc "Plug-ins" editor > "Position". You configure the X and Y position of screen objects in "Position" property through SiVArc expressions, and add the screen object to the master copy. You define the screen rule, which comprises of the object placed under the master copy, and the screen to be generated. During SiVArc generation, the expression defined for X and Y position must resolve to integer values, and generates the corresponding screen objects at that position.

You can also configure dynamic tag values for X and Y position through "Printout of tags". During SiVArc generation, the expression defined under "Printout of tags" column resolves to a tag name, and generates the corresponding screen objects. For more information on generation, refer to Overview for positioning of generated objects (Page 57).

See also

Overview for positioning of generated objects (Page 57)

Basics on generation (Page 198)

6.1.1.4 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.

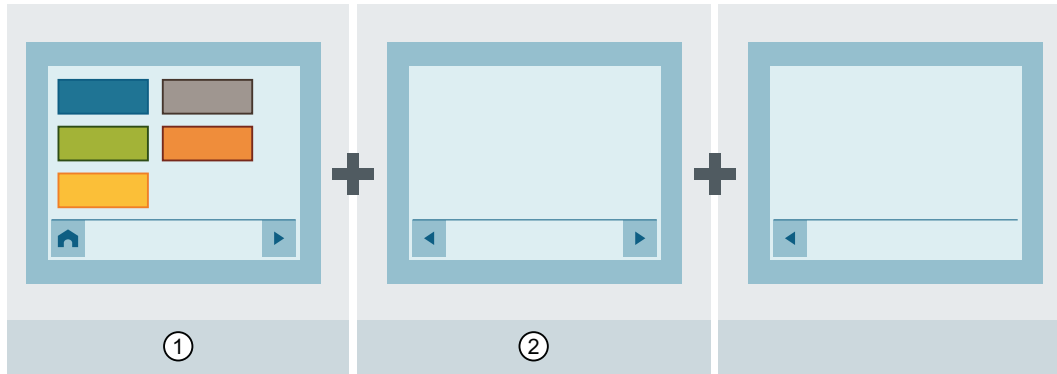
Plantsection1 [Screen]				
SiVArc properties		SiVArc animations	SiVArc events	Generation overview
Name	Printout of the static value	Printout of tags		
▶ General				
▶ Screen as content of screen window				
▼ Positioning scheme				
X position	150			
Y position	150			
Row spacing	200			
Column spacing	600			
▶ Layout				

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.

6.1 Plan layout

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 207)

Overview for positioning of generated objects (Page 57)

Overflow mechanisms (Page 71)

Example: Using a layout with free positioning (Page 82)

6.1.1.5 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

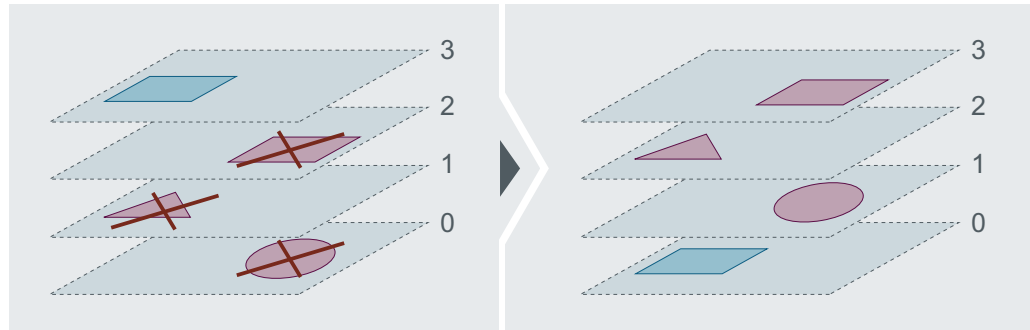
Changing the layer

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.

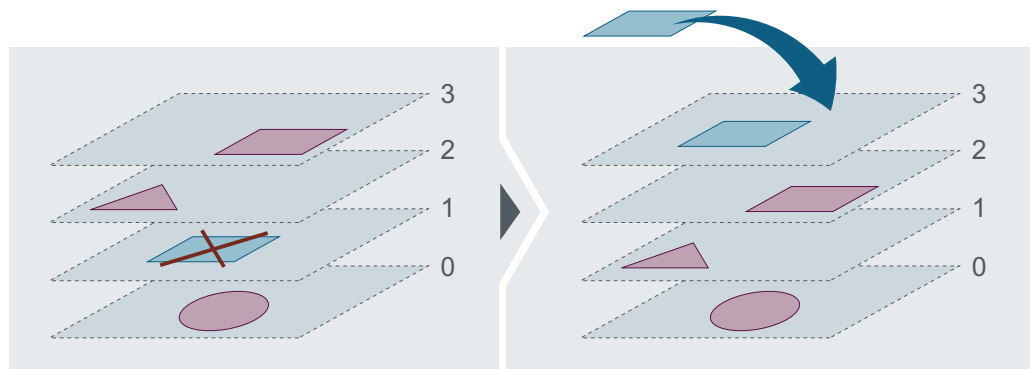


Generated display and operating object



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

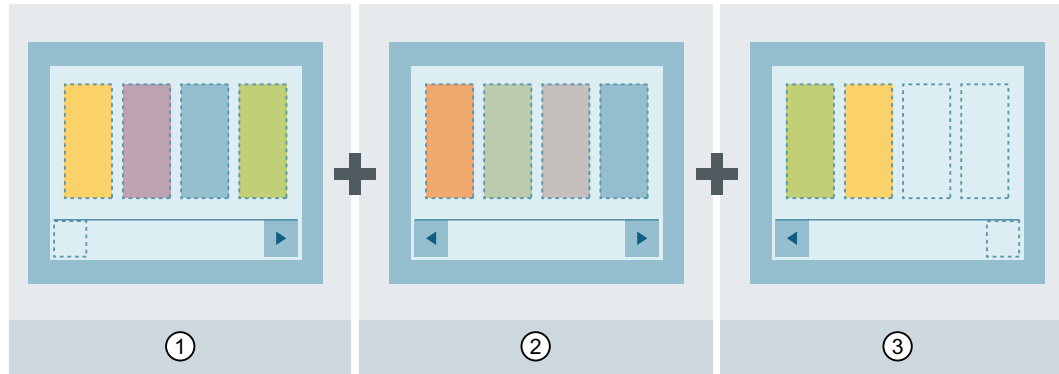
6.1.2 Overflow mechanisms

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.

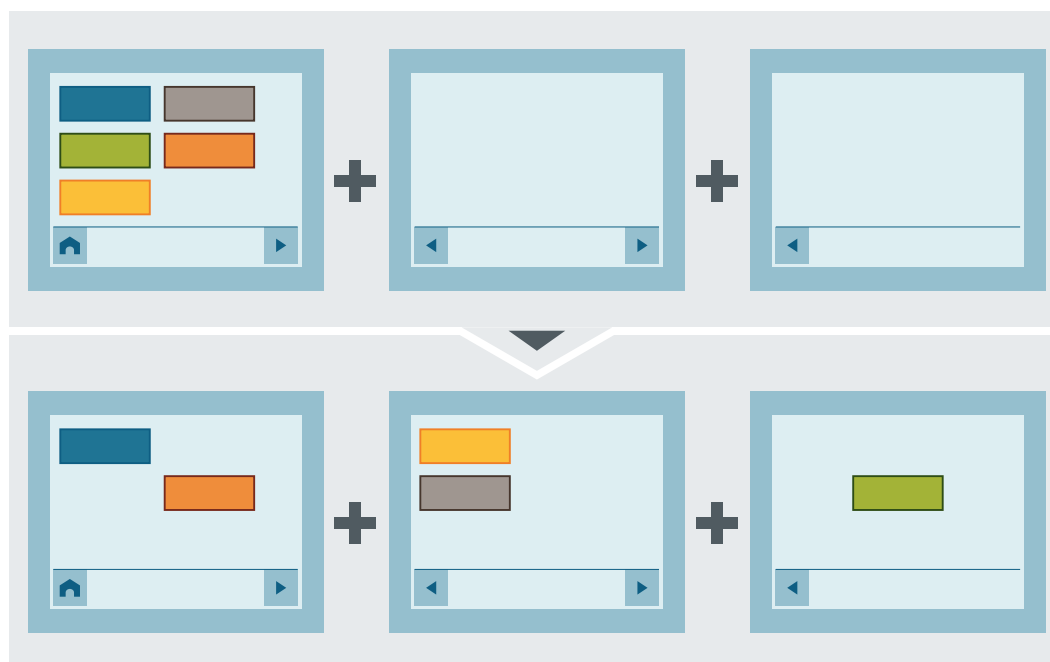


- ① Base screen
- ② First overflow screen with positioning scheme of the base screen
- ③ Second overflow screen with positioning scheme of the base screen

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

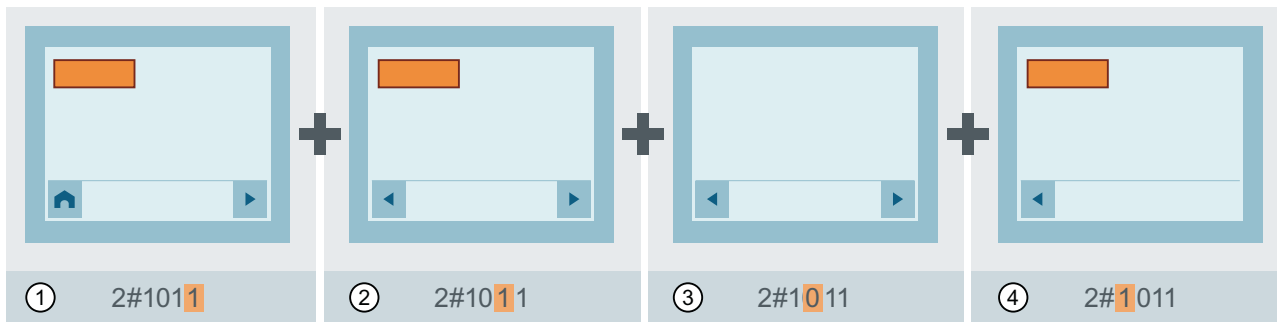
You define the bit mask at the program block or in the generation template of the screen. To do so, use a static value or a tag.

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

6.1 Plan layout

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

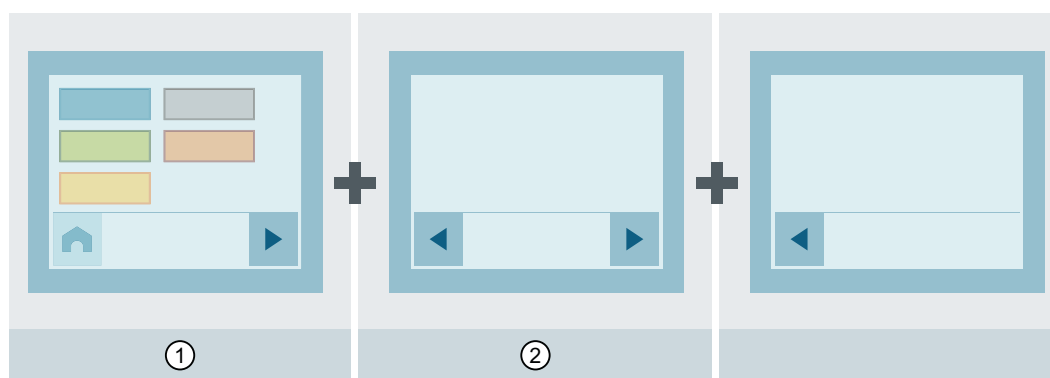
Note the following when you configure overflow screens with a bit mask:

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 90)"

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 69)

Example: Using overflow mechanisms (Page 82)

Configuring overflow screens without screen objects (Page 81)

6.1.3 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
 - WinCC Comfort Unified/ Unified Scada RT

6.1.4 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 57)

6.1.5 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".

SiVArc properties		SiVArc animations	SiVArc events	Generation overview
Name	Printout of the static value	Printout of tags		
▼ General				
Use as layout field	<input checked="" type="checkbox"/>			
Layoutfield name	Monitoring			
Layout field index	1/4			
Font size name	12			
▼ Alignment				
Horizontal alignment	Left			
Vertical alignment	Top			

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.

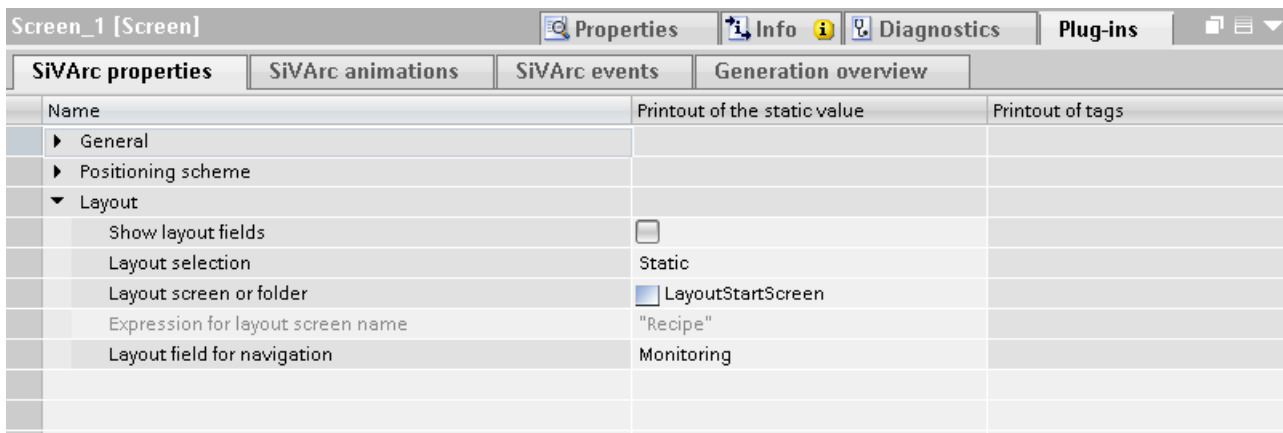
Result

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 59)

6.1.6 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.

6.1 Plan layout

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 82)

Overflow mechanisms (Page 71)

6.1.7 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 69)

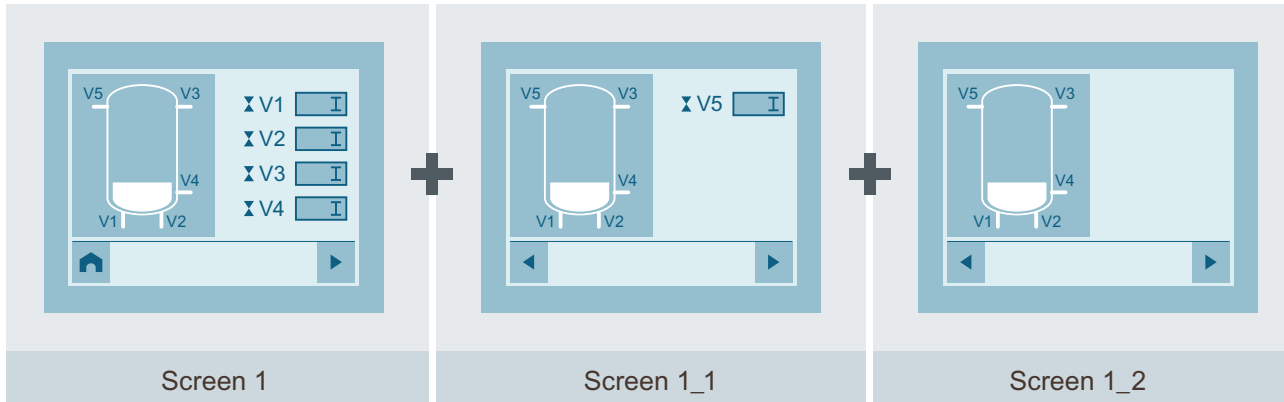
6.1.8 Example: Using overflow mechanisms

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.



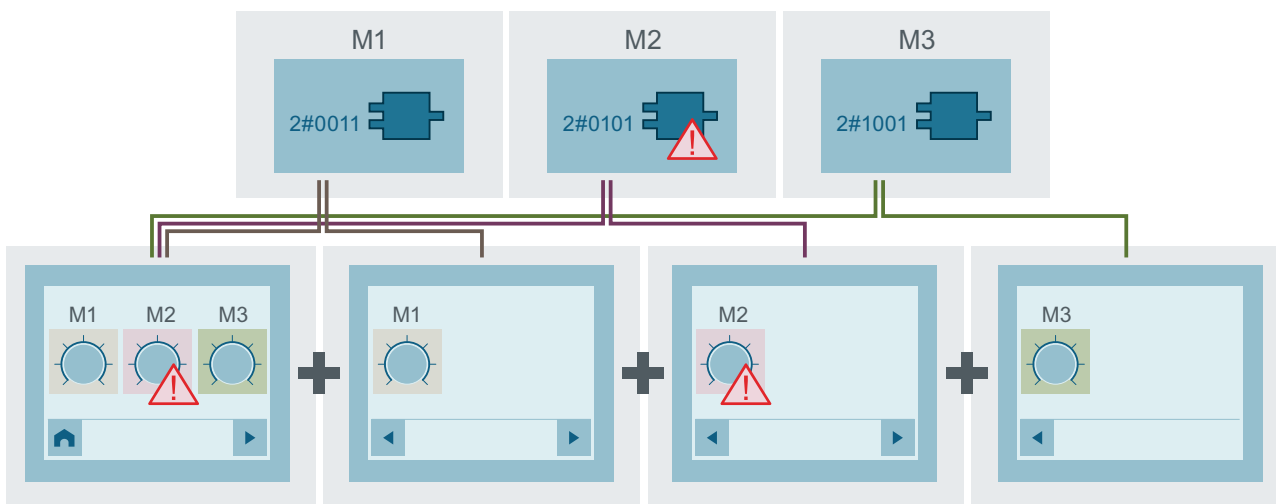
Example scenario overflow screen with bit mask

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 71)

Configuring overflow screens without screen objects (Page 81)

6.1.9 Example: Using a dynamic layout

Example scenario

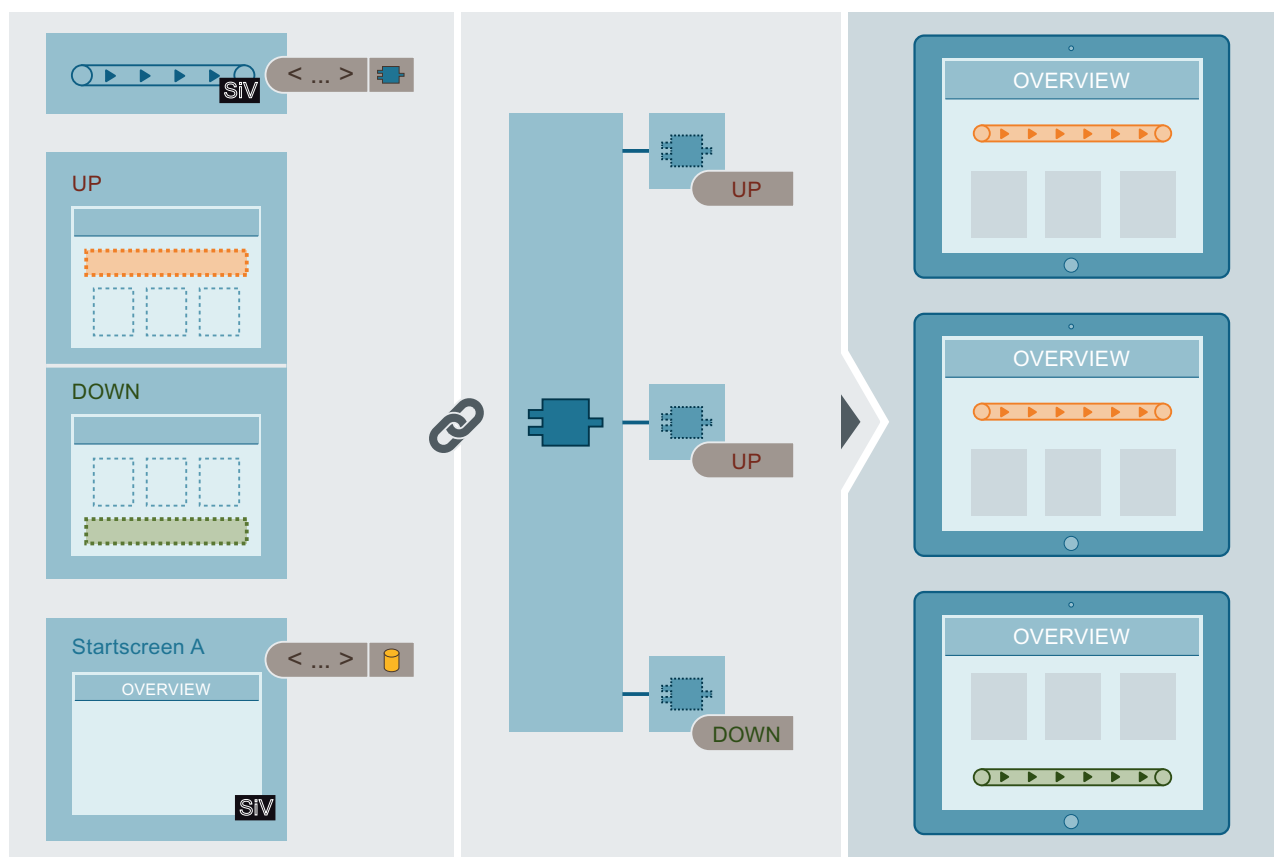
Two conveyor belts require a different screen arrangement but have the same operating objects.

Requirement

The number of screen rules shall be reduced.

Implementation concept

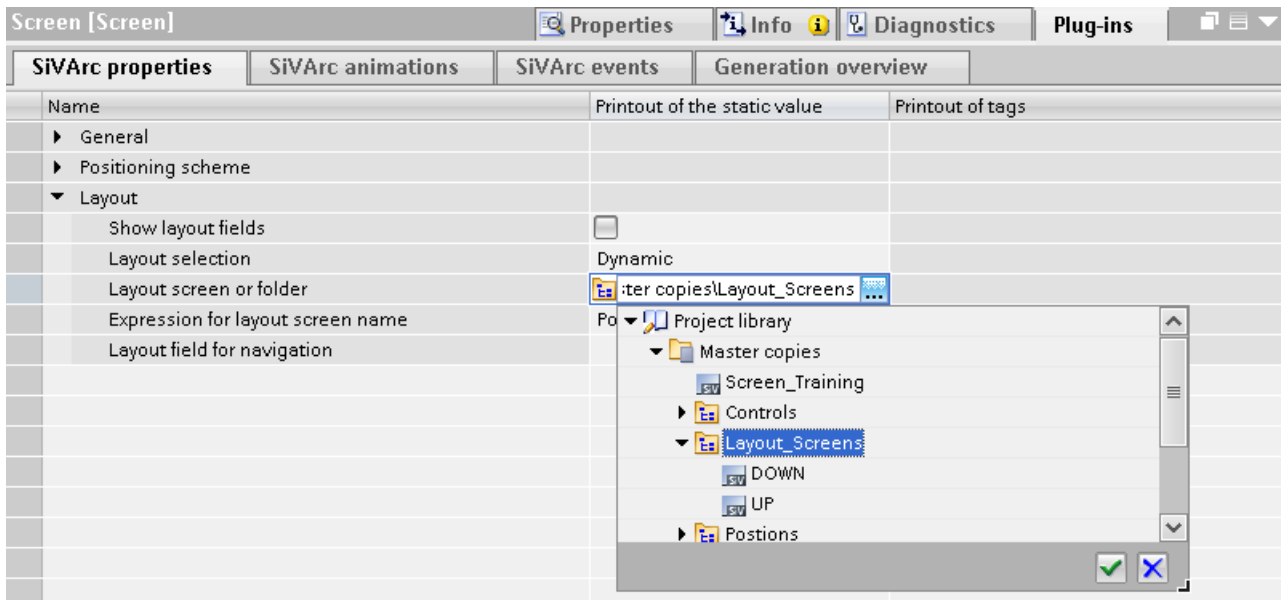
Two positioning schemes are created and stored in a library folder. A SiVArc tag is defined on the network. The SiVArc tag controls where the belt is to be arranged in the screen. The matching positioning scheme is selected during evaluation of the screen rule. One of the belts is arranged at the top 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 59)

Picture legends (Page 266)

6.1.10 Example: Using a combined layout**Example scenario**

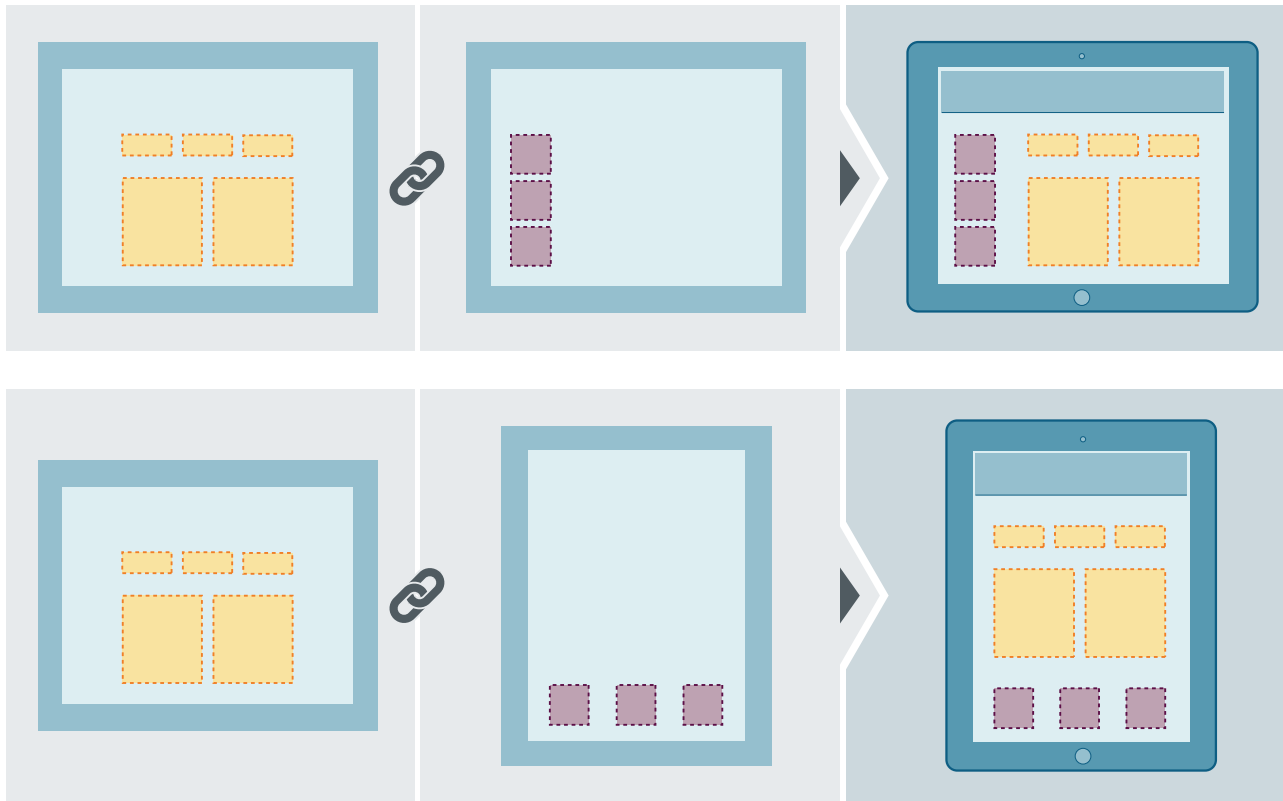
The overview screens of the production line for placing, soldering and packaging have a similar layout and differ from each other in the use of two different HMI devices (landscape format and portrait format). This results in a different partitioning of the screen.

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 59)

6.1.11 Example: Using generated screen navigation

Example scenario

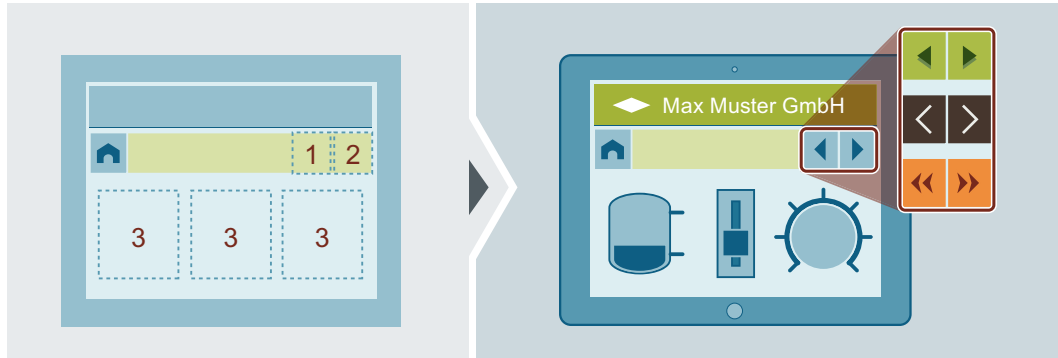
The screen navigation in an existing visualization object deviates from the new guidelines for corporate design and standardization in the company.

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".

Screen_1 [Screen]				
SiVArc properties		SiVArc animations	SiVArc events	Generation overview
Name			Printout of the static value	Printout of tags
▶ General				
▶ Positioning scheme				
▼ Layout				
Show layout fields		<input type="checkbox"/>		
Layout selection			Static	
Layout screen or folder			LayoutStartScreen	
Expression for layout screen name			"Recipe"	
Layout field for navigation			Monitoring	

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 59)

6.2 Creation of generation templates

6.2.1 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 screenshot shows the SIMATIC Manager interface with the 'SiVArc properties' tab selected. The main workspace displays three text objects on a grid. The properties table below shows the configuration for 'Text OFF'.

Name	Printout of the static value	Printout of tags
▼ General		
Text OFF	Block.DB.SymbolicName	
▼ Miscellaneous		
Name		
Layer		
Tooltip text		
▼ Position		
X position		
Y position		

6.2 Creation of generation templates

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.

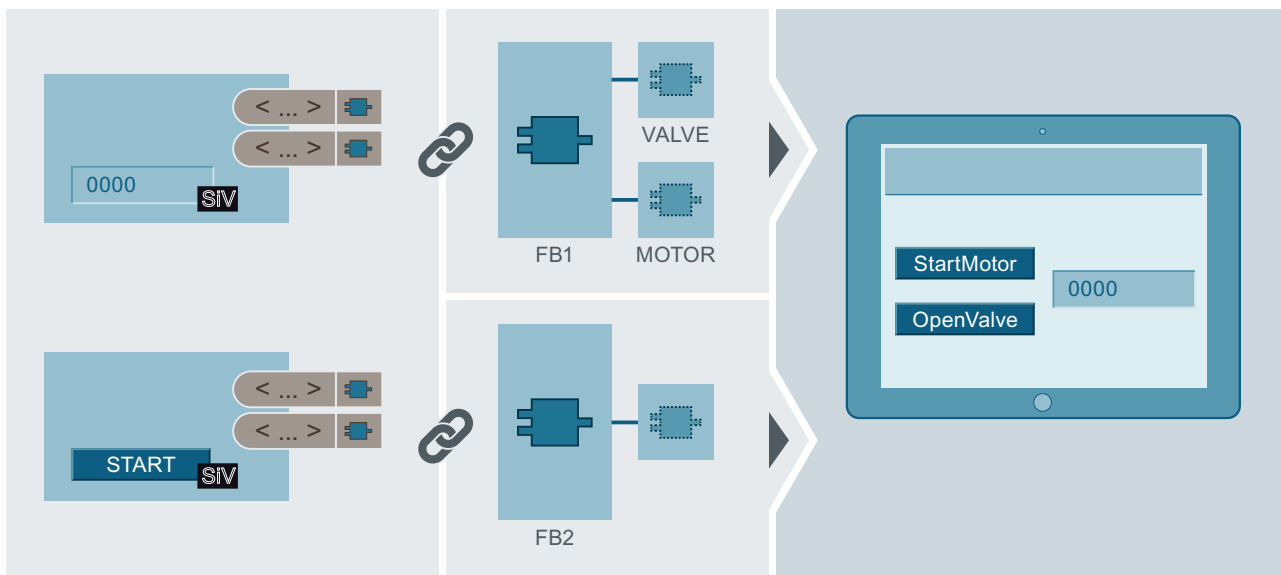
You then store the configured generation template in the project library and interconnect it later to a function block. You create a separate interconnection for each generation template. The SiVArc properties are evaluated during the generation of the visualization.





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.



	SiVArc generation template
	SiVArc property with referenced text source
	Function block
	Instantiated function block

Generating screen objects on template screens

SiVArc supports the generation of screen items on template screens, and creation of templates. You can configure the templates using the **plug-ins** editor. You can perform the following in the template screens:

- Full generation and merge scenarios behave similar to normal screens
- Configure rules by choosing templates as master copies.
- Update the screen's template based on the configured template property.

Note

Template screen is supported on all other devices, except for professional devices.

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

6.2 Creation of generation templates

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 Advanced devices.

See also

Requirements for a generation template (Page 109)

Example: Generating template screen using copy rule (Page 130)

Example: Generating HMI screen using template property (Page 131)

6.2.2 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	WinC Comfort Unified/ Unified SCADA RT
External tag ¹	x	x	x	---
Following master copies in a library:				
Bar	x	x	x	x
Screen ¹	x	x	x	x
Screen window	---	---	x	---
Template screen	---	x	---	---
I/O field	x	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	---	x
Button	x	x	x	x
Slider	---	x	x	x

HMI object	Basic Panels	Comfort Panels / Mobile Panels /2nd Generation RT Advanced	RT Professional	WinC Comfort Unified/ Unified SCADA RT
Symbolic I/O field	x	x	x	x
Text field	x	x	x	x
Text lists	x	x	x	x
Gauge	---	x	x	x
Following types in a library:				
Screen as screen window ¹	---	---	x	---
Alarms	x	x	x	---
Faceplates	---	x	x	x
Events	x	x	x	x
Animation	x	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	WinCC Unified SCADA RT / Unified Comfort Panel
Screen	x	x	x	
Tags				x
Internal tag	x	x	x	x
Tag table	x	x	x	x
Scripts				
C script	---	---	x	
VB script	x	x	x	x
Text list	x	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	WinCC Unified Comfort Panel	WinCC Unified SCADA RT
Text Off (length)	320	500	500	1000	1000
ToolTip (length)	320	1000	1000	320	320
Text field (text)	320	32767	32767	30000	30000
Text list entry (text)	320	320	320	320	255

6.2.3 Sources for texts

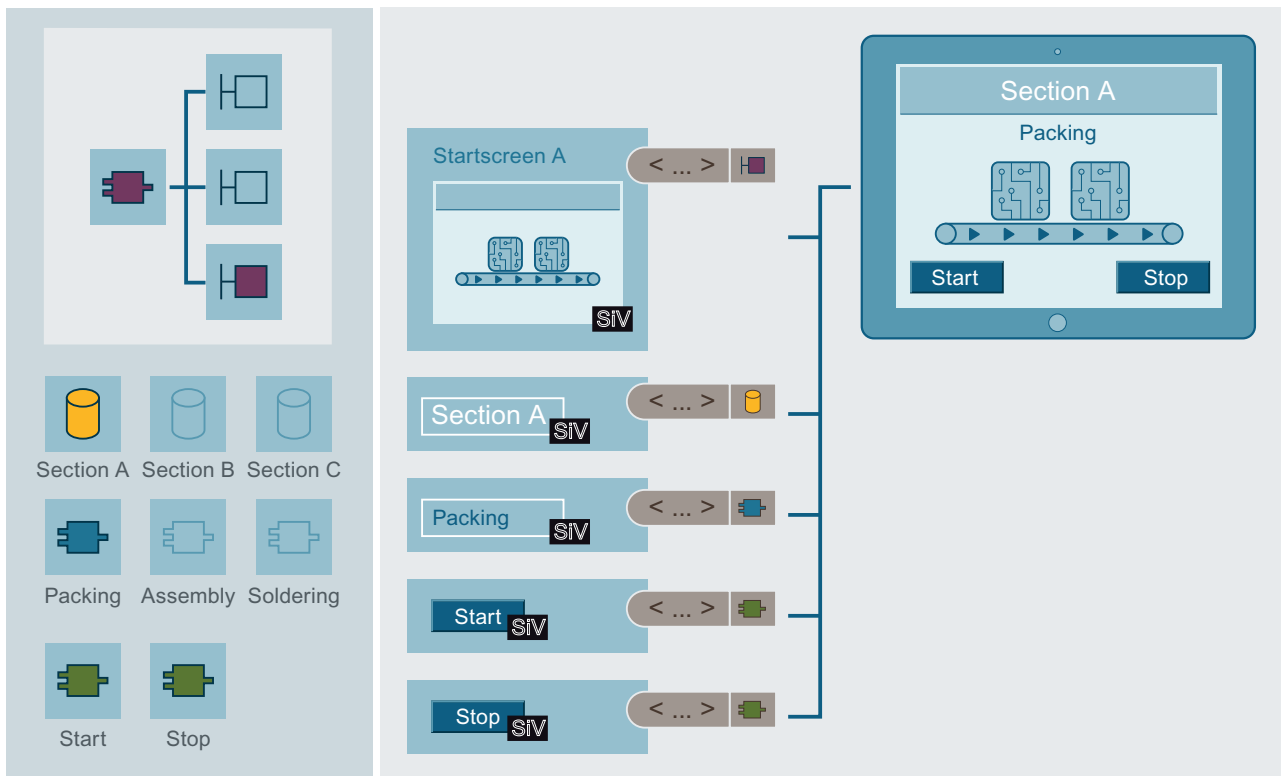
6.2.3.1 Overview of text source in SiVArc project

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.

**SiVArc**

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
- Main block (OB)
 - Symbolic name
 - Symbolic address
 - Version
 - Folder path
 - Type
 - Number
- I/O device
 - Article number
 - Invariant type
 - Name

- PLC Tags
 - HMI prefix
- HMI device
 - Device name/resolution/type
- HMI application
 - Name
 - 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 117)

SiVArc expression (Page 106)

6.2.3.2 SiVArc texts

Introduction

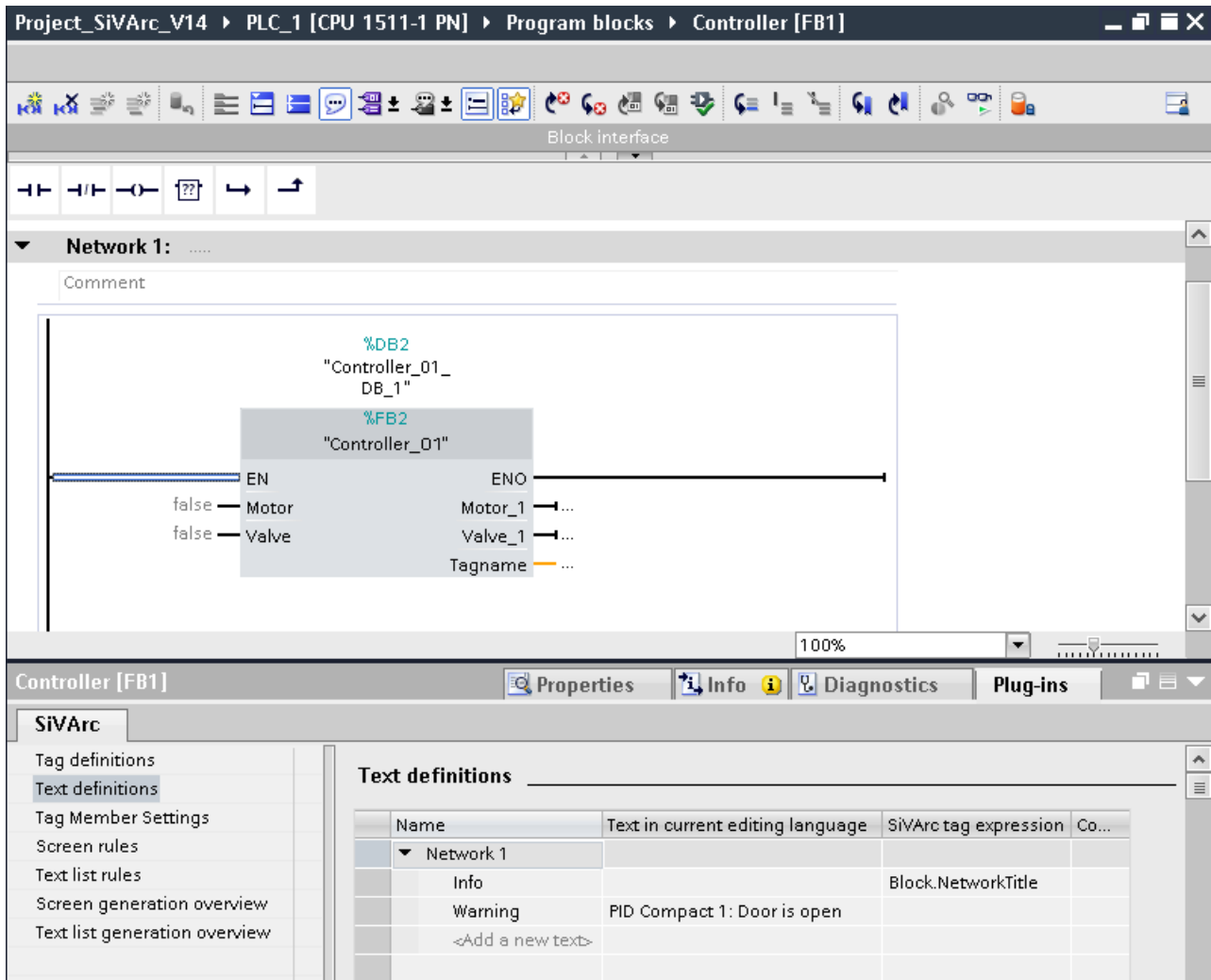
With SiVArc you can define texts as text list entries for the generation of your visualization. This functionality is integrated in the user program in STEP 7 and is available in any network title and block.

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 107)

Influence of multilingualism on a generation template (Page 120)

6.2.4 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
- SCL

Technology objects

SiVArc supports the following technology objects:

- PID Control:
 - Compact PID
 - PID basic functions

See also

Supported data types for PLC tags (Page 262)

6.2.5 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. The expressions entered are also auto saved in the editor.
- **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.

You can drag and drop scripts within SiVArc plug-in editor.

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#111100 = 29`
- When necessary, you use binary code and other constant values, e.g. `25 * 2#111100 = 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 225)

6.2.6 SiVArc expression

6.2.6.1 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 107)
SiVArc object properties (Page 246)
"Format" function (Page 249)
Structure of SiVArc expressions (Page 115)

6.2.6.2 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.

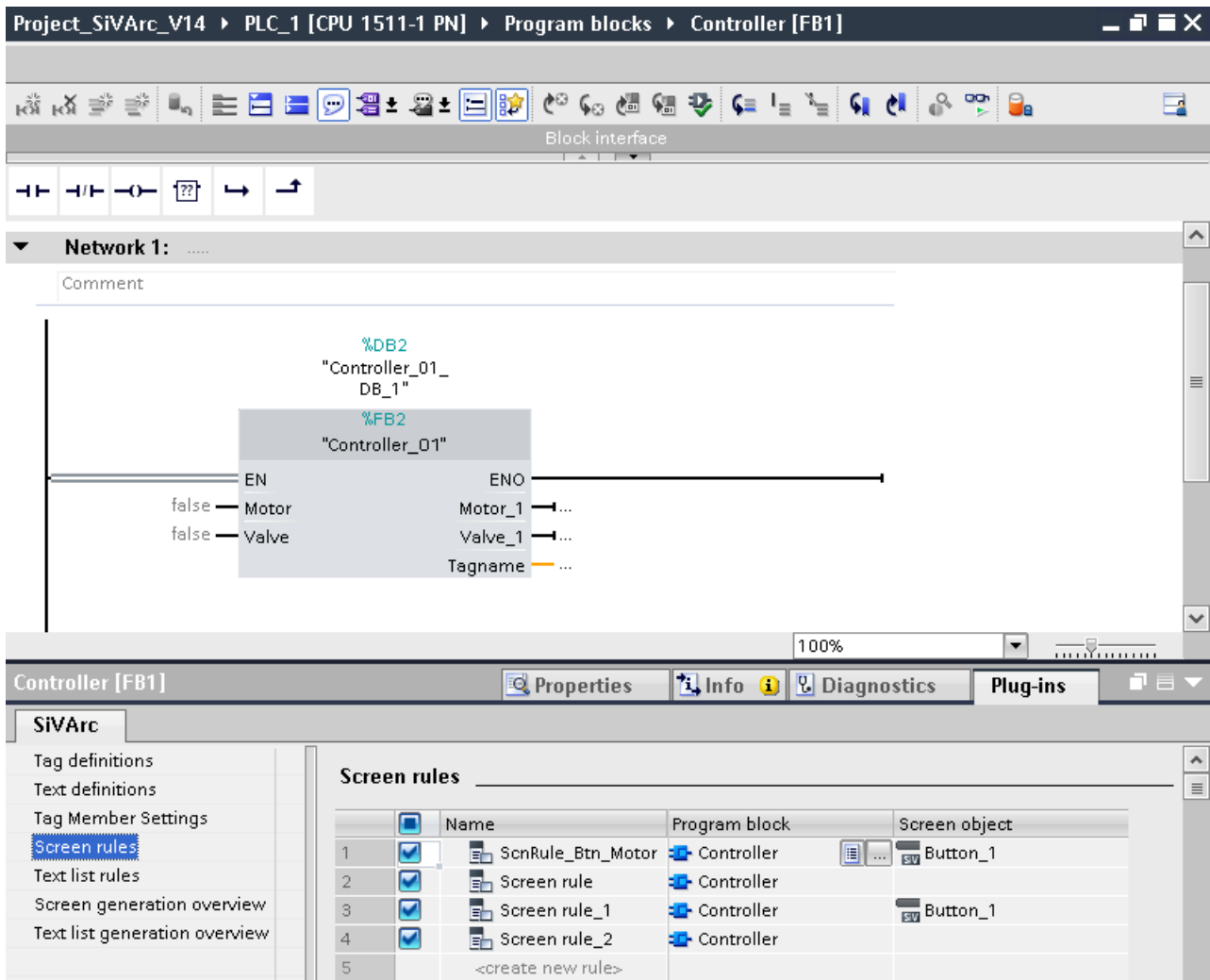
Note**Using SiVArc tags in screen rules**

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 106)

"Screen rules" editor (Page 26)

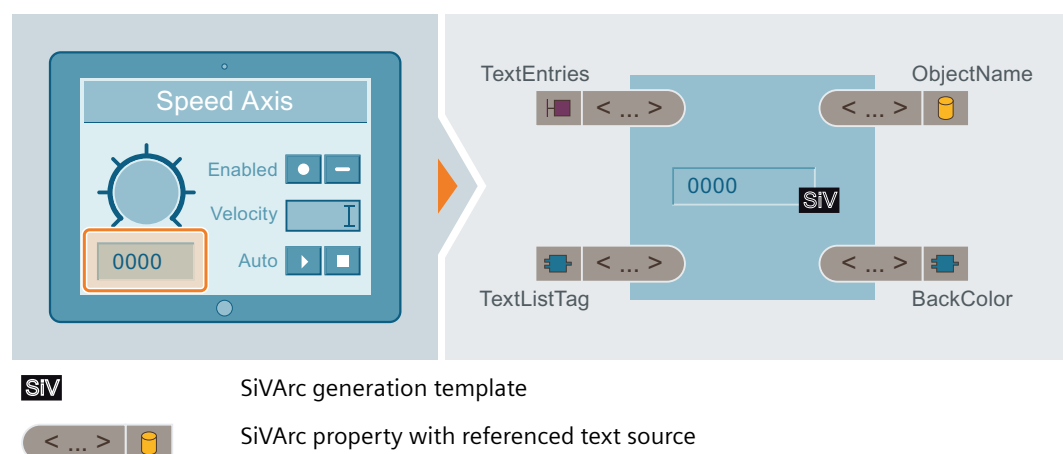
SiVArc texts (Page 101)

6.2.7 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.



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 90)

6.2.8 Parameterization concept

6.2.8.1 Example: Generating Unified faceplates

Faceplates in unified screens

SiVArc supports the following faceplate properties for a unified device:

- Tag Interface - allows you to configure tag interface name, data types (supported by tags), user data types (UDT supported by tags)
- Property interface - allows you to configure interface property, and data type (color, resource list)

Requirement

- Generation template of the faceplate type is stored in the library
- Unified screen configured in screen rule

Procedure

1. Create a faceplate for a unified device. For more information on faceplates, refer Generating faceplates. (Page 144)
2. Add a button to the unified faceplate. The plug-in properties of a button are updated automatically.
3. You can configure the tag interface and property interface for the faceplates, and the same is reflected in the plug-in properties area.
4. Configure the screen rule with screen object as unified faceplate.
5. Upon SiVArc generation, unified faceplate is generated as faceplate container. The container is linked to the faceplate type, and refers to the latest available faceplate type in the Library. The interface properties set during the configuration of faceplates is available in the generated unified faceplate property.

See also

Example: Generating faceplates with animations (Page 144)

6.2.8.2 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.

6.2.8.3 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.

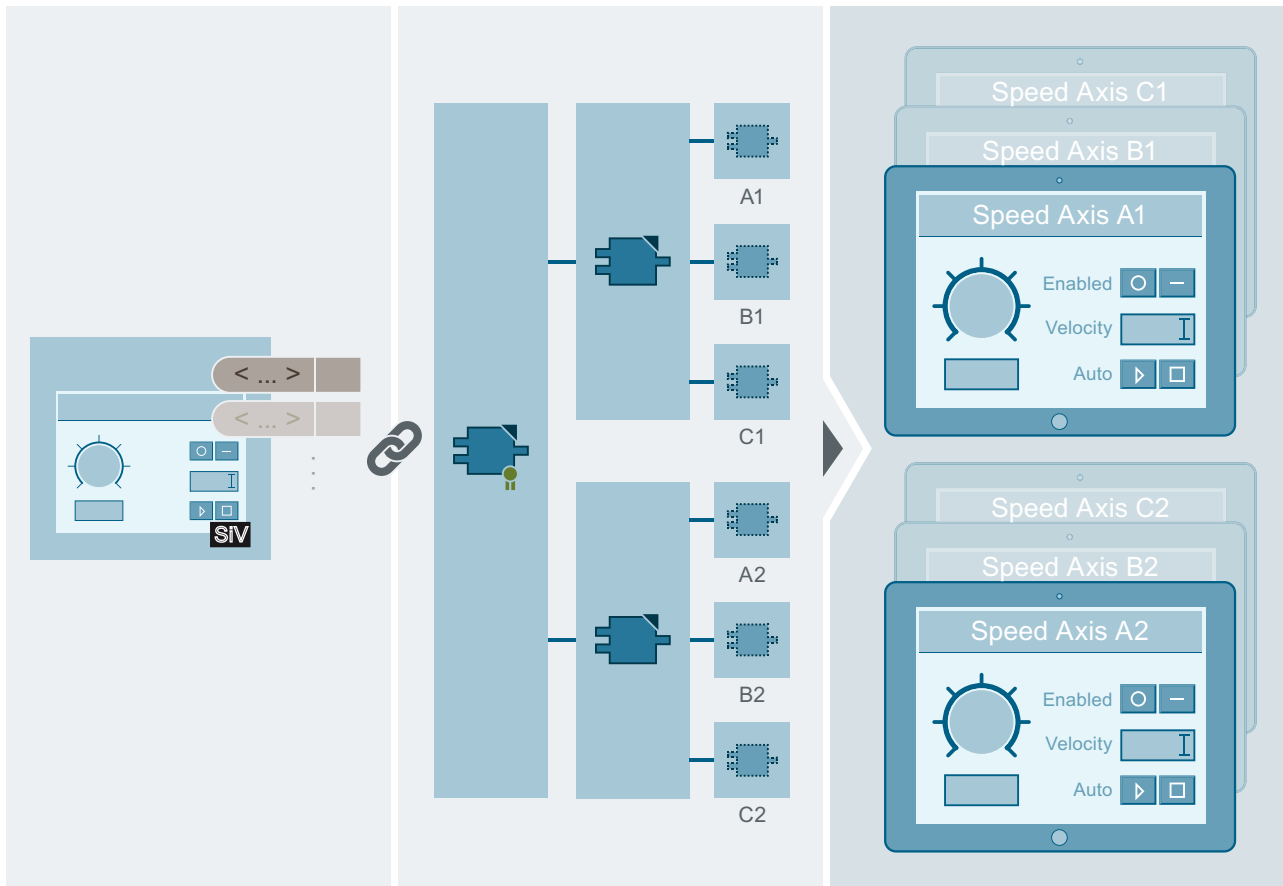
SiVArc concept





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.

6.2 Creation of generation templates

- 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.

6.2.8.4 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.

SiVArc supports symbolic expressions for organization blocks. You can define expressions using relative addressing for an organization block. The dots in the expression "...SymbolicName" indicates that in the call hierarchy, the dots lead to the main block, and hence the expression be resolved until the main block. If the dots in the expression does not lead or resolve to any value, an error is displayed.

SiVArc supports the use of absolute addressing for an organization block.

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`

6.2 Creation of generation templates

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_Instance_1_DB"	Label 01
ModuleBlock	Support function	"ProductionLine"	"ProductionLine_Instance_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 106)

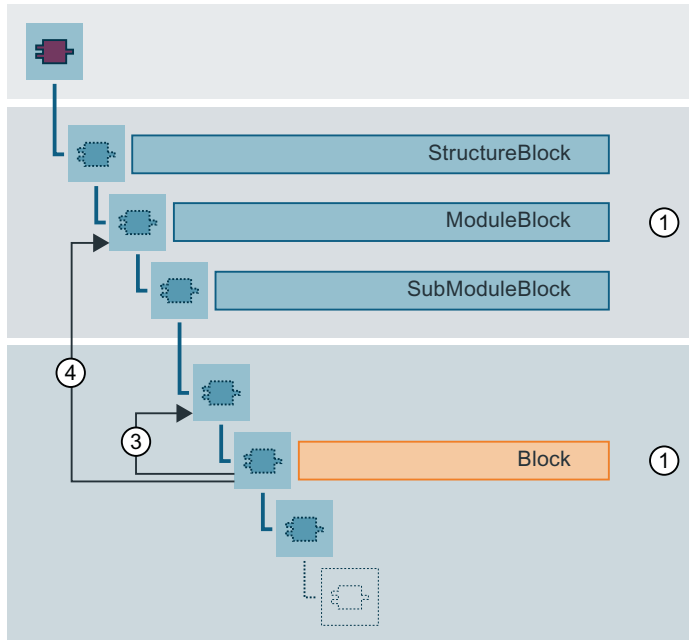
6.2.9 Influence of the user program on a generation template

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

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 246)

Overview of text source in SiVArc project (Page 98)

6.2.10 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 244)
- SymbolComment (Page 243)
- Comment (Page 237)
- DB (Page 228)
- NetworkTitle (Page 241)
- NetworkComment (Page 241)
- SiVArc texts (Page 101)

6.2.11 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.

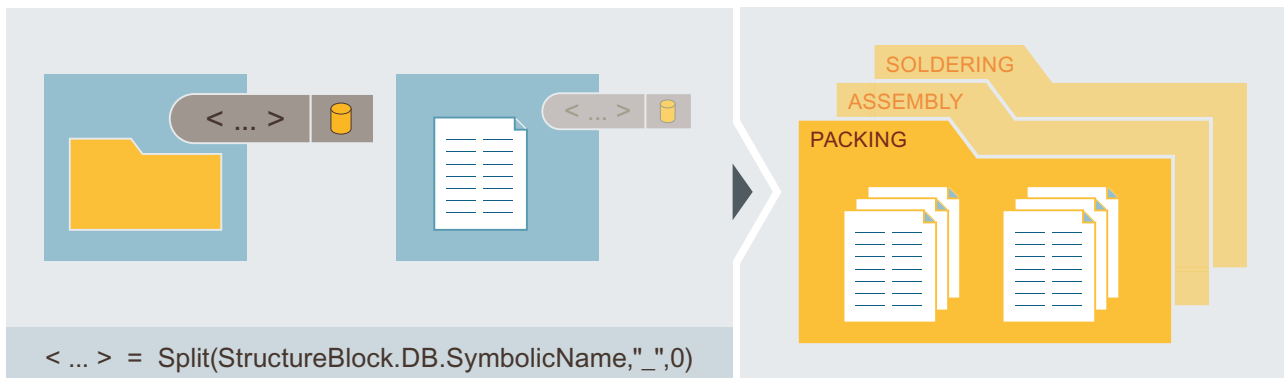
Advantage

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.



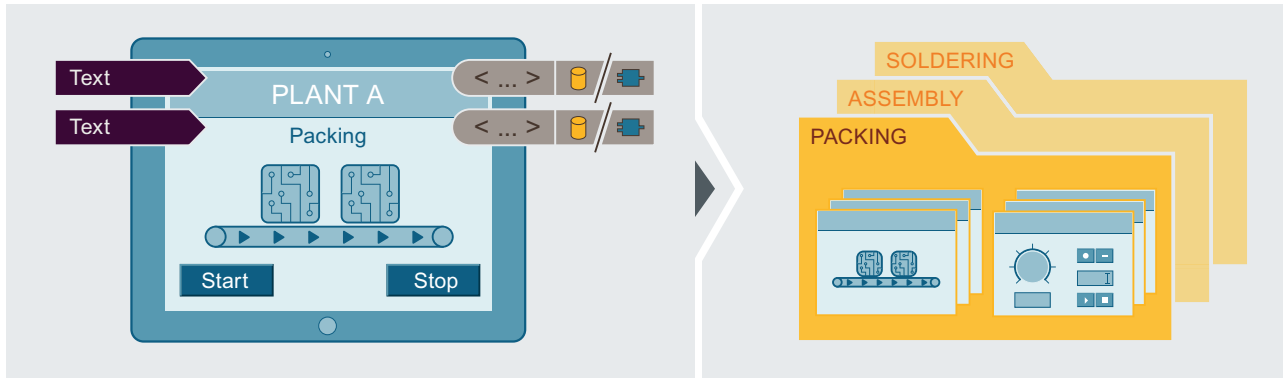
Double-click "Common data > SiVArc > tag rules" in the project tree to open the "Tag rules" editor.

Project_Sivarc > Common data > SiVArc > Tag rules					
	Name	Index	Tag group hierarchy	Tag table	Condition
1	Tag rule	0	HmiTag.DB.FolderPath	HmiTag.DB.SymbolicName	
2	<create new rule>				

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.

SIVArc properties		SIVArc animations	SIVArc events	Generation overview
Name	Printout of the static value		Printout of tags	
▼ General	Name	Split(StructureBlock.DB.SymbolicName,"_",0)		
	Display name			
	Comment			
	Screen group	Split(StructureBlock.SymbolicName,"_",0)		
	Number of overflow screens			
	Evaluate number of overflow screens as bit ...	<input type="checkbox"/>		
	Navigation button	<input checked="" type="checkbox"/>		
	Navigation button "Next"			
	Navigation button "Back"			
▶	Screen as content of screen window			
▶	Positioning scheme			
▶	Layout			

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 180)

Creating a generation template for a screen (Page 155)

Name (Page 240)

FolderPath (Page 238)

LibraryObject (Page 231)

HMITag (Page 230)

6.2.12 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.

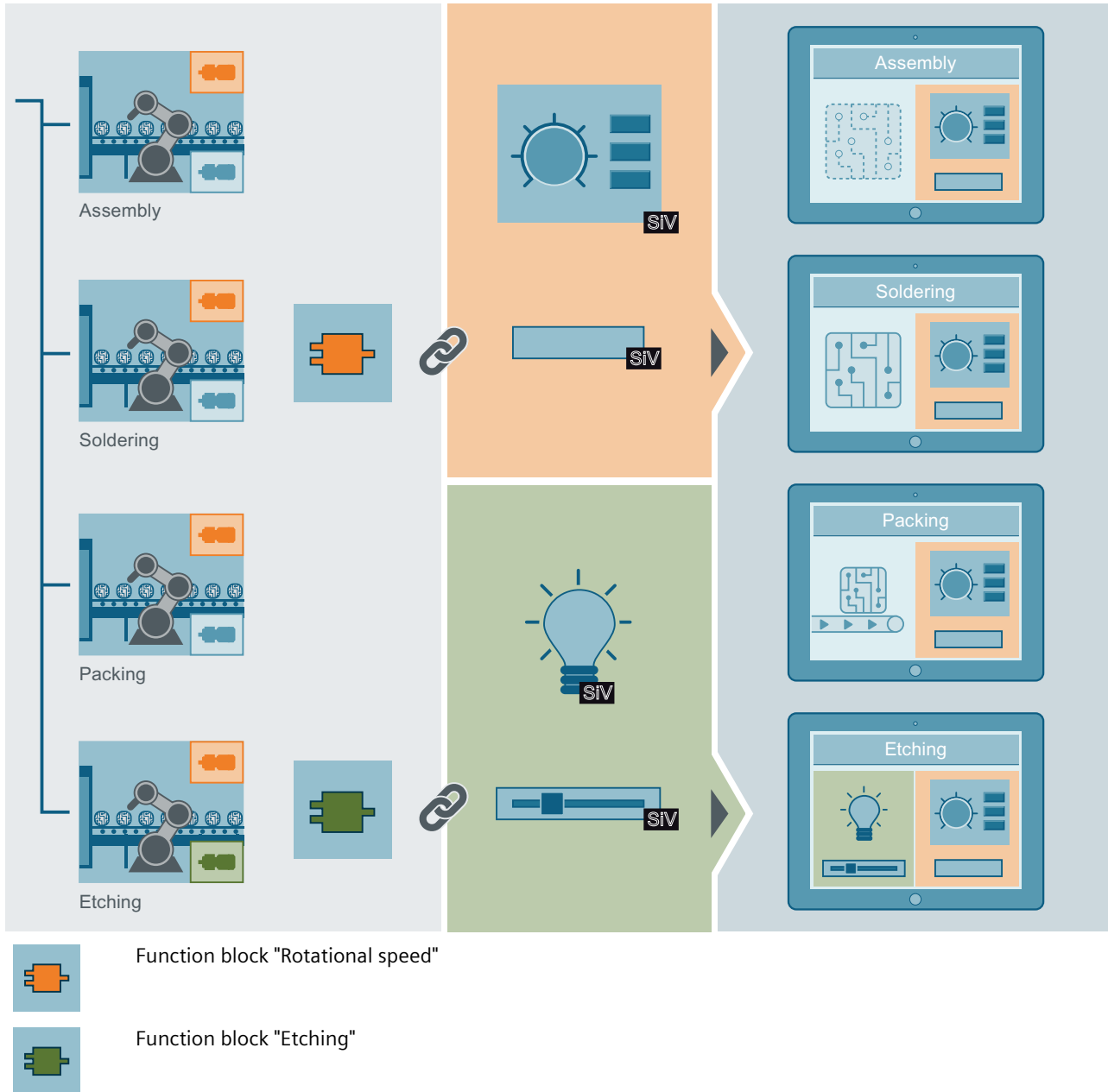
Once the new plant section is created, the modules are tested and optimized for operation.

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.

6.2 Creation of generation templates

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.



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.

6.2.13 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.

6.2.14 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.

Screen_Training [Screen]			
SiVArC properties		SiVArC animations	SiVArC events
Name		Printout of the static value	Printout of tags
▼ General			
Name		Block.DB.SymbolicName&"_SWC"	
Display name		Soldering	
Comment			
Screen group			
Number of overflow screens			
Evaluate number of overflow screens as bit ma...	<input type="checkbox"/>		
Navigation button	<input checked="" type="checkbox"/>		
Navigation button "Next"			
Navigation button "Back"			
▼ Screen as content of screen window			
Name of the screen window		Training_SWC_01	
Tag prefix or process tag		Soldering_	
Title			
Generate additional screens	<input type="checkbox"/>		
▶ Position			
▶ Positioning scheme			
▶ Layout			

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.

GettingStartedSiVArCv2.0_Complete_V14_1 ▶ Common data ▶ SiVArC ▶ Screen rules					
	Name	Program block	Screen object	Master copy of a screen	Comment
1	Plantsection_Title	Plantsection_Soldering	Plantsection_Title	Plantscreen	
2	Plantsection_Stat...	Plantsection_Soldering	PlantStatus_Symb...	Plantscreen	
3	Productionline_Title	Productionline	Productionline_title	Plantscreen	
4	Conveyor	Conveyor	Conveyor	Plantscreen	
5	Productionline_Po...	Productionline	Position_IO	Plantscreen	
6	Processing_Unit	Processing	ProcessingUnit	Plantscreen	
7	Activate_Btn	Activate	Function_Activate	Plantscreen	
8	Stop_Btn	Stop	Function_Stop	Plantscreen	
9	Station_Training	Plantsection_Solder...	Startscreen	Screen_Training	Training_Only
10	<create new rule>				

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 130)

6.2.15 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 90)

6.2.16 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.

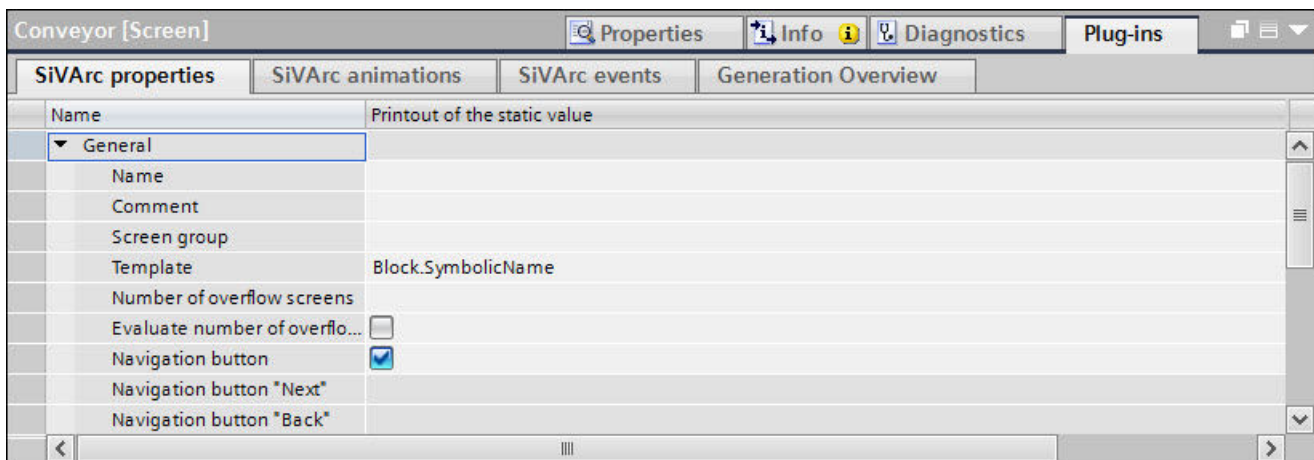
Requirement

- 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 expression 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.

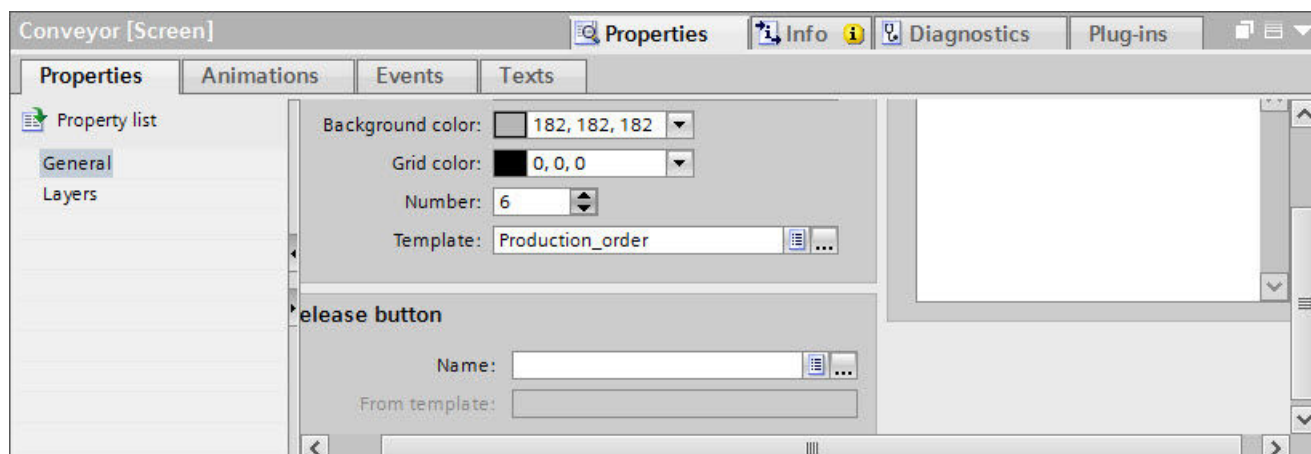


5. Place the configured HMI screen in the "Master copies" folder.
6. Under "SiVArc > Screen rules", click "Create new rule". Enter the name of the screen rule as "Conveyor rule".
7. Under "Library object", browse for a PLC block , and the screen template from the "Master copies" folder. Library object 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 90)

6.2.17 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

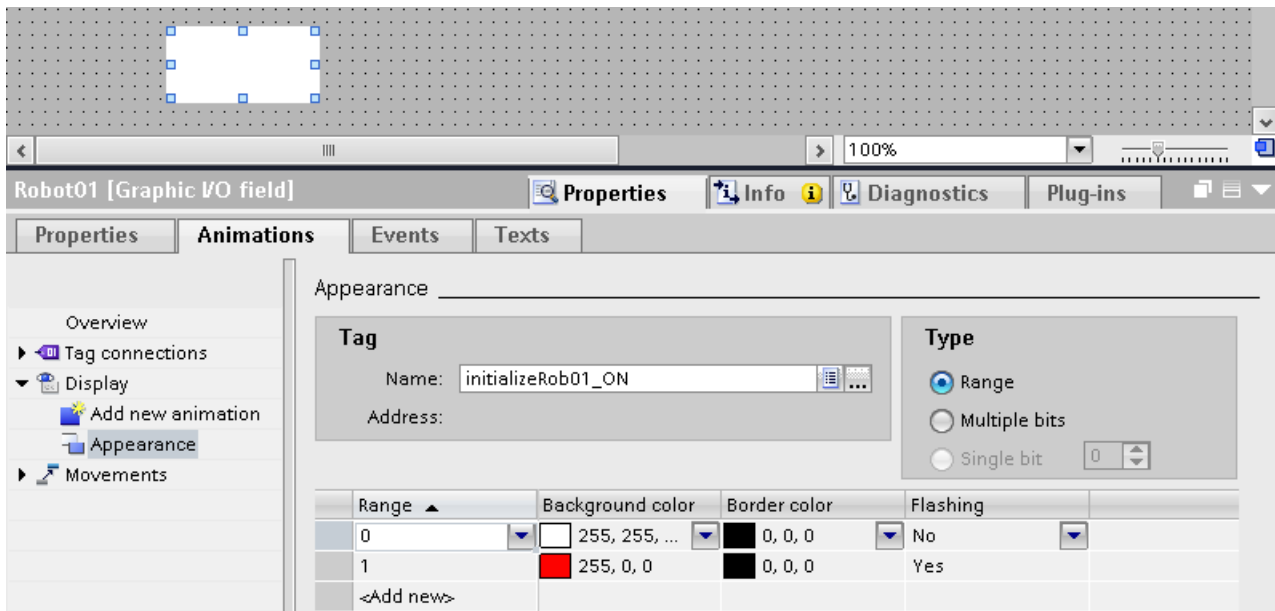
- A "Robot01" graphic I/O field is configured as generation template to display the robot.
- The PLC tags are synchronized with the HMI tags.

6.2 Creation of generation templates

- 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.

6.2.18 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.

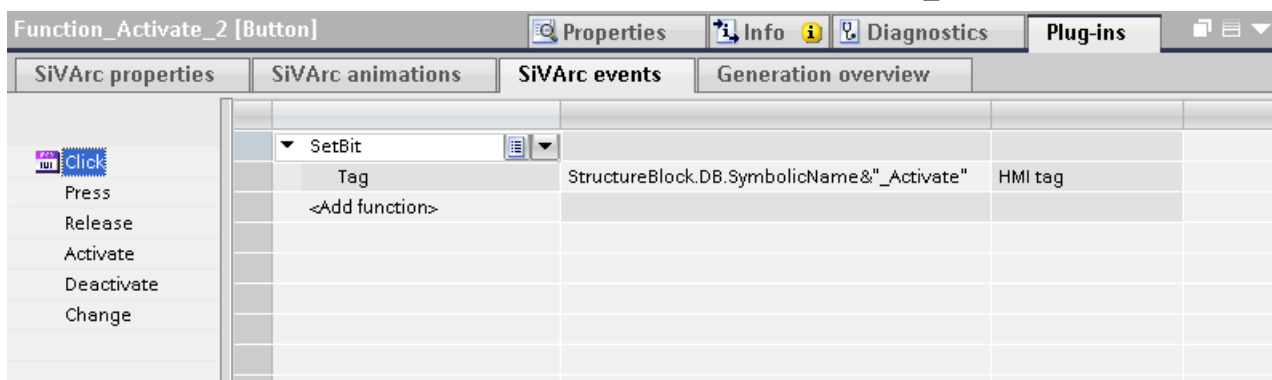
Requirement

- 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.

Events for WinCC Unified devices

- Events support script function for screen and screen items. It is not mandatory to configure the parameters of script functions. By default, the parameter type will be "none" if the function parameter is not configured.
- You can configure "SetLanguage" parameters through "LCID parameter with language ID or language notation. Example Language ID as "1033"; language notation as "en-US.
- For parameters of "SelectionType", you must manually configure the supported values else SiVArc displays an error message.
- In case of invalid values system displays warning message, and no changes are done to the engineering system.

6.2.19 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.

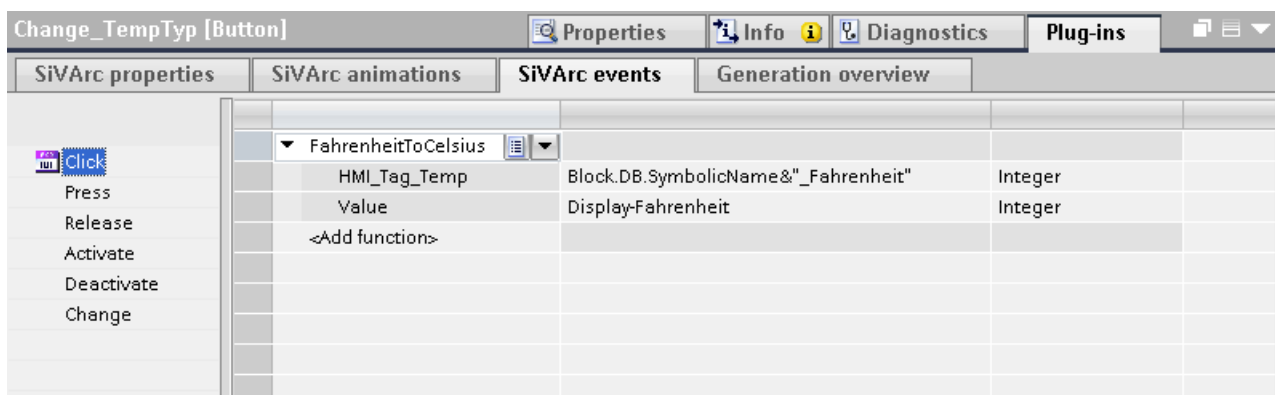
Additional information on device dependency of events is available in the online help of the TIA Portal in the section "Working with system functions and Runtime scripting" in the reference.

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

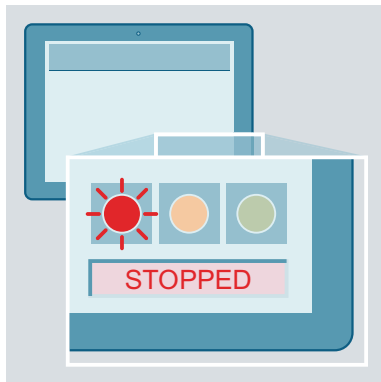
When the visualization is generated, the "Change_TempTyp" button and the "Display_Fahrenheit" I/O field are created for each call of the relevant function block.

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.

6.2.20 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.

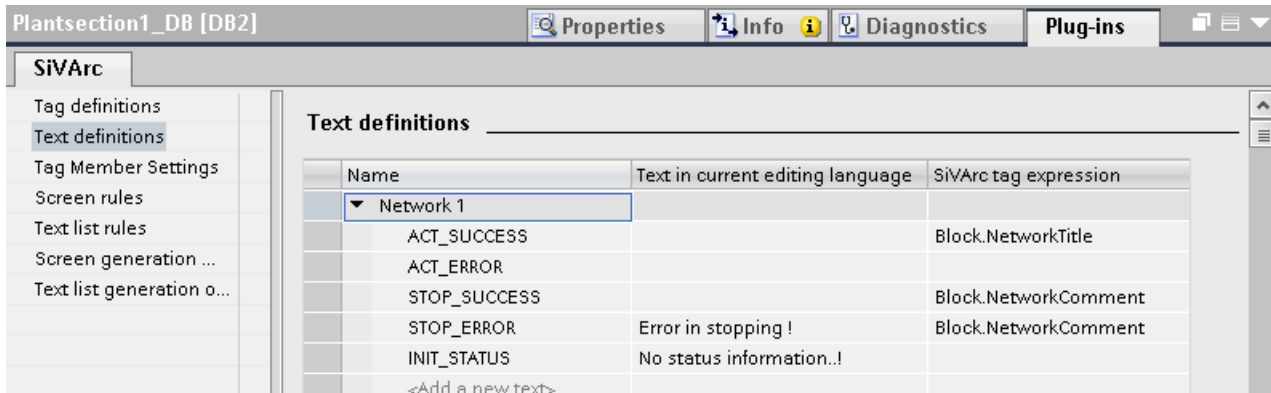
Requirement

- 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.
You can configure the text definition with SiVArc expression that would resolve to the text list names if placed in mastercopy. Generation of text lists with similar names of default text lists cannot be performed.

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.

6.2 Creation of generation templates

- 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".

The screenshot shows the 'Text list entries' table and the 'SiVArc properties' configuration window.

Default	Value ▲	Name	Text
<input type="radio"/>	0 - 1	ACT_SUCCESS	RUN
<input type="radio"/>	2 - 3	ACT_ERROR	ERROR
<input type="radio"/>	4 - 5	STOP_SUCCESS	STOP
<input type="radio"/>	6 - 7	STOP_ERROR	STOP ERROR
<input type="radio"/>	8 - 9	INIT_STATUS	START
	<Add new>		

Name	Printout of the static value	Printout of tags
General		
Name	Split(StructureBlock.DB.SymbolicName,"_",0)"_Textlist"	
Comment		
Use block par... <input type="checkbox"/>		
Block para...		
I/O type	Input	
<Add new dat...		

- 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.

For Unified devices, you can configure the tag table for a PLC with multilingual comments, and simultaneously configure text list which is placed as master copy, System compares the tag table with multilingual comments and the tag table in the function block. The matched entries are available in a newly created HMI text list table

Points to remember

- For WinCC Unified devices, you can configure the tag table for a PLC with multilingual comments, and simultaneously configure text list which is placed as master copy, System compares the tag table with multilingual comments and the tag table in the function block. The matched entries are available in a newly created HMI text list table.
- PLC connected to WinCC Unified and Runtime Advanced devices, you can create a text list entry on WinCC Unified device, and add the text list to master copy. Under text list rule creation, if the device specific columns are selected for WinCC Unified and Runtime Advanced devices, upon SiVArc generation the text lists are generated only on Unified device, and an error is displayed on the Runtime Advanced device. The same applies for copy rule.

6.2.21 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.

6.2.22 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.

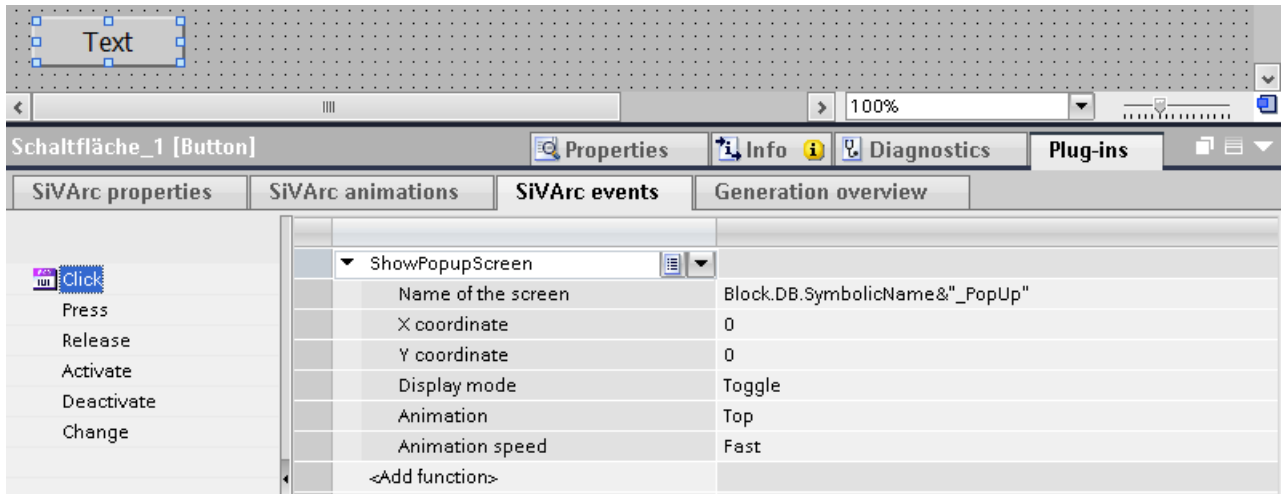
Create generation template for the calling button

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.

6.2 Creation of generation templates

- 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.

6.2.23 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.

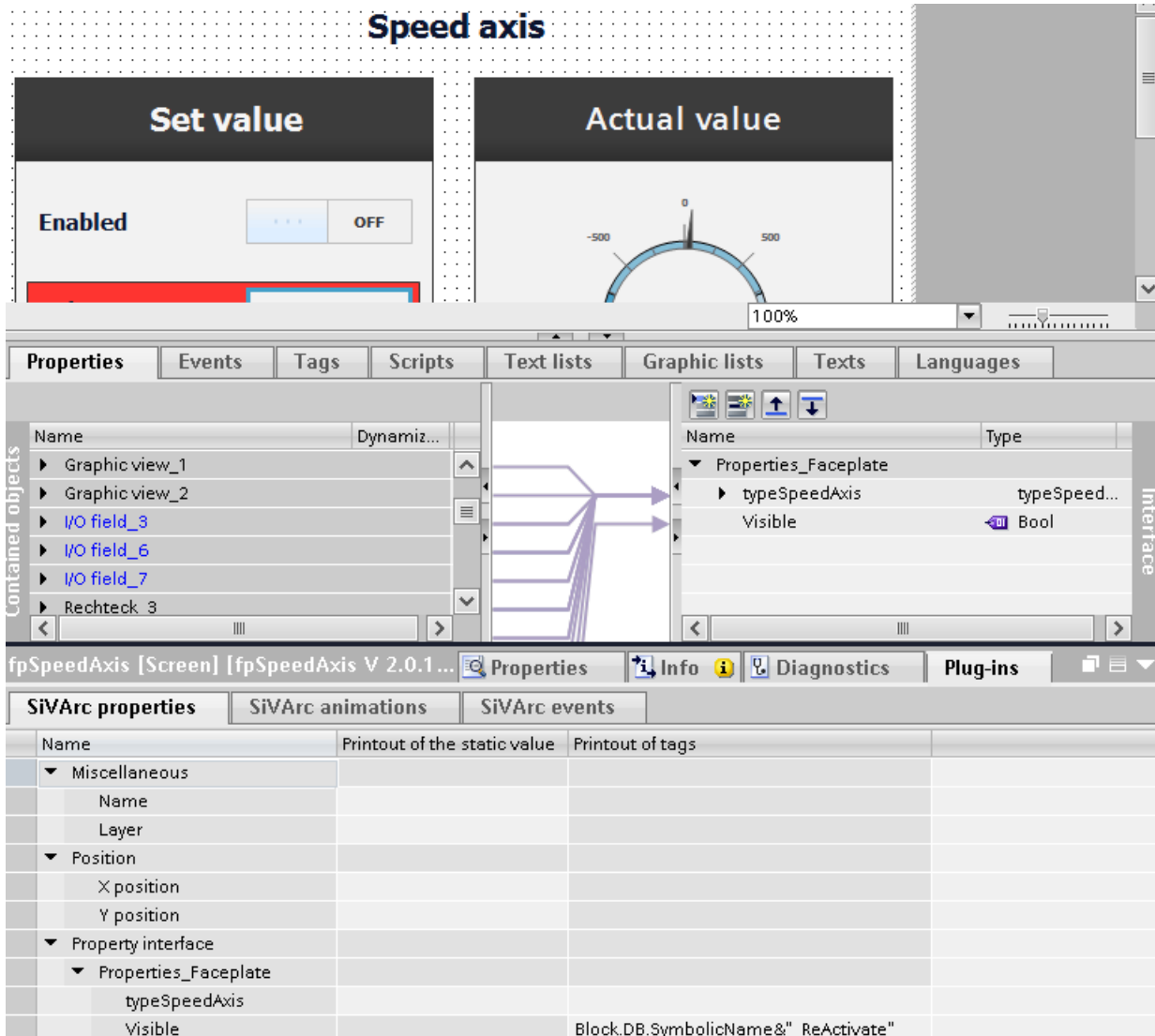
Procedure

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.

6.2 Creation of generation templates

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



- Create a new faceplate type version as a generation template.
- 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.

Merging interface properties for faceplates

While configuring a faceplate with interface properties of screen items, you can retain the changes made for static and dynamic values. The **Retain manual changes** checkbox in faceplates **Plug-ins** editor allows you to retain the changes made for interface property values on the current faceplate instance.

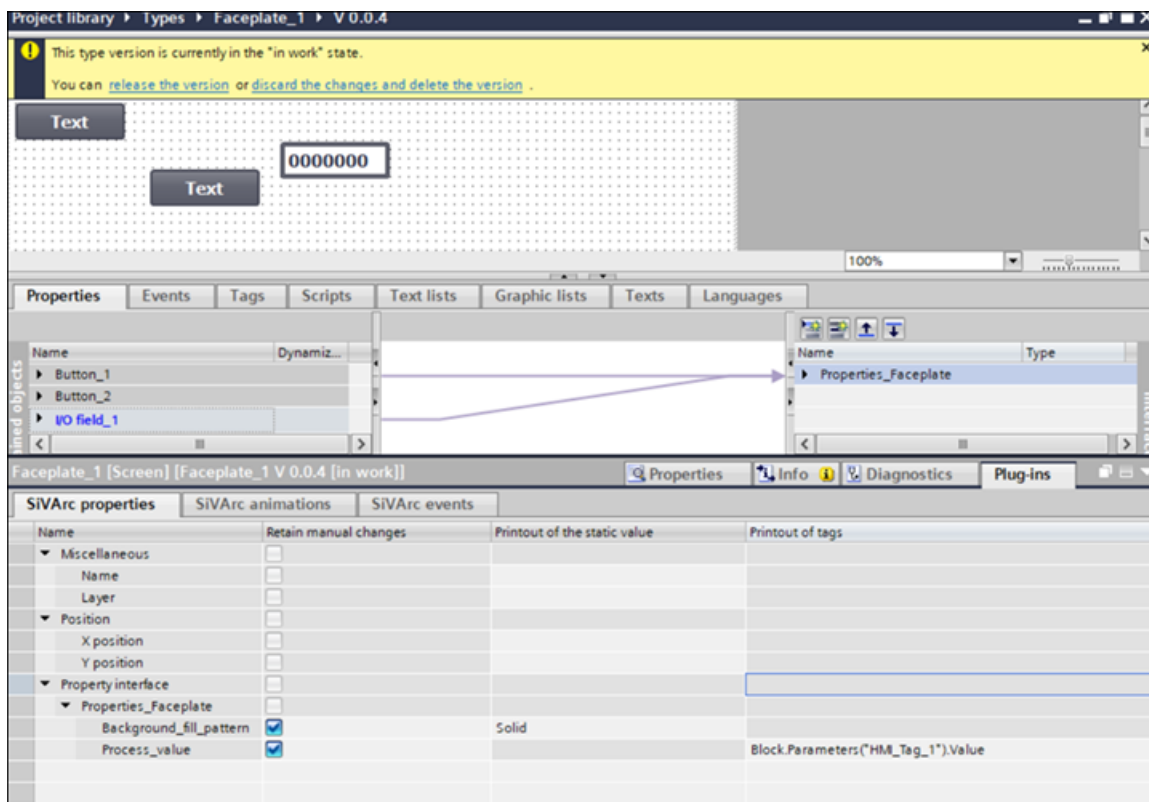
Post SiVArc generation, when the faceplate instance is created, and **Retain manual changes** checkbox is selected :

- If a user manually changes the property value in the generated faceplate instance, upon SiVArc generation, the values are retained.

Procedure

To generate SiVArc using faceplates interface properties in screens, perform the following steps:

1. Configure a faceplate's interface properties with screen items.
2. Select the "Retain manual changes" checkbox, and configure the value for static column and "Tag expression" for dynamic column.

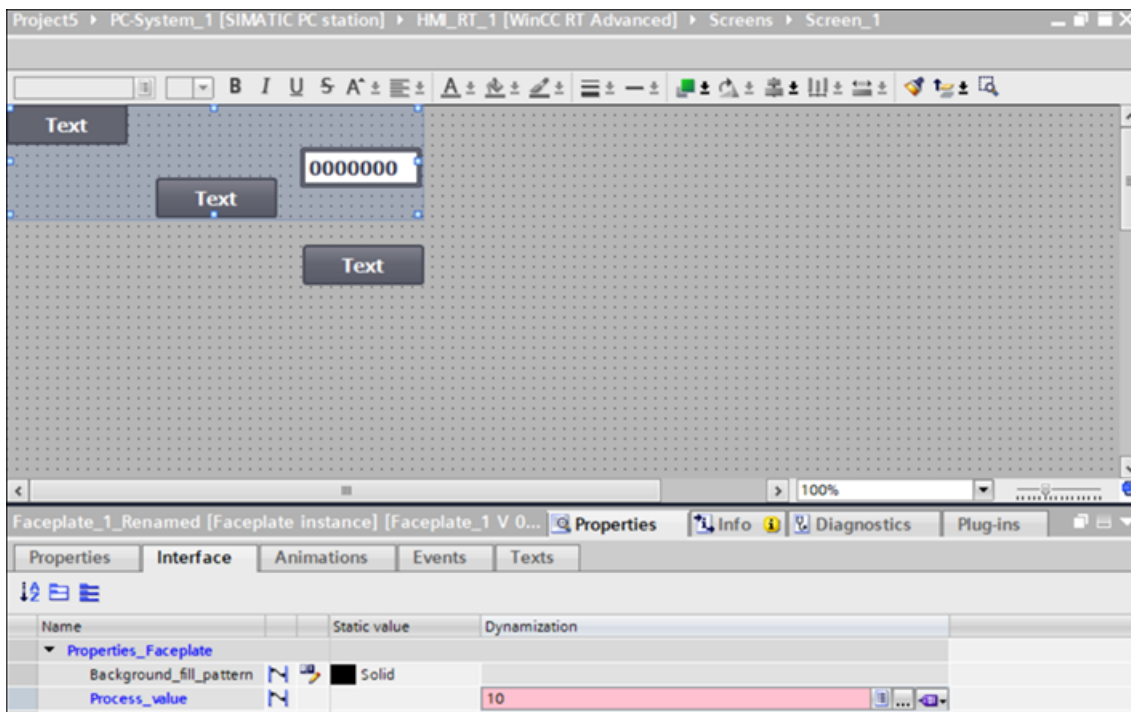


3. Create a screen rule with screen object as faceplate, and screen as "Master copy of a screen". For more details on configuring screen rules, see creating rules (Page 41).
4. Generate SiVArc visualization.
5. The generated screen object contains the faceplate's interface properties. For more information on generation, see Generation (Page 41).

Editing the instance of Faceplate properties

Post SiVArc generation, you can edit the instance of a faceplate that is generated on the screen. When "Retain manual changes" check box is selected, any changes performed on the fly to the Interface property of faceplate instance that is generated will remain unchanged for the next generation.

Consider the following scenario where the faceplate interface property contains "Process_value" as 10 that resulted from first generation. After generation, the faceplate's interface property is modified with "Retain manual changes" checkbox selected in the faceplate's library window, and SiVArc generation is triggered. For all subsequent generations, upon changing the interface property, the "Process_value" will change according to the manual changes. This is applicable for Static properties. Eg: Background_fill-pattern.



Exceptional cases:

- Manual changes will be applicable only if similar versions are available in the faceplate library and the generated faceplate.
- If faceplate is deleted, the "Retain manual changes" will not be applicable, and the faceplate will be generated from the latest available version.
- Static interface properties are not supported for Advanced faceplates where dynamic properties are supported.
- For Advanced faceplate, some of the properties which cannot be retained by SiVArc will have values set by Engineering system.
- For dynamic values, input type supported is of HMI tags.

See also

"Screen rules" editor (Page 26)

"Generation matrix" editor (Page 35)

Generation overview (Page 41)

6.2.24 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.

6.2 Creation of generation templates

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".

6.2.25 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.

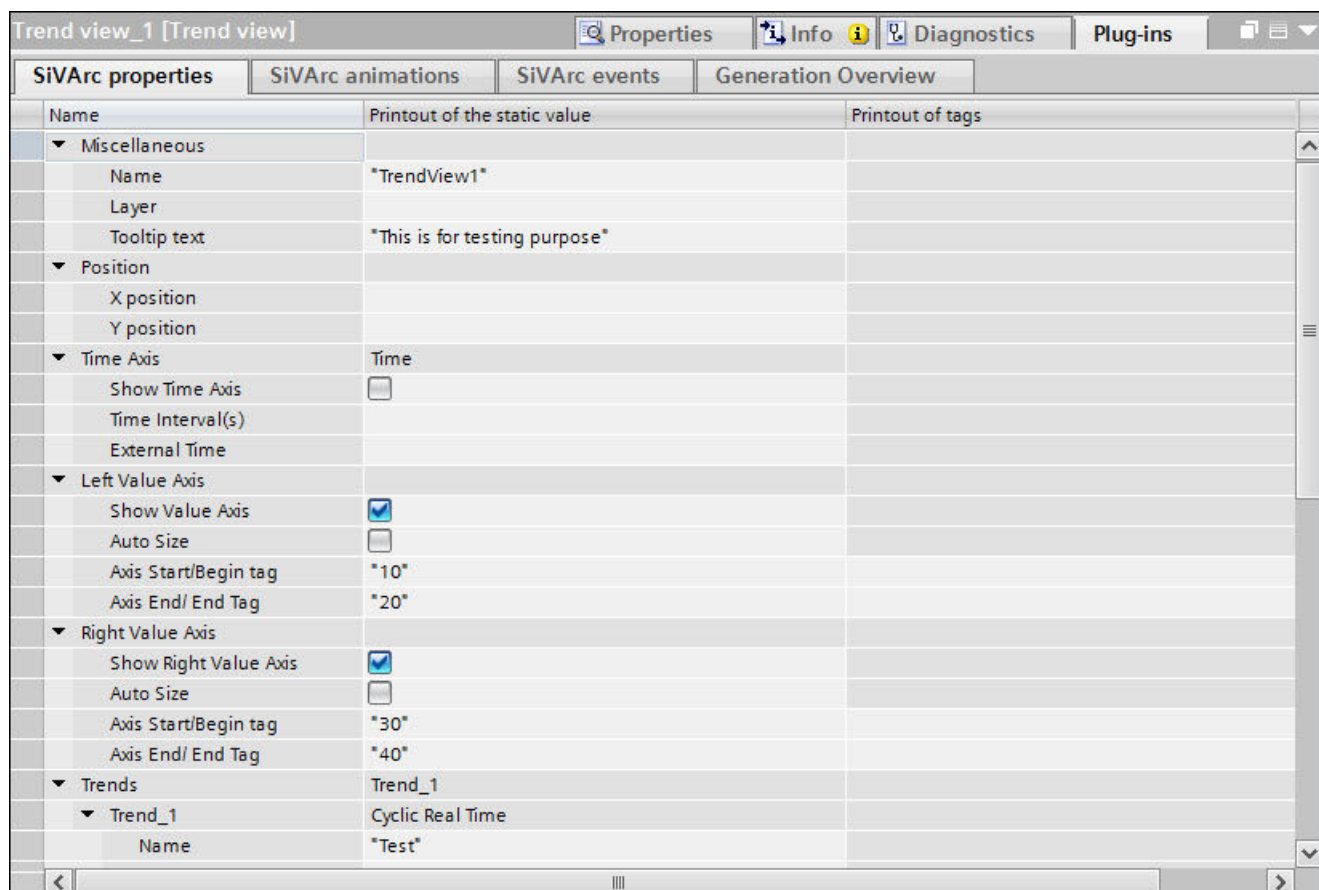
Example: Creating generation templates for trend views

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.

6.2.26 Example: Hardware configuration for screen

Introduction

You can configure IO devices connected to the PLC, and can generate screens (through expressions) that result in IO device name, Invariant type, and Article number. IO devices connected to the PLC can be of similar or different Invariant type. When an IO device is assigned to a PLC, and you try to configure SiVArc rules, the Rule Trigger column displays the Invariant type available for the connected PLC. SiVArc generation results in creating screens for single or multiple instances of IO device names, invariant type, and Article number.

Example scenario

Consider programming the assemblies of parts in an automotive manufacturing plant, where PLC is connected to IO device with similar or different Invariant types.

Objective

In automotive manufacturing plant, multiple HMI screens are used to visualize the assembling of parts. You configure SiVArc properties with expressions that result in generation of screens with names of IO devices that are assigned to the PLC. Thus, resulting in increase productivity and scalability across multiple HMI devices.

Requirement

- PLC assigned to single or multiple IO devices (ET200 Profinet only) with similar or different invariant types
- HMI device connected to PLC
- SiVArc expression that supports IO device name, Invariant type, Article number

Procedure

To generate HMI screens using IO devices, follow these steps:

1. Assign PLC with single or multiple IO devices with Invariant type of your choice.
2. Add a screen. In the inspector window, under "Plug-ins > SiVArc properties > Name", configure the screen property with expression that results in IO device name, Invariant type and Article number. Eg: "`Device.Name`". The resolving expression name will be displayed in the target screen.
3. Add the screen into the Master copies folder with the screen object.
4. Under "SiVArc > Screen rules", click "Create new rule".
5. In the "Rule Trigger column, browse for Invariant folder of your choice, and the screen object from the "Master copies" folder.
6. Generate SiVArc visualization on the HMI device.

Note

- Screens will not be generated for IO devices that are under "Master Copies" > "Types".
 - In case of screen rules containing blocks and invariant types, screen rules with blocks are executed first, followed by the Invariant types.
-

Result

An HMI screen "<<device_name>>" is generated. Eg: IO device_1.

6.2.27 Example: Hardware configuration for alarms

Introduction

You can configure IO devices connected to the PLC, and can generate alarms, alarm groups, and alarm classes (through expressions) that result in IO device name, Invariant type, and Article number. IO devices connected to the PLC can be of similar or different invariant type. When an IO device is assigned to a PLC, and you try to configure SiVArc rules, the Rule Trigger column displays the Invariant type available for the connected PLC. SiVArc generation results in creating alarms for single or multiple instances of IO device names, Invariant type, and Article number.

Example scenario

Consider monitoring the status of a manufacturing plant. An alarm with a specific device name indicates that the device needs important attention.

Objective

In automotive manufacturing plant, multiple HMI alarms are used to monitor the statuses of assembling parts. You configure SiVArc properties with expressions that result in generation of alarms with names of IO devices that are assigned to the PLC.

Requirement

- PLC assigned to single or multiple IO devices (ET200 Profinet only) with similar or different Invariant types
- HMI device connected to PLC
- SiVArc expression that supports IO device name, Invariant type, Article number

Procedure

To generate HMI alarms using IO devices, follow these steps:

1. Assign PLC with single or multiple IO devices with Invariant type of your choice.
2. Select "HMI alarms" and click "Add new" to add a new alarm.
3. In the inspector window, under "Plug-ins > SiVArc properties > Name", configure the alarm name property with expression that results in IO device name, Invariant type and Article number. Eg: "Device.Name". The resolving expression name will be displayed in the target alarm.
4. Add the alarm into the "Master copies" folder.
5. Under "SiVArc > Alarm rules", click "Create new rule".
6. In the "Rule Trigger column, browse for Invariant folder of your choice, and the alarm object from the "Master copies" folder.
7. Generate SiVArc visualization on the HMI device.

Note

- Alarms will not be generated for IO devices that are under "Master Copies" >"Types".
 - In case of alarm rules containing blocks and Invariant types, alarm rules with blocks are executed first, followed by the Invariant types.
-

Result

An HMI alarm "<<device_name">>" is generated. Eg: IO device_1.

6.2.28 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 181)

Generating visualization (Page 207)

SiVArc object properties (Page 246)

Storage strategies for generated objects (Page 122)

6.3 Creating rules

6.3.1 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
 - Alarms
 - Tag tables
 - Graphic lists
 - Graphics table
 - Graphics
 - Scheduled tasks

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. While configuring screen rules, you can browse for "Screen type" in Master copy/Type of a screen". Screen types can be configured with properties and events.

You can add a new rule/rule group at any desired index within any rule editor by right clicking the index > insert a new rule/insert a new rule group.

- While adding new rule at an index point, the new rule will be inserted immediately below the selected index, and named as "*selected Index name_ <<incremental number>>*".
- While adding a new rule group, the new rule group will be added at the first index point within the selected rule group, and named as "*index group name_ << <<incremental number>>*".

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.

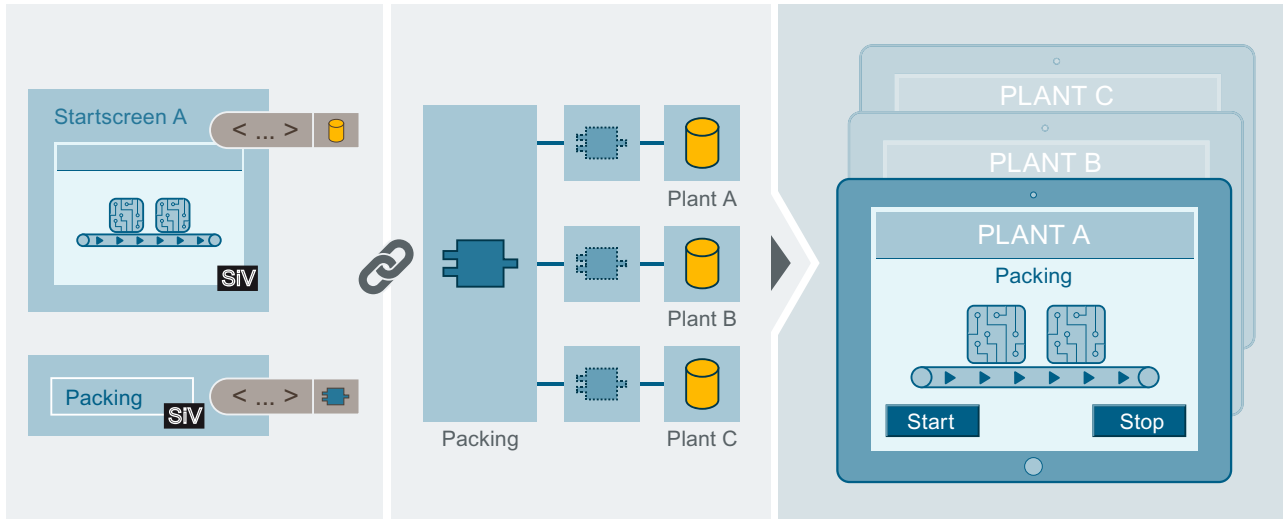
You can also create the visualization separately for individual devices using the SiVArc rules.






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:



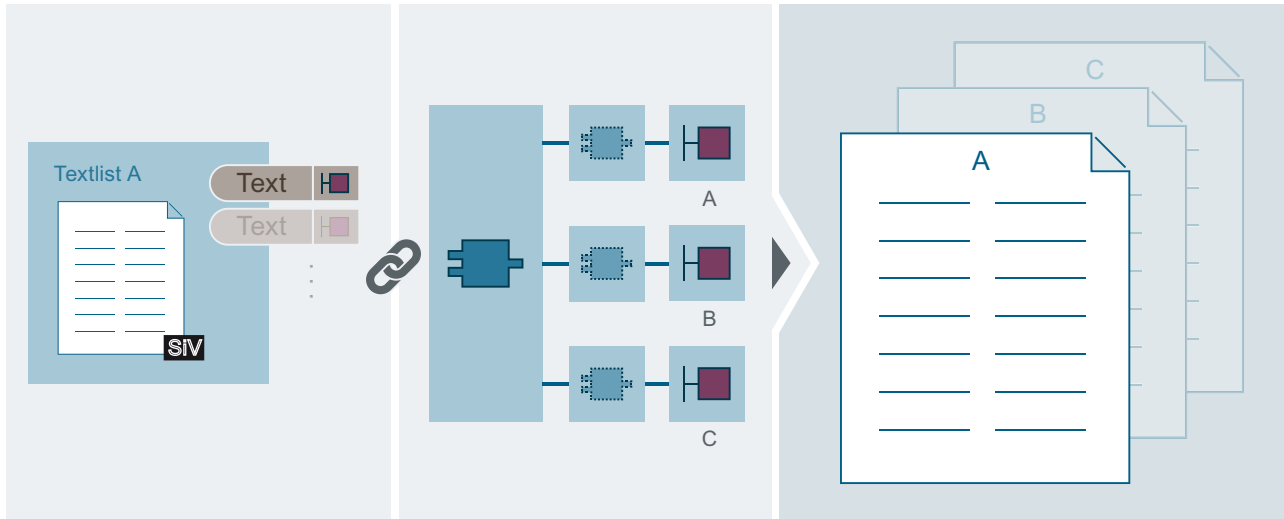
-  SiVArc generation template
-  SiVArc property with referenced text source
-  Process instruction
-  Instance of an instruction in Main OB
-  Data block






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:



	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 197)

Changing SiVArc rules (Page 205)

Creating SiVArc rules (Page 160)

6.3 Creating rules

Creating a screen rule (Page 181)

Creating text list rules (Page 183)

6.3.2 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.

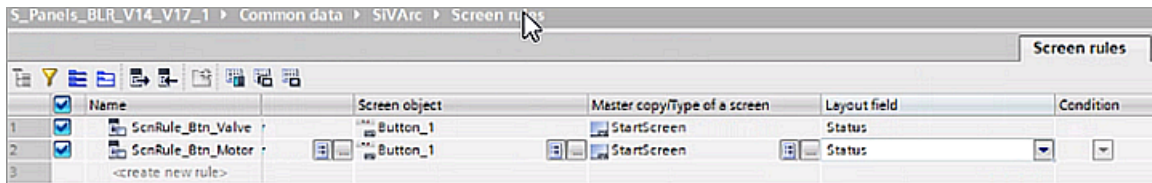
The SiVArc editors are table editors. Each type of SiVArc rule has its own editor. The editors have different objectives.

Access to the SiVArc rule editors

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

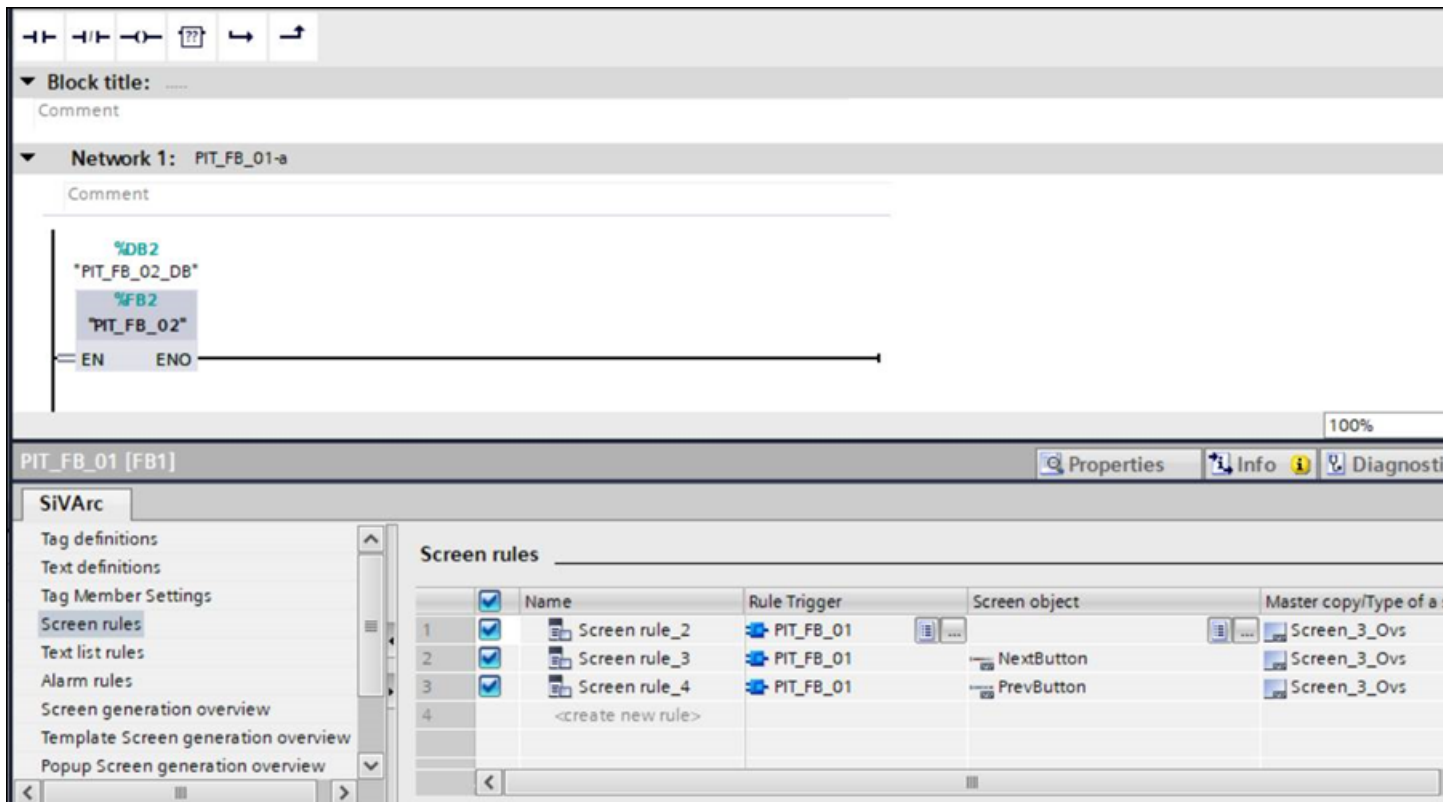
You hide or show individual columns using the icons in the toolbar, for example, the columns "PLC", "HMI device" or "HMI device type".

You can choose to generate screens by configuring expressions in the "Condition" column that resolves to the target device's width and height.



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.

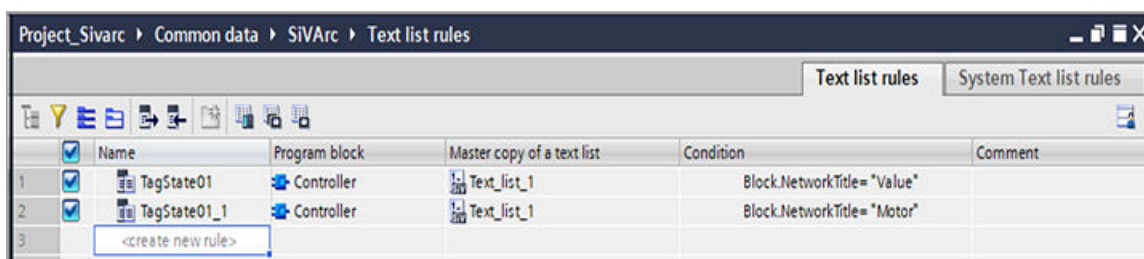
You only remove the know-how protection of SiVArc rules in STEP 7 with the commands in the shortcut menu in the project tree under "Common data > SiVArc".

The "Template screen generation overview" and the "Popup screen generation overview" tabs are 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.



6.3 Creating rules

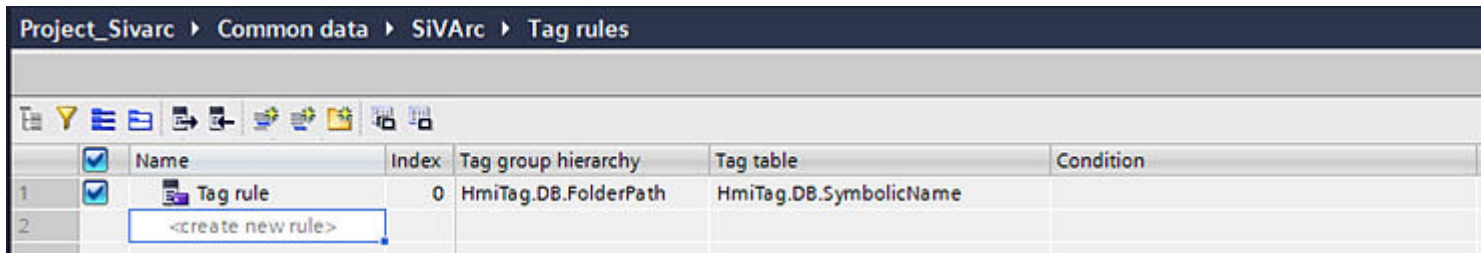
In doing so, you create the rules and specify the following for each rule:

- 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.

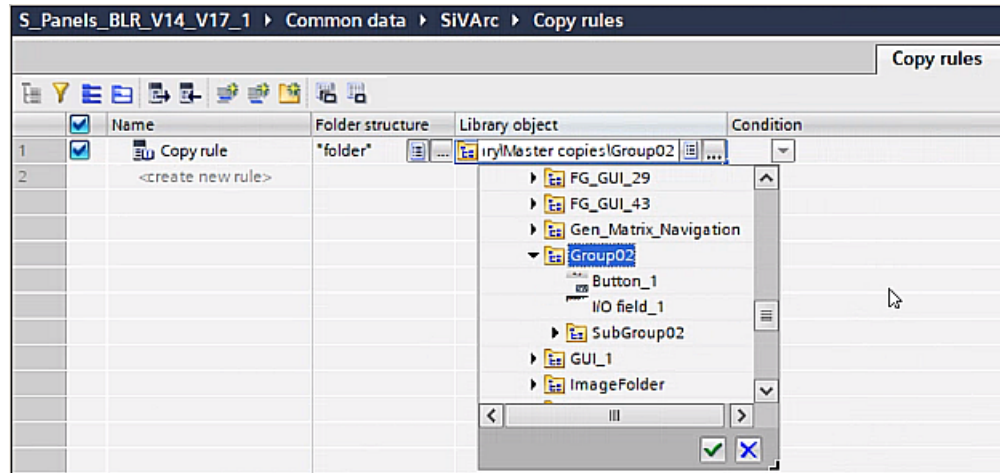


In doing so, you create the rules and define the following for the generated tags:

- 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 194)
- Setting up know-how protection for a SiVArc project (Page 197)
- Editing and managing SiVArc rules (Page 192)
- Creating a screen rule (Page 181)
- Creating text list rules (Page 183)
- Editing the view in the SiVArc editors (Page 264)

6.3.3 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.

6.3 Creating rules

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

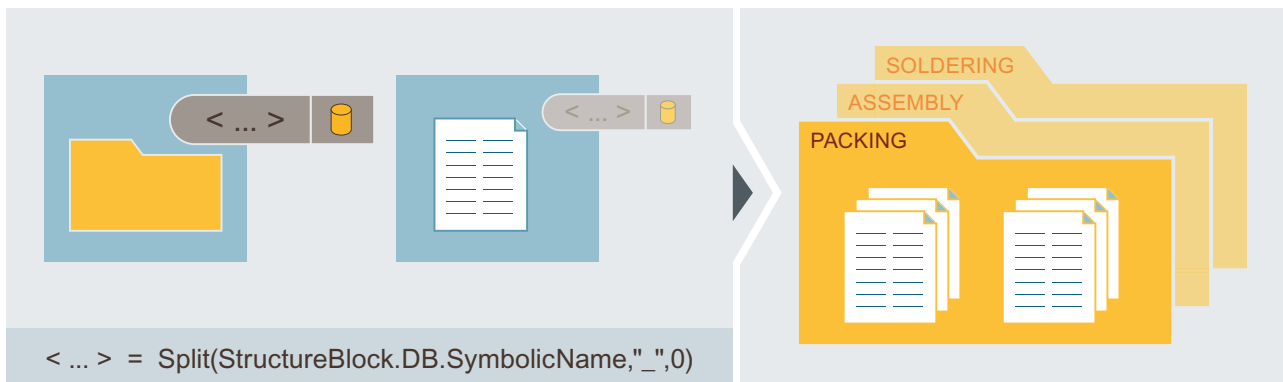
Conditions in SiVArc rules

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.

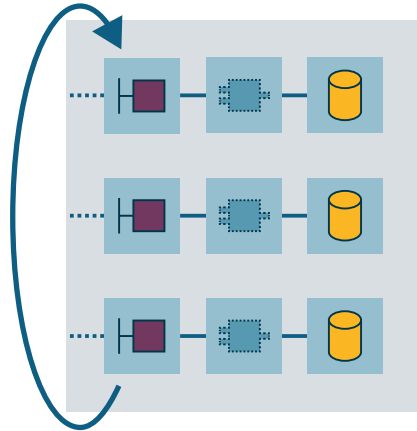
Only in the "Tag rules" editor do you use SiVArc expressions that address the SiVArc object `HMI Tag`.

	<input checked="" type="checkbox"/>	Name	Index	Tag group hierarchy	Tag table	Condition
1	<input checked="" type="checkbox"/>	Plantsection_1	0	HmiTag.DB.FolderPath	HmiTag.DB.SymbolicName	
2		<create new rule>				

6.3.4 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.

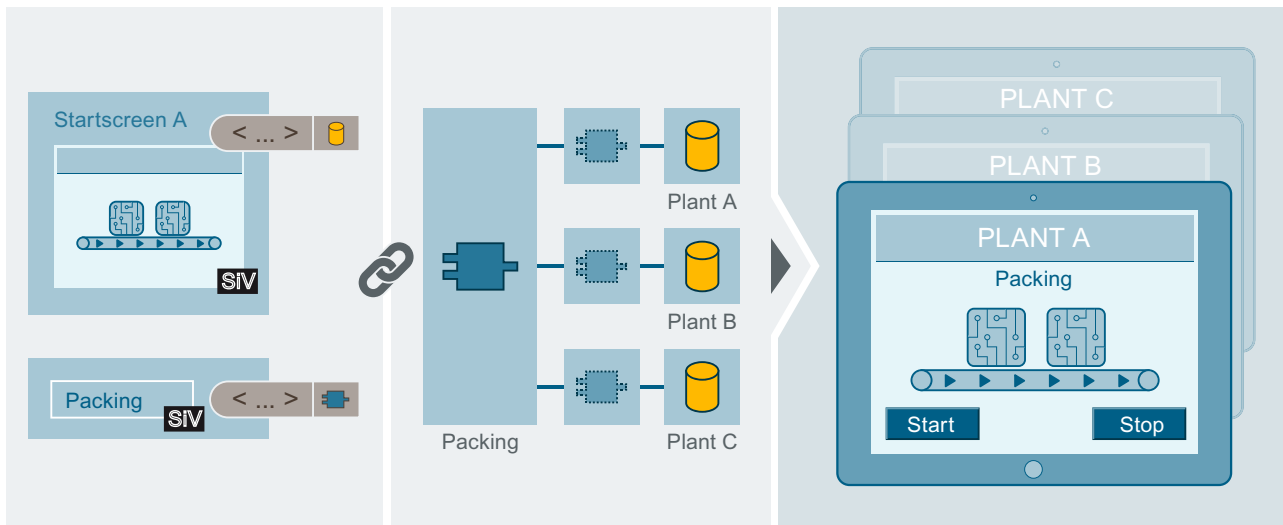
All screen and text list rules of a project that are enabled and meet a configured condition are executed.

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.

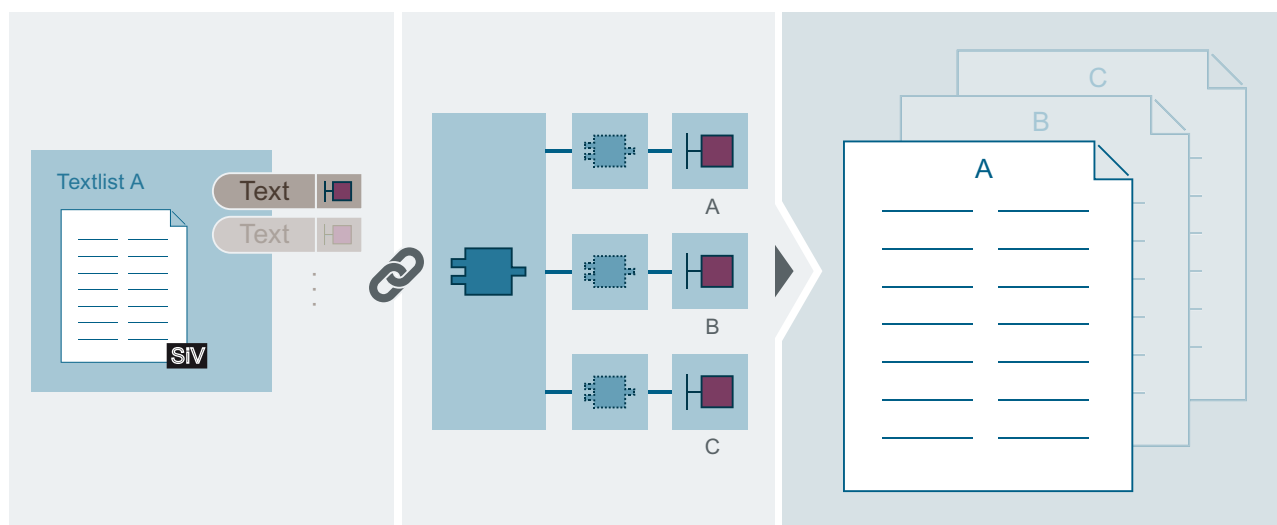
Evaluation of the text list rules

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

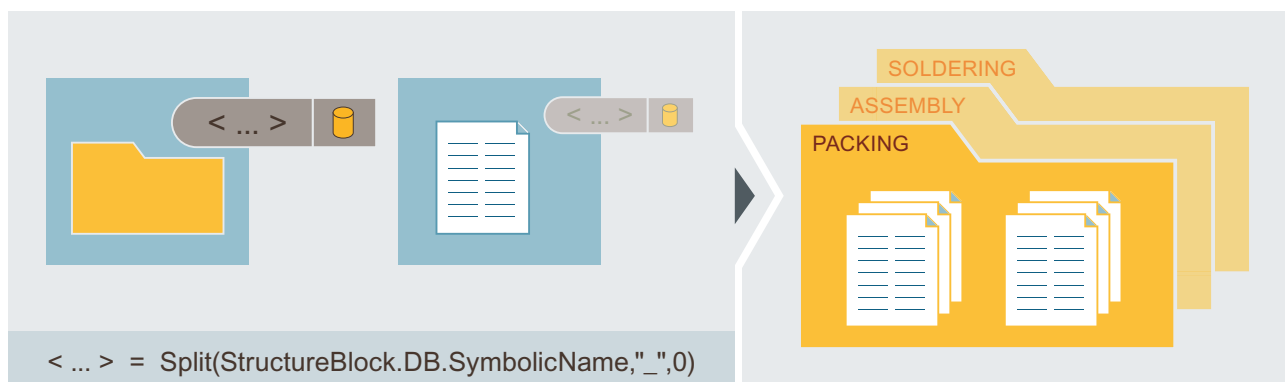
Creating a screen rule (Page 181)

Creating text list rules (Page 183)

Picture legends (Page 266)

6.3.5 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.

For unified devices, acquisition cycle of 250ms is available.

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 "Principle of the tag generation (Page 175)".

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.

Tag generation scenarios

PLC tags are used on HMI device through the "Accessible from HMI" checkbox in the PLC tags window. While creating screens, the input and output fields of a screen item are configured with the PLC tags, and the same is used in HMI generation. For more information on tags, see Principle of tag generation.

Case 1: When All HMI tags are selected

In the "PLC tags" window, after configuring the PLC tags, if the "Accessible from HMI" checkbox is selected, and generation is triggered with the Tag generation Mode set to "All HMI tags", system generates the HMI tags under the folder "HMI tags". The generated HMI tags are generated under default tag table with the tag name as "Block name"_"input/output tag name".

Case 2: When Used HMI tags are selected

Add an input/output device, and configure the input/output field with dynamic tag name in the SiVArc screen editor. PLC Tags are configured for a PLC and Dynamic tags are configured for screens or screen Items. Dynamic tag name refers to any expression that resolves to a PLC name suffixed with the PLC tag value. If the "Accessible from HMI" checkbox is selected in the PLC tags window, and SiVArc generation is triggered with the "Tag generation" mode set to "Used HMI tags", only those PLC tags that are used in the HMI device are generated under "HMI tags" >"Default tag table".

Case 3: With multiple PLCs, and Tag rule

When multiple PLCs are connected, configure tag rules with expressions "S7.Control.Name" and "HmiTag.SymbolicName" and "Accessible from HMI" checkbox selected; configure the SiVArc screens, and trigger SiVArc generation with the "Tag generation mode" set to "All HMI Tags" or

"Used HMI Tags"; the HMI tags are generated with naming as "PLC Name" _ "PLC Tag name". You can find the PLC tags generated and available under "Generation Overview". The HMI Tags gets generated under respective Tag Folder and Tag Table based on the the Expressions configured in the Tag rules. "Generation overview" supports the display of PLC tags along with block tags. For more details on Generation, see Generate visualization. (Page 198)

See also

Example: Adapting tag names (Page 190)

Creating tag rules (Page 180)

6.3.6 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
- Alarms
- Templates
- Screens
- Scripts
- Graphic lists
- Graphics
- Graphics table
- Scheduled tasks

Copy rule supports 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.

Using scheduled tasks as a library object, you can create rules through copy rules.

Using copy rules with library objects:

- You can perform SiVArc generation With scheduled task, which will result in generating objects as scheduled tasks, and is extended to WinCC Unified devices. Multiple copy rules with same library object will be treated as a single rule.
- If scheduled tasks generated by SiVArc and user created scheduled tasks are having identical names, SiVArc automatically renames the user created scheduled task into "Task <<incremental number>>_Renamed".
 - Professional devices support both scheduled tasks with Type "Function list" and "Print job" only
- Creating scheduled task with type as "Print job" in WinCC Runtime device, and during generation if you choose to generate object in Advanced device, SiVArc displays an error.
- As a folder combining scheduled tasks, screen, and tag table, you can perform SiVArc generation that will result in generating objects as scheduled tasks, screen, and tag table in the specified HMI device of TIA portal.
- You can choose to select Energy Suite rules from "Select stations and controllers for SiVArc generation" for the above mentioned points.

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.

You can copy graphics which are available in project library and global library to the folderpath "Languages & resources > Project graphics" through copy rules/system copy rules. While

configuring a copy rule/ system copy rule, you can choose to browse graphics as a library objects. You can perform the following using graphics in copy rule/system copy rule:

- In a TIA portal project:.
 - If copy rule/system copy rule is configured with graphics, upon SiVArc generation, the system automatically creates the folder Project graphics within Languages & resources.
 - Copy rules/system copy rules configured with graphics can generate graphics objects even if you do not choose HMI/PLC device in the station selection dialog. The generated graphic objects will be available under folderpath "Languages & resources > Project graphics.
 - Copy rule/system copy rule during second generation renames the user defined graphic object.
 - If copy rule/system copy rule is configured with folder path consisting of SiVArc objects ; upon SiVArc generation, system ignores the non supported objects, and generates the graphics in the Project graphics folder.
 - While configuring copy rule/system copy rule, you can choose to browse for the already generated graphic object. if the generated graphic object is renamed and you perform a second generation, the system generates a new graphics object.

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 156)
- Creating copy rules (Page 186)

6.3.7 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.

	<input checked="" type="checkbox"/>	Name	Index	Tag group hierarchy	Tag table	Condition
1	<input checked="" type="checkbox"/>	Plantsection_1	0	"Plantsection1"	"Plantsection1 "	InStr(HmiTag.SymbolicName, "Plantsection1 ")
2	<input checked="" type="checkbox"/>	Plantsection_2	1	"Plantsection2"	"Plantsection2"	InStr(HmiTag.SymbolicName, "Plantsection2")
3	<input checked="" type="checkbox"/>	Function_Vars	2	"Standard Functionality"	"Standard Functi...	InStr(HmiTag.SymbolicName, "Result")
4	<input checked="" type="checkbox"/>	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":

	<input checked="" type="checkbox"/>	Name	Index	chy	Tag table	Condition	Comment
1	<input checked="" type="checkbox"/>	Plantsection_1	0	...	"Plantsection..."	InStr(HmiTag.SymbolicName, "Plantsection1 ")	
2	<input checked="" type="checkbox"/>	Plantsection_2	1		"Plantsection2"		
3	<input checked="" type="checkbox"/>	Function_Vars	2	ionality"	"Standard Functi...		
4	<input checked="" type="checkbox"/>	Plantsection_3	3		"Plantsection3"		
5		<create new rule>					

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 103)

If conditions (Page 261)

SiVArc expression (Page 106)

6.3.8 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 190)

Generating tags (Page 168)

Creating tag rules (Page 180)

Example: Using SiVArc to generate tags (Page 18)

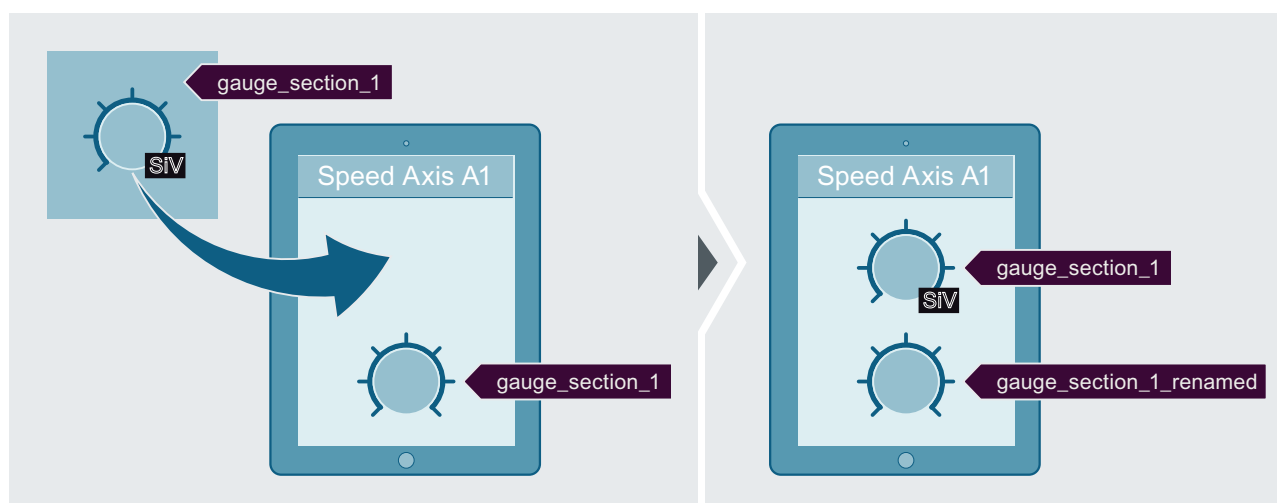
6.3.9 Avoiding conflicts during generation

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.

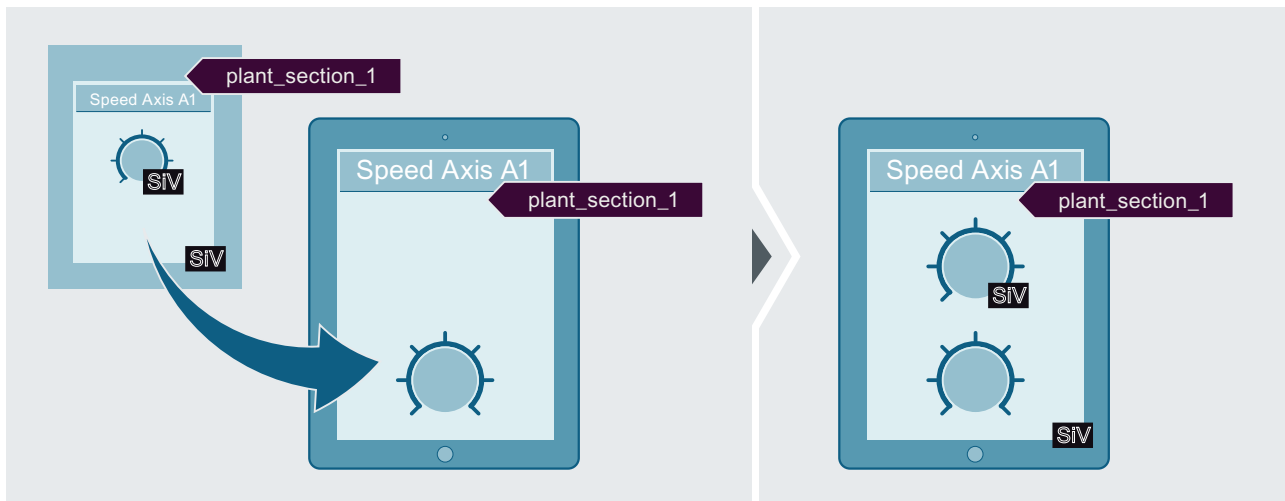
Identical names for screens and text lists

For screens and text lists SiVArc processes identical names differently:

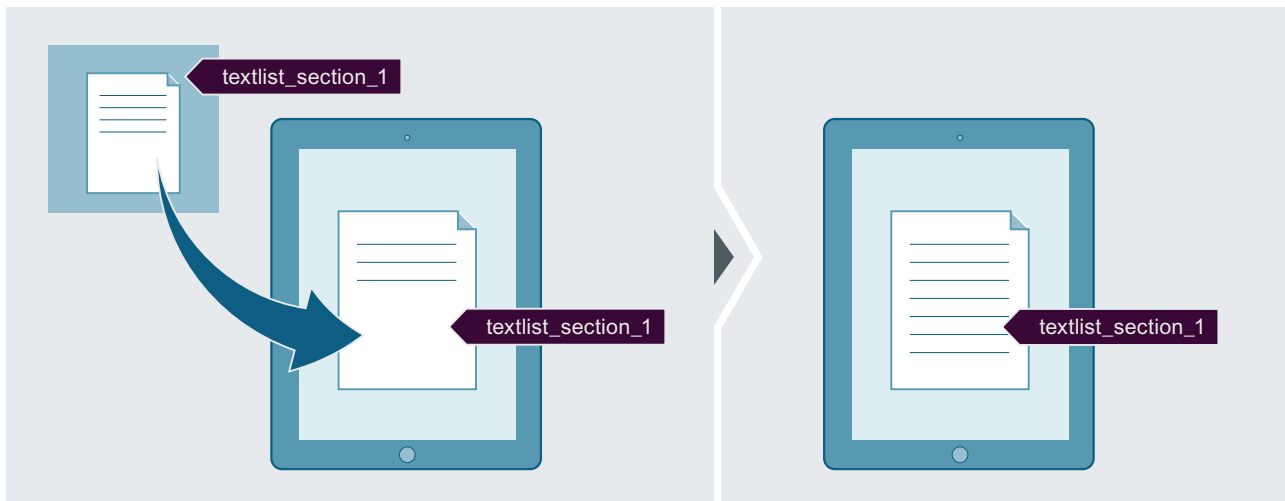
If generated screens or text lists with the same name already exist, SiVArc captures these screens or text lists despite the naming conflict during the generation. Keep in mind that an error message is output for text lists but not for screens.

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

6.3 Creating rules



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 194)

6.3.10 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 168)

Principle of the tag generation (Page 175)

Example: Adapting tag names (Page 190)

Changing SiVArc rules (Page 205)

Storage strategies for generated objects (Page 122)

Editing the view in the SiVArc editors (Page 264)

6.3.11 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.

6.3 Creating rules

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 207)

Supported objects in the user program (Page 103)

Example: Creating screen rule with condition (Page 187)

Example: Organizing screen and text list rules (Page 188)

SiVArc rules (Page 156)

Creating SiVArc rules (Page 160)

Processing of rules (Page 165)

Changing SiVArc rules (Page 205)

Editing the view in the SiVArc editors (Page 264)

6.3.12 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 188)

Example: Creating generation templates for text lists (Page 138)

Example: Create generation template for a text list for block parameters (Page 141)

SiVArc rules (Page 156)

Creating SiVArc rules (Page 160)

Processing of rules (Page 165)

Changing SiVArc rules (Page 205)

Editing the view in the SiVArc editors (Page 264)

Creating alarm rules (Page 184)

6.3.13 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 "Rule Trigger" column, browse for a respective program block or program type.
4. In the "Master copy of Alarms/Classes/Groups" column, browse for single or multiple objects for 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 "Rule Trigger" area. In SiVArc "Plug-in" editor, choose alarm rules, which displays filtered list of alarm rules for a block.
 - Go to reference will be available if you have made single selection under "Master copy of Alarms/Classes/Groups" column.
-


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 198).

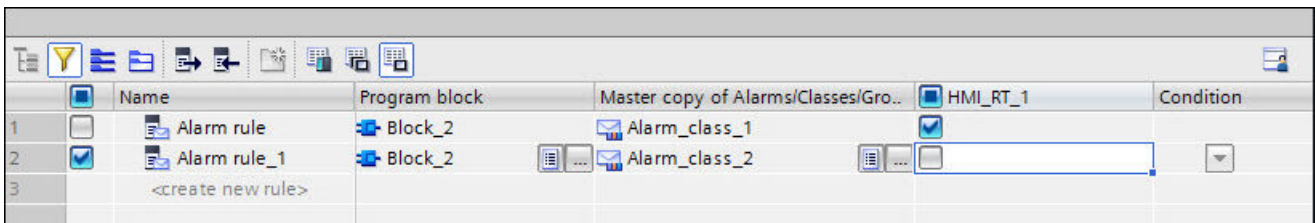
General notes

- 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 or 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.



	Name	Program block	Master copy of Alarms/Classes/Gro..	HMI_RT_1	Condition
1	Alarm rule	Block_2	Alarm_class_1	<input checked="" type="checkbox"/>	
2	Alarm rule_1	Block_2	Alarm_class_2	<input type="checkbox"/>	
3	<create new rule>				

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 198)

6.3.14 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 171)

Changing SiVArc rules (Page 205)

Editing the view in the SiVArc editors (Page 264)

6.3.15 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.

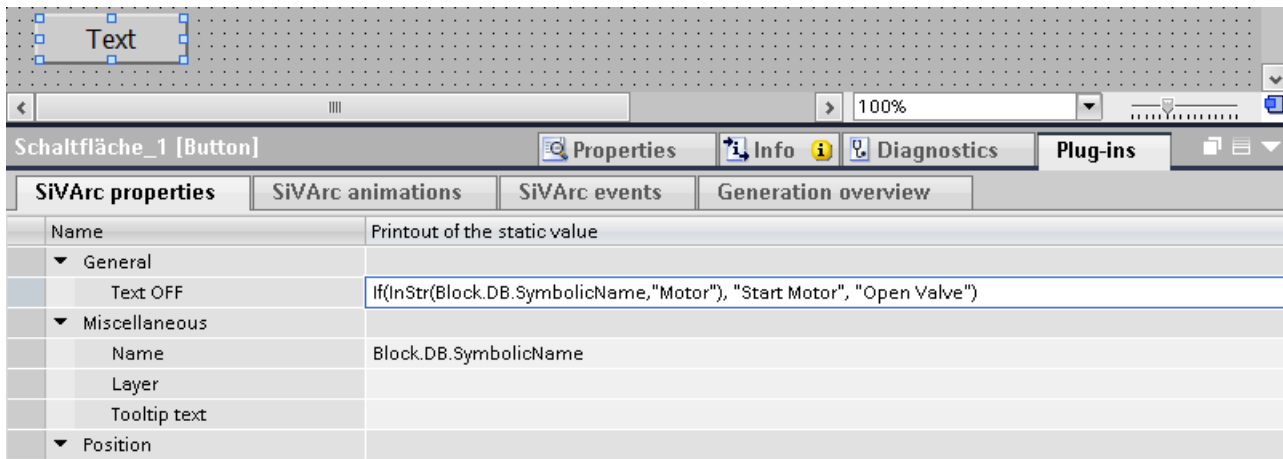
Implementation concept

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>".

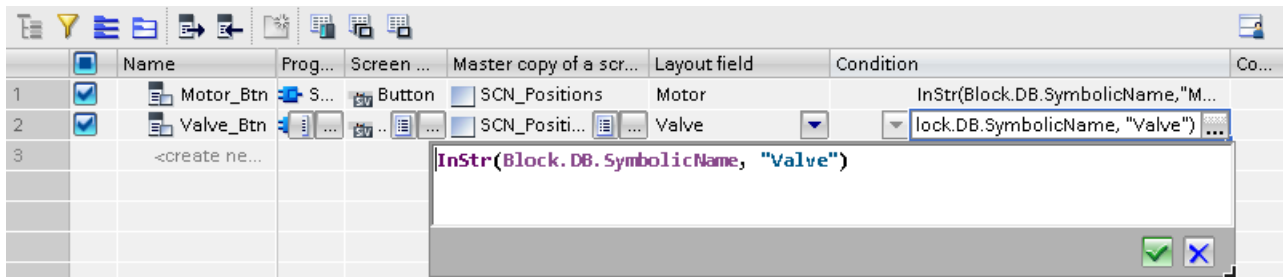
To control the labeling of the button, the configuration engineer programs a condition in the SiVArc property "Text OFF":

6.3 Creating rules



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.

To 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 181)

6.3.16 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 \(Page 181\)](#)

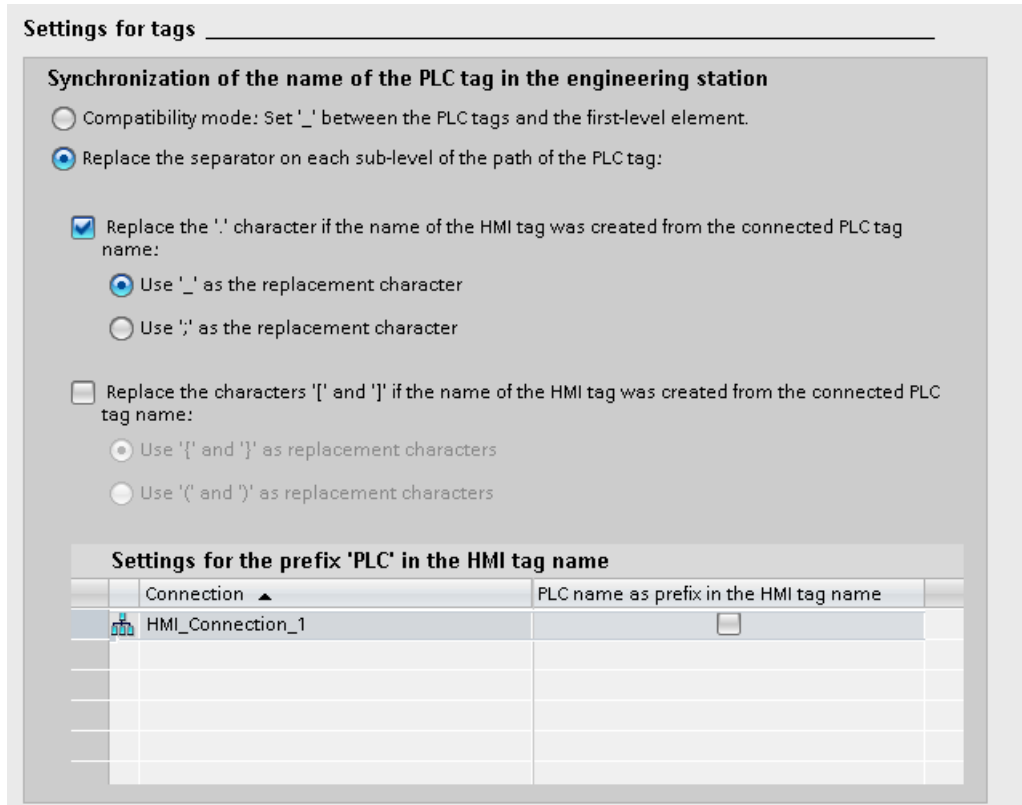
[Creating text list rules \(Page 183\)](#)

6.3.17 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:

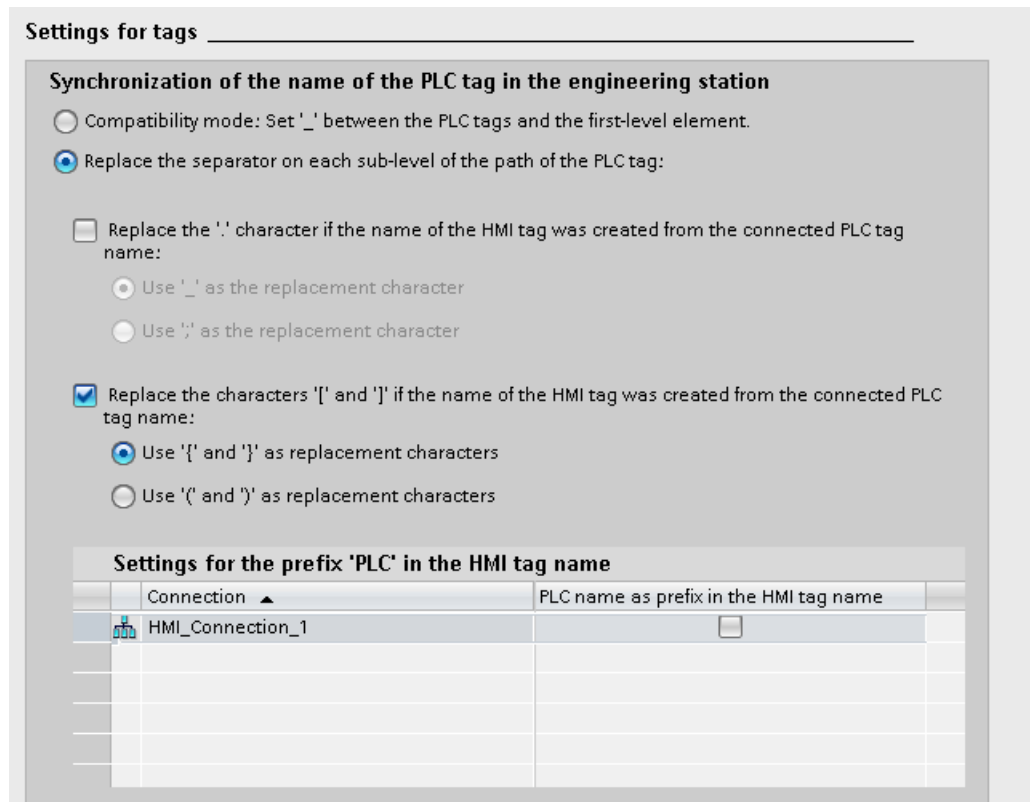
Runtime settings for HMI device 1:



Synchronized HMI tags:

Plantsection1_DB	
Name	
Plantsection1_DB_Activate	
Plantsection1_DB_Productionline_Instance_1_Container_Position	
Plantsection1_DB_Productionline_Instance_1_Dispatchunit_Instance_1_Conveyor_Instance_Speed	
Plantsection1_DB_Productionline_Instance_1_Dispatchunit_Instance_1_Conveyor_Instance_State_A	

Runtime settings for HMI device 2:



Synchronized HMI tags

Plantsection1_DB	
Name	
	Plantsection1_DB.Activate
	Plantsection1_DB.Productionline_Instance_1.Container_Position
	Plantsection1_DB.Productionline_Instance_1.Dispatchunit_Instance_1.Conveyor_Instance.Speed
	Plantsection1_DB.Productionline_Instance_1.Dispatchunit_Instance_1.Conveyor_Instance.State_A

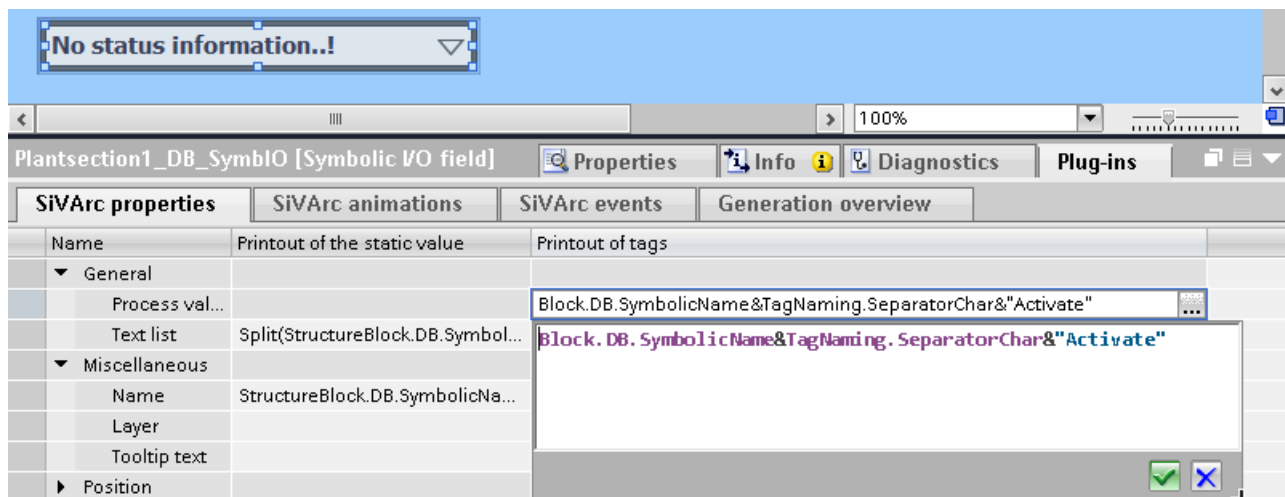
Requirement

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 236)
- Principle of the tag generation (Page 175)
- SiVArC expression (Page 106)

6.3.18 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.

6.3 Creating rules

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 264)

6.3.19 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
- AlarmRules
- 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
The rule/rule group in the existing rule editor gets aligned to the imported file's rule/rule group hierarchy.
- Renaming rules to be imported if rule name already exists
In case of naming conflicts, the imported rules and rule groups are appended with an underscore. For example, "Rule" after renaming becomes "Rule_1".
- Deleting all existing rules prior to the import
After the import, the rule editor deletes the existing rules, and the rules/rule groups from the imported file are displayed.

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".

6.3.20 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.

6.4 Generate visualization

6.4.1 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

If screen rules that uses master copies of Unified devices; consists of Unified and HMI devices when undergoes SiVArc generation, screens are generated only on Unified devices. Unified configured screens are generated on Unified devices only.

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 (Unified) objects based on the SiVArc rules and saves them according to the configuration. The generated objects will have the plug-in properties updated with the configured data.

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 "Visualization (SiVArc)". Choose from the following options:
 - Generate the visualization (SiVArc)
 - Generate with station selection
 - Clear the visualization (SiVArc)

Alternatively, press the shortcut <ALT+Shift+G>.

When you configure faceplates with multi instance DB, during generation the warning messages are not displayed if you enable "Common data > SiVArc Settings > Warning Settings > Hide warnings if screen object already exists". By default, the checkbox "Hide warnings if screen object already exists" is disabled. The warning messages will be available under "Common data > Logs" folder by default.

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 207)

Avoiding conflicts during generation (Page 177)

Subsequent changes (Page 202)

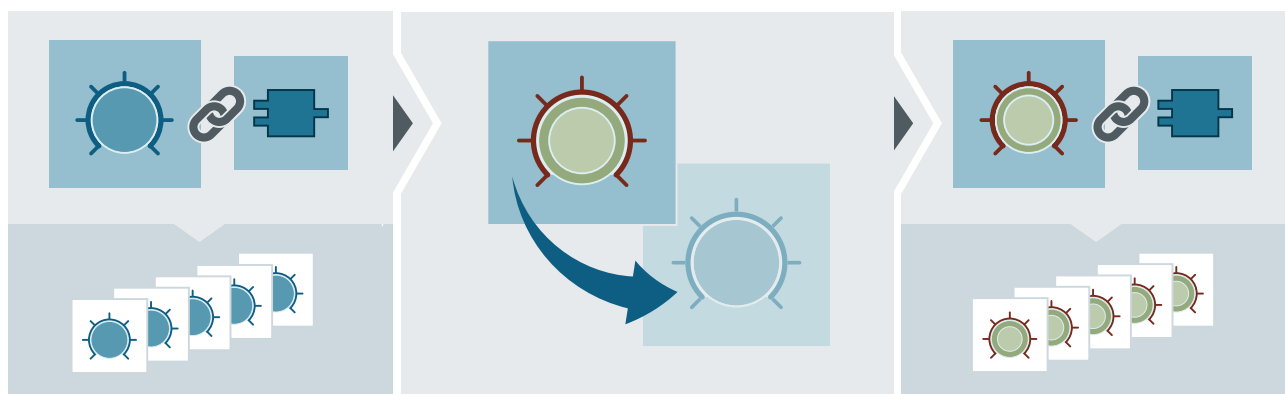
6.4.2 Subsequent changes

6.4.2.1 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.

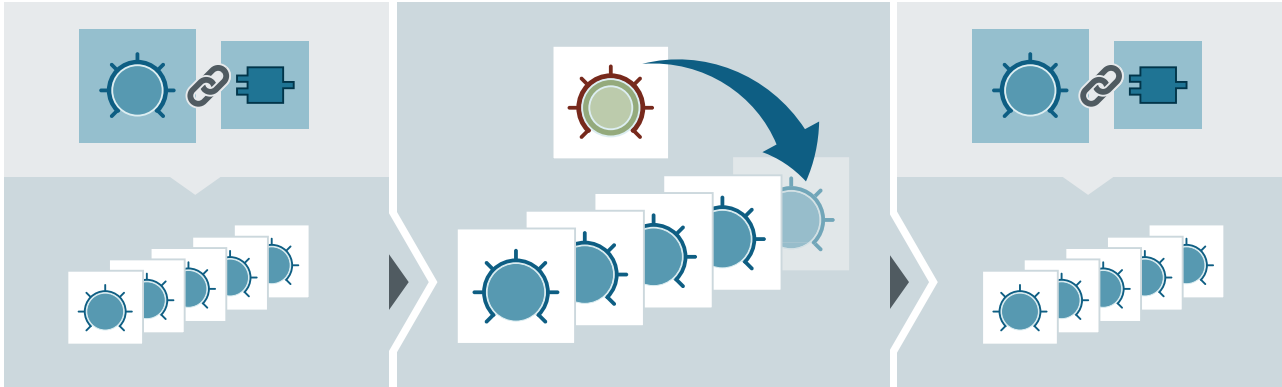
If the display and operating elements are to be adapted graphically at a later time, the visualization engineer merely changes the associated generation template. The configuration engineer stores the template in the library under the same name as the previous one. During the next generation the display 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". "Entry_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 216)
Identifications in the SiVArc project (Page 206)
Picture legends (Page 266)

6.4.2.2 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.

When SiVArc screen rule and copy rule contains similar library object, upon SiVArc generation, copy rule takes precedence and generates the library object along with the folder structure defined within copy rule.

When SiVArc tag rule and copy rule contains similar library object (tag table), upon SiVArc generation, copy rule takes precedence and generates the library object along with the folder structure defined within copy rule. Tag table with PLC tags are generated as defined in the copy rule. A warning message is displayed when tag rule is used for generation.

If copy rule is unavailable, and you choose SiVArc screen rule to be generated first then screen object is generated. Upon adding copy rule for second generation, a warning message is displayed.

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 156)

Creating tag rules (Page 180)

Creating a screen rule (Page 181)

Creating text list rules (Page 183)

Creating copy rules (Page 186)





6.4.2.3 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".

6.4.3 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.

6.4 Generate visualization

- Text list rules are defined.
- All used instances of types are updated to the latest version.

Note

Changes to the controller require a compilation

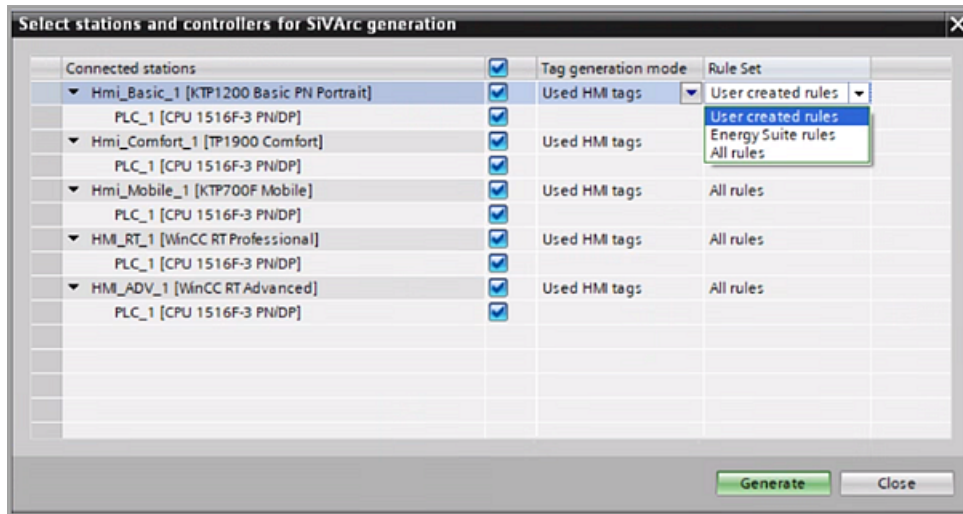
Changes in the user program or in the hardware configuration must be compiled before you generate the visualization.

Generation without station selection

1. Click "Visualization (SiVArc) >" in the shortcut menu of Runtime or the HMI device in the project tree. Choose from the options:
 - Generate the visualization (SiVArc)
 - Generate with station selection
 - Clear the visualization (SiVArc)

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 "Generate the visualization (SiVArc) > 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".

Clear the generation data

You can choose to clear the generation data for a device by performing the following:

1. Right-click on an "HMI device >Visualization (SiVArc)>Clear the Visualization (SiVArc)".
2. Choose the rule set from the station selection pop-up for a selected HMI device:
 - User created rules - system will delete all occurrences of SiVArc generated objects that were generated using user created rules only
 - Energy Suite rules - system will delete SiVArc objects that were generated using Energy Suite rules only
 - All rules - system will delete all occurrences of SiVArc objects that were generated using the above mentioned

Note

- HMI tags for a chosen HMI device would have been generated from user created rules or Energy suite rules. When you perform the "**Clear the Visualization (SiVArc)**" task, irrespective of the mode of generation, all the HMI tags get deleted.
 - During cleaning up of the generation data, SiVArc license and Energy Suite licenses are validated.
-

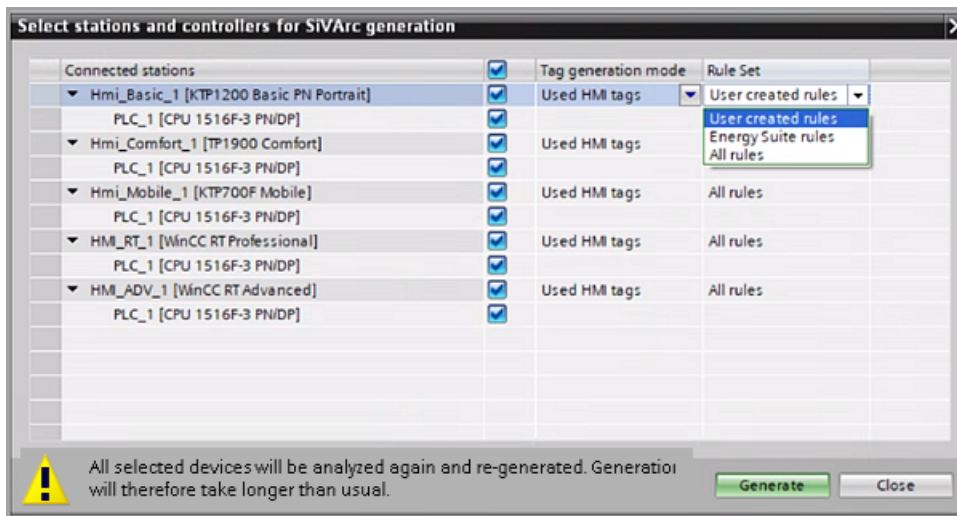
3. The SiVArc clean up dialog box prompts you with an alert message stating the data from "Generation overview" and "Generation matrix" would be deleted.
4. At any point of time during the cleaning up of generation data, you may choose to cancel the process by clicking the "Cancel" button available in the clean up progress dialog box.
5. Upon clicking "Cancel" button, no data will be deleted and transaction will be rolled back/ aborted.

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.

6.5 Checking result

6.5.1 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.

The generation overview is also available at several places in the SiVArc project:

The screenshot shows the 'Generation overview' window in SiVArc. The window title is 'GettingStartedSiVArcV2.0_Complete_V14 > Common data > SiVArc > Generation overview'. The window has several tabs: 'Screens / screen objects', 'Template screens/Screen items', 'Popup screens/Screen items', 'Tags', 'Text lists', and 'Alarms'. The 'Screens / screen objects' tab is active, showing a table with the following columns: Screen, Screen object, Master copy / type, HMI device, PLC device, Program block, Screen rule, and Generated by mo... The table contains 13 rows of data, with the first 11 rows grouped under 'Plantsection1' and the last two rows under 'Plantsection2' and 'Plantsection3'.

Screen	Screen object	Master copy / type	HMI device	PLC device	Program block	Screen rule	Generated by mo...
1	Plantsection1	Plantscreen	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	<input type="checkbox"/>
2	Plantsection1	Plantsection_Title	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	<input type="checkbox"/>
3	Plantsection1	PlantStatus_Symb_IO	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_St...	<input type="checkbox"/>
4	Plantsection1	Function_Activate	HMI_RT_1	PLC_1	Activate, Activate_DB	Activate_Btn	<input type="checkbox"/>
5	Plantsection1	Productionline_title	HMI_RT_1	PLC_1	Productionline, #Produc...	Productionline_...	<input type="checkbox"/>
6	Plantsection1	Position_IO	HMI_RT_1	PLC_1	Productionline, #Produc...	Productionline_...	<input type="checkbox"/>
7	Plantsection1	Conveyor	HMI_RT_1	PLC_1	Conveyor	Conveyor	<input type="checkbox"/>
8	Plantsection1	ProcessingUnit	HMI_RT_1	PLC_1	Processing	Processing_Unit	<input type="checkbox"/>
9	Plantsection1	Conveyor	HMI_RT_1	PLC_1	Conveyor	Conveyor	<input type="checkbox"/>
10	Plantsection1	ProcessingUnit	HMI_RT_1	PLC_1	Processing	Processing_Unit	<input type="checkbox"/>
11	Plantsection1	Function_Stop	HMI_RT_1	PLC_1	Stop, Stop_DB	Stop_Btn	<input type="checkbox"/>
12	Plantsection2	Plantscreen	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	<input type="checkbox"/>
13	Plantsection3	Plantscreen	HMI_RT_1	PLC_1	Plantsection, Plantsecti...	Plantsection_Tit...	<input type="checkbox"/>

6.5 Checking result

- 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 213)

Example: Using the generation matrix (Page 216)

Editing and managing SiVArc rules (Page 192)

Editing the view in the SiVArc editors (Page 264)

6.5.2 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.

The screenshot shows the 'Generation matrix' editor window. The title bar reads 'GettingStartedSiVArcV2.0_Complete_V14 > Common data > SiVArc > Generation matrix'. The 'Target device' is set to 'HMI_RT_1 [WinCC RT Advanced] (1)'. The main table displays the following data:

Call structure	Screen rule	Name of the screen object	HMI devices	Plantsection1	Plantsection2	Plantsection3
1 PLC_1						
2 Main						
3 Plantsection, Plan...	Plantsection_Status_SymbIO Plantsection_Title	Plantsection1_DB_SymbIO Plantsection1_DB	For all	X		
4 Plantsection, Plan...	Plantsection_Status_SymbIO Plantsection_Title	Plantsection2_DB_SymbIO Plantsection2_DB	For all		X	
5 Plantsection, Plan...	Plantsection_Status_SymbIO Plantsection_Title	Plantsection3_DB_SymbIO Plantsection3_DB	For all			X
6						
7						
8						

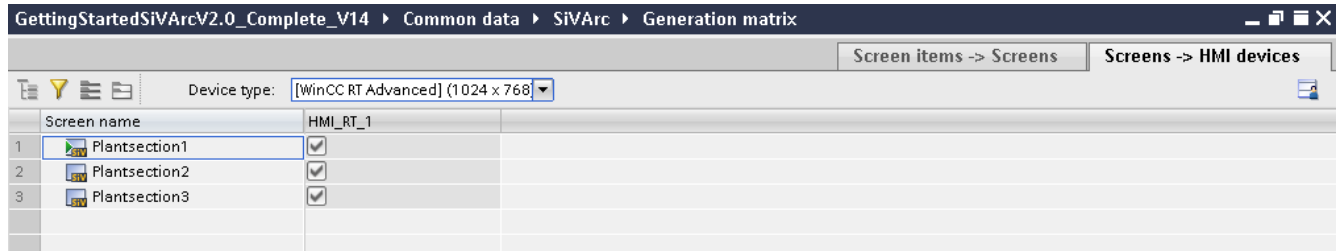
Below the main table, there is a smaller table with the following data:

Screen name	Layout field
1 Plantsection1	
2 Plantsection2	
3 Plantsection3	X

"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.

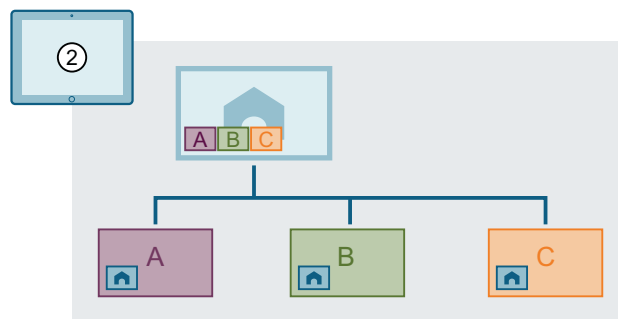
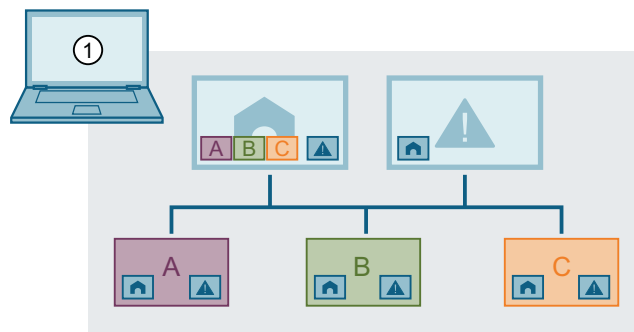
The screens and navigation buttons leading to this screen are generated again.

Configure the property "Number of overflow screens" of a screen and move the property to library types. Using the screen type as a "Master copy/Type of a screen", the navigation objects are generation accordingly.

Example: Moving screens with navigation to other devices with the generation matrix

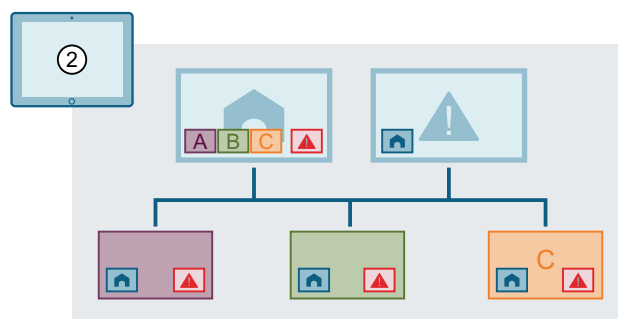
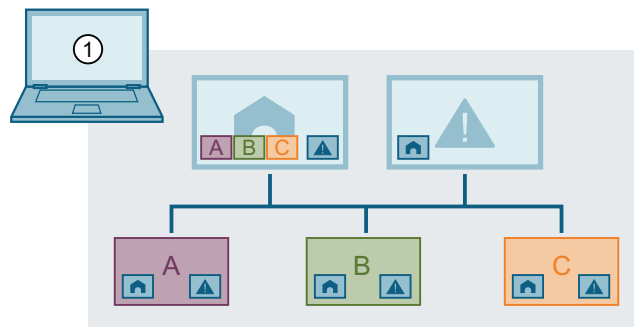
You have generated a start screen, a diagnostic screen and lower-level screens on HMI device 1. The start screen and the diagnostic screen can be displayed from each lower-level screen with the help of navigation buttons.

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 216)

Editing the view in the SiVArc editors (Page 264)

6.5.3 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 213)

SiVArc Openness

7.1 Introduction

Introduction

TIA portal openness application allows you to instantiate SiVArc. You must need a client application to access TIA portal, and through openness feature launch SiVArc services. For more details on set up, and accessing Openness, see TIA portal user guide.

Setting up the application

To set up a client application, perform the following steps:

1. Create a console application. Add reference of Public API (Siemens.Engineering.dll) available at _deployed\TIAPV15SP1_11010001\PublicAPI\15.1\936 Siemens.Engineeringin.dll or from installed binaries location PublicAPI\15.1\937 Siemens.Engineeringin.dll
2. Add configuration details to the configuration file. For detailed information of configuration details and steps to access the public API, see TIA openness wiki.
3. To access Sivarc service, use the below mentioned API:

```
using (TiaPortal tia = new
TiaPortal (TiaPortaMode.WithUserInterface))
{
    Project myProject = tia.Projects.Open(new FileInfo(@"C:\Users
\z003exve\Documents\Automation\Project_Demo\Project_Demo.ap15));
    //if SiVArc is not installed, user will not be able to access
SiVArc service (compiler error)
    Sivarc sivarc =myproject?.GetService<Sivarc>():
    if (sivarc !=null)
    {
    }
}
```

7.2 SiVArc service properties

SiVArc service properties

The table below lists the supported properties and methods for SiVArc:

Property Name	Description	Data Type
AlarmRules	Anchor object for all alarm rule objects	AlarmRulesBrowsable
ScreenRules	Anchor object for all screen rule objects	ScreenRulesBrowsable

Property Name	Description	Data Type
TextlistRules	Anchor object for all textlist rule objects	TextlistRulesBrowsable
TagRules	Anchor object for all tag rule objects	TagRulesBrowsable
CopyRules	Anchor object for all copy rule objects	CopyRulesBrowsable
Alarm Rules	Enumerate of all immediate first level alarm rules	AlarmRuleComposition
Groups	Enumerate of all immediate first level alarm rule groups	AlarmRuleGroupComposition
ScreenRules	Enumerate of all immediate first level screen rules	ScreenRuleComposition
ScreenRulesGroups	Enumerate of all immediate first level screen rule groups	ScreenRuleGroupComposition
TextlistRules	Enumerate of all immediate first level textlist rules	TextlistRuleComposition
TextlistGroups	Enumerate of all immediate first level textlist rule groups	TextlistRuleGroupComposition
TagRules	Enumerate of all immediate first level tag rules	TagRuleComposition
TagRulesGroups	Enumerate of all immediate first level tag rule groups	TagRuleGroupComposition
CopyRules	Enumerate of all immediate first level copy rules	CopyRuleComposition
CopyRulesGroups	Enumerate of all immediate first level copy rule groups	CopyRuleGroupComposition

Following table lists the AlarmRule and AlarmRuleGroup composition. The same applies to other SiVArc objects.

Method Name	Parameter	Description	Data Type
Find	String- alarm rule /rule-group name	Finds the alarm rule/ alarm rule group from alarm rule/alarm group collection	AlarmRule
CreateFrom	MasterCopy – alarm rule/alarm rule group master copy	Copy alarm rule/alarm rule group master copy from library to project with default replace option	AlarmRule
CreateFrom	MasterCopy – alarm rule/alarm rule group master copy, CreateOptions- Rename/Replace	Copy alarm rule /alarm rule group master copy from library to project with create option	AlarmRule

7.3 Copying rules or groups from library

Requirement

- Launch TIA portal openness application. For more information on connections, see TIA portal user guide.
- An existing TIA portal project containing screen rule editor, screen rules group and master copy.

Case 1: When you copy rules/rule groups from master copy to screen rules editor

"The `CreateFrom`" global library allows rules and rule groups to be copied from the global library to the SiV Arc rule editor. If successful, the API function will return `ScreenRule/ScreenRuleGroup`. The following code explains how the rules or rule groups are copied from the master copy to the SiV Arc editor:

```
// Finds screen rule master copy "Screen rule_1"
MasterCopy screenRuleMasterCopy =
    myProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");

if (screenRuleMasterCopy != null)
{
    var rule = sivarcs.ScreenRules.Rules.CreateFrom(screenRuleMasterCopy);
    if (rule != null)
    {
        Console.WriteLine("Copied Screen Rule Name: " + rule.Name);
        Console.WriteLine("Copied Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Copied Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Copied Screen Rule Condition: " + rule.Condition);
    }
}
```

By default, the behaviour will be replaced.

Case 2: When you copy rules/rule groups from master copy, the existing rules/rule groups can be renamed or replaced based on the second parameter input

If rules/rule groups in the master copy are already existing in the SiV Arc editor, and if you are trying to copy them, the rules/rule groups gets renamed. The "CreateOptions" API will create the rules/rule groups in the SiV Arc editor if they are not existing, else they replace the existing rules/rule groups. If successful, the API function will rename `ScreenRule/ScreenRuleGroup`. The following code snippet shows the replacing of rules/rule groups:

```

// Finds screen rule master copy "Screen rule_1"
MasterCopy screenRuleMasterCopy =
    myProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");

if (screenRuleMasterCopy != null)
{
    var rule = sivarC.ScreenRules.Rules.CreateFrom(screenRuleMasterCopy, CreateOptions.Rename);
    if (rule != null)
    {
        Console.WriteLine("Copied Screen Rule Name: " + rule.Name);
        Console.WriteLine("Copied Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Copied Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Copied Screen Rule Condition: " + rule.Condition);
    }
}

```

7.4 Finding a screen rule and screen rule group

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA portal project consisting of screen rules and screen rule groups.

Finding screen rules within screen groups

Case 1: Finding screen rules within screen rules editor.

The API `sivarC.ScreenRules.Rules` allows you to find the available rules within the SiVArc screen rule editor as shown below:

```

// Collection of all immediate first level screen rules
ScreenRuleComposition screenRules = sivarC.ScreenRules.Rules;
if(screenRules != null && screenRules.Count > 0)
{
    // Finds screen rule
    ScreenRule rule = screenRules.Find("Screen rule_7");
    if(rule != null)
    {
        Console.WriteLine("Screen Rule Name: " + rule.Name);
        Console.WriteLine("Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Screen Rule Condition: " + rule.Condition);
    }
}

```

Case 2: Finding screen rule groups within screen rules editor.

The API `sivarc.ScreenRule.Groups` allows you to find the available rules and rule groups within the SiVArc screen rule editor as shown below:

```
var groups = sivarc.ScreenRules.Groups;
if (groups != null && groups.Count > 0)
{
    var rule = groups.Find("Screen rule group").Rules.Find("Screen rule_2");
    if (rule != null)
    {
        Console.WriteLine("Found Screen Rule Name: " + rule.Name);
        Console.WriteLine("Found Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Found Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Found Screen Rule Condition: " + rule.Condition);
    }

    var group = groups.Find("Screen rule group").Groups.Find("Screen rule group_2");
    if (group != null)
    {
        Console.WriteLine("Found Screen Rule Group Name: " + group.Name);
        Console.WriteLine("Found Screen Rule Group Comment: " + group.Comment);
        Console.WriteLine("Found Screen Rule Group Enabled: " + group.Enabled);
        Console.WriteLine("Found Screen Rule Group Condition: " + group.Condition);
    }
}
}
```

7.5 Deleting rules and rule groups

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA project containing screen rules and screen rule groups.

Deleting rules and rule groups

To delete a rule or a rule group, the following API is used:

```
sivarc.ScreenRules.Rules.Find("Screen rule_1").Delete();
```

The `ScreenRules` is an anchor object for all screenrule objects. To perform deletion, it is mandatory that you find the rule within the rule editor. For more information on finding the screen rule, refer section Finding a screen rule and screen rule group.

To delete screens from screen group, use the following API:

```
sivarc.ScreenRules.Rules.Find("Screen rule group_1").Delete();
```

7.6 UMAC set up for openness

About UMAC

To access UMAC through openness, ensure that you have the UMAC credentials and access privileges. If you are not a valid UMAC user, the application will return `NULL` values for all SiVArc anchor rule objects. For more information on UMAC, refer UMAC topic in SiVArc user guide.

7.7 SiVArc generation

Requirement

- Launch TIA portal openness application. For more information on connections, see TIA portal user guide.
- An existing TIA portal project connected to an HMI device, and PLC configured.

Important points to notice

- Ensure SiVArc license is installed on your PC, else during generation an exception is thrown - *"SiVArc license is missing; it is mandatory to have SiVArc license for modifying data"*.
- Ensure valid device name is used, else an exception is thrown - *"HMI device 'deviceName' not found"*.
- Ensure valid PLC name is called, else an exception is thrown - *"PLC device 'plcDeviceName' not found"*.
- Ensure supported device name is called, else an exception is thrown - *"HMI device 'deviceName' is not supported"*
- Ensure supported PLC name is called, else an exception is thrown - *"PLC device 'plcDeviceName' is not supported"*
- Ensure you pass valid GenerationOption parameter. If none is passed, SiVArc generation will happen and by default the TIAP project settings will be used for SiVArc generation
- Ensure you valid PLC name which was not used for generation in the previous generation, else the system freezes.

GenerationOptions- Enum (Flag)

SiVArc supports Enum options, and you can pass combination of two values in the Generate API. Following table displays the enum options:

SN	Values	Description
1	None	Nothing is selected, takes default setting for generation

2	AllTags	Generate all tags
3	UsedHmi- Tags	Generate only relevant (used) tags
4	FullGenera- tion	In case when FullGeneration option is not selected, SiVArc will decide internally based on the configuration if it has to perform full generation or delta generation. When you pass FullGeneration as parameter, SiVArc will be generated with force full generation.

To generate SiVArc, use the following API:

```
sivarc.Generate("HMI_1", new List<string> {PLC_1},
GenerateOptions.AllTags | GenerateOptions.FullGeneration);
```

SiVArcGenerationResult and SivarcFeedbackMessage

SiVArc generation accesses the following properties on successful generation:

- IsGenerationSuccessful - Informs if SiVArc generation is successful.
- WarningCount - Total warning count after SiVArc generation
- ErrorCount - Total error count after SiVArc generation
- Messages - Composition of feedback message

To generate SiVArc result, use the following API:

► Print Sivarc generated feedback messages

```
private void WriteSivarcGenerationResults(SivarcGenerationResult result)
{
    sb.Append("Is SiVArc generation successful:" +
result.IsGenerationSuccessful);
    sb.Append(Environment.NewLine);
    sb.Append("Total Warning Count:" + result.WarningCount);
    sb.Append(Environment.NewLine);
    sb.Append("Total Error Count:" + result.ErrorCount);
    sb.Append(Environment.NewLine);

    RecursivelyWriteMessages(result.Messages);
}
```

SiVArc generation accesses the following feedback messages on successful generation:

- Path: Header text of feedback message(Header messages will always have empty description field)
- DateTime: DateTime of feedback message

- MessageType: Feedback message type
- Description: Description/Content of Feedback message (Only when Path is empty to ensure not a header message)
- WarningCount: Warning count for a header message
- ErrorCount: Error count for a header message
- Messages: Composition of feedback message (SivarcFeedbackMessage)

You can view recursive feedback messages by using the following code snippet:

```
private void RecursivelyWriteMessages(SivarcFeedbackMessageComposition messages)
{
    foreach (SivarcFeedbackMessage message in messages)
    {
        sb.Append("Path: " + message.Path);
        sb.Append(Environment.NewLine);
        sb.Append("DateTime: " + message.DateTime);
        sb.Append(Environment.NewLine);
        sb.Append("State: " + message.MessageType);
        sb.Append(Environment.NewLine);
        sb.Append("Description: " + message.Description);
        sb.Append(Environment.NewLine);
        sb.Append("Warning Count: " + message.WarningCount);
        sb.Append(Environment.NewLine);
        sb.Append("Error Count: " + message.ErrorCount);
        sb.Append(Environment.NewLine);
        sb.Append(Environment.NewLine);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```


Reference

8.1 SiVArc objects

8.1.1 PLC Tags

Description

PLC tags represents the tags supported by the PLC.

Use

You can use the PLC Tag object to store generated external PLC supported tags in the project tree in structured form.

Use the "PLC Tag" object as follows:

- "SymbolicName" object property
`PLCTag.DB.SymbolicNam`
Access the user-defined name of the data block.

8.1.2 I/O device

Description

The I-device is linked as an IO device to a "higher-level" IO controller.

Use

You can use the IODevice object to access an IO device in the project.

Use the "Device.name" object as follows:

- "Name" object property
`IODevice.Name`
Access the user-defined name of a IO device, e.g. HMI_1.

8.1.3 Software units

Description

You can configure expressions for various properties of HMI objects, which resolves to the name of the software unit containing the PLC block used in SiVArc rule creation.

Use

8.1 SiVArc objects

You can use the softwareunit object to access the software unit device in the project.

Use the "Device.name" object as follows:

- "Name" object property
SoftwareUnit.Name
Access the user-defined name of a software unit.

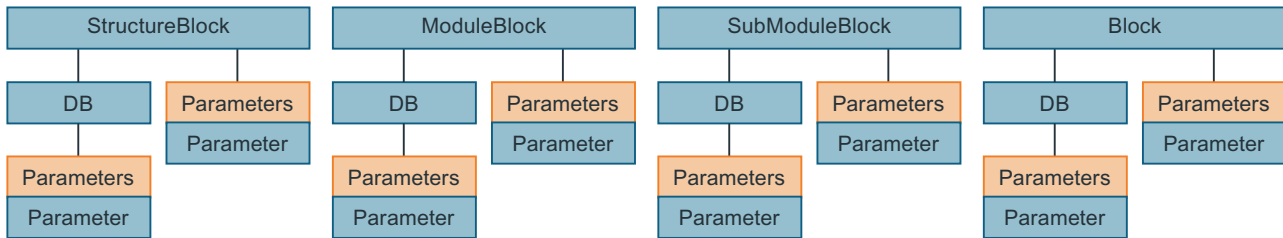
8.1.4 Object hierarchy

Introduction

You can use SiVArc expressions to directly address data from different areas of the TIA Portal.

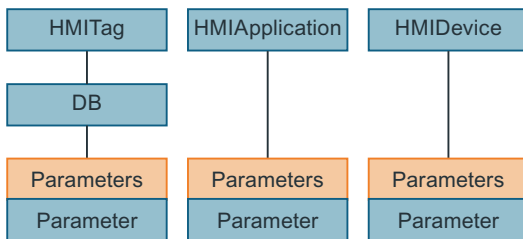
Program call in STEP 7

You can use keywords to access the blocks in the user program, the associated data blocks and their parameters.



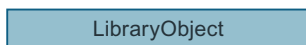
WinCC data

You can use the following key words to access external tags, devices and applications of the visualization.



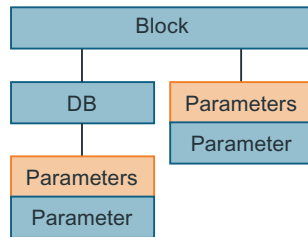
Library data

You can use the keyword LibraryObject to access the storage location of a generation template in the library.



8.1.5 Block (Panels, Comfort Panels, RT Advanced, RT Professional)

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.

8.1 SiVArc objects

- "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.

8.1.6 DB (Panels, Comfort Panels, RT Advanced, RT Professional)

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 120)

8.1.7 HMIApplication (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

HMIApplication

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.

8.1.8 HMIDevice (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

HMIDevice

Represents the HMI device in the project.

Use

You can use the HMIDevice object to access an HMI device in the project.

8.1 SiVArc objects

Use the "HMIDevice" object as follows:

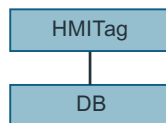
- "Name" object property
`HMIDevice.Name`
Accesses the user-defined name of an HMI device, e.g. HMI_1.
- "Type" object property
`HMIDevice.Type`
Accesses the type of HMI device, e.g. KTP400.

Note

If your HMI device is a panel, the HMIDevice and HMIApplication objects are the same.

8.1.9 HMITag (Panels, Comfort Panels, RT Advanced, RT Professional)

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 122)

8.1.10 LibraryObject (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

LibraryObject

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 use a multi-level storage hierarchy, you can use the expression `LibraryObject.FolderPath` as substitute for the backslash.

- "Name" object property

`LibraryObject.Name`

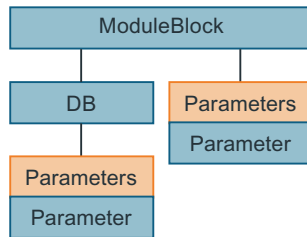
References the name of the screen type of the library. If you use the SiVArc property "Screen group" in the SiVArc expression, the screen is stored in a folder with the name of the screen type in the project tree. If you use the SiVArc expression in the SiVArc property "Name", the screen is named after the screen type.

See also

Storage strategies for generated objects (Page 122)

8.1.11 ModuleBlock (Panels, Comfort Panels, RT Advanced, RT Professional)

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.

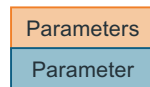
Use the "ModuleBlock" object as follows:

- "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.

8.1.12 Parameters (Panels, Comfort Panels, RT Advanced, RT Professional)

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.

8.1.13 S7Control (Panels, Comfort Panels, RT Advanced, RT Professional)

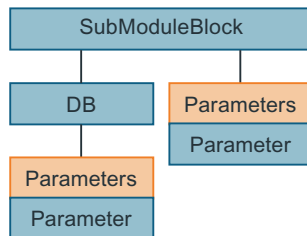
Description

Represents the PLC in the project.

Use

You use the S7Control object to access the name of a PLC:

- "Name" object property
`S7Control.Name`

8.1.14 SubModuleBlock (Panels, Comfort Panels, RT Advanced, RT Professional)**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.

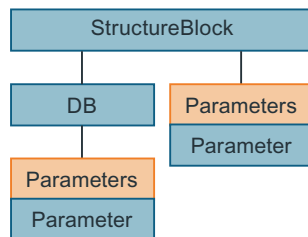
Use the "SubModuleBlock" object as follows:

- "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.

8.1.15 StructureBlock (Panels, Comfort Panels, RT Advanced, RT Professional)

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.

8.1 SiVArc objects

- "NetworkTitle" object property
`SubModuleBlock.NetworkTitle`
Accesses the title of the network in which the block is instantiated.
- "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.

8.1.16 TagNaming (Panels, Comfort Panels, RT Advanced, RT Professional)

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 = "."	Db1_a[1].b.c[3]	Compatibility mode
TagNaming.IndexStartChar = "["	Plc1.Db1.a[1].b.c[3]	PLC prefix
TagNaming.IndexEndChar = "]"	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 = ";"	Db1;a(1);b;c(3)	Replace the period and bracket with ; ()
TagNaming.IndexStartChar = "("		
TagNaming.IndexEndChar = ")"		
TagNaming.SeparatorChar = "_"	Plc1_Db1_a{10}_b_c{3}	Replace the period and bracket with _ { }
TagNaming.IndexStartChar = "{"		PLC prefix
TagNaming.IndexEndChar = "}"		

8.2 SiVArc object properties

8.2.1 Assigned

Description

Returns TRUE if there is an assignment at the specified block parameter.

Syntax

```
<Object>.Assigned
```

Object

- Parameter

8.2.2 Comment

Description

Returns the entered comments.

8.2 SiVArc object properties

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 120)

8.2.3 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 122)

8.2.4 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

8.2.5 IndexEndChar

Description

Returns the closing bracket set in the Runtime settings when structuring external tags.

Syntax

<Object>.IndexEndChar

Object

- TagNaming

8.2.6 IndexStartChar

Description

Returns the opening bracket set in the Runtime settings when structuring external tags.

Syntax

<Object>.IndexStartChar

Object

- TagNaming

8.2.7 InitialValue

Description

Returns the default value of a parameter.

Syntax

<Object>.InitialValue

Object

- Parameter

8.2.8 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 122)

8.2.9 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 120)

8.2.10 NetworkTitle

Description

Returns the network title.

Syntax

```
<Object>.NetworkTitle
```

8.2 SiVArc object properties

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 120)

8.2.11 Number

Description

Returns the block number.

Syntax

`<Object>.Number`

Object

- StructureBlock
- ModuleBlock
- SubModuleBlock
- Block
- DB

8.2.12 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

8.2.13 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 120)

8.2.14 SymbolicName**Description**

Returns the user-defined name of a block or tag.

Syntax

<Object>.SymbolicName

8.2 SiVArc object properties

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.

8.2.15 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 120)

8.2.16 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".

8.2.17 Value

Description

Returns the value.

Syntax

```
<Object>.Value
```

Object

- Parameter

8.2.18 Version

Description

Returns the version of a block type.

8.3 SiVArc object properties

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.

8.3 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:

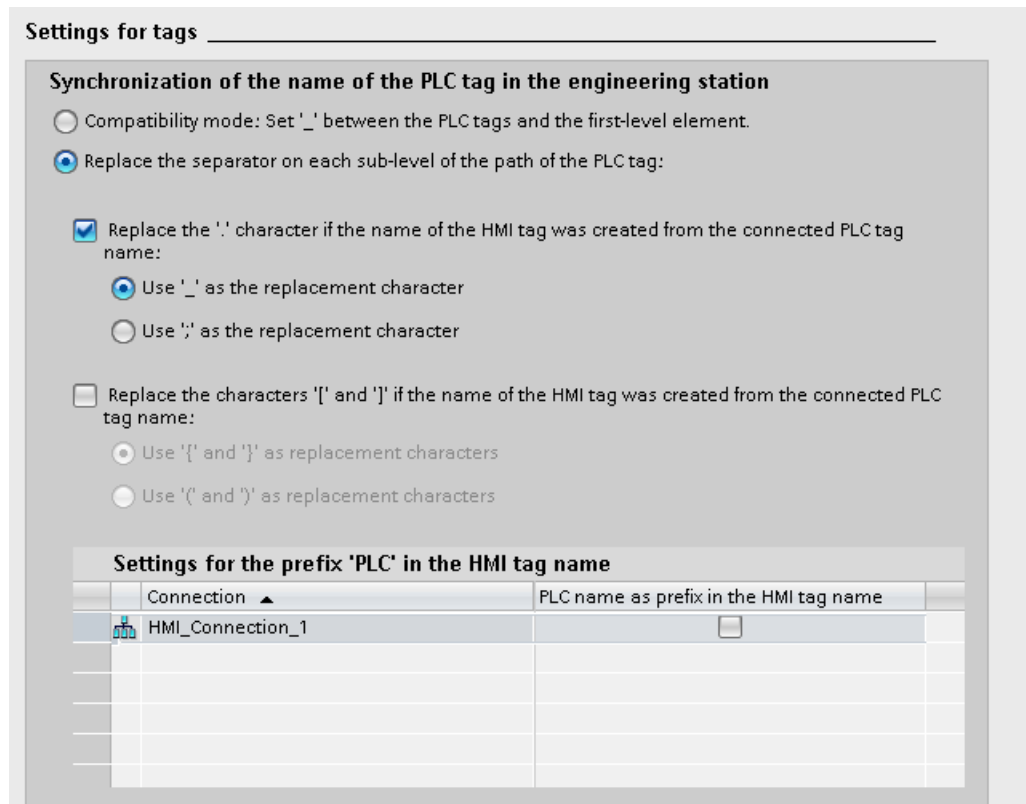
You can only use the SiVArc expressions HmiTag.SymbolicName and HmiTag.DB.SymbolicName in the "Tag rules" editor.

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

SiVArc object property	Referenced object	Formulation in the SiVArc expression
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 28)

8.4 Functions

8.4.1 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.

8.4.2 "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
<code>Contains("ButtonText", "Text")</code>	True
<code>Contains("ButtonText", "ttonT")</code>	True
<code>Contains("ButtonText", "butt")</code>	False
<code>Contains("ButtonText", "txeT")</code>	False
<code>Contains("ButtonText", "Text")</code>	False
<code>Contains("ButtonText", "Text ")</code>	False
<code>Contains("ButtonText", "Te xt")</code>	False
<code>Contains("ButtonText", "on")</code>	False
<code>Contains("ButtonText 1", "ButtonText 2")</code>	False

8.4.3 "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
<code>EndsWith("ButtonText", "Text")</code>	True
<code>EndsWith("ButtonText", "ButtonText")</code>	True
<code>EndsWith("ButtonText", "butt")</code>	False
<code>EndsWith("ButtonText", "Butt")</code>	False
<code>EndsWith("ButtonText", "Text")</code>	False
<code>EndsWith("ButtonText", "Text ")</code>	False
<code>EndsWith("ButtonText", "Te xt")</code>	False
<code>EndsWith("ButtonText", "t")</code>	True

Function	Result
<code>EndsWith("ButtonText", "T")</code>	False
<code>EndsWith("ButtonText ", "Text")</code>	False
<code>EndsWith("ButtonText 1", "ButtonText 2")</code>	False

8.4.4 "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
<code>Format(5, "0.00")</code>	5.00
<code>Format((VAR_1 Or 2#11100), "b")</code>	2#11101

You can find more information on the format string by searching for "Strings.Format method" in the Microsoft Developer Network.

8.4.5 "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. <code>FormatNumber ("12,4", 3, -2, -2, -2) ("12 comma 4") = 12,400</code>	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. <code>FormatNumber ("0,4", 3, -1, -2, -1) = 0,400</code>	The possible settings are listed under "List of constants".
4	UseHigherLevelAsNegativeNumbers	Number that specifies whether or not negative values are displayed in brackets. <code>FormatNumber ("-12", 1, -2, -1, 0) = (12,0)</code>	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. <code>FormatNumber ("1288,4", 3, -2, -2, 0) = 1288,400</code>	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 8-1 ApplyLeadingNumber

ApplyLeading-Number	Value	
TRUE	-1	Display leading 0.
FALSE	0	Do not display leading 0.
UseDefault	-2	Use country settings of the computer.

Table 8-2 UseHigherLevelAsNegativeNumbers

ApplyLeading-Number	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 8-3 UseHigherLevelAsNegativeNumbers

ApplyLeading-Number	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
<code>FormatNumber("12,4",3,-2,-2,-2)</code> ("12 comma 4")	12.400
<code>FormatNumber("12.4",3,-2,-2,-2)</code> ("12 point 4")	124.000
<code>FormatNumber("1288,4",3,-2,-2,-1)</code>	1.288,400
<code>FormatNumber("1288,4",3,-2,-2,0)</code>	1288.400
<code>FormatNumber("-12",1,-2,-2,0)</code>	-12.0
<code>FormatNumber("-12",1,-2,-1.0)</code>	(12.0)

8.4.6 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
<code>InStr("Hello","Hello")</code>	True
<code>InStr("Hello","hello")</code>	False

Function	Result
InStr("Hello","el")	True
InStr("12345",3)	True
InStr("12345","6")	False

8.4.7 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

8.4.8 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

8.4.9 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
<code>Left("ButtonText", 6)</code>	"Button"
<code>Left("ButtonText", 0)</code>	"" (Empty string)
<code>Left("ButtonText", "10")</code>	"ButtonText"
<code>Left("ButtonText", 11)</code>	Error (Number is greater than the number of characters in the string)

8.4.10 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
<code>Len("ButtonText")</code>	10
<code>Len("")</code>	0
<code>Left("ButtonText", Len("ButtonText"))</code>	"ButtonText"

8.4.11 Function "LTrim"

LTrim function

The `LTrim` function removes leading spaces from a string. The function expects a string as a function parameter.

Function	Result
<code>LTrim (" ButtonText")</code>	"ButtonText"
<code>LTrim ("ButtonText")</code>	"ButtonText"

8.4.12 Function "Max"

Max function

The `Max` function expects two numbers as a parameter and returns the higher of the two.

Function	Result
<code>Max(12, 3)</code>	12
<code>Max(3, 123)</code>	123

8.4.13 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
<code>Mid("ButtonText", 5, 3)</code>	"nTe"
<code>Mid("ButtonText", 0, 10)</code>	"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)

8.4.14 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

8.4.15 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
<code>Replace("a", "", "b")</code>	"a"
<code>Replace("aA", "a", "b")</code>	"bA"

8.4.16 "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
<code>Right("ButtonText", 4)</code>	"Text"
<code>Right("ButtonText", 0)</code>	"" (Empty string)
<code>Right("ButtonText", 10)</code>	"ButtonText"
<code>Right("ButtonText", 11)</code>	Error (Number is greater than the number of characters in the string)

8.4.17 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
<code>RTrim("ButtonText ")</code>	"ButtonText"
<code>RTrim("ButtonText")</code>	"ButtonText"

8.4.18 "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
<code>Split("SG19_FG97_ST090", "_") (0)</code>	SG19
<code>Split("SG19.FG97.ST090", ".") (1)</code>	FG97
<code>Split("42", ".") (0)</code>	42
<code>Split(".", ".") (0)</code>	"" (Empty string)

The following examples show the number of substrings that the `Split` function produces:

Function	Result
<code>Split("SG19_FG97_ST090", "_").Length</code>	3
<code>Split("SG19.FG97.ST090", ".").Length</code>	3

8.4.19 "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
<code>StartsWith("ButtonText", "Butt")</code>	True
<code>StartsWith("ButtonText", "butt")</code>	False
<code>StartsWith("ButtonText", "Text")</code>	False
<code>StartsWith("ButtonText", "ButtonText")</code>	True
<code>StartsWith("ButtonText", " Butt")</code>	False
<code>StartsWith("ButtonText", "Butt ")</code>	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

8.4.20 "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`

8.4.21 "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
<code>TrailNum("42")</code>	42
<code>TrailNum("Anzahl42")</code>	42
<code>TrailNum("Anzahl0042")</code>	42
<code>TrailNum("Anzahl-42")</code>	42
<code>TrailNum("Minimum42_Maximum84")</code>	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.

8.4.22 "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
<code>Trim("ButtonText")</code>	"ButtonText"
<code>Trim("ButtonText")</code>	"ButtonText"
<code>Trim("ButtonText")</code>	"ButtonText"
<code>Trim("ButtonText")</code>	"ButtonText"

8.4.23 "UBound" function

UBound function

The `UBound` functions expects an array as a parameter and returns the largest possible index.

Function	Result
<code>UBound(Split("SG19_FG97_ST090", "_"))</code>	2
<code>UBound(Split("SG19_FG97", "_"))</code>	1
<code>UBound(Split("", "."))</code>	0

8.5 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.

8.6 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

8.7 If conditions

You formulate logical conditions in SiVArc expressions with the If operator.

If operator

The If operator has the following syntax:

8.8 Supported data types for PLC tags

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

8.8 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.

8.9 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	---

8.10 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.

8.11 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
SiV	SiVArc configuration
	SiVArc expression that accesses a data block
	SiVArc expression that accesses a function block
	SiVArc expression that accesses a network entry

Tooltips

9.1 UMAC

9.1.1 CTHMISivarc

See also

[Creating SiVArc rules \(Page 160\)](#)

[SiVArc rules \(Page 156\)](#)

[User Management Control \(UMAC\) in SiVArc \(Page 26\)](#)

Index

"

- "SiVArc animations" tab
 - Layout, 52
- "SiVArc properties" tab
 - Structure, 91

A

- Acquisition type, 176
- Animation, 51
- Application
 - Generation matrix, 213
- Assigned property (SiVArc), 237

B

- Basic installation, 24
- Bit mask
 - Overflow screens, 74
- Block object (SiVArc), 227
- Block parameter, 94
- Button
 - SiVArc event, 136

C

- Call hierarchy, 117
- Changing
 - Rotation angle, 203
 - SiVArc rules, 205
 - Size, 203
- Comment
 - Copy rule, 34
 - Screen rule, 27
 - Tag rule, 28
 - Text list rule, 29
- Comment property (SiVArc), 237
- Condition
 - Screen rule, 27
 - Tag rule, 28
 - Text list rule, 29
- Conditions, 174
- Configuration
 - Text list entries, 94
- Contains function, 248

- Controller, (Screen rule)
 - Screen rule, 181
- Copy rule
 - Comment, 34
 - HMI device, 34
 - HMI device type, 34
 - Library object, 34
 - Name, 33
- Copy rules, 166, 172
 - Naming conflicts, 173

D

- DB object (SiVArc), 228
- Definition
 - Tag generation, 168
- Devices
 - Supported, 76

E

- Edit
 - Expressions, 104
- EndsWith function, 248
- Evaluation
 - Rules, 165
- Example
 - Screen rule, 27
- Exporting
 - SiVArc rules, 194
- Expressions
 - Edit, 104

F

- Faceplate, 181
- Faceplates
 - System functions, 136
- Fixed positioning, 61, 68, 203
- FolderPath property (SiVArc), 238
- Format function, 249
- FormatNumber function, 249
- Formulation rules
 - SiVArc expression, 104
- Function, 51
- Functional scope, 13
- Functions, 248
 - Contains, 248

- EndsWith, 248
- Format, 249
- FormatNumber, 249
- InStr, 251
- IsDefined, 252
- LBound, 252
- Left, 253
- Len, 253
- LTrim, 254
- Max, 254
- Mid, 254
- Min, 255
- Replace, 255
- Right, 256
- RTrim, 256
- Split, 257
- StartsWith, 257
- StrComp, 258
- TrailNum, 258
- Trim, 259
- UBound, 259

G

- Generated objects
 - Nesting depth, 71
- Generation
 - Visualization, 91
 - Visualization., 47
- Generation matrix, 203, 213, 216
 - Application, 213
- Generation overview
 - Program block, 42
 - Screen rule, 42
- Generation template
 - Screen rule, 27
- Generation templates, 174
- Global data block, 180
- Graphic lists, 171
- Group, 155

H

- HMI device
 - Copy rule, 34
 - Screen rule, 27, 181
- HMI device type
 - Copy rule, 34
- HMI object
 - Screen rule, 181
 - Text list rule, 183

- HMI objects, 96
- HMIApplication object (SiVArc), 229
- HMIDevice object (SiVArc), 229
- HMI Tag object (SiVArc), 230
- HMI Tag Prefix property (SiVArc), 239

I

- Import options, 179, 196
- Importing
 - Rule groups, 196
 - SiVArc rules, 196
- Index, 28
- IndexEndChar property (SiVArc), 239
- IndexStartChar property (SiVArc), 240
- InitialValue property (SiVArc), 240
- Instance data block, 180
- InStr function, 251
- Interconnections, 169
- Internal HMI tags, 168
- Internal tags, 171
- IsDefined function, 252

K

- Know-how protection
 - Setting up, 197

L

- Languages
 - Program blocks, 103
- Layout field
 - Screen rule, 27, 182
- LBound function, 252
- Left function, 253
- Len function, 253
- Library
 - SiVArc rules, 205
- Library object
 - Copy rule, 34
- LibraryObject object (SiVArc), 231
- LTrim function, 254

M

- Manual changes, 202
- Manually created HMI objects, 203
- Manually created text list entries, 204

Master copy
 Text list rule, 29
 Max function, 254
 Mid function, 254
 Min function, 255
 ModuleBlock object (SiVArc), 232

N

Name
 Copy rule, 33
 Screen rule, 26
 Tag rule, 28
 Text list rule, 29
 Name changes, 204
 Name property (SiVArc), 240
 Naming conflicts
 Copy rules, 173
 Priority, 167
 Naming convention, 170
 Navigation buttons, 74
 Nesting depth
 Generated objects, 71
 NetworkComment property (SiVArc), 241
 NetworkTitle property (SiVArc), 241
 New positioning, 203
 Number property (SiVArc), 242

O

Objects (SiVArc)
 Block, 227
 DB, 228
 HMIApplication, 229
 HMIDevice, 229
 HMI Tag, 230
 LibraryObject, 231
 ModuleBlock, 232
 Parameters, 233
 S7Control, 233
 StructureBlock, 235
 SubModuleBlock, 234
 TagNaming, 236
 Overflow screens
 Bit mask, 74
 with screen objects, 73

P

Parameters
 System function, 50

Parameters object (SiVArc), 233
 Password, 197
 Plant structure, 155
 Positioning methods, 58
 Positioning scheme, 69
 Priority
 Naming conflicts, 167
 Program block
 Generation overview, 42
 Screen rule, 26
 Text list rule, 29
 Program blocks, 103
 Languages, 103
 Properties (SiVArc)
 Assigned, 237
 Comment, 237
 FolderPath, 238
 HMI TagPrefix, 239
 IndexEndChar, 239
 IndexStartChar, 240
 InitialValue, 240
 NamGe, 240
 NetworkComment, 241
 NetworkTitle, 241
 Number, 242
 SeparatorChar, 242
 SymbolComment, 243
 SymbolicName, 243
 Title, 244
 Type, 245
 Value, 245
 Version, 245

R

Replace function, 255
 Right function, 256
 Rotation angle
 Changing, 203
 RTrim function, 256
 Rule groups
 Importing, 196
 Rules
 Evaluation, 165

S

S7Control object (SiVArc), 233
 Scope
 Tag generation, 176

- Screen
 - Screen rule, 27
 - Screen object
 - Screen rule, 27
 - Screen rule, 181
 - Comment, 27
 - Condition, 27
 - Controller, 27
 - Example, 27
 - Generation overview, 42
 - Generation template of a screen, 27
 - HMI device, 27, 181
 - HMI object, 181
 - Layout field, 27, 181
 - Name, 26
 - Program block, 26
 - Screen, 27
 - Screen object, 27
 - Screen rules, 165
 - Screen storage, 155
 - Screen type, 181
 - Screen window
 - Changing screen, 203
 - Screens, 156, 171
 - Script, 136
 - Scripts, 156, 171
 - SeparatorChar property (SiVArc), 242
 - Settings for tags, 175
 - SiVArc
 - Use, 13
 - SiVArc event, 136
 - Button, 136
 - SiVArc expression, 47, 51, 91
 - Formulation rules, 104
 - Syntax, 106
 - SiVArc object properties, 106
 - SiVArc objects, 105, 106
 - SiVArc positioning scheme, 61
 - SiVArc property, 47, 91
 - SiVArc reference, 203
 - SiVArc rule master copies, 206
 - SiVArc rules
 - Changing, 205
 - Exporting, 194
 - Importing, 196
 - Library, 205
 - SiVArc tags, 106, 168
 - SiVArc texts, 94
 - Size
 - Changing, 203
 - Spaces in tag names, 170
 - Split function, 257
 - StartsWith function, 257
 - Storage structures, 168, 176
 - StrComp function, 258
 - Structure
 - "SiVArc properties" tab, 91
 - StructureBlock object (SiVArc), 235
 - SubModuleBlock object (SiVArc), 234
 - Supported
 - Devices, 76
 - Symbol table, 94
 - SymbolComment property (SiVArc), 243
 - SymbolicName property (SiVArc), 243
 - Syntax
 - SiVArc expression, 106
 - System function, 136
 - Parameters, 50
 - System functions, 264
 - Faceplates, 136
- ## T
- Tag generation, 168
 - Definition, 168
 - Scope, 176
 - Tag group
 - Tag rule, 28
 - Tag rule, 174
 - Comment, 28
 - Condition, 28
 - Name, 28
 - Tag group, 28
 - Tag table, 28
 - Tag rules, 28, 123, 166
 - Index, 28
 - Tag table
 - Tag rule, 28
 - Tag tables, 156
 - TagNaming object (SiVArc), 236
 - Text list
 - Text list rule, 29
 - Text list entries, 203
 - Configuration, 94
 - Text list rule
 - Comment, 29
 - Condition, 29
 - HMI object, 183
 - Master copy of a text list, 29
 - Name, 29
 - Program block, 29
 - Text list, 29
 - Text list rules, 167
 - Text lists, 156, 171

Text sources, 174
Title property (SiVArc), 244
TrailNum function, 258
Trim function, 259
Type property (SiVArc), 245

U

UBound function, 259
Update cycle, 176
Use
 SiVArc, 13
User-defined positioning scheme, 61

V

Value property (SiVArc), 245
Version property (SiVArc), 245
Visualization
 Generation, 47, 91

