

SoMachine

Functions and Libraries User Guide

06/2017

EIO0000000735.13

www.schneider-electric.com



The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2017 Schneider Electric. All Rights Reserved.

Table of Contents



	Safety Information	5
	About the Book	7
Chapter 1	General Description of Libraries	11
	Generalities About Libraries	11
Chapter 2	Library Management	15
2.1	Library Referencing Methods	16
	Introduction	17
	Direct Version	18
	Newest Version	20
	Placeholder Mechanism	22
	Forward Compatible Libraries	24
2.2	Using Libraries	26
	Different Ways to Declare a Library in a Project	27
	Adding Libraries to a SoMachine Project	28
	Updating Libraries and Library References	30
	Creating Your Own Forward Compatible Library	34
Chapter 3	Library Manager Editor	37
	Library Manager Editor	38
	Add Library	41
	Properties	45
	Try to Reload Library	47
	Placeholders	48
	Library Repository	49
	Version Mapping Tab	56
Chapter 4	Create Your Own Library	59
	General Information	60
	Step 1: Setup the Project	61
	Step 2: Fill-in the Project Information	62
	Step 3: Reference Other Libraries, if Needed	65
	Step 3.1: Use Placeholder References	66
	Step 4: Design and Program the Library Modules	67
	Step 5: Design the Interface	68
	Step 6: Implement Error Handling Routines	69
	Step 7: Set up a Reasonable Deployment (Usage Restriction)	70

Chapter 5 Libraries Overview	71
Schneider Electric Libraries	72
Other Libraries Used in SoMachine	90
Chapter 6 Function and Function Block Representation	93
Differences Between a Function and a Function Block	94
How to Use a Function or a Function Block in IL Language	95
How to Use a Function or a Function Block in ST Language	99
Glossary	103
Index	109

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book



At a Glance

Document Scope

This document describes the libraries, libraries management and functions implementations in SoMachine.

Validity Note

This document has been updated for the release of SoMachine V4.3.

Product Related Information

WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.¹
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

¹ For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
EN 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2008	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
EN 1088:2008 ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2006	Safety of machinery - Emergency stop - Principles for design
EN/IEC 62061:2005	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2008	Digital data communication for measurement and control: Functional safety field buses.

Standard	Description
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive (2006/42/EC)* and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Chapter 1

General Description of Libraries

Generalities About Libraries

Content of Libraries

Libraries provide the following items that are executed in the runtime system of your controller:

- functions and function blocks
- data type definitions
- global variables
- system variables
- visualization objects

The management of the libraries in a project is done by using **Library Managers** for the whole project or controller-specific libraries. The installation of libraries is accomplished during the installation of those elements (devices, solutions, controllers) you select to install with SoMachine Configuration Manager. For user-defined libraries, they are managed by you via SoMachine directly.

Library Information

In the **Library Manager**, you find the included libraries and in the **Library Repository**, you find the available libraries:

Information	Description	Example
Name	name of the library	Altivar Library (ATV)
Version	version of the library	1.0.1.8
Company	The primary provider or group name defined by the primary provider of the library, as shown in the Library Manager and Library Repository dialog box.	Schneider Electric
Namespace	The default namespace of the library for accessing functions of the library. NOTE: Best practice is to always use the default namespace as the namespace used in your application. If qualified-access-only is listed after the default namespace, this indicates that the usage of the namespace in your application is mandatory.	SEC
Category	The category (or categories) which this libraries belongs to, as shown in the Library Manager and Library Repository dialog box.	Devices → ATV31/ATV312

Namespace

A library namespace is a symbol that allows the unique access to the attached library components (functions, function blocks, variables...). The namespace is necessary when two components of two different libraries used in the same project have the same name. The usage of the namespace in your application is mandatory if the library has set the attribute qualified-access-only . To ensure unique access to the correct component, use the full name <namespace>.<component> format, including the namespace.

Case	Description
1	There is a function block GEN in the library Util. The namespace of the library Util is Util . An instance of the function block GEN can be declared with or without the library namespace if the name GEN is unique within the project: MyGenerator: Util.GEN; or MyGenerator: GEN;
2	A function block GEN has been created within the project. The use of the library Util namespace will allow the system to access the function block GEN of the library Util. Without namespace, the project function block GEN will be accessed: MyGenerator_Util: Util.GEN; MyGenerator_Project: GEN;
3	Another library, also containing a function block called GEN, is declared in the project with namespace NewLib. The use of the namespace becomes mandatory to identify the correct function block GEN to be accessed: MyGenerator_Util: Util.GEN; MyGenerato_NewLib: NewLib.GEN;

A default namespace is defined for each library.

Library Repository

The **Library Repository** is the editor that manages libraries installed in SoMachine. The **Library Repository** allows you to install or remove user-defined libraries. A library can be used in a SoMachine project only if it is installed in the **Library Repository**. With the installation of SoMachine, a set of libraries is installed by default. You can install new libraries or new versions of existing libraries with the SoMachine Configuration Manager.

Managing Libraries by Using Library Managers

Libraries declared in a project are managed in the **Library Manager** editor.

Within a SoMachine project, two different **Library Managers** are available in the **Tools tree** (see *SoMachine, Programming Guide*) for different use cases of libraries:

Use Case	Location of the Library Manager
libraries available for a specific controller	In the Tools tree below the Application node for each controller.
libraries available for a project	In the Tools tree under Global , add a Library Manager .

For more information about library management, **Library Repository** and **Library Manager Editor**, refer to Library Management (see [page 37](#)).

For more information on finding a function or function block of libraries with the **FFB Finder**, refer to How to Find a Function or Function block with the FFB Finder (see *SoMachine, Programming Guide*).

Chapter 2

Library Management

Overview

This chapter provides information to help you managing libraries and library versions of referenced libraries.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Library Referencing Methods	16
2.2	Using Libraries	26

Section 2.1

Library Referencing Methods

Overview

This section provides general information about library referencing methods, their advantages, and disadvantages as well as their use in SoMachine.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction	17
Direct Version	18
Newest Version	20
Placeholder Mechanism	22
Forward Compatible Libraries	24

Introduction

Overview

Both projects and libraries themselves can use external functionalities which are stored in other, separate libraries. To use those functionalities, these separate libraries have to be included into the project in the **Library Manager** as referenced libraries. Libraries which are referenced by libraries used in a project or library are referred to as indirectly referenced libraries. These indirectly referenced libraries are included automatically when compiling and should not be used in a project itself.

Depending on the used referencing method, the possibility and/or affect of a change of the library version varies. Changing an indirect referenced library version for example is not possible.

Instead, the library versions are changed by the different library management mechanisms. Due to these mechanisms an identical library version is used in all references in the entire project.

Libraries can be referenced in various ways:

- Direct version (*see page 18*)
- Newest version (*see page 20*)
- Placeholder mechanism (*see page 22*)
- Forward compatible library (FCL) (*see page 24*)

Direct Version

Overview

The simplest way of referencing libraries is to define explicitly within the **Library Manager** of an application or library project which library and exact library version should be used.

If a library X embeds another library Z using a direct reference, then library Z will be loaded exactly in the version that is embedded by library X.

This can lead to several versions of the same library being loaded in one **Library Manager** of a project.

NOTE: The method allows referencing several versions of a library within the same project. Although this method is advantageous for certain types of libraries, care must be taken when applying it to others to avoid complications. Therefore, this method is generally discouraged except where indicated.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Verify whether the versions of the libraries contained in your program are correct, after updating the software.
- Verify whether the versions of the updated libraries are consistent with your application specifications.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Example

Situation: The following libraries are added in the **Library Manager**:

Library	Version
Library X	1.0.0.0
Library Y	1.0.0.0
Library Z	1.0.0.0

Dependencies:

Library...	Requires Library Z in Version...
X	1.0.1.0
Y	1.0.2.0

Assuming that library Z was added in all libraries as a direct library version, this would mean:

- Library Z will be loaded in 3 different versions:
 - The POU's of the project are directly using the functionalities of version 1.0.0.0.
 - The POU's of library X are using the functionalities of version 1.0.1.0.
 - The POU's of library Y are using the functionalities of version 1.0.2.0.
- This can cause potential compiler errors, for example, if the system tries to exchange data between the POU's of library Z used by library X and between the POU's of library Z used by library Y. This can occur, even if the data structure is identical.
- When the library reference is a direct version reference, a change of the indirect library dependencies is not possible after the library creation.

Newest Version

Overview

This referencing method is similar to the direct version referencing method.

Instead of an exact library version, a symbol (*) is defined as referenced library version. Every time when compiling, the newest locally installed version of a library is used in the current project as referenced library version.

NOTE: When a new version of a library gets installed in the library repository, all projects and libraries referencing this library with the newest version will get changed without further notice when compiled. Normally this behavior is not desirable in your application because it may introduce unintended changes, either during build or during program execution.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Verify whether the versions of the libraries contained in your program are correct, after updating the software.
- Verify whether the versions of the updated libraries are consistent with your application specifications.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Example

Situation: The following libraries are added in the **Library Manager**:

Library	Version
Library X	1.0.0.0
Library Y	1.0.0.0
Library Z	1.0.0.0

Dependencies:

Library...	Requires Library Z in Version...
X	1.0.1.0
Y	1.0.2.0

Additionally the following library version is installed in the **Library Repository**:

On Computer A	On Computer B
Library Z. 1.0.3.0	Library Z. 1.0.2.0

Assuming that library Z was added by using the newest version method, this would mean:

On Computer A:	On Computer B:
Library X uses version 1.0.3.0.	Library X uses version 1.0.2.0.
Library Y uses version 1.0.3.0.	Library Y uses version 1.0.2.0.

Placeholder Mechanism

Overview

The placeholder mechanism uses a location outside of the library, a so-called placeholder, for referencing another library.

When devices are upgraded or when new devices are installed, the information concerning which library versions are compatible with the device version is also installed. The compiler inserts this information where placeholders are used.

In contrast to a newest version reference, a placeholder reference is not changed unintentionally. The placeholder reference only gets changed when you intentionally switch to a newer version of the following components:

- Library profile (depending on the compiler version)
- Device description (for device-specific libraries)
- Project profile (**Placeholder** dialog box in the **Library Manager**)

The search order for the placeholder definition is (highest priority first):

1. **Placeholder** dialog box
2. Device description
3. Library profile

The placeholder resolution is already correctly defined by the installation of SoMachine.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Verify whether the versions of the libraries contained in your program are correct, after updating the software.
- Verify whether the versions of the updated libraries are consistent with your application specifications.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTE: The placeholders are defined by the controller versions. When you create a new library, there is no placeholder defined by the controller versions for this new library. The placeholder mechanism cannot be used when adding this custom library to the device library manager.

NOTE: Library names and placeholder references are case-sensitive.

Example

Situation: The following placeholders are added in the **Library Manager** instead of libraries:

Placeholder	Library
Placeholder X	1.0.0.0
Placeholder Y	1.0.0.0
Placeholder Z	1.0.0.0

Dependencies:

Library...	Requires Library Z by Placeholder Z
X	1.0.1.0
Y	1.0.2.0

Additionally the following placeholders are replaced by controller A:

Library Version Defined in Controller A:	Default Library Version in the Placeholder Tab of the Library Manager:
Placeholder Z. 1.0.2.0	Library Z. 1.0.1.0

Forward Compatible Libraries

Overview

A *forward compatible library (FCL)* is developed in such a way that its functionalities are forward compatible. This means that every version of a forward compatible library contains all functionalities of the previous version and a newer library version can be easily used in already existing projects without any changes.

Dependencies to libraries (library X uses library Z) are read as minimum compatible version.

If a library X requires another, forward compatible library Z, for example in version 1.0.0.0, library X will work with version 1.0.0.0 or any newer version of library Z.

Only one single version of a forward compatible library is selected and used in a project **Library Manager** on request (when clicking the **Automatic** button). This compatible version of the library selected in the Version mapping tab of the Library Manager (*see page 56*) is used in the direct and indirect references in the libraries within this project.

This referencing method has the following advantages:

- It supports a parallel independent development process of several libraries.
- It eases library updates by forward compatible development ruleset.

It is assumed, that once a version of a library is marked as a forward compatible library, then all future versions will also be forward compatible.

For detailed information about creating forward compatible libraries, refer to Create Your Own Forward Compatible Library (*see page 34*).

Example

Situation: The following libraries are added in the **Library Manager**:

Library	Version
Library X	1.0.0.0
Library Y	1.0.0.0
Library Z	1.0.0.0

Dependencies:

Library...	Requires Minimum Version of Library Z...
X	1.0.1.0
Y	1.0.2.0

On the local system the following versions of library Z are installed:

- 1.0.0.0
- 1.0.1.0

- 1.0.2.0
- 1.0.3.0

Assuming that the installed versions of library Z are marked as forward compatible, this would mean:

- Only one version of library Z is loaded.
- Compatible versions of library Z in this project are 1.0.2.0 and 1.0.3.0, which meet the minimum required dependencies.
- You can configure which version is to be used in this case (however, it makes sense to use the newest installed compatible version).
- After clicking the **Automatic** button in the **Version mapping** tab of the project, version 1.0.3.0 of library Z will be chosen because it is the newest installed compatible version.
- The POUs of the projects of library X and library Y use the same versions of POUs of library Z.
- The exchange of POUs from library Z between the project and other libraries is possible.

Advanced Example

Situation:

- A new version 1.0.3.1 of library Z uses some features of a system library referenced via placeholder V.
- This library is only compatible to controller A since version 2.0.0.0. This is indicated in library Z version 1.0.3.1 by a minimum controller firmware (*see page 34*) requirement.

This would mean:

- If a project uses controller A version 1.0.0.0, the following libraries are compatible:
 - 1.0.2.0
 - 1.0.3.0
- If a project uses controller A version 2.0.0.0, the following library is compatible:
 - 1.0.3.1
- When clicking the **Automatic** button in the **Version mapping** tab, version 1.0.3.1 of library Z will be selected, if the controller A used in the project has been updated to version 2.0.0.0. Otherwise version 1.0.3.0 of library Z will be selected.

Section 2.2

Using Libraries

Overview

This section provides information on situations you can encounter, when adding libraries to an existing SoMachine project, updating libraries or creating a forward compatible library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Different Ways to Declare a Library in a Project	27
Adding Libraries to a SoMachine Project	28
Updating Libraries and Library References	30
Creating Your Own Forward Compatible Library	34

Different Ways to Declare a Library in a Project

Overview

There are different ways to declare a library in a project. The libraries are automatically declared when adding the following:

- a controller:
 - IEC 61131 basic libraries: Standard and Util libraries
 - Controller PLCSystem library
 - Other libraries to manage embedded controller features
- specific controller features
- a fieldbus manager
- a fieldbus device

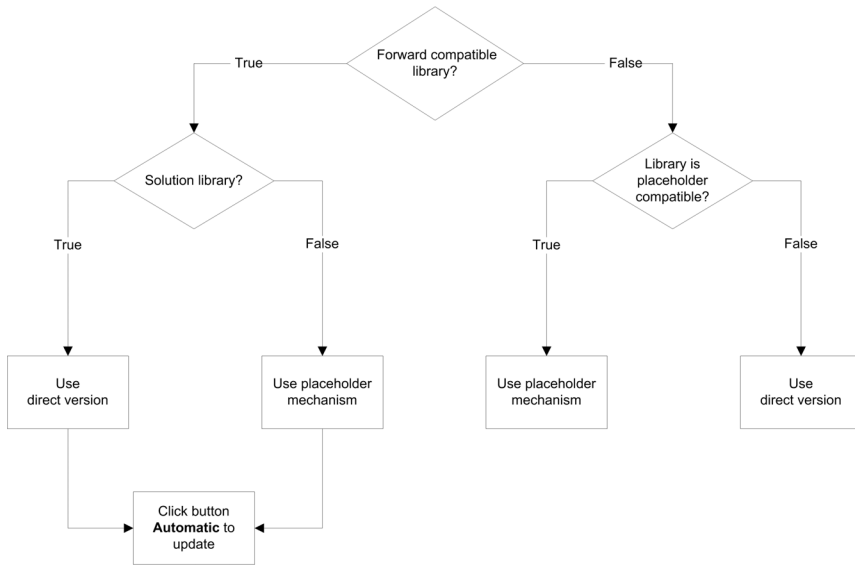
Some libraries must be added manually.

NOTE: Some functions and function blocks are not intended to be used within your program, but may be required by the system or by a device. You should use only functions and function blocks that are documented as program applicable, or functions or function blocks that you have thoroughly tested with your application.

Adding Libraries to a SoMachine Project

Overview

The following flowchart and table represent which referencing mechanism should be used, when adding a library in the **Library Manager** to a SoMachine project:



If the library is...	and...	then use the...
forward compatible ¹⁾	it belongs to the library category Solution ²⁾ or is a user-defined library	direct version: <ul style="list-style-type: none"> Set an explicit library version in the Library Manager → Library tab, for example Packaging, V.1.0.3.0 (Schneider Electric) For updating this kind of library, use the Automatic button in the Version mapping tab of the Library Manager
forward compatible ¹⁾	it does not belong to the library category Solution ²⁾ and is not a user-defined library	placeholder mechanism: <ul style="list-style-type: none"> Choose the respective placeholder in the Placeholder tab of the Library Manager, for example SE_ToolboxAdvance for the Toolbox_Advance library. Set a default library version in the Placeholder tab of the Library Manager. The default library version will be used if no device description is available. For updating this kind of library, use the Automatic button in the Version mapping tab of the Library Manager
not forward compatible ¹⁾	the library is placeholder compatible ¹⁾	placeholder mechanism: <ul style="list-style-type: none"> Choose the respective placeholder in the Placeholder tab of the Library Manager, for example SE_PTOPWM for the XBTGC PTOPWM library. Set a default library version in the Placeholder tab of the Library Manager. The default library version will be used if no device description is available.
not forward compatible ¹⁾	the library is not placeholder compatible ¹⁾	direct version: <ul style="list-style-type: none"> Set an explicit library version in the Library Manager → Library tab, for example SE_Packaging, V.1.0.3.0)

- To check if the library you want to add is forward compatible or placeholder compatible, refer to Schneider Electric Libraries ([see page 72](#)) and Other Libraries Used in SoMachine ([see page 90](#)).
- To check if the library you want to add belongs to the category **Solution**, refer to Schneider Electric Libraries ([see page 72](#)).

NOTE: If a library or a later version of the library is forward compatible, it will be automatically recognized by SoMachine and be respected in the automatic update mechanism (**Automatic** button in the **Version mapping** tab).

Updating Libraries and Library References


Overview

This topic describes the situations you can encounter when updating libraries.


Update Project Dialog

When opening an existing project in SoMachine, the **Update project** dialog box is displayed, if one of the following elements has been installed on the local system:


- a newer compiler version,
- a newer visualization version,
- a newer device version,
- a newer library version of one of the used libraries.

Update Project 

Compiler version

Current version: 3.5.3.0 (CoDeSys version, not released) New version: 4.0.0.0 

Visualization profile

Current profile: New profile: SoMachine V4.0 

Device	Current device type and version	New device type and version
HMIGTO5310_5315	HMIGTO5310/5315 (2.0.4.1)	HMIGTO5310/5315 (3.5.3.10)
MyController	TM258LF42DT4L (2.0.40.10)	TM258LF42DT4L (2.0.40.19)

Libraries

Update all libraries

Don't show this dialog again.

The following effects will occur when you update your libraries using the **Update project** dialog box:

If...	Then...
the option Update all libraries in the Update project dialog box is activated and you click OK .	<ul style="list-style-type: none"> ● all direct referenced libraries are updated. ● all libraries with at least one forward compatible library version installed in the Library Repository and the version mapping of former legacy versions will be updated to the newest forward compatible library version.
the option Update all libraries in the Update project dialog box is deactivated and you click OK .	<ul style="list-style-type: none"> ● no existing library reference is changed. ● a manual update of the libraries can be executed in the Version mapping tab of the Library Manager ● legacy libraries will be listed in the Version mapping tab, but marked as legacy, if a newer, forward-compatible library version exists. ● the Update project dialog box is shown again next time you open this project. You can omit this by setting the option Don't show this dialog again in the Update project dialog box or in the Project Settings.
you click Cancel .	<ul style="list-style-type: none"> ● no existing library reference is changed. ● legacy libraries will be listed in the Version mapping tab, but marked as legacy, if a newer, forward compatible library version exists. ● a manual update of the libraries, the device, the visualization, or the compiler can be executed in the project. ● the Update project dialog box is shown again next time you open this project. You can omit this by setting the option Don't show this dialog again in the Update project dialog box or in the Project Settings.

Manual Library Update

If	Then...
<p>you want to update forward compatible libraries manually,</p>	<p>this manual update can be executed in the Version mapping tab of the Library Manager. When using the Automatic button for updating, the following effects will result:</p> <ul style="list-style-type: none"> ● All forward compatible libraries will be updated to the newest forward compatible library version. ● All legacy libraries are changed into forward compatible libraries and will be updated to the newest forward compatible library version. <p>For updating only one library, right-click on the respective library in the Version mapping tab and select Edit version mapping (selected library).</p>
<p>you want to update non forward compatible libraries manually,</p>	<p>this manual update can be executed in the Libraries tab of the Library Manager:</p> <ol style="list-style-type: none"> 1. In the Libraries tab select a library and click Menu → Commands → Library Manager → Properties. Result: The Properties dialog box is displayed. 2. In the Properties dialog box select one of the versions installed on the local system and click OK.

Manual Device Update

When manually updating a device description using **Update Device...** the following libraries are updated too:

- libraries which are automatically included by the device,
- libraries which are included as placeholders,
- forward compatible libraries.

To update a device (*see SoMachine, Programming Guide*) right-click on the device in the **Devices tree** and select **Update Device...**

Project Created with Previous Version

In a project created with a previous version of SoMachine software, the versions of the libraries declared in the project will be changed as follows:

- The library versions are kept unchanged for libraries declared with a direct version (*see page 18*).
- They are automatically updated with the newest version (*see page 20*) for libraries declared using the *newest* version method (version identified with * in the **Library Manager**).
- They are automatically updated with the versions defined in the controller *Device Description File* after a controller device update command for libraries declared using the placeholder mechanism (*see page 22*).

Creating Your Own Forward Compatible Library

Overview

When creating your own library, it is by default defined as not forward compatible.

Set the attribute `ForwardCompatibleLibrary` in the **Project Information** of your library project to create your own forward compatible library. For further details, refer to the procedure below.

NOTE: Setting the `ForwardCompatibleLibrary` key means that your library and its subsequent versions meet the following requirements:

The functionality of a previous versions is available in a later version:

- All POUs of the previous versions are available.
- The behavior of the POUs is identical in the previous and the later versions of your library.
- The visibility of the POUs of the previous version is equal or greater than in the later version.
- Names and data types of your inputs and outputs are unchanged in the different versions.

Additionally, an optional attribute (`MinimumControllerFirmware`) can be set if a device dependency exists. The attribute indicates the necessary minimum firmware version for the supported controllers. It is only necessary if a library uses POUs from another library that is referenced as a placeholder and the placeholder resolution is not a forward compatible library.

The `MinimumControllerFirmware` attribute consists of a manufacturer and device identification number, as well as the necessary minimum firmware version.

Procedure

NOTE: The following procedure describes the additional steps that are necessary for creating a forward compatible library when creating your own library.

Step	Action
1	Follow the procedure of Creating Your Own Library (<i>see page 59</i>) until step 3.
2	In the menu Project → Project Information open the Project Information dialog box, select the Properties tab, and enter or choose the following entries: <ul style="list-style-type: none"> ● Key field: ForwardCompatibleLibrary ● Type field: Boolean ● Value field: True
3	Click Add for adding the key to the Project Information of your library.

Step	Action
4	<p>If required, also add the optional attribute <code>MinimumControllerFirmware</code>. In the File Project → Project Information enter or choose the following entries:</p> <ul style="list-style-type: none">● Key field: <code>MinimumControllerFirmware</code>● Type field: <code>Text</code>● Value field: device category "/" vendor and device id "/" version, for example <code>4096/1003 0082/1.33.2.0</code> <p>If several devices shall be considered in the <code>MinimumControllerFirmware</code> attribute, then the identification numbers have to be entered into the Value field together, separated with a .</p> <p>For example:</p> <ul style="list-style-type: none">● <code>4096/1003 0082/1.33.2.0 4096/1003 009D/1.35.1.1</code>
5	<p>Continue with step 3 of the procedure of Creating Your Own Library (<i>see page 59</i>).</p>

Chapter 3

Library Manager Editor

What Is in This Chapter?

This chapter contains the following topics:

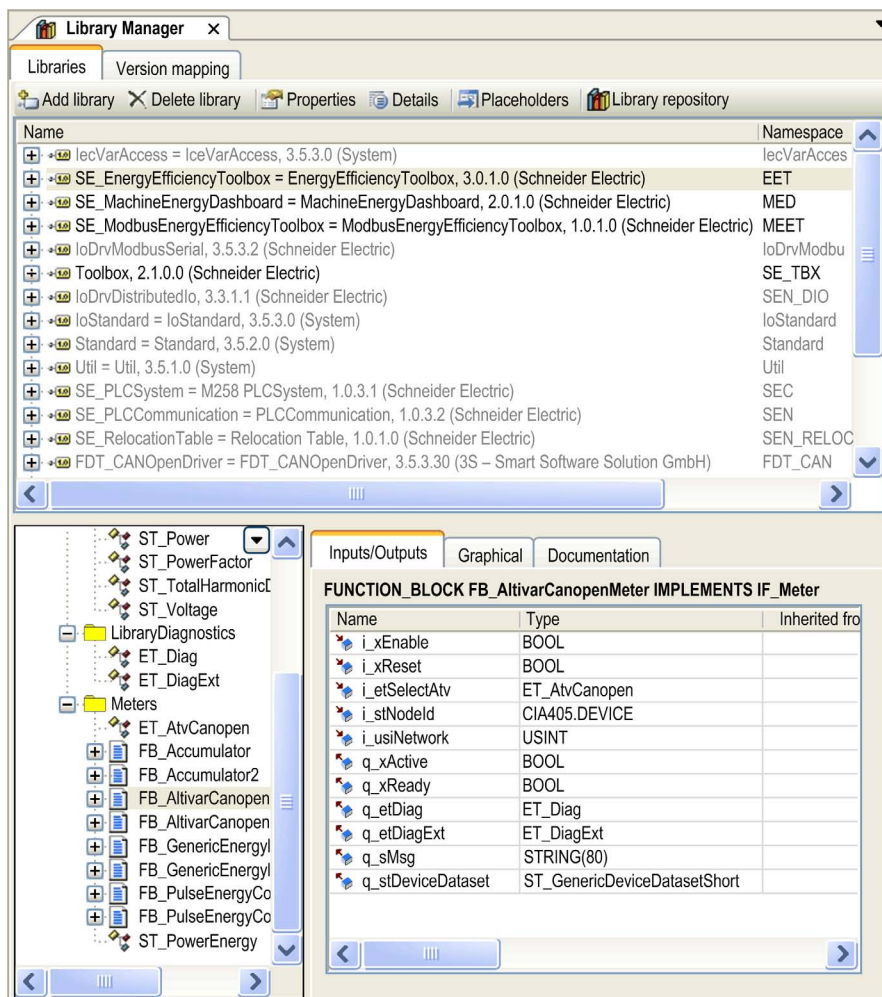
Topic	Page
Library Manager Editor	38
Add Library	41
Properties	45
Try to Reload Library	47
Placeholders	48
Library Repository	49
Version Mapping Tab	56

Library Manager Editor

Overview

Double-click a library node in the **Tools tree** to open the **Library Manager** in an editor view.

Library Manager editor view, **Libraries** tab



The editor window view consists of 4 parts:

- The upper part, displaying libraries currently included in the project
- The lower left part, displaying particular modules of the library selected in the upper part

- The lower right part, displaying further information on the library selected in the upper part in different tabs
- a toolbar with buttons and commands

Description of the Upper Part





The upper part of the view displays the libraries currently included in the project.

The following information is provided:

Parameter	Description
Name	Indicates the title, version, and optionally company name as specified in the Summary tab of the Project Information dialog box of the library.
Namespace	The default setting for the namespace of a library is <library name> . An exception is made by libraries that have another namespace in their Project Information . Use the library namespace (<i>see SoMachine, Programming Guide</i>) as a prefix of the identifier. This helps to access a module which is plurally available in the project, or if the use of this prefix is enforced by the library property LanguageModelAttribute qualified-access-only . You can modify the standard namespace of an included library for local use (within the project) in the Properties dialog box (<i>see page 45</i>).
Effective Version	The effective version is the library version which is currently used due to the definition in the library Properties dialog box (<i>see page 45</i>).

Libraries which have been automatically included in a project are displayed as gray-colored, those which have been added manually (Add library...) are displayed in black color.

An icon before the library name indicates the type of the library:

Icon	Description
	Library with version information
	Referenced library (<i>see page 17</i>)
	The referenced library file could not be found or is not a valid library (see the corresponding message in the Library Manager list of the Messages view). For further information, refer to the description of the Try to Reload Library command (<i>see page 47</i>).
	Licensed library for which currently no valid license is available

If a library has dependencies on other libraries (referenced library (*see page 17*)), those are automatically included - if available - and are displayed with icon for referenced libraries in a subtree of the entry. Expand or collapse a subtree by clicking the respective plus or minus sign.

Description of the Lower Part

In the lower left part of the editor, the particular modules of the currently selected library are displayed.

The lower right part of the editor can contain the following tabs:

Tab	Description
Documentation	The components of the module currently selected in the left part are displayed in a table, showing the variable Name , data Type and the Comment which may have been added to the component declaration when creating the library. If a folder is selected, the associated documentation is displayed. Thus, inserting such comments is an easy way to automatically providing module documentation.
Inputs/Outputs	The components of the currently selected library module are listed in a table with variable Name , data Type , Address , Initial value, and Comment , as defined in the library.
Graphical	Graphical representation of the module
Library Parameters	This tab is only available for libraries containing a parameter list. You can edit the parameter values in column Value (editable) . For details, refer to the paragraph GVL for Configurable Constants (Parameter List) in Libraries (<i>see SoMachine, Programming Guide</i>).

Buttons and Commands

The following commands, which are provided in the editor view when one or several libraries are selected, correspond to those of the libraries. This is by default available in the menu bar as long as the **Library Manager** editor is active:

Tab	Description
Add library	For including a library into the project The precondition is that the library is installed on the local system. An appropriate message appears in case of trying to insert a library which is already available in the project (<i>see page 41</i>).
Delete library	The library currently selected in the library list is deleted from the project.
Properties	For settings on the namespace and - with notice that the library will later be available as a referenced library in another project - for settings on version handling, visibility and accessing. Refer to the Properties... chapter (<i>see page 45</i>).
Details	Displays a dialog box with details about the library (general information, contents, properties, license information).
Placeholders	For redirecting any placeholder and library group to a different version. Refer to the Placeholders... chapter (<i>see page 48</i>).
Library repository	For defining library locations and for installing or uninstalling libraries. Refer to the Library Repository chapter (<i>see page 49</i>).

Add Library

Overview

In the **Library Manager** editor view, click the **Add library** button to add libraries which are already installed on your system to the project.

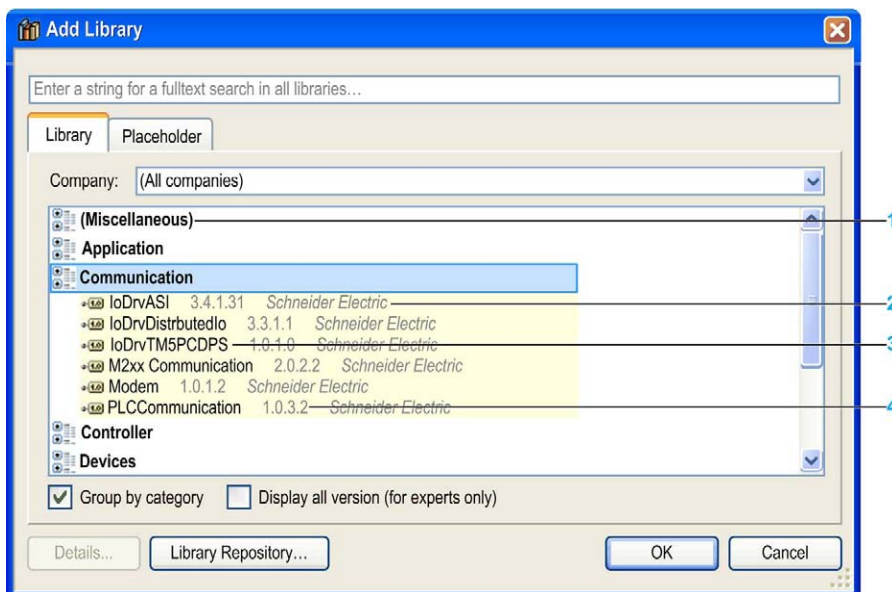
The **Add Library** dialog box opens. It consists of 2 tabs:

- The **Library** tab that allows you to add a definite library by specifying a fixed version.
- The **Placeholder** tab that allows you to reference a library using a placeholder.

Consider the general information concerning library management ([see page 15](#)).

Library Tab

Library tab of the **Library Manager** editor view



- 1 Category
- 2 Company
- 3 Title
- 4 Version

The **Add Library** tab allows you to search in all installed libraries, for example for a certain function block. Enter the text you want to find in the text box in the upper part of the dialog box. Only those objects (library names, POUs, data types, and comments inside these objects) that contain the searched text is displayed in the lists of the **Library** and **Placeholder** tab.

The **Library** tab lists the libraries currently installed on your system with title, version, company, and category as defined in the project information of the library. You can filter the display by setting a certain providing **Company** from the selection list. **(All companies)** lists all available libraries.

If the option **Group by category** is activated, the libraries of the currently set company is listed according to the available categories. The categories appear as nodes, the libraries (or further categories) are displayed indented below. If the option **Group by category** is not activated, the libraries are displayed in alphabetical order.

NOTE: For designing and referencing libraries, follow the guidelines for creating libraries (*see page 65*).

Choose the desired library.

If the option **Display all versions (for experts only)** is activated, all installed versions of the libraries display indented below the currently selected library entry. In addition to the explicit version identifiers, an asterisk * is available, which means **latest version**. This allows you to choose among the versions. By default, this option is deactivated and only the latest version is displayed.

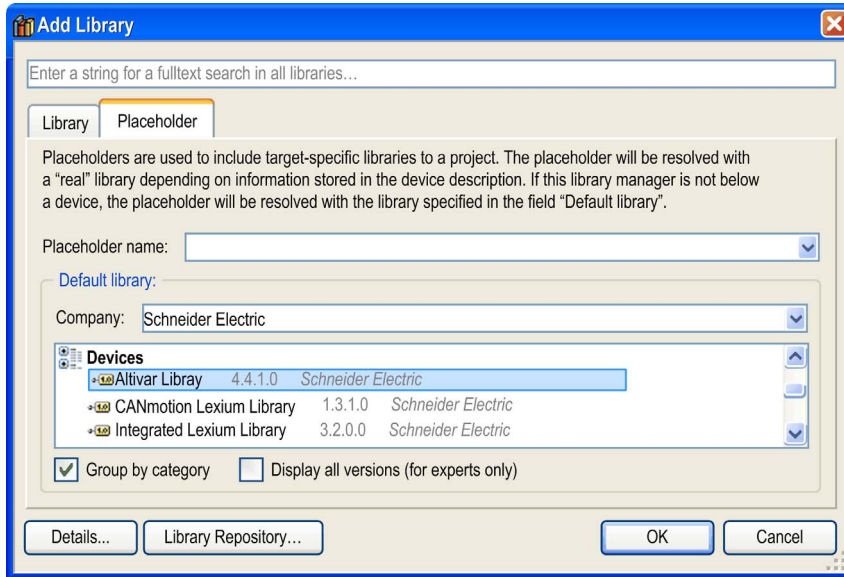
In this case, a multiple selection of libraries is possible: keep the CTRL or SHIFT key pressed while selecting the desired libraries.

After having confirmed by clicking **OK**, the selected libraries will be added to the list in the **Library Manager** view.

If you want to include a library which is not yet installed on the local system, click the **Library Repository** button. It opens the **Library Repository** dialog box (*see page 49*) for performing the required installation.

Placeholder Tab

Placeholder tab of the Library Manager editor view



Use placeholders in the following cases:

- To achieve compatibility of projects for multiple interchangeable target devices.
- If your project is a library project referencing other, device-specific libraries.

Then include these specific libraries in the library manager via placeholders, which are defined in the device descriptions. Also consider the practices for creating your own library ([see page 66](#)).

NOTE: In case of library development, the placeholder cannot be resolved. The selected default library is taken into account.

NOTE: Also consider the assignment of library placeholder resolutions depending on the currently used compiler version, defined via library profiles ([see page 54](#)).

Adding a Library Via Placeholder

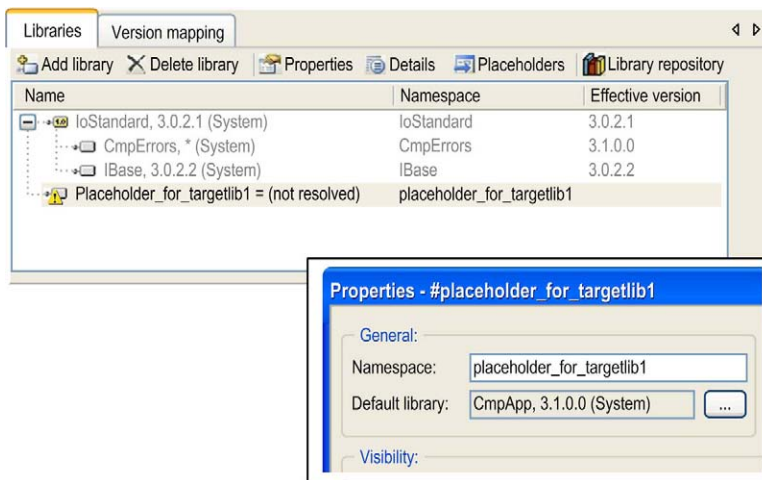
Enter a name in the **Placeholder name** text box. To help to ensure correct insertion of the name, click the arrow button to open the selection list offering all placeholder names currently defined in device descriptions.

Choose a default library from the currently installed libraries that are displayed in the list of the **Default library** area. This default library is used if for any reason no device is available. It allows compilation of the currently edited library project without detecting errors. Select a default library as described for adding a library in the **Library** tab. You can also activate the option **Display all versions (for experts only)** to display all currently installed versions of a library.

As soon as you close the **Placeholder** tab by clicking **OK**, the placeholder library is inserted in the tree structure of the **Library Manager**. Select this entry and open the **Properties** dialog box (see page 45) to display information on the currently set default library.

In the following example, the placeholder is not yet replaced (resolved). As soon as the **Library Manager** belongs to an application of a device that references this library, the name, and the version of this device-specific library is displayed in the respective columns.

Example for a library placeholder inserted in a **Library Manager**

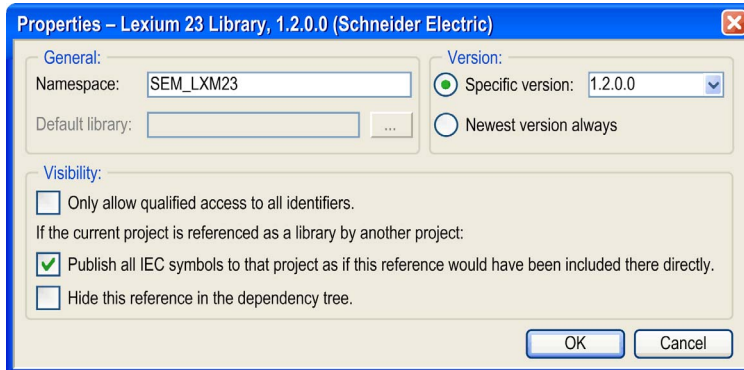


Properties

Overview

In the **Library Manager** editor view, click the **Properties** button to open the **Properties** dialog box for the selected library. It allows you to configure the namespace, version handling, availability, and visibility of library references.

Properties dialog box for a library



NOTE: Before working on library projects and modifying namespaces, version and visibility settings, read the guidelines for creating libraries ([see page 59](#)).

General Area of the Dialog Box

Parameter	Description
Namespace	The current namespace of the library is displayed. By default, the namespace of a library primarily is identical to the library name, except another string has been defined in the Project Information when creating the library project. You can edit the namespace. For further information, refer to the Library Manager editor view description (see page 39).
Default library	If a library placeholder is selected in the Library Manager , this field contains the name of the library which replaces the placeholder if no device-specific library is available. Refer to the Placeholder tab (see page 43).

Version Area of the Dialog Box

Configure the version of the library that is used in the project:

Parameter	Description
Specific version	Enter the version or select one from the list that is used in the project. To be used for container libraries (<i>see page 61</i>).
Newest version always	The newest version found in the library repository is used. The modules used can change because a newer version of the library is available. To be used for interface libraries (<i>see page 61</i>). For common libraries, do not specify any version constraints, but use a placeholder reference (<i>see page 66</i>).

Visibility Area of the Dialog Box

These settings are of interest as soon as the library is included and referenced by another library. By default, they are deactivated:

Parameter	Description
Only allow qualified access to all identifiers	If this option is enabled, the usage of the namespace is mandatory.
Publish all IEC symbols to that project as if this reference would have been included there directly	This option is relevant if a library B was added to a library A inside a project that uses library A: If the option is activated, you can access components of library B by using the namespace of library A. Example: <code>NamespaceLibA.ComponentOfLibB</code> If the option is deactivated, you have to type the full namespace path. Example: <code>NamespaceLibA.NamespaceLibB.ComponentOfLibB</code> Also refer to the <i>Library Management</i> chapter (<i>see page 15</i>) for general information on library handling. NOTE: Only activate this option if you want to use container libraries, not containing own modules, but just including other libraries for packaging them. This packaging for example allows you to include multiple libraries in a project at once just by including the container library. In this case, it can be desired to have the particular libraries on top level of the Library Manager of the project. At top level, the modules can be accessed directly. The namespace of the container library can be left out. To achieve this, activate this option.
Hide this reference in the dependency tree	If this option is activated, the current library will not be displayed later when its father library is included in a project. This allows you to include hidden libraries. This requires careful use because if library detected error messages are issued, it may be difficult to identify the causing library.

Try to Reload Library

Overview

If a library included in a project is for any reason not available at the defined path when opening the project in the programming system, an appropriate message will be generated.

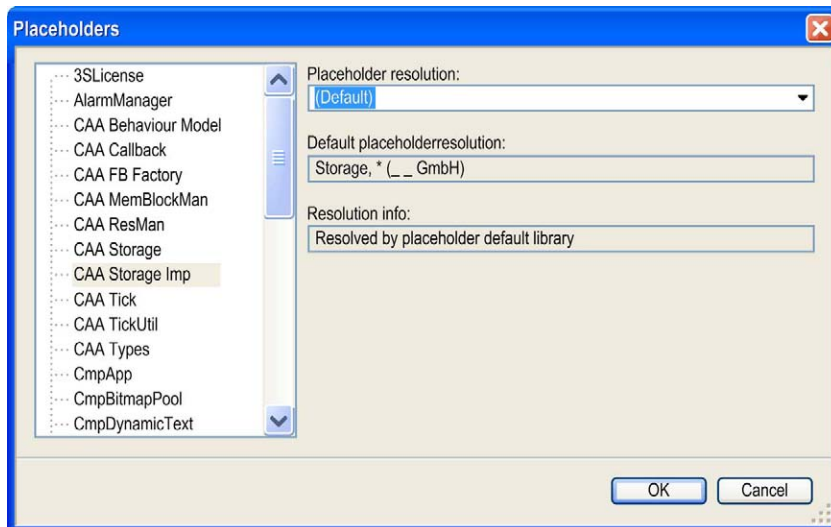
After you have made the library available again, execute the **Try to Reload Library** command in the **Library Manager** editor view when the library entry is selected. The command is also available in the context menu if you right-click a library entry. Thus, the library can be reloaded without the necessity of leaving the project.

Placeholders

Overview

In the **Library Manager** editor view, click the **Placeholders** button to redirect any placeholder (primarily defined via the library profile and the target device) to a different version. This functionality is only available if the feature **Enable simplified library handling** is not active.

NOTE: Carefully consider the possible effects of changing the library referencing. Also consider the guidelines for creating libraries ([see page 59](#)).



Parameter	Description
Placeholder resolution	Redirects the resolution for the placeholder. You can enter the new resolution manually or by selecting (Browse...). This opens a dialog box for choosing one of the available libraries.
Default placeholder resolution	The resolution shown here is applied when the Placeholder resolution parameter is set to (Default).
Resolution info:	Information about where the current placeholder resolution is defined (by device description, by library profile, default library, by placeholder resolution, and so forth).

Library Repository

Overview

NOTE: This functionality is only available if selected in the currently used feature set (**Options** → **Features** → **Predefined feature sets**).

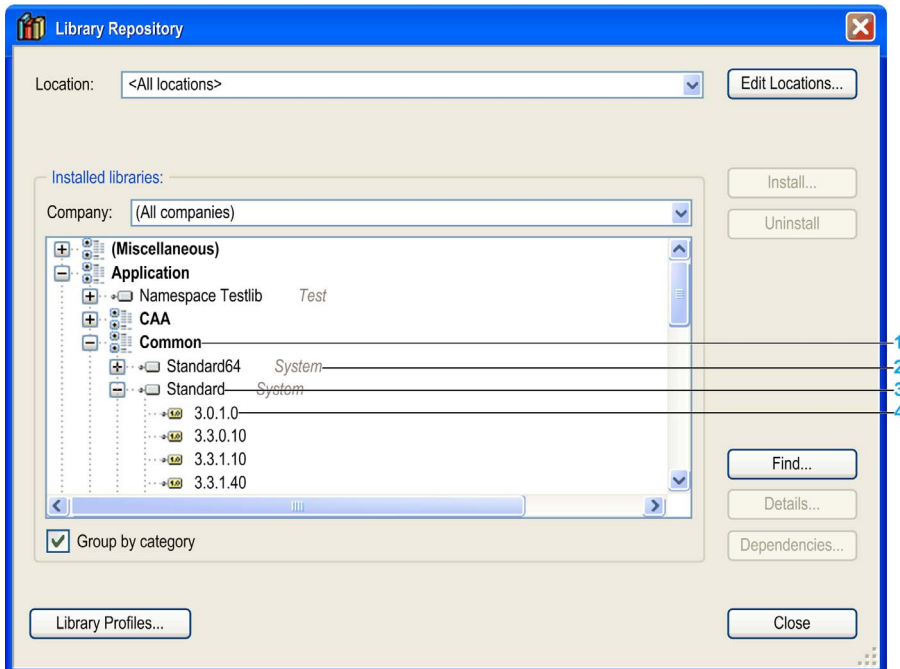
Open the **Library Repository** dialog box via the **Tools** → **Options** → **Library Repository...** command or by clicking the **Library repository** button in the **Library Manager** editor.

To access the **Library Repository** from the SoMachine Central, open the **System Options** dialog box by clicking the **System Options** button in the toolbar or by clicking the **Settings** button in the **Versions** screen. In the **System Options** dialog box, click the **Library Repository** button (see *SoMachine Central, User Guide*).

A library repository is a database for libraries which have been installed on the local system in order to be available for SoMachine projects.

NOTE: A library project **.library*, which is stored in a library repository, cannot be opened there for editing or viewing in the programming system.

Library Repository dialog box



- 1 Category
- 2 Company
- 3 Name
- 4 Version

The dialog box shows the currently defined library **Locations** (repositories) and the currently installed libraries.

It allows you to:

- Add, modify, or remove repositories
- Install and uninstall libraries

The **Library Repository** dialog box contains the following elements:

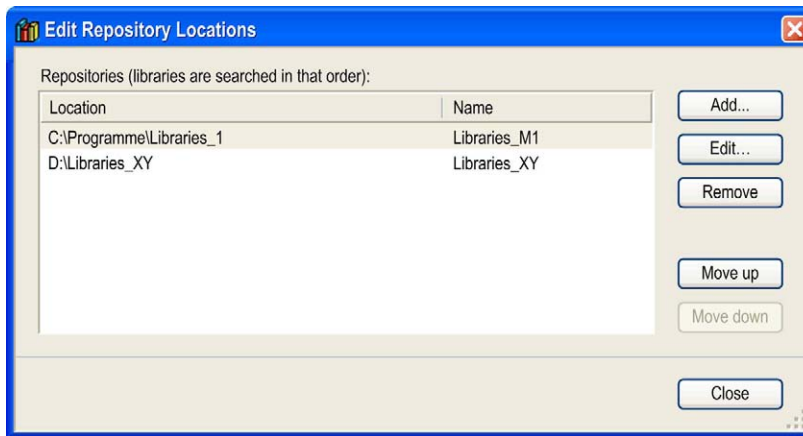
Parameter	Description
Location list	Select a directory on the local system where library files are stored. Select System to display the standard libraries provided by SoMachine in the Installed libraries list. Select User to display only those libraries you defined by yourself in the Installed libraries list.
Company list	Select a name to display only those libraries provided by a primary provider (or group name defined by the primary provider) in the list of Installed libraries . To display all libraries available at the selected Location , select (All companies) from the list.
Installed libraries list	The list shows libraries that are available at the selected Location from the selected Company by their names (title), version number, and company name as provided by the project information of the library.
Group by category option	Activate the option Group by category to sort the Installed libraries list according to the library categories. The category names are displayed as nodes which can be folded or unfolded to show/hide the libraries. Deactivate this option to list the libraries alphabetically. The library categories are defined by external description files <i>*.libcat.xml</i> (<i>see page 62</i>).
Edit Locations... button	Refer to the paragraph <i>Library Locations (Repositories)</i> (<i>see page 50</i>).
Install... / Uninstall buttons	Refer to the paragraph <i>Library Installation and Uninstallation</i> (<i>see page 52</i>).
Find... button	Refer to the paragraph <i>Find Libraries</i> (<i>see page 52</i>).
Details... / Dependencies... buttons	Refer to the paragraph <i>Further Information on Particular Libraries</i> (<i>see page 54</i>).
Library Profiles... button	Refer to the paragraph <i>Library Profiles</i> (<i>see page 54</i>).

Library Locations (Repositories)

You can use several repositories to manage libraries. The defined repositories are shown in the **Location** list. By default, it contains the entries **System** for standard libraries provided by SoMachine and **User** for those libraries you defined by yourself.

To edit the path or name of a repository, click the **Edit Locations...** button.

The **Edit Repository Locations** dialog box displays:



The **Edit Repository Locations** dialog box contains the following elements:

Parameter	Description
Repositories list	Lists the defined locations. They are searched later for a library in the given order from up to down.
Move up / Move down buttons	Modifies the order of the libraries. The selected location is moved up or down in the list.

Consider that the setting **Location: <All locations>** displays the libraries of the previously defined locations. No installation is possible in this view.

Defining a New Repository or Modifying Path and Name of an Existing Repository

Click the **Add...** button of the **Edit Repository Locations** dialog box to add a new repository. The **Repository Location** dialog displays.

To modify an existing repository, select the respective entry in the **Edit Repository** dialog box, and click the **Edit** button. The **Repository Location** dialog box displays.

The **Repository Location** dialog box contains the following elements:

Parameter	Description
Location text box	Enter the path of the new repository or edit the path of the selected repository. To browse for a folder or to create a new folder, click the ... button. Verify that the selected folder is empty.
Name text box	Enter a symbolic name for the location, for example Libraries for System1 .

NOTE: The folder that is selected as a repository folder has to be empty. The **System** repository is not editable. This is indicated by the entry being displayed in italic font.

Removing an Existing Repository

To remove a repository, select it in the **Edit Repository Locations** dialog box, and click the **Remove** button. A message is displayed asking you whether just the entry in the list should be deleted, or if also the folder containing the library files should be removed from the file system.

Library Installation and Uninstallation

You can only include those libraries in a project that are installed on the local system (**Library Repository**). As a precondition for the installation, it must be assigned at least a **Title**, **Version** info, and a **Company** name in the **Summary** tab of the **Project Information** dialog box.

To install a library, select the repository to which the library should be added in the **Library Repository** dialog box, and click the **Install...** button.

The **Select Library** dialog box opens. This is a standard dialog box for browsing for a file. By default, the **Files of type** filter is set to **Compiled library files**. You can change the filter to **Library files**, or to **All files**.

Select the desired library and click **Open**. The library is added to the list of currently installed libraries in the **Library Repository** dialog box.

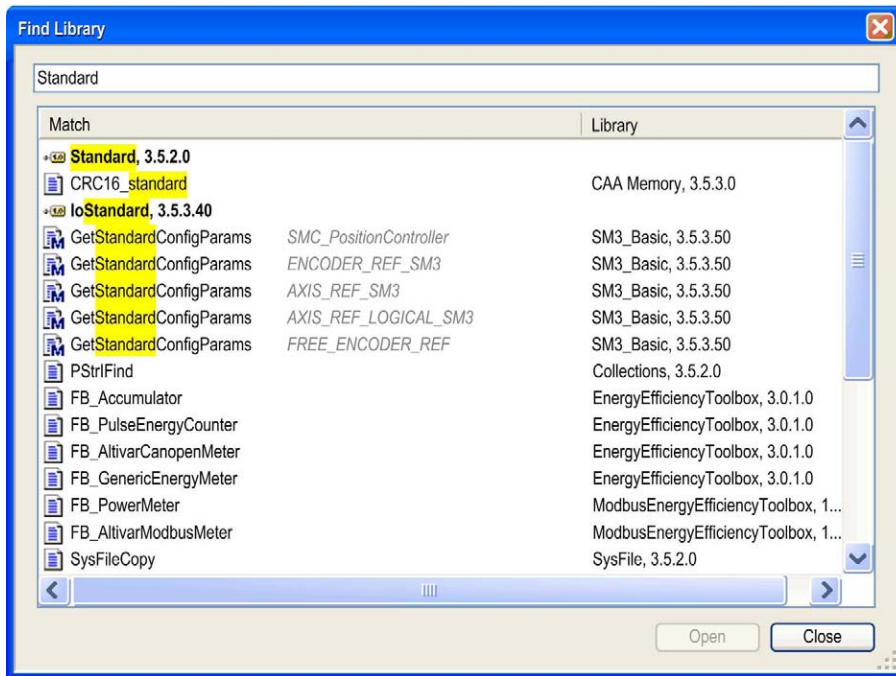
If you select a library which cannot be installed because it does not provide the obligate **Project Information** (title, version, company), an appropriate message displays.

To uninstall a library, select it in the list of currently installed libraries in the **Library Repository** dialog box, and click the **Uninstall** button.

Find Libraries

To search for a library in the specified storage location, click the **Find** button in the **Library Repository** dialog box. The **Find Library** dialog box opens. It allows you to search for function blocks and the corresponding libraries.

Find Library dialog box



The Find Library dialog box contains the following elements:

Parameter	Description
Find what text box	Enter the search string for the function block and the corresponding libraries. In addition, you can also enter the wildcards * and ?.
Match case option	Activate this option to execute a case-sensitive search referring to the search string.
Include comments into search option	Activate this option to scan not only the function block name, but also the comments of the function blocks.

Further Information on Particular Libraries

To display further details for the currently selected library, select the line of the library containing the version, and click the **Details...** button. A **Details** dialog box displays, providing further information from the **Project Information** of the library. To get even more detailed information, click the **More...** button.

To display the dependencies concerning other libraries for the currently selected library, select the line of the library containing the version and click the **Dependencies...** button. A **Dependencies** dialog box displays, listing those libraries which are included in the selected library with **Title**, **Version**, and **Company** information.

References working via placeholders are listed according to the following syntax:

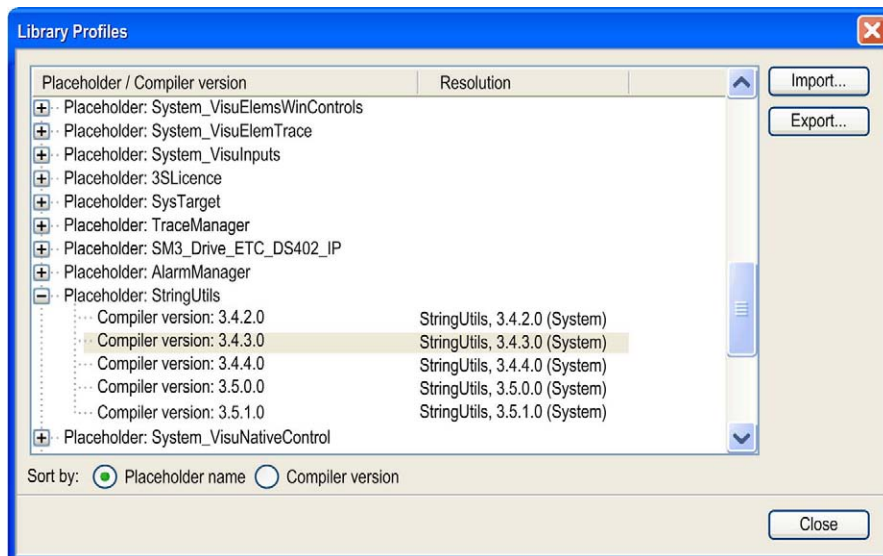
#<placeholder name>.

For details on library placeholders, refer to the *Placeholder Mechanism* chapter (*see page 22*).

Library Profiles

A library profile defines the library version that is used to resolve a library placeholder when a certain compiler version is set for the project. To open the **Library Profiles** dialog box, click the **Library Profiles...** button in the **Library Repository** dialog box.

Library Profiles dialog box



The **Library Profiles** dialog box contains the following elements:

Parameter	Description
Export button	Select 1 or multiple of the listed Placeholder entries or just a single Compiler version entry below a Placeholder entry. Click this button to export the assignments between placeholder and library version to an XML file with the extension <i>*.libraryprofile</i> .
Import button	Click this button to import an XML file with the extension <i>*.libraryprofile</i> . If the import provides already available placeholder entries, you are asked whether the existing ones should be overwritten.

Consider that you can additionally define placeholder resolutions by the currently set target device and even by locally specified relocations in the **Placeholder** tab of the **Add Library** dialog box (*see page 41*).

Version Mapping Tab

Overview

The **Version mapping** tab of the **Library Manager** allows you to edit the version mapping for forward compatible libraries (FCL) (*see page 24*).

Version mapping tab of the **Library Manager**:

Library Manager x

Libraries Version mapping

Environment information
Used device description: TM258LF42DT4L (Schneider Electric), Version 2.0.40.19

Edit the version mapping for forward compatible libraries. Automatic

Title	Manufacturer	Referenced directly	Configured version	Hints
(Library manager)				Shows all depen
Altivar Library	Schneider Electric	Yes	4.4.1.0	
EnergyEfficiencyToolbox	Schneider Electric	Yes	3.0.1.0	Some requireme
Integrated Lexium Library	Schneider Electric	Yes	3.2.0.0	
IoDrvModbusSerial	Schneider Electric	Yes	3.5.3.2	
Lexium 32i Library	Schneider Electric	Yes	1.2.1.0	
Lexium Library	Schneider Electric	Yes	3.5.1.0	
M258 PLCSystem	Schneider Electric	Yes	1.0.3.1	
M2xx Communication	Schneider Electric	No	2.0.2.2	
MachineEnergyDashboard	Schneider Electric	Yes	2.0.1.0	
ModbusEnergyEfficiencyToolbox	Schneider Electric	Yes	1.0.1.0	
PLCCommunication	Schneider Electric	Yes	1.0.3.2	
Relocation Table	Schneider Electric	Yes	1.0.1.0	
Toolbox	Schneider Electric	Yes	2.1.0.0	

Details of the selected library

Dependency	Required version	Actual version
[-] Accepted device descriptions		
... TM258LD42DT (Schneider Electric)	2.0.40.0.4	
... TM258LD42DT4L (Schneider Electric)	2.0.40.0.4	
... TM258LF42DT (Schneider Electric)	2.0.40.0.4	
... TM258LF42DT4L (Schneider Electric)	2.0.40.0.4	2.0.40.19
... TM258LF42DR (Schneider Electric)	2.0.40.0.4	
... TM258LF66DT4L (Schneider Electric)	2.0.40.0.4	

The **Version mapping** tab consists of two tables:

- The upper table lists the forward compatible libraries that are included in the SoMachine project. It also lists legacy libraries if a newer, forward compatible version of these libraries exists. Legacy libraries are libraries that have been created with a SoMachine version 3.1 or lower and that are not yet forward compatible. You can recognize these libraries by the entry **Legacy** in the **Configured version** column of the table.
- The lower part of the table provides further information on the library selected in the upper table. It shows dependencies on other forward compatible libraries and device descriptions.

Changing the Version of a Library

The **Version mapping** tab of the **Library Manager** allows you to edit the version mapping of the libraries in different ways:

- You can automatically perform the version mapping for all libraries listed in the table by clicking the **Automatic** button. This has the following effects:
 - The forward compatible libraries are updated to the newest forward compatible library version available.
 - The legacy libraries are changed into forward compatible libraries and are updated to the newest forward compatible library version available.
- You can automatically perform the version mapping for one specific library. To achieve this, right-click the library and execute the command **Edit version mapping (selected library)**.
- You can edit the version of each library manually. To achieve this, click the **Configured version** column to open a list displaying the available versions. Select the desired version from the list.

Library with Unresolved Dependencies

FCL libraries with unresolved dependencies are displayed with a red background in the upper table. Select this library to display further information about required dependencies and compatible versions in the lower table. Unresolved dependencies are marked red in the lower table. For example, if no compatible device version is used in the project, then the line **Accepted device description** is marked red. If a required library dependency is missing, then the line **library dependencies** and the unfulfilled library dependency are marked red.

Unsupported Devices

If a device is listed in the device dependencies details table as compatible, but is not installed in the version of SoMachine that you are using, then the text **Unsupported device** is displayed in the line, together with the device ID.

Chapter 4

Create Your Own Library

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
General Information	60
Step 1: Setup the Project	61
Step 2: Fill-in the Project Information	62
Step 3: Reference Other Libraries, if Needed	65
Step 3.1: Use Placeholder References	66
Step 4: Design and Program the Library Modules	67
Step 5: Design the Interface	68
Step 6: Implement Error Handling Routines	69
Step 7: Set up a Reasonable Deployment (Usage Restriction)	70

General Information

Overview

For compatibility reasons, consider the guidelines described in the following chapters when creating a library.

When creating a library, consider the following main items:

- Define a library name that is consistent with the contents of your library (required).
- Use a familiar and as far as possible consistent project structure (optional).
- Fill in the **Project Information** (required).
- Select a unique library namespace (required).
- Use the right way for referencing other libraries (required).
- Design usable external and internal interfaces (required).
- Implement an error handling routine (required).
- Use the appropriate method (protection) for deployment (required).
- Follow a naming convention for code that easier to read and debug (optional).

When creating a library project, follow the steps described in the following chapters.

Step 1: Setup the Project

Overview

In SoMachine Central, create a new library project in the **Get started** screen by executing the commands **New Project** → **New Library**. Define the **Name** and **Details** in the **New Library** dialog box as described in the SoMachine Central User Guide (*see SoMachine Central, User Guide*).

Step 2: Fill-in the Project Information

Overview

Specify information on the project in the **Project** → **Project Information** dialog box. This dialog box consists of several tabs. Pieces of information required for creating your own library are described here.

Summary Tab

Specify the following information in the **Summary** tab of the **Project Information** dialog box:

Parameter	Description
Company	Obligatory: Enter a company for the project.
Title	Obligatory: Enter a name for the project.
Version	Obligatory: Enter a version for the project.
Default namespace	Allows you to make each symbol unique, even if the same symbol name is used in different libraries If you do not define a namespace, the library name is used as namespace. It is better to define a unique one. After the library has been included in a project, you can still modify the namespace in the Properties dialog box of the Library Manager - if necessary. For enforcing the namespace, see the separate section in this chapter.
Library Categories	Provides helpful sorting of the entries in the Library Repository and Library Manager If possible specify another category than Miscellaneous from the following categories: <ul style="list-style-type: none"> ● Application libraries which must be inserted explicitly in a project. Designed for direct access from end-user code. ● Internal libraries which are automatically inserted in a project and normally should not be inserted or removed manually. ● System libraries depending directly on the runtime system or implement parts of it. They should not be inserted directly in the project, but be referenced by other libraries. They are not intended to be included in the user application directly. (No resource management is provided like, for example, deleting of resources on the controller in order to get serial interfaces closed before a further download is done).
Author	Enter the name of the author of this project.
Description	Enter a short description of the library content.
Automatically generate POUs for property access	Not used. NOTE: Do not use within libraries as it may provoke application errors.

Enforcing the Namespace

It can be enforced that the namespace must be preceding in any case when accessing library modules or variables, from the application as well as from another library. For this purpose, set the property `LanguageModelAttribute (Text) := 'qualified-access-only'` in the **Properties** tab.

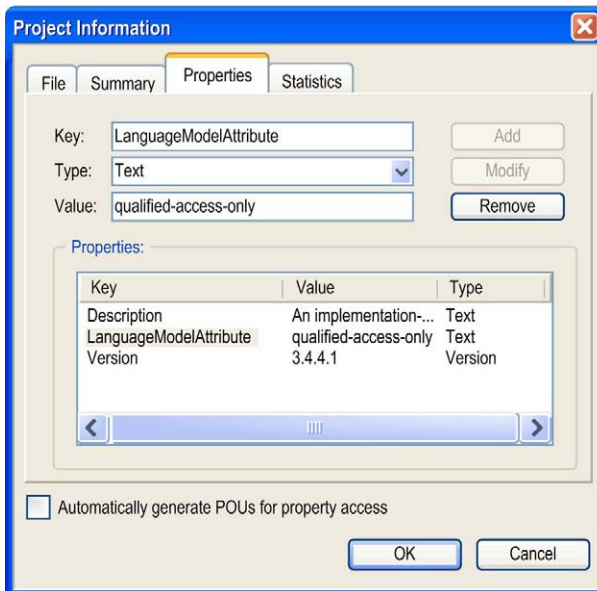
Key name: `LanguageModelAttribute`

Type: `Text`

Value: `qualified-access-only` (see figure below).

Compile errors are detected if a library module or variable is used without preceding namespace.

Properties tab: Settings for enforcing the namespace:



Properties Tab

List of configurable properties:

Property	Type	Value	Description
Company	Text	<company name>	Serves to filter the libraries in the Add Library dialog box
Title	Text	<library name> #	–
Version	Text	<library version>	–
Released	Boolean	TRUE FALSE	Library must not be modified after release
Author	Text	<author name>	Author of the current library version
DefaultNamespace	Text	<namespace>	Worldwide unique namespace prefix
Description	Text	<description>	Short description on the scope of the library
Placeholder	Text	<placeholder>	–
IsContainerLibrary	Boolean	TRUE FALSE	This library follows the rules for a container library
IsInterfaceLibrary	Boolean	TRUE FALSE	This library follows the rules for a container library
LanguageModelAttribute	Text	'qualified-access-only'	Symbols of this library can only be accessed via the namespace prefix
IsEndUserLibrary	Boolean	TRUE FALSE	This library is especially designed regarding the needs of an end user

Step 3: Reference Other Libraries, if Needed

Overview

When you include other libraries, define their visibility and the version to be used.

Visibility

In the **Properties** (*see page 45*) of each referenced library, define the behavior when being inserted in a project along with its parent library.

Consider whether a referenced library should be hidden (not visible in the **Library Manager**), or if it is useful to create a container library (a library project containing library references):

Hidden library	Deactivate the visibility of the library in the Library Manager below the parent library by selecting the option Hide this reference in the dependency tree in the Properties dialog box (<i>see page 45</i>). Thus, hidden libraries can be available in a project.
Container library	A container library (<i>*_Cnt.library</i>) is a library project which does not define modules, but references other libraries. With container libraries, the access to the modules of these libraries is simplified. Create a container library if you want to include a set of libraries in a project at one time by including that container library. In this case, you can simplify the access to the modules of these libraries by making them top-level libraries. This allows you to leave out the namespace of the container library in the access path. To achieve this, select the option Publish all IEC symbols to that project as if this reference would have been included there directly in the Properties dialog box (<i>see page 45</i>). Select this option only for a container library.

Version Constraints

- Interface libraries are referenced with the constraint **newest**. To achieve this, select the * instead of a specific version when you include the library via the **Add Library** dialog box.
- Container libraries are referenced with a **version** constraint. To achieve this, select a specific version of the library in the **Add Library** dialog box.
- Common libraries are referenced by a placeholder reference. This provides consistent relationships between different libraries. An update of an indirectly referenced library does not require modifications of each affected library but a modification of the placeholder definition (*see page 48*).

Step 3.1: Use Placeholder References

Overview

For referencing common libraries, use placeholder references.

Use library placeholders to make a project compatible for interchangeable target devices or for different compiler versions. When adding a library to a project, consider using a placeholder instead of a specific library version. Then the currently required version of a library is referenced, according to the currently valid definition of the placeholder resolution.

For the final, effective resolution definition, the following locations are regarded in ascending order. Item 3 has top priority:

1. The library profile, by default provided with the programming system installation, defining the resolution of a placeholder depending on the compiler version. For details, refer to the *Library Profiles* description (*see page 54*).
2. The device description of the currently set target, defining the device-specific placeholder resolution (if no device is available, the default resolution specified for the placeholder is performed).
3. The **Placeholders** dialog box (*see page 48*) that is accessible from the **Library Manager**, where you can modify the resolution of the placeholder specifically for the current project.

NOTE: This is the final and effective resolution. You can only edit it if the option **Enable simplified library handling** is not activated. Modify it only after careful consideration.

Step 4: Design and Program the Library Modules

Consideration

NOTE: Use the proposed POUs structure as provided with the library template and do not forget to add sufficient documentation of the library modules.

Step 5: Design the Interface

Overview

Basically distinguish 2 types of library interfaces:

Interface	Description
External interfaces (user interfaces)	Are accessed by the end-user application. Use a reduced set of parameter types so that other programmers need not work with troublesome or complicated data types like pointers or the <code>ADR ()</code> operator.
Internal interfaces	All other interfaces

Interfaces should be optimized for the use with CFC.

If applicable, reuse the common behavior model to build function block interfaces for end users.

Mark libraries especially designed for the usage by end users with the property `IsEndUserLibrary` (refer to the library type **End-User Library**, described in the chapter *Step 1: Set up the Project*).

Step 6: Implement Error Handling Routines

Overview

Consider the following for creating efficient error handling routines:

- Only error codes that you have documented should be returned.
- Do not simply return error codes that have passed through from other libraries, but document them in your routines and return a proper code as the documented source.

Step 7: Set up a Reasonable Deployment (Usage Restriction)

Overview

In order to help protect the library source, set up an appropriate protection and access rights for the library. Then, use an appropriate type of usage restriction.

- Supply the library only in compiled format (**.compiled-library*). Therefore, the project source cannot be restored. For this purpose, use the command **Save Project as...** in the SoMachine Central toolbar (*see SoMachine Central, User Guide*).
- Increase the restrictions of the access rights for user **Everyone**. For further information, refer to the chapter *User Management and Access Rights* (*see SoMachine, Programming Guide*).
- Reserve the right for viewing the content of the objects **Project Information** and **Library Manager**. For further information, refer to **Access Control** (*see SoMachine, Menu Commands, Online Help*).
- Restrict the usage of the library to certain platforms or devices: you can implement a check code which has the effect that the library will only work if the appropriate device is used.

Chapter 5

Libraries Overview

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Schneider Electric Libraries	72
Other Libraries Used in SoMachine	90

Schneider Electric Libraries

Introduction

All libraries from Schneider Electric are listed below.

The following table explains how to use the tables describing the Schneider Electric libraries below:

Category	The category which this library belongs to, as shown in the Library Manager dialog box.
Namespace	The default namespace of the library for accessing functions of the library. NOTE: Best practice is to always use the default namespace as the namespace used in your application. If <i>qualified-access-only</i> is listed after the default namespace, this indicates that the usage of the namespace in your application is mandatory. For more information, refer to the General Description of Libraries (see page 11).
Reference Library as	The way to add a reference to the library: <ul style="list-style-type: none"> ● <i>PH</i> indicates the inclusion of the library via placeholder mechanism (see page 22). The corresponding placeholder name to be used for this library is listed. ● <i>direct version</i> indicates the inclusion of the library via direct version mechanism (see page 18). ● <i>FCL</i> indicates that this is a forward compatible library (see page 24).
USER	<ul style="list-style-type: none"> ● <i>YES</i>: Functions and function blocks of the library can be used in the user application (program). ● <i>NO</i>: Intended only for specific system or administrative functions. Typically, these functions and function blocks have no user documentation.

For more information on how to add a library to a SoMachine project, refer to Add Libraries ([see page 41](#)).

For more information on the use of libraries and the reference mechanism refer to Library Management in SoMachine ([see page 15](#)).

Category: (Miscellaneous)

The following table describes the library available in the **(Miscellaneous)** category:

Name	Namespace	Reference Library as	Description	USER
FeatureNotSupported	FeatureNotSupported	direct version	Virtual empty library used by the system	NO

Category: Application

The following table describes the library available in the **Application** category:

Name	Namespace	Reference Library as	Description	USER
PD_ETest refer to PD_ETest Library Guide (<i>see SoMachine, PD_ETest, Library Guide</i>)	PD_ETest	direct version (FCL)	Contains functions used in the ETEST framework.	NO
SE_LMC_Utility	SE_LMC_Utility	PH (FCL)SE_LMC_Utility	Utility functions used by the PROFIBUS DP slave library	NO

NOTE: For the other library descriptions, see the categories **Communication**, **Solution** and **Util**.

Category: Communication

The following table describes the libraries available in the **Communication** category:

Name	Namespace	Reference Library as	Description	USER
EEmailHandling refer to EEmailHandling Library Guide (<i>see SoMachine, EEmailHandling, Library Guide</i>)	SE_Email (qualified-access-only)	direct version (FCL)	The library allows your controller to send an email to one or several recipients with the possibility to customize the content. The protocol type used is TCP as standard for email traffic. It is also possible to receive or delete emails from a server using the Post Office Protocol 3 (POP3). NOTE: The communication is implemented using the TcpUdpCommunication library.	YES
EtherNet/IP Explicit Messaging refer to SoMachine Industrial Ethernet User Guide (<i>see SoMachine Industrial Ethernet, User Guide</i>)	EIPXM (qualified-access-only)	direct version (FCL)	Explicit Messaging over EtherNet/IP to communicate with generic devices (e.g. cameras) for which SoMachine does not offer a device integration.	YES

Name	Namespace	Reference Library as	Description	USER
EtherNet/IP Remote Adapter refer to SoMachine Industrial Ethernet User Guide (see <i>SoMachine Industrial Ethernet, User Guide</i>)	EIPRA (qualified-access-only)	direct version (FCL)	Used by EtherNet/IP device objects to support user parameter and connection handling.	NO
EtherNet/IP Scanner refer to SoMachine Industrial Ethernet User Guide (see <i>SoMachine Industrial Ethernet, User Guide</i>)	EIPSC	direct version (FCL)	Function blocks to establish and close CIP connections and to build Explicit Messaging request over EtherNet/IP.	YES
FtpRemoteFileHandling refer to FtpRemoteFileHandling Library Guide (see <i>SoMachine, FtpRemoteFileHandling, Library Guide</i>)	SE_FTP (qualified-access-only)	direct version (FCL)	The library offers FTP (File Transfer Protocol) client functionalities to a controller to access and handle files remotely from and to an FTP server. NOTE: The communication is implemented using the TcpUdpCommunication library.	YES
IoDrvASI refer to AS-Interface (see <i>Modicon M238 Logic Controller, Programming Guide</i>)	SEN_ASI	PH (FCL) SE_ASi	AS-Interface bus management functions	YES
IoDrvDistributedIo	SEN_DIO	PH SE_Distributed_IO	Bus management functions for distributed I/O on CANopen	NO
IoDrvTM5PCDPS	SEN_PROFIBUS	PH (FCL) IoDrvTM5PCDPS	PROFIBUS DP slave library	NO
IoDrvTM4PDPS1	IoDrvTM4DPS1Lib	direct version (FCL)	M241 PROFIBUS DP slave library	NO
M2xx Communication refer to Functions to Get/Set Serial Line Configuration (see <i>Modicon M238 Logic Controller, Programming Guide</i>)	SEN_COM	PH (FCL) SE_M2XXCommunication	Serial Line port configuration getting and setting on M238, M258 and LMC058 controllers	YES

Name	Namespace	Reference Library as	Description	USER
Modem refer to Modem Library <i>(see SoMachine, Modem Functions, Modem Library Guide)</i>	SE_MOD	PH (FCL) SE_Modem	Modem configuration on M238, M258 and LMC058 controllers	YES
PLCCommunication refer to PLCCommunication Library <i>(see SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)</i>	SEN	PH (FCL) SE_PLCCommunication	Management of explicit data exchanges between controller and devices through Modbus or ASCII protocols NOTE: This library is not supported by the ATV IMC Drive Controller.	YES
SE_NetVarUdp refer to SE_NetVarUdp Library <i>(see SoMachine, Network Variable Configuration, SE_NetVarUdp Library Guide)</i>	NetVarUdp	direct version (FCL)	Data exchange between controllers within a network via network variables.	YES
SnmpManager refer to SnmpManager Library Guide <i>(see SoMachine, SnmpManager, Library Guide)</i>	SE_SNMP (qualified-access-only)	direct version (FCL)	The library implements a manager for the Internet-standard protocol SNMP (Simple Network Management Protocol). The SNMP manager allows your controller to collect and organize information about managed devices on IP (Internet Protocol) networks and to modify this information in order to modify the configuration of the device. NOTE: The communication is implemented using the TcpUdpCommunication library.	YES

Name	Namespace	Reference Library as	Description	USER
SqlRemoteAccess refer to SqlRemoteAccess Library Guide <i>(see SoMachine, SqlRemoteAccess, Library Guide)</i>	SE_SQL (qualified-access-only)	direct version (FCL)	The library provides SQL (Structured Query Language) client function blocks that allow your controller to connect to an SQL database in order to run SQL queries for reading and writing data. NOTE: The communication is implemented using the TcpUdpCommunication library.	YES
ModbusTCPIOScanner	SE_IOS	direct version (FCL)	Provides Modbus TCP IOScanner function blocks..	NO
TcpUdpCommunication refer to TcpUdpCommunication Library <i>(see SoMachine, TcpUdpCommunication, Library Guide)</i>	TCPUDP	direct version (FCL)	Core functionality for implementing communication protocols based on TCP (Client and Server) or UDP (including broadcast and multicast) via Internet Protocol Version 4 (IPv4).	YES
TimeSync refer to TimeSync Library Guide <i>(see SoMachine, TimeSync, Library Guide)</i>	TIMS (qualified-access-only)	direct version (FCL)	The library offers services related to time synchronization. With the SNTP client, the clock of the controller can be synchronized with time servers located in the same network via SNTP (Simple Network Time Protocol). NOTE: The communication is implemented using the TcpUdpCommunication library.	YES

Category: Controller

The following table describes the libraries available in the **Controller** category:

Name	Namespace	Reference Library as	Description	USER
DataLogging refer to DataLogging Library (see <i>SoMachine, Data Logging Functions, DataLogging Library Guide</i>)	SEDL	PH (FCL) SE_DataLogging	Data logging management of controllers that support file management operations	YES

The following table describes the libraries available in the **Controller → ATV IMC** category:

Name	Namespace	Reference Library as	Description	USER
ATV- IMC HSC refer to High Speed Counting (see <i>Altivar ATV IMC Drive Controller, High Speed Counting, ATV IMC HSC Library Guide</i>)	SEC_HSC	PH (FCL) SE_HSC	ATV IMC High Speed Counting management	YES
ATV-IMC PLCSystem refer to PLCSystem Library (see <i>Altivar ATV IMC Drive Controller, System Functions and Variables, ATV-IMC PLCSystem Library Guide</i>)	SEC	PH (FCL) SE_PLCSysSystem	ATV IMC System Functions and Variables	YES
ATV-IMC SysLib V2.3	SEC_SL23	PH (FCL) SE_SysLibv23	Functions and Function Blocks for compatibility with Controller Inside applications (CoDeSys v2.3)	YES
ATV-IMC UserLib refer to ATV IMC - UserLib Library Guide (see <i>Altivar ATV IMC Drive Controller, ATV IMC UserLib Library Guide</i>)	SEC_USER	PH (FCL) SE_UserLib	ATV IMC - ATV71 host interface management	YES

The following table describes the libraries available in the **Controller → LMC058** category:

Name	Namespace	Reference Library as	Description	USER
IoDrvTM5SEAI SG	SEC_TM5SEAI SG	PH IoDrvTM5SEAI SG	TM5SEAI SG device library provides calibration, tarring and calibrated measure services	YES
LMC058 Expert IO <ul style="list-style-type: none"> ● refer to High Speed Counting (<i>see Modicon LMC058 Motion Controller, High Speed Counting, LMC058 Expert I/O Library Guide</i>) ● refer to Pulse Width Modulation (<i>see Modicon LMC058 Motion Controller, Pulse Width Modulation, LMC058 Expert I/O Library Guide</i>) 	SEC_EXP	PH (FCL) SE_ExpertIO	LMC058 Expert I/O management	YES
LMC058 Motion refer to High Speed Counting (<i>see Modicon LMC058 Motion Controller, High Speed Counting, LMC058 Expert I/O Library Guide</i>)	SEC_MC	PH (FCL) SE_Motion	Functions used to get present motion axis value and to reset the detected error.	YES
LMC058 PLCSystem refer to PLCSystem Library (<i>see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 PLCSystem Library Guide</i>)	SEC	PH (FCL) SE_PLCSys tem	LMC058 System Functions and Variables	YES
Relocation Table	SEC_RELOC	PH (FCL) SE_RelocationTable	System library for Relocation Table (<i>see Modicon LMC058 Motion Controller, Programming Guide</i>) management.	NO
Ethernet Util	SEC_ETH_UTIL	PH (FCL) SE_EthernetUtil	Library for Ethernet utilities.	YES

The following table describes the libraries available in the **Controller → M238** category:

Name	Namespace	Reference Library as	Description	USER
M238 ASi Interface	SEC_ASIITF	PH (FCL) SE_ASiInterface	System library for AS-Interface bus management	NO
M238 HSC refer to High Speed Counting <i>(see Modicon M238 Logic Controller, High Speed Counting, M238 HSC Library Guide)</i>	SEC_HSC	PH (FCL) SE_HSC	M238 High Speed Counting management	YES
M238 PLCSystem refer to PLCSystem Library <i>(see Modicon M238 Logic Controller, System Functions and Variables, M238 PLCSystem Library Guide)</i>	SEC	PH (FCL) SE_PLCSytem	M238 System Functions and Variables	YES
M238 PTO/PWM refer to Pulse Train Output / Pulse Width Modulation <i>(see Modicon M238 Logic Controller, Pulse Train Output, Pulse Width Modulation, M238 PTO/PWM Library Guide)</i>	SEC_PTOPWM	PH (FCL) SE_PTOPWM	M238 PTO and PWM management	YES

The following table describes the libraries available in the **Controller → M241** category:

Name	Namespace	Reference Library as	Description	USER
M241 HSC refer to the M241 HSC Library Guide (<i>see Modicon M241 Logic Controller, High Speed Counting, HSC Library Guide</i>)	SEC_HSC	direct version (FCL)	M241 High Speed Counting management	YES
M241 PLCSystem refer to the M241 PLCSystem Library Guide (<i>see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide</i>)	SEC	PH (FCL) SE_PLCSysyem	M241 System Functions and Variables	YES
M241 PTO/PWM refer to M241 PTO/PWM Library Guide (<i>see Modicon M241 Logic Controller, PTO/PWM, Library Guide</i>)	SEC_PTO/PWM	direct version (FCL)	M241 PTO and PWM management	YES

The following table describes the libraries available in the **Controller → M251** category:

Name	Namespace	Reference Library as	Description	USER
M251 PLCSystem refer to M251 PLCSystem Library Guide (<i>see Modicon M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide</i>)	SEC	PH (FCL) SE_PLCSysyem	M251 System Functions and Variables	YES

The following table describes the libraries available in the **Controller → M258** category:

Name	Namespace	Reference Library as	Description	USER
IoDrvTM5SEAI SG	SEC_TM5SEAI SG	PH IoDrvTM5SEAI SG	TM5SEAI SG device library provides calibration, tarring and calibrated measure services	YES
M258 Expert IO <ul style="list-style-type: none"> ● refer to High Speed Counting (<i>see Modicon M258 Logic Controller, High Speed Counting, M258 Expert I/O Library Guide</i>) ● refer to Pulse Width Modulation (<i>see Modicon M258 Logic Controller, Pulse Width Modulation, M258 Expert I/O Library Guide</i>) 	SEC_EXP	PH (FCL) SE_ExpertIO	M258 Expert I/O management	YES
M258 PLCSystem refer to PLCSystem Library (<i>see Modicon M258 Logic Controller, System Functions and Variables, M258 PLCSystem Library Guide</i>)	SEC	PH (FCL) SE_PLCSytem	M258 System Functions and Variables	YES
Ethernet Util	SEC_ETH_UTIL	PH (FCL) SE_EthernetUtil	Library for Ethernet utilities.	YES

The following table describes the libraries available in the **Controller → XBTGC** category:

Name	Namespace	Reference Library as	Description	USER
XBTGC HSC refer to High Speed Counting (see <i>Magelis XBTGC, XBTGC HMI Controller, High Speed Counting, XBTGC HSC Library Guide</i>)	SEC_HSC	PH (FCL) SE_HSC	XBTGC High Speed Counting management	YES
XBT PLCSystem refer to PLCSystem Library (see <i>Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide</i>)	SEC	PH (FCL) SE_PLCSys- tem	XBTGC System Functions and Variables	YES
XBT PTO/PWM refer to Pulse Train Output / Pulse Width Modulation (see <i>Magelis XBTGC HMI Controller, Pulse Train Output, Pulse Width Modulation, XBTGC PTO/PWM Library Guide</i>)	SEC_PTO/PWM	PH (FCL) SE_PTO/PWM	XBTGC PTO and PWM management	YES

The following table describes the libraries available in the **Controller → HMISCU** category:

Name	Namespace	Reference Library as	Description	USER
HMISCU HSC refer to HMISCU HSC Library (see <i>Magelis SCU, HMI Controller, HSC Library Guide</i>)	HMISCU_HSC	PH (FCL) HMISCU_HSC	HMISCU High Speed Counting management	YES
HMISCU PLCSystem refer to HMISCU PLCSystem Library (see <i>Magelis SCU, HMI Controller, PLCSystem Library Guide</i>)	SEC	PH (FCL) SE_PLCSys- tem	HMISCU System Functions and Variables	YES
HMISCU PTO/PWM refer to HMISCU PTO/PWM Library	HMISCU_PTO/PWM	PH (FCL) HMISCU_PTO/PWM	HMISCU PTO and PWM management	YES

Name	Namespace	Reference Library as	Description	USER
SE_ModbusTCP_Slave refer to Magelis SCU - HMI Controller - Programming Guide (<i>see Magelis SCU, HMI Controller, Programming Guide</i>)	MTS	direct version (FCL)	Provides a function block to manage ModbusTCP request from any clients to the HMISCU controller.	YES

Category: Devices

The following table describes the libraries available in the **Devices** category:

Name	Namespace	Reference Library as	Description	USER
Altivar Library refer to <i>Altivar Library Guide</i>	SE_ATV	PH (FCL) SE_Altivar	Control of ATV variable speed drives.	YES
CANmotion Lexium Library refer to <i>CANmotion Lexium Library Guide</i>	SEM_LXM_SM	direct version (FCL)	Functions SM_Servo_Startup and SM_Stepper_Startup and associated Visualization components to ease motion drives and steppers commissioning.	YES
GMC Independent Altivar refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GIATV (qualified-access-only)	direct version (FCL)	Altivar specific function blocks, for General Motion Control (GMC) of independent axes via Industrial Ethernet.	YES
GMC Independent Base refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GIB (qualified-access-only)	direct version (FCL)	Base implementations (reserved for internal use only), for General Motion Control (GMC) of independent axes via Industrial Ethernet.	NO
GMC Independent Base Device refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GIBD (qualified-access-only)	direct version (FCL)	Core device implementations (reserved for internal use only), for General Motion Control (GMC) of independent axes via Industrial Ethernet.	NO

Name	Namespace	Reference Library as	Description	USER
GMC Independent CANopen refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GICAN (qualified-access-only)	direct version (FCL)	PLCopen Motion Control function blocks, for General Motion Control (GMC) of independent axes via CANopen.	NO
GMC Independent EtherNetIP refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GIEIP (qualified-access-only)	PH GMC Independent EtherNetIP	EtherNet/IP backend (reserved for internal use only), for General Motion Control (GMC) of independent axes.	NO
GMC Independent EtherNetIP CIFX refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GIEIP (qualified-access-only)	PH GMC Independent EtherNetIP	EtherNet/IP backend (reserved for internal use only), for General Motion Control (GMC) of independent axes with LMC078.	NO
GMC Independent Interfaces refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GIITF (qualified-access-only)	direct version (FCL)	Interfaces (reserved for internal use only), for General Motion Control (GMC) of independent axes via Industrial Ethernet.	NO
GMC Independent ModbusTCP refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GITCP (qualified-access-only)	direct version (FCL)	Modbus TCP backend (reserved for internal use only), for General Motion Control (GMC) of independent axes.	NO
GMC Independent Lexium refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GILXM (qualified-access-only)	direct version (FCL)	Lexium specific function blocks, for General Motion Control (GMC) of independent axes via Industrial Ethernet.	YES
GMC Independent PLCopen MC refer to Motion Control Library Guide (<i>see SoMachine, Motion Control Library Guide</i>)	GIPLC (qualified-access-only)	direct version (FCL)	PLCopen Motion Control function blocks, for General Motion Control (GMC) of independent axes via Industrial Ethernet.	YES
Integrated Lexium Library refer to <i>Integrated Lexium Library Guide</i>	SEM_ILX	PH (FCL) SE_IntegratedLexium	Control of Lexium Integrated Drives.	YES
Lexium Library refer to <i>Lexium Library Guide</i>	SEM_LXM	PH (FCL) SE_Lexium	Control of Lexium 32, Lexium 05 and Lexium SD3 drives on CANopen fieldbus.	YES

Name	Namespace	Reference Library as	Description	USER
Lexium 23 Library refer to <i>Lexium 23 Library Guide</i>	SEM_LXM23	direct version (FCL)	Control of Lexium 23 drives on CANopen fieldbus.	YES
Lexium 28 Library refer to Lexium 28 Library Guide	SEM_LXM28	direct version (FCL)	Control of Lexium 28 drives on CANopen fieldbus.	YES
Lexium 32i Library	SEM_LXM32i	direct version (FCL)	Control of Lexium32i drives on CANopen fieldbus.	YES
TeSys Library refer to TeSys Library (see <i>SoMachine, TeSys Motor Starters Functions, TeSys Library Guide</i>)	SE_TESYS	direct version (FCL)	Control of TeSys U motor starter-controller and TeSys T motor management system. The library also offers function blocks for generic control of 1 or 2 directions / 1 or 2 speed motors.	YES
Harmony ZBRN refer to Harmony ZBRN Library (see <i>SoMachine, Scan for Buttons Linked to ZBRN Modules, Harmony ZBRN Library Guide</i>)	ZBRN	direct version (FCL)	Provides functionality to include Harmony wireless and batteryless push-buttons for remote control into the application.	YES
FieldBusDevicesModbusTcp	SEMFDM (qualified-access-only)	direct version (FCL)	Fieldbus Devices library with fieldbus and vendor specific function blocks.	YES
FieldBusDevicesPlcOpen	SEMFDP (qualified-access-only)	direct version (FCL)	PLCopen library for fieldbus devices with function blocks for motion control.	YES
TM3Safety	TMSF	direct version (FCL)	The TM3 safety module provides safety functionality embedded in the TM3 environment of the M2•• platforms.	YES
TM3TesysU	TSU	direct version (FCL)	TeSys modules library for motor starters.	YES

Category: Intern

The following table describes the libraries available in the **Intern** → **IoDrivers** category:

Name	Namespace	Reference Library as	Description	USER
IoDrvEtherNetIPAdapter	IoDrvEtherNetIPAdapter	PH (FCL) IoDrvEtherNetIPAdapter	EtherNet/IP adapter	NO
IoDrvPROFIBUSDPV1Slave	IoDrvPROFIBUSDPV1SlaveLib	PH (FCL) IoDrvPROFIBUSDPV1Slave	PROFIBUS DP slave library used to manage the TM5PCDPS module	NO
IoDrvSlave	IoDrvSlaveLib	PH (FCL) IoDrvSlave	Slave device library extended by IoDrvPROFIBUSDPV1Slave	NO
IoDrvModbusSerial	IoDrvModbusSerial	PH (FCL) SE_ModbusIOScanner	Modbus devices I/O scanning management for Modbus_IOScanner manager (for further information about Modbus Manager configuration, refer to online help <i>Device Editors / Modbus Configuration Editor / Modbus Configuration Editor / Modbus Device Editor</i>)	NO

Category: Solution

The following table describes the libraries available in the **Solution** category:

Name	Namespace	Reference Library as	Description	USER
Hoisting refer to Hoisting Library Guide (<i>see SoMachine, Hoisting Application Functions, Hoisting Library Guide</i>)	SE_HOIST	direct version (FCL)	Hoisting applications function blocks	YES
Hoisting_Basic refer to Hoisting Library Guide (<i>see SoMachine, Hoisting Application Functions, Hoisting Library Guide</i>)	SE_HOIST	direct version (FCL)	Core hoisting applications function blocks	YES

Name	Namespace	Reference Library as	Description	USER
PackML refer to PackML Library (<i>see PackML, Library Guide</i>)	PackML (qualified-access-only)	direct version (FCL)	Function blocks, structures and visualization frames for data management and operation mode management conforming to PackML, as defined by ISA-TR88.00.02.	YES
Packaging refer to Packaging Library Guide (<i>see SoMachine, Packaging Application Functions, Packaging Library Guide</i>)	SE_PACK	direct version (FCL)	Packaging applications function blocks	YES
Pumping refer to Pumping Library Guide (<i>see SoMachine, Booster Pumping Application Functions, Pumping Library Guide</i>)	Pump	direct version	Pumping applications function blocks	YES
EnergyEfficiencyToolbox refer to Energy Efficiency Toolbox Library Guide (<i>see Energy Efficiency Toolbox, Library Guide</i>)	EET	direct version (FCL)	Function Blocks to collect energy data from individual devices or equipment and to summarize energy usage for the machine.	YES
MachineEnergyDashboard refer to the Machine Energy Dashboard Library Guide (<i>see SoMachine, Machine Energy Dashboard Library Guide</i>)	MED	direct version (FCL)	MED information, based on the data provided by the EnergyEfficiencyToolbox library, dedicated to providing meaningful metrics concerning the energy efficiency of a machine.	YES
ModbusEnergyEfficiencyToolbox refer to Modbus Energy Efficiency Toolbox Library Guide (<i>see Modbus Energy Efficiency Toolbox, Library Guide</i>)	MEET	direct version (FCL)	Function Blocks to collect energy data from individual Modbus devices or equipment and to summarize energy usage for the machine.	YES

Category: System

The following table describes the libraries available in the **System** category:

Name	Namespace	Reference Library as	Description	USER
FtpBase	SE_FtpBase	PH SE_FtpBase	Core functionalities for FTP (File Transfer Protocol). The user interface is the library FtpRemoteFileHandling.	NO
LMC078 PLCSystem refer to LMC078 Motion Controller - PLCSystem Library Guide (<i>see Modicon LMC078 Motion Controller, System Functions and Variables, PLCSystem Library Guide</i>)	SEC	direct version	LMC078 System functions and variables	YES
LMC078 Sercos3	S3	direct version	Sercos 3 library for LMC078	YES
SystemConfigurationAddExtends	SystemConfigurationAddExtends	PH SystemConfigurationAddExtends	Provides base POU implementations from which POU's in SystemConfiguration are derived.	NO
SystemConfigurationItf	SystemConfigurationItf	PH SystemConfigurationItf	System configuration interface library used by the IoDrvTM5PCDPS library	NO
SystemInterface	SystemInterface	PH SystemInterface	System library used by the IoDrvTM5PCDPS library	NO

Category: Util

The following table describes the libraries available in the **Util** category:

Name	Namespace	Reference Library as	Description	USER
FileFormatUtility refer to FileFormatUtility Library (see <i>SoMachine, FileFormatUtility, Library Guide</i>)	FFU (qualified-access-only)	direct version (FCL)	The library offers functionalities to generate or read files coded in different structured formats (e.g. XML, CSV). These formatted files can be used to exchange data between the controller application and external systems (other controllers, web services, Manufacturing Enterprise System...).	YES
Toolbox refer to Toolbox Library Guide (see <i>SoMachine, Miscellaneous Functions, Toolbox Library Guide</i>)	SE_TBX	direct version (FCL)	Set of utility Functions and Function Blocks complementary to the automatically declared Standard and Util libraries	YES
Toolbox_Advance refer to Toolbox_Advance (see <i>SoMachine, Manage a Cyclic Task Interval, Toolbox_Advance Library Guide</i>)	SE_TBXADV	PH (FCL) SE_ToolboxAdvance	Functions used to get and set the interval of a Cyclic Task.	YES
FieldBusDevicesBase	SEMFDB (qualified-access-only)	direct version (FCL)	Fieldbus Devices base library	NO
FieldBusDevicesItf	SEMFDI (qualified-access-only)	direct version (FCL)	Fieldbus Devices interface library	NO
TwidoEmulationSupport refer to TwidoEmulationSupport Library (see <i>Twido Emulation Support Library, Library Guide</i>)	TES (qualified-access-only)	direct version (FCL)	Function blocks used by projects converted from SoMachine Basic and Twido into SoMachine.	NO

Other Libraries Used in SoMachine

Introduction

Libraries from companies other than Schneider Electric that are useful are listed below.

For more information on how to use the tables below, refer to the introduction for the Schneider Electric Libraries (*see page 72*).

3S - Smart Software Solution GmbH Libraries

Libraries contained in **Intern** and **System** are for internal system use and not available for user applications. Libraries are indicated as such in the column USER of the following tables by the word NO. Those that are intended for user applications are indicated as such by the word YES.

Libraries available in **Use Cases** category: container libraries that gather CAA libraries about a common topic (for further information about CAA functions, please refer to *CoDeSys Libraries / CAA Libraries*).

The following table describes the libraries available in the **Intern** → **SoftMotion** category:

Name	Namespace	Reference Library as	Description	USER
SM3_Basic	SM3_Basic	PH SM3_Basic	Functions for SoftMotion basic management (for further information about basic management, please refer to <i>CoDeSys Libraries / SoftMotion Libraries</i>) NOTE: This library is only supported on Modicon LMC058 Motion Controller.	YES
SM3_CNC	SM3_CNC	PH SM3_CNC	Functions for SoftMotion CNC management (for further information about CNC management, please refer to <i>CoDeSys Libraries / SoftMotion Libraries</i>) NOTE: This library is only supported on Modicon LMC058 Motion Controller.	YES
SM3_Drive_***	–	–	SoftMotion system management libraries (for further information, please refer to <i>CoDeSys Libraries / SoftMotion Libraries</i>)	NO

System Libraries

The following table describes the library available in the **(Miscellaneous)** category:

Name	Namespace	Reference Library as	Description	USER
CmpEventMgr	CmpEventMgr	PH CmpEventMgr	Event system management library	NO

The following table describes the libraries available in the **Application → Common** category:

Name	Namespace	Reference Library as	Description	USER
Standard64	Standard64	PH Standard64	Functions for wide characters strings management and long timers	YES
Standard	Standard	PH Standard	IEC programming standard functions and function blocks (for further information about standard functions, please refer to <i>CoDeSys Libraries / Standard Library</i>).	YES
Util	Util	PH Util	Programming additional functions and function blocks (for further information about additional functions, please refer to <i>CoDeSys Libraries / Util Library</i>)	YES

Libraries available in **Intern** category: some of these libraries are for internal system use only.

The following table describes the libraries available in the **System → SysLibs** category:

Name	Namespace	Reference Library as	Description	USER
Cmp*** Sys*** ISys***	–	–	System management libraries	NO
SysTime refer to Real Time Clock management (see <i>SoMachine, Getting & Setting Real Time Clock, SysTimeRtc and SysTimeCore Library Guide</i>)	SysTime	PH SysTime	Controller Real Time Clock management	YES

The following table describes the library available in the **System** → **SysLibs23** category:

Name	Namespace	Reference Library as	Description	USER
Sys***23	–	–	System management libraries for CoDeSys v2.3 projects migration	NO

CAA Technical Workgroup Libraries

The following table describes the libraries available in the **Application** → **CAA** category:

Name	Namespace	Reference Library as	Description	USER
CAA***	–	–	Libraries from CoDeSys Automation Alliance workgroup (for further information about CAA functions, please refer to <i>CoDeSys Libraries / CAA Libraries</i>).	NO
CAA CiA 405	CIA405 (qualified-access-only)	PH CAA CiA405	Functions blocks for CANopen fieldbus management from application	YES

The following table describes the library available in the **Intern** → **CAA** category:

Name	Namespace	Reference Library as	Description	USER
CAA***	–	–	Libraries from CoDeSys Automation Alliance workgroup (for further information about CAA functions, please refer to <i>CoDeSys Libraries / CAA Libraries</i>).	NO
CAA Net Base Services	NBS (qualified-access-only)	PH CAA NetBase Srv	This library implements a TCP server, TCP client and a UDP peer as a collection of core services for Ethernet.	YES *
* Do not use this library together with the TcpUdpCommunication library (Schneider Electric) or with libraries which implements the TcpUdpCommunication library. Use the library TcpUdpCommunication instead of library CAA Net Base Services.				

Chapter 6

Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Differences Between a Function and a Function Block	94
How to Use a Function or a Function Block in IL Language	95
How to Use a Function or a Function Block in ST Language	99

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (AND), calculations, conversion (BYTE_TO_INT)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, Timer_ON is an instance of the function block TON:

```
1  PROGRAM MyProgram_ST
2  VAR
3      Timer_ON: TON; // Function Block Instance
4      Timer_RunCd: BOOL;
5      Timer_PresetValue: TIME := T#5S;
6      Timer_Output: BOOL;
7      Timer_ElapsedTime: TIME;
8  END_VAR

1  Timer_ON(
2      IN:=Timer_RunCd,
3      PT:=Timer_PresetValue,
4      Q=>Timer_Output,
5      ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POU's (<i>see SoMachine, Programming Guide</i>).
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: <ul style="list-style-type: none"> type the name of the function in the operator column (left field), or use the Input Assistant to select the function (select Insert Box in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: <code>IsFirstMastCycle</code>	
with input parameters: <code>SetRTCDrift</code>	

In IL language, the function name is used directly in the operator column:

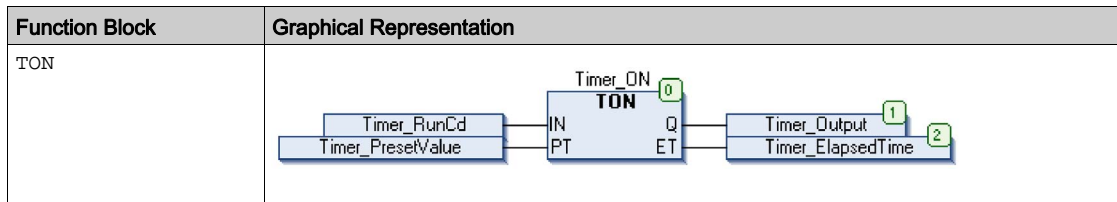
Function	Representation in POU IL Editor															
IL example of a function without input parameter: IsFirstMastCycle	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 FirstCycle: BOOL; 4 END_VAR 5 </pre> <hr/> <table border="1" data-bbox="391 467 975 581"> <tr> <td data-bbox="391 467 432 500">1</td> <td data-bbox="432 467 738 500">IsFirstMastCycle</td> <td data-bbox="738 467 975 500"></td> </tr> <tr> <td data-bbox="391 500 432 532"></td> <td data-bbox="432 500 738 532">ST</td> <td data-bbox="738 500 975 532">FirstCycle</td> </tr> </table>	1	IsFirstMastCycle			ST	FirstCycle									
1	IsFirstMastCycle															
	ST	FirstCycle														
IL example of a function with input parameters: SetRTCDrift	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 myDrift: SINT (-29..29) := 5; 4 myDay: DAY_OF_WEEK := SUNDAY; 5 myHour: HOUR := 12; 6 myMinute: MINUTE; 7 myDiag: RTCSETDRIFT_ERROR; 8 END_VAR 9 </pre> <hr/> <table border="1" data-bbox="391 992 926 1170"> <tr> <td data-bbox="391 992 432 1024">1</td> <td data-bbox="432 992 683 1024">LD</td> <td data-bbox="683 992 926 1024">myDrift</td> </tr> <tr> <td data-bbox="391 1024 432 1057"></td> <td data-bbox="432 1024 683 1057">SetRTCDrift</td> <td data-bbox="683 1024 926 1057">myDay</td> </tr> <tr> <td data-bbox="391 1057 432 1089"></td> <td data-bbox="432 1057 683 1089"></td> <td data-bbox="683 1057 926 1089">myHour</td> </tr> <tr> <td data-bbox="391 1089 432 1122"></td> <td data-bbox="432 1089 683 1122"></td> <td data-bbox="683 1089 926 1122">myMinute</td> </tr> <tr> <td data-bbox="391 1122 432 1154"></td> <td data-bbox="432 1122 683 1154">ST</td> <td data-bbox="683 1122 926 1154">myDiag</td> </tr> </table>	1	LD	myDrift		SetRTCDrift	myDay			myHour			myMinute		ST	myDiag
1	LD	myDrift														
	SetRTCDrift	myDay														
		myHour														
		myMinute														
	ST	myDiag														

Using a Function Block in IL Language

This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POU's (<i>see SoMachine, Programming Guide</i>).
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a <code>CAL</code> instruction: <ul style="list-style-type: none"> ● Use the Input Assistant to select the FB (right-click and select Insert Box in the context menu). ● Automatically, the <code>CAL</code> instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: <ul style="list-style-type: none"> ● Values to inputs are set by " := ". ● Values to outputs are set by " => ".
4	In the <code>CAL</code> right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the `TON` Function Block graphically presented below:



In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	<pre>1 PROGRAM MyProgram_IL 2 VAR 3 Timer_ON: TON; // Function Block instance declaration 4 Timer_RunCd: BOOL; 5 Timer_PresetValue: TIME := T#5S; 6 Timer_Output: BOOL; 7 Timer_ElapsedTime: TIME; 8 END_VAR 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000</pre>

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

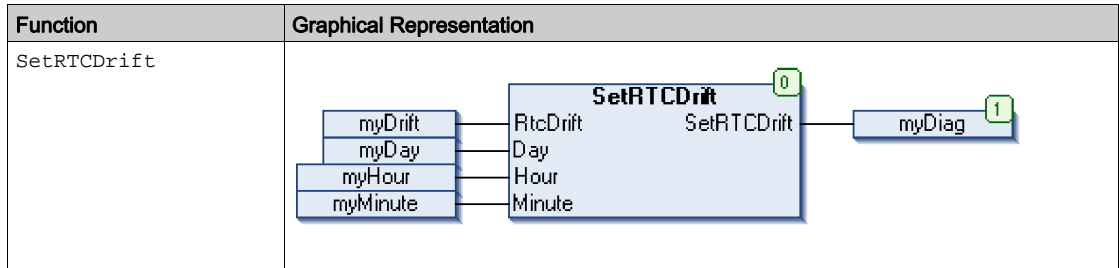
Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POU's (<i>see SoMachine, Programming Guide</i>).
2	Create the variables that the function requires.
3	Use the general syntax in the POU ST Editor for the ST language of a function. The general syntax is: <code>FunctionResult := FunctionName(VarInput1, VarInput2, .. VarInputx);</code>

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:



The ST language of this function is the following:

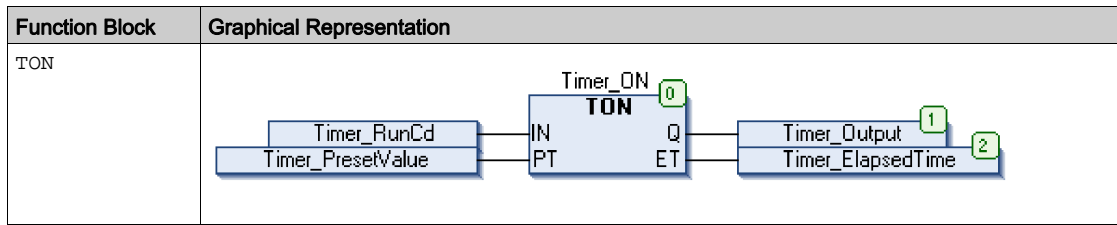
Function	Representation in POU ST Editor
SetRTCDrift	<pre>PROGRAM MyProgram_ST VAR myDrift: SINT(-29..29) := 5; myDay: DAY_OF_WEEK := SUNDAY; myHour: HOUR := 12; myMinute: MINUTE; myRTCAdjust: RTCDRIFT_ERROR; END_VAR myRTCAdjust := SetRTCDrift(myDrift, myDay, myHour, myMinute);</pre>

Using a Function Block in ST Language

This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. NOTE: The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation (see <i>SoMachine, Programming Guide</i>).
2	Create the input and output variables and the instance required for the function block: <ul style="list-style-type: none"> • Input variables are the input parameters required by the function block • Output variables receive the value returned by the function block
3	Use the general syntax in the POU ST Editor for the ST language of a Function Block. The general syntax is: <pre>FunctionBlock_InstanceName (Input1:=VarInput1, Input2:=VarInput2, ... Ouput1=>VarOutput1, Ouput2=>VarOutput2, ...);</pre>

To illustrate the procedure, consider this example with the TON function block graphically presented below:



This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	<pre> 1 PROGRAM MyProgram_ST 2 VAR 3 Timer_ON: TON; // Function Block Instance 4 Timer_RunCd: BOOL; 5 Timer_PresetValue: TIME := T#5S; 6 Timer_Output: BOOL; 7 Timer_ElapsedTime: TIME; 8 END_VAR Timer_ON(IN:=Timer_RunCd, PT:=Timer_PresetValue, Q=>Timer_Output, ET=>Timer_ElapsedTime); </pre>



A

application

A program including configuration data, symbols, and documentation.

ASCII

(American standard code for Information Interchange) A protocol for representing alphanumeric characters (letters, numbers, certain graphics, and control characters).

ATV

The model prefix for Altivar drives (for example, ATV312 refers to the Altivar 312 variable speed drive).

B

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

calibration

The process of setting or maintaining the accuracy of a measuring device by comparing its value to a known and correct standard.

CFC

(continuous function chart) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

CiA405

The CANopen interface and device profile for IEC 61131-3 programmable controllers.

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

control network

A network containing logic controllers, SCADA systems, PCs, HMI, switches, ...

Two kinds of topologies are supported:

- flat: all modules and devices in this network belong to same subnet.
- 2 levels: the network is split into an operation network and an inter-controller network.

These two networks can be physically independent, but are generally linked by a routing device.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

cyclic task

The cyclic scan time has a fixed duration (interval) specified by the user. If the current scan time is shorter than the cyclic scan time, the controller waits until the cyclic scan time has elapsed before starting a new scan.

E

equipment

A part of a machine including sub-assemblies such as conveyors, turntables, and so on.

expansion bus

An electronic communication bus between expansion I/O modules and a controller.

F

FB

(function block) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

FCL

(forward compatible library) A forward compatible library is developed in such a way that its functionalities are forward compatible. This means that every version of a forward compatible library contains all functionalities of the previous version and a newer library version can be easily used in already existing projects without any changes.

FTP

(file transfer protocol) A standard network protocol built on a client-server architecture to exchange and manipulate files over TCP/IP based networks regardless of their size.

function

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

Examples: boolean (AND) operators, calculations, conversions (BYTE_TO_INT)

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

I**I/O**

(input/output)

IEC

(international electrotechnical commission) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

IL

(instruction list) A program written in the language that is composed of a series of text-based instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand (refer to IEC 61131-3).

INT

(integer) A whole number encoded in 16 bits.

IP

(Internet protocol) Part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

L**LD**

(ladder diagram) A graphical representation of the instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller (refer to IEC 61131-3).

LMC

(Lexium motion controller)

LMC058

Lexium motion controller A Modicon logic controller

M**M258**

Modicon M258 logic controller

machine

Consists of several *functions* and/or *equipment*.

Modbus

The protocol that allows communications between many devices connected to the same network.

N

network

A system of interconnected devices that share a common data path and protocol for communications.

P

POU

(program organization unit) A variable declaration in source code and a corresponding instruction set. POU's facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POU's are available to one another.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

protocol

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

PTO

(pulse train outputs) A fast output that oscillates between off and on in a fixed 50-50 duty cycle, producing a square wave form. PTO is especially well suited for applications such as stepper motors, frequency converters, and servo motor control, among others.

PWM

(pulse width modulation) A fast output that oscillates between off and on in an adjustable duty cycle, producing a rectangular wave form (though you can adjust it to produce a square wave).

S

SNMP

(simple network management protocol) A protocol that can control a network remotely by polling the devices for their status and viewing information related to data transmission. You can also use it to manage software and databases remotely. The protocol also permits active management tasks, such as modifying and applying a new configuration.

SNTP

(simple network time protocol) A simplified version of the NTP (network time) protocol. It is used to synchronize computer clock times in a network of computers. It uses Coordinated Universal Time (UTC) to synchronize computer clock times to a millisecond, and sometimes to a fraction of a millisecond.

SQL

SQL (Structured Query Language) is a programming language for managing data stored in relational database management systems.

ST

(structured text) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

string

A variable that is a series of ASCII characters.

symbol

A string of a maximum of 32 alphanumeric characters, of which the first character is alphabetic. It allows you to personalize a controller object to facilitate the maintainability of the application.

system variable

A variable that provides controller data and diagnostic information and allows sending commands to the controller.

T**TCP**

(transmission control protocol) A connection-based transport layer protocol that provides a simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

U**UDP**

(user datagram protocol) A connectionless mode protocol (defined by IETF RFC 768) in which messages are delivered in a datagram (data telegram) to a destination computer on an IP network. The UDP protocol is typically bundled with the Internet protocol. UDP/IP messages do not expect a response, and are therefore ideal for applications in which dropped packets do not require retransmission (such as streaming video and networks that demand real-time performance).

V**variable**

A memory unit that is addressed and modified by a program.



F

functions

- differences between a function and a function block, *94*

- how to use a function or a function block in IL language, *95*

- how to use a function or a function block in ST language, *99*

L

- LanguageModelAttribute, *39, 63*

- libraries, *11*

- Library Manager, *11*

- Library Repository, *11*