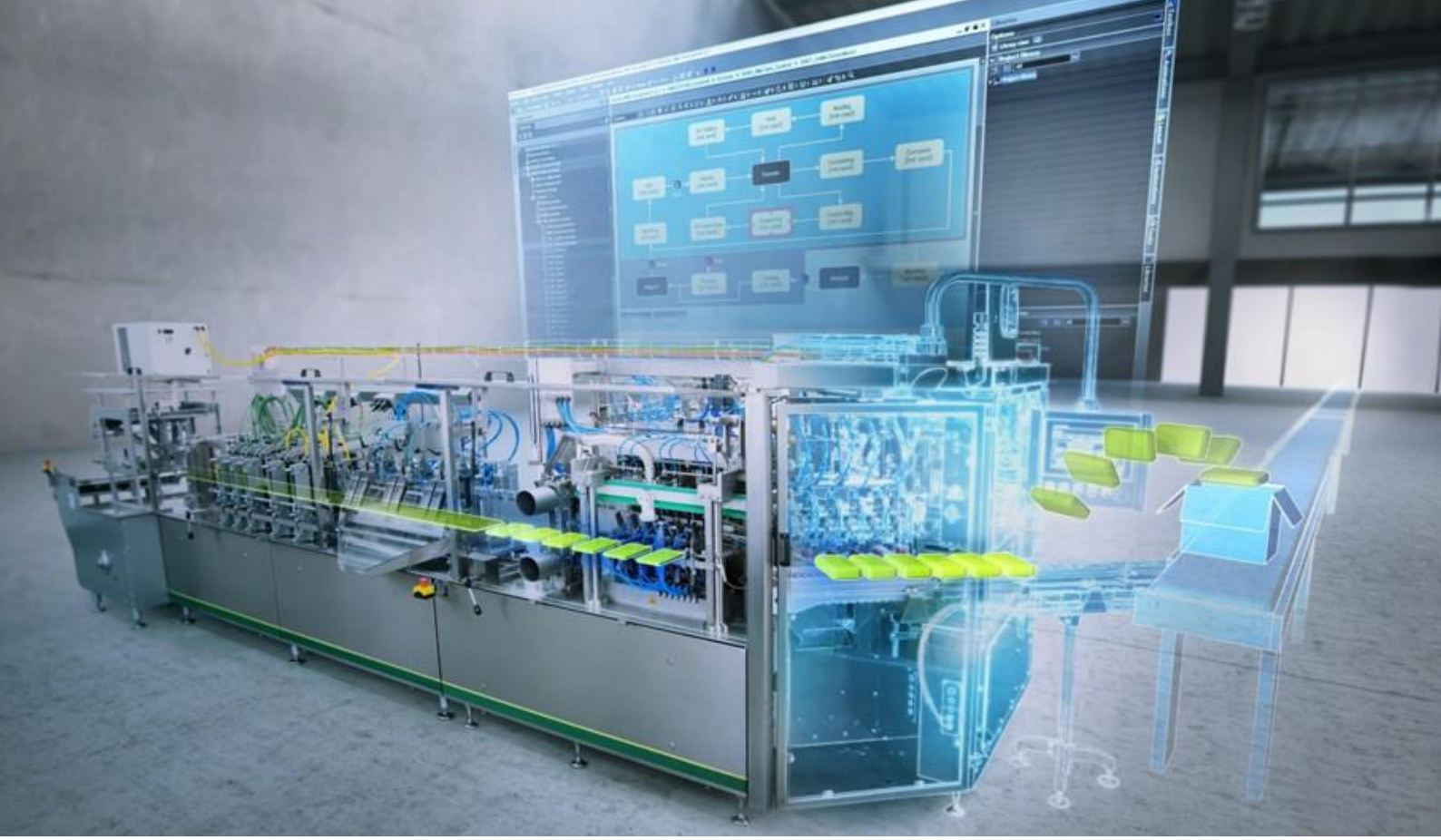


Siemens
Industry

AF LIBRARY

Automation Framework V1.2

SIEMENS

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

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

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under https://www.siemens.com/cert.

Table of contents

1. What's new

1.1. New features

Issue	Date	Description
1.0	2023/12/22	First version
1.1	2024/06/28	• Advanced unit control features (LUC-Library extensions) • Enhancements of sequencer (Graph, LAD and SCL) • Holistic alarm handling from CM to Unit • Manual operation lines • Multi-Unit support (>=1 Unit with >= 1 operation panel) • Favorites for screen selection
1.2	2024/09/30	• Simulation with S7-PLCSIM V19 in addition to S7-PLCSIM Advanced is supported • Function "LAF_OmacToSequence" renamed to "LAF_MapUnitStatusToSequence" and configuration inputs have been added to provide more flexibility in configuring the sequence behavior.

Table 1-1: New features

1.2. Change log

See Chapter Change documentation

2. Introduction

2.1. Overview

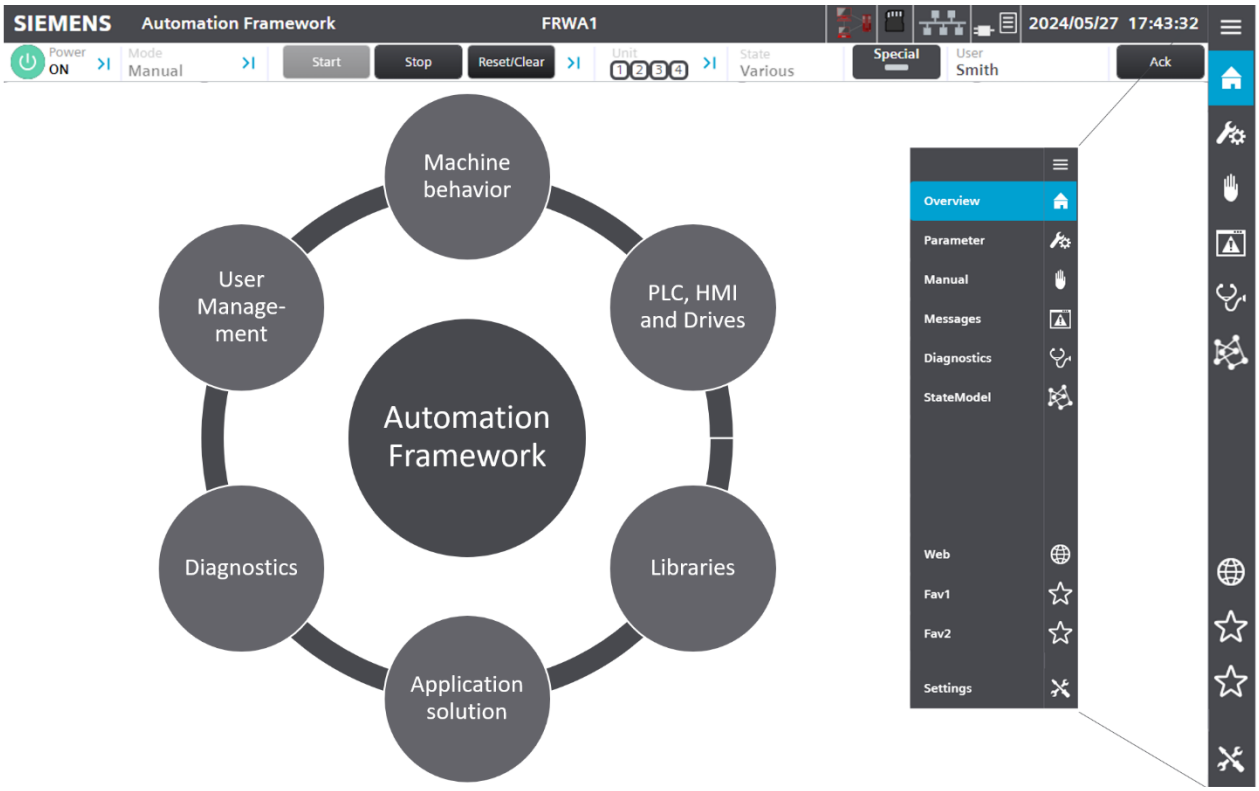


Figure 2-1: Siemens Automation Framework.

A core offering of the Siemens Automation Framework (in following named as AF) is a comprehensive collection of TIA Portal libraries, application examples, HW configurations, and good programming practices in a consolidated, preconfigured TIA Portal project. It offers basic functionality to develop a generic automation application. It comes with a fully configured operator panel which contains main navigation and operating functions. On this basis, the user can easily build and extend his own project modularly using additional PLC and HMI objects from the library.

The project introduces but does not dictate a best practice for intuitive TIA Portal project structuring and provides application examples for PLC, Motion and HMI programming ready to use.

This document describes the content and structure of the TIA Portal project Start_Up_AF_Library, short AF project, and its libraries. The user is free to modify the project or start over with an empty project to avoid unused code in his application.

Key Features

- Modularized PLC and HMI TIA Portal project conform to the physical modell according to ISA-88
- User-friendly implementation of OMAC PackML compliant mode and state manager
- Pre-configured operator panel and guidance where and how to extend the configuration
- Introduction on easy integration and handling of libraries and updates
- Various TIA Portal libraries with application solutions (examples)
- Harmonised process diagnostics via system function ProDiag
- Manual operations with HMI templates for example to execute jog operations

Goals

- Holistic TIA Portal project & libraries that suits various machine types
- Guidance and best practice sharing how to implement various application uses cases
- Easy extention by utalizing additional Siemens technology offerings, out-of-the-box libraries, application examples, templates, and how to integrate them

Benefits for the machine builder and equipment end users

- Modularity allows reusability and better return of invest
- Isolation of equipment increases flexibility
- Fast implementation and harmonized operation of different machine types
- Flexibility to manage updates, feature extension and change request
- Preconfigured operation key performance indicators (KPI)
- Harmonized process and system diagnostics speeds up commissioning and minimizes production downtime

NOTE The intention of the AF project is to be used as a TIA Portal project template for various machine types. It does not claim to be a fully comprehensive out of the box solution. The user remains responsible developing his own application, implementing the needed use cases, and incorporating updates and patches where necessary.

For existing machine code implementation, the user can import and utilize elements of the AF library where appropriate.

Required knowledge

A general knowledge of the following areas is needed to understand these manuals:

- Automation engineering within
  - SIMATIC and SINAMICS Hardware
  - SIMATIC STEP 7 (TIA Portal)
  - SIMATIC WinCC Unified (TIA Portal)
  - SIMATIC Safety (TIA Portal)
  - SINAMICS Startdrive (TIA Portal)
- Machine and plant diagnostics
- Industrial Communication and Networks
- OPC UA

NOTE Please be aware that the usage of the TIA Portal project requires the latest Unified Panel image version (from V19.0.0.2).

2.2. Scope of Delivery

The Version 1.2 of Automation Framework includes:

- This user documentation "109817223_Automation_Framework_DOC_V1_2_en.pdf"
- TIA Portal Project 109817223_Automation_Framework_PROJ_V1_2.zap
- TIA Portal Project 109817223_Automation_Framework_Startup_PROJ_V1_2.zap

2.3. Used Libraries

The following TIA Portal libraries are used in the AF project:

AF Global library

The AF Global library contains all sub-libraries, master copies and user documentation recommended for the AF. Upcoming releases, patches, and updates will be provided via this Global library. Update procedures based on TIA Portal standard features are covered in chapter Library handling and updates.

Library Name	Description	Link
LGF	The Library of General Functions (LGF) extends the STEP 7 instructions in TIA Portal for PLC programming (mathematical functions, times, counters, etc.). The library can be used unrestricted and contains features such as FIFO, search function, matrix calculations, astro timer, etc.	SIOS-ID: 109479728
LPML	The OMAC PackML library (LPML) provides a user-friendly basis for the configuration and use of an OMAC-compliant mode and state manager for SIMATIC.	SIOS-ID: 109821198
LUC	The Library of Unit Control (LUC) provides function blocks that simplify LPML OMAC state machine handling, preprocesses operator or remote commands (e.g. via OPC UA) and offers a stack light implementation. It uses blocks and data types from the LPML and extends its functionality.	SIOS-ID: 109974940
LAF	The Library of Automation Framework (LAF) provides blocks to easily implement an ISA-88 structure and common functions. It uses blocks and data types from the LUC library and extends its functionality.	SIOS-ID: not planned
LDrvSafe	The Library of Safe Drive Control (LDrvSafe) includes blocks to realize safety applications, e.g. simple control of SINAMICS Safety Functions via PROFIsafe as well as failsafe diameter detection, up to Safety Integrity Level 2 (EN 62061) and Performance Level d, Category 3 (EN ISO 13849-1).	SIOS-ID: 109485794
LSafe	The TÜV-certified library LSafe can be used to implement basic safety functions for electromechanical or electronic sensors and actuators. The library facilitates the acceptance of your application software since it can be based on already tested modules.	SIOS-ID: 109793462
LSNTP (LCom)	The use of a SIMATIC S7 CPU as an SNTP server enables flexible and simple synchronization of systems and subsystems, for example, to obtain meaningful timestamps for error messages and logging data system wide.	SIOS-ID: 109780503
LAxisCtrl	This library provides an axis function block with very extensive functions for simple control of axes. It is used in the LBC library blocks for axis control or be used standalone.	SIOS-ID: 109749348
LBC	The "Library of Basic Controls" (LBC) provides basic controls that are programmed in a standardized concept according to the Siemens programming style guide and the "PLC Open" guideline.	SIOS-ID: 109792175
LSicar V5	The library "LSicar" contains useful and well proven features for production data, shift model and diagnostics.	SIOS-ID: 109804254
LPD	The "Library of PLC data types" (LPD) contains PLC data types which describe the data structure of the address spaces and the data records of peripheral / technology modules and PROFIdrive drives.	SIOS-ID: 109482396

Table 2-1: Libraries used in AF project

2.4. Hardware and Software Requirements

The AF project was developed based on the following components:

Hardware

- SIMATIC S7-1500 (F/ TF) with Firmware Version V3.0.3 or higher– for more information consult the SIOS-ID: 109478459
- WinCC Unified Panel display size 12" (from Panel Image V19.0.0.2, SIOS-ID:109825605)
- SINAMICS S200, S210 and G120
- ET 200SP

Software

- TIA Portal V19 Update 3 or higher
- SIMATIC STEP 7 Professional V19 Update 3 or higher for PLC engineering
- SIMATIC WinCC Unified V19 Update 3 or higher for HMI engineering
- SINAMICS Startdrive Basic V19 SP1 or higher for configuration and commissioning of drives, PLC connection, diagnosis, optimization and parametrization
- TIA Portal Add-In Code2Docu \4\ for generating PLC-Block documentation

Optional products to be used

- S7-PLCSIM V19 or higher for simulation of real PLC firmware
- S7-PLCSIM Advanced V6.0 or higher for simulation of real PLC firmware with virtual network adapter
- SIMATIC WinCC Unified PC Runtime V19 Update 2 or higher for simulation of the HMI application
- SIMATIC Visualization Architect (SiVarc) for automatic generation of "ready to use" screens
- TIA Portal Test Suite for automated code unit testing
- SINAMICS Startdrive Advanced for safety acceptance and safety activation tests, measurement functions and long-term trace
- SIMIT to simulate the behavior of drives, sensors, actors, I/Os, etc. and the connectivity to periphery and PLCs for validation of PLC program functionality and communication
- NX Mechatronics Concept Designer (MCD) 2406 or higher for 3D based physics simulation to plan and validate machine sequences, collision detection and virtual commissioning

Licenses

The following licenses are used in the AF project:

Mandatory	See current AF component release list for order and version information	License which must be delivered to the end user
Yes	SIMATIC STEP 7 Professional V19	-
Yes	SIMATIC S7, Safety programming tool	-
Yes	STEP 7 Safety Advanced	-
No	SIMATIC S7-PLCSIM V19	-
No	SIMATIC S7-PLCSIM Advanced V6.0	1x License for SIMATIC S7-PLCSIM Advanced
Yes	SIMATIC WinCC Unified V19 Comfort Engineering	-
No	SIMATIC WinCC Unified PC Runtime V19	1x Runtime License for SIMATIC WinCC Unified PC (10k) RT
Yes	SINAMICS Startdrive Basic V19	-
No	SINAMICS Startdrive Advanced V19	1x Engineering-License for SINAMICS Startdrive Advanced
Yes	ProDiag for PLC Runtime	1x Runtime-License ProDiag unlimited for each CPU
No	ProDiag for HMI (PLC Code Viewer)	1x SIMATIC ProDiag for WinCC Unified Runtime Controls. Licenses needed for each control panel
No	OPC UA	1x Runtime-License OPC UA for each CPU, type of license (small, medium, large) depending on the interface type
No	SIMIT	1x Engineering-License for SIMIT, type of license (S, M, L, XL) depending on the simulation scope
No	NX MCD	1x License for MCD Designer (model editing & virtual commissioning) or 1x License for MCD Player (only virtual commissioning)

Table 2-2: Licenses for AF project

NOTE OPC UA and HMI power tag licenses might vary with your specific application. Please ensure corresponding license size if needed.

ProDiag for PLC Runtime is used in the AF project to simplify the engineering for the user. Alternatively, it can also be easily replaced with an applicative solution.

2.5. User-defined documentation - Online Help

Over time, you create your own contents in your project or library, for example, blocks, tags, or library types. While the operating principle of the TIA Portal is described in the supplied help system (F1), there is no help for the contents you have created yourself.

TIA Portal provides a system function to open a document (Shift+F1) related to the block. You can create your own user-defined documentation to explain to other employees how your project and functions work or how to use individual library types or blocks.

NOTE

User-defined documentation is not copied together with a type to another library. You need to copy the user-defined documentation for types to the corresponding directory.

For additional help on using the user-defined documentation, refer to the section "Using user-defined documentation" in TIA Portal documentation.

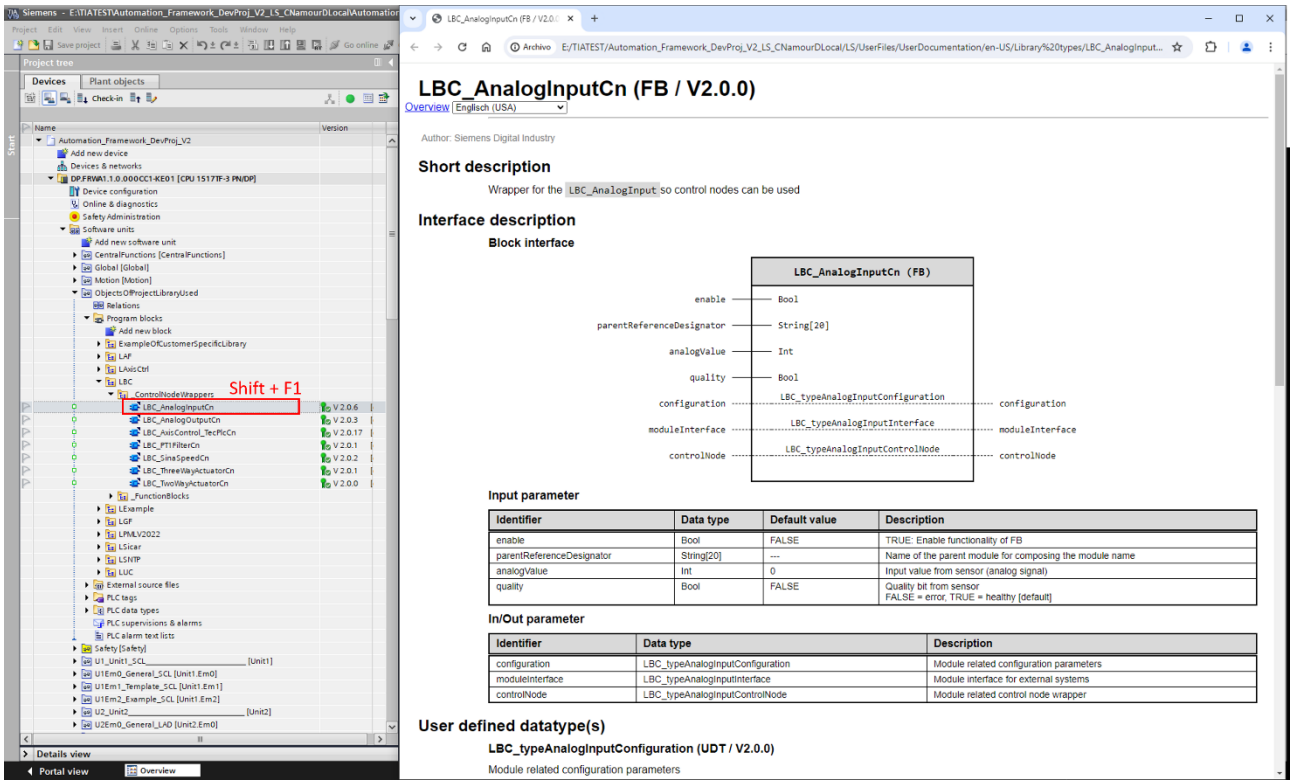


Figure 2-2: How to open the online help for library blocks.

2.5.1. Directories for user-defined documentation

User-defined documentation can be saved in the following directories:

TIA Portal project folder

Save the documentation under [project folder]\UserFiles\UserDocumentation\[language]. This way the documentation can be forwarded with the TIA Portal project.

Directory of a Global library

Save the documentation in the directory of the [Global library]\UserFiles\UserDocumentation\[language]. This way the user-defined documentation can be forwarded with the Global library.

Central directory on the hard drive or a network drive

You can store the documentation in a central directory on the hard disk or on a network drive. This way you can access to the documentation from different projects or share the documentation via network drive. The directory path for the user-defined documentation must be specified in the TIA Portal XML configuration file or via the TIA Portal settings.

2.5.2. Code2Docu

In AF the user documentation is written in the code section of each block (Comment of Network 1 and 2 as multi-language for LAD blocks and a comment in Region Header Info and Description for SCL blocks).

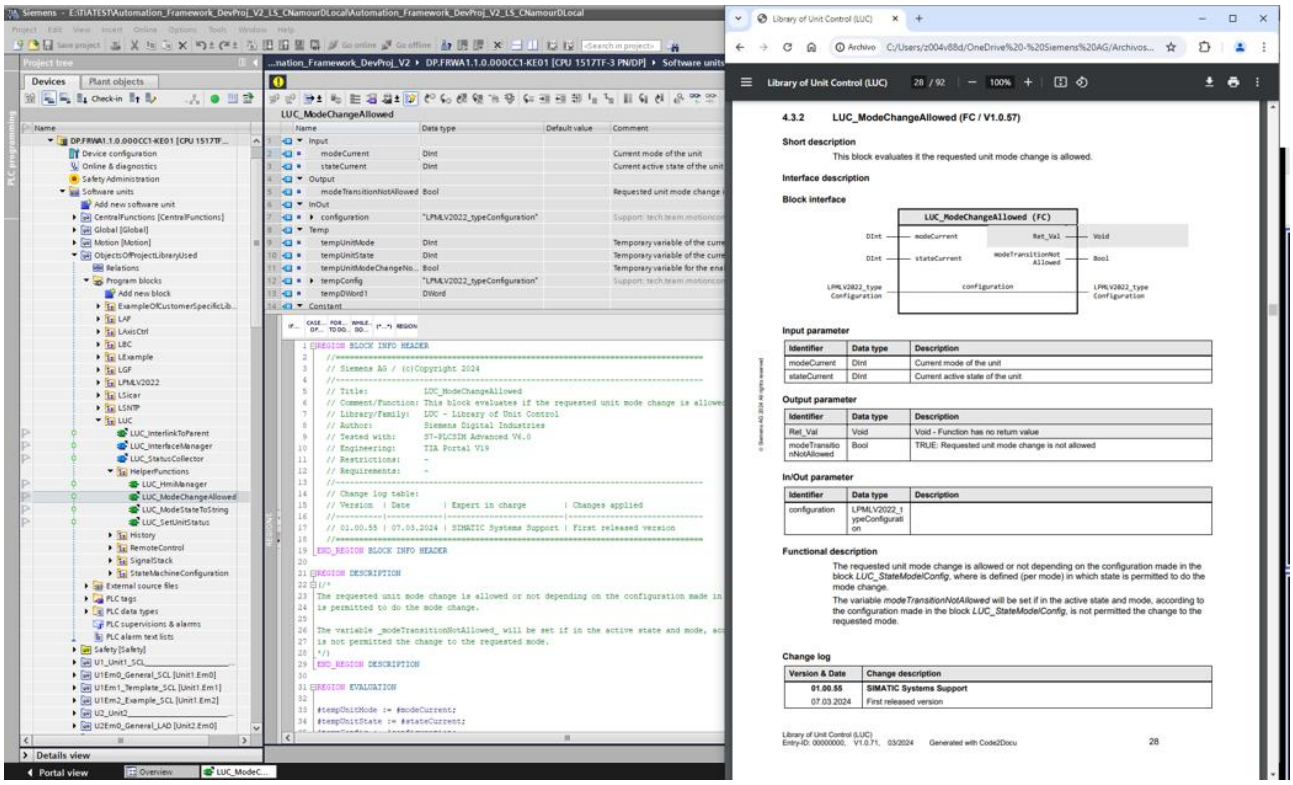


Figure 2-3: Example of a block documentation generated with the add-in Code2Docu.

With the help of the TIA Portal AddIn Code2Docu \4\ a document can be created for each block automatically for all used languages.

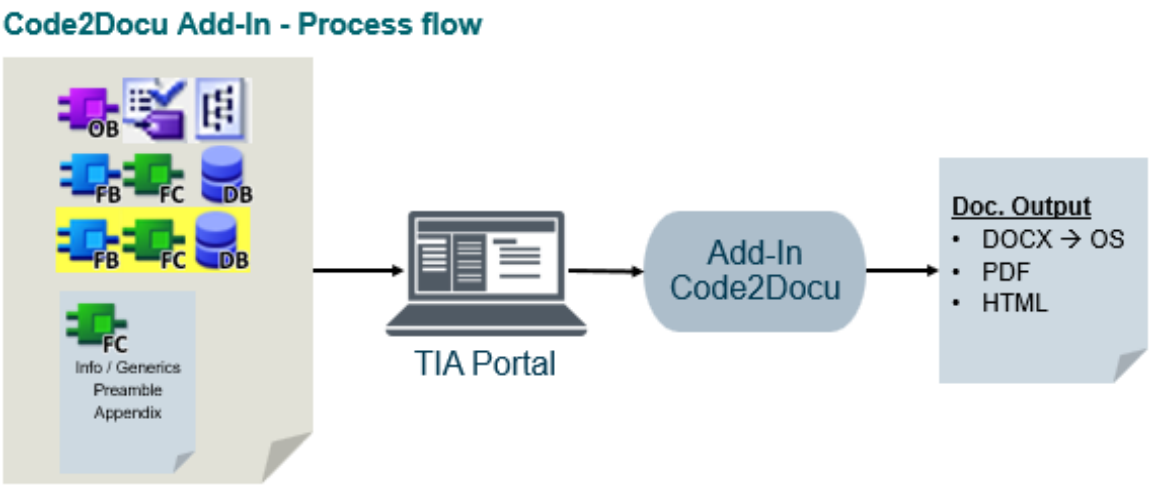


Figure 2-4: Code2Docu – Process flow.

2.6. Programming conventions

AF follows the conventions of the Siemens Programming Styleguide for S7-1200/S7-1500 and the rules of the Siemens Programming Guideline, Siemens Programming Guideline for Safety and Engineering Guidelines for WinCC Unified.

More information about the Programming Styleguide available in the following link:
Siemens Guidelines

Programming Styleguide

Deviation from these guidelines apply to the following guideline style rules:

- **GL003 Rule : Supply texts in all project languages**.

  All project texts must be available in English and in all other languages used within the project. While the AF project supports Chinese, English (UK), French, German, Italian, Korean, and Spanish, the texts are currently provided only in English (United States). Users are responsible for editing and supplying the translations needed for their specific project languages.

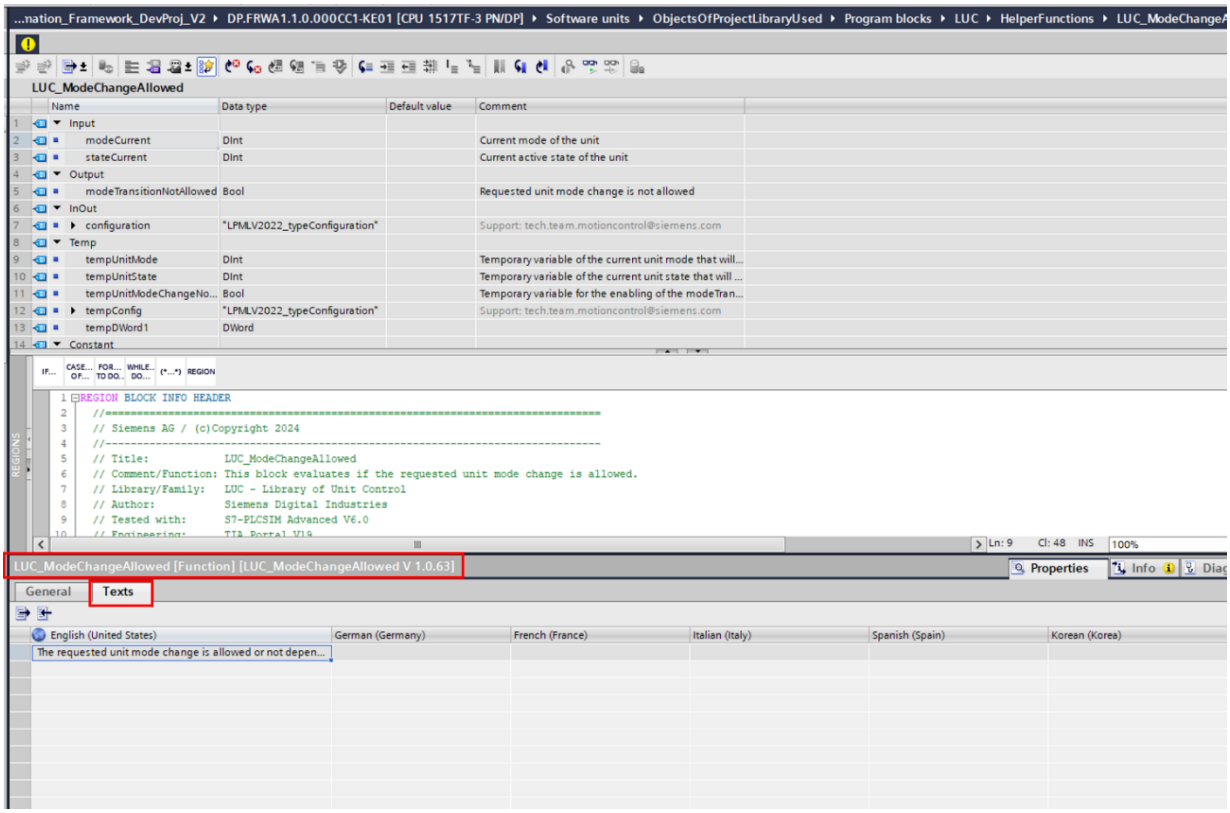


Figure 2-5: In the block editor the texts and their translations can be easily managed in the tab "Texts".

- **NF002 Rule : Use meaningful comments and properties**

  Comment and property fields should be filled with meaningful information in the active languages of the project. Although the AF project supports Chinese, English (UK), French, German, Italian, Korean, and Spanish, the texts are currently provided only in English (United States). Users are required to add translations in the necessary project languages.

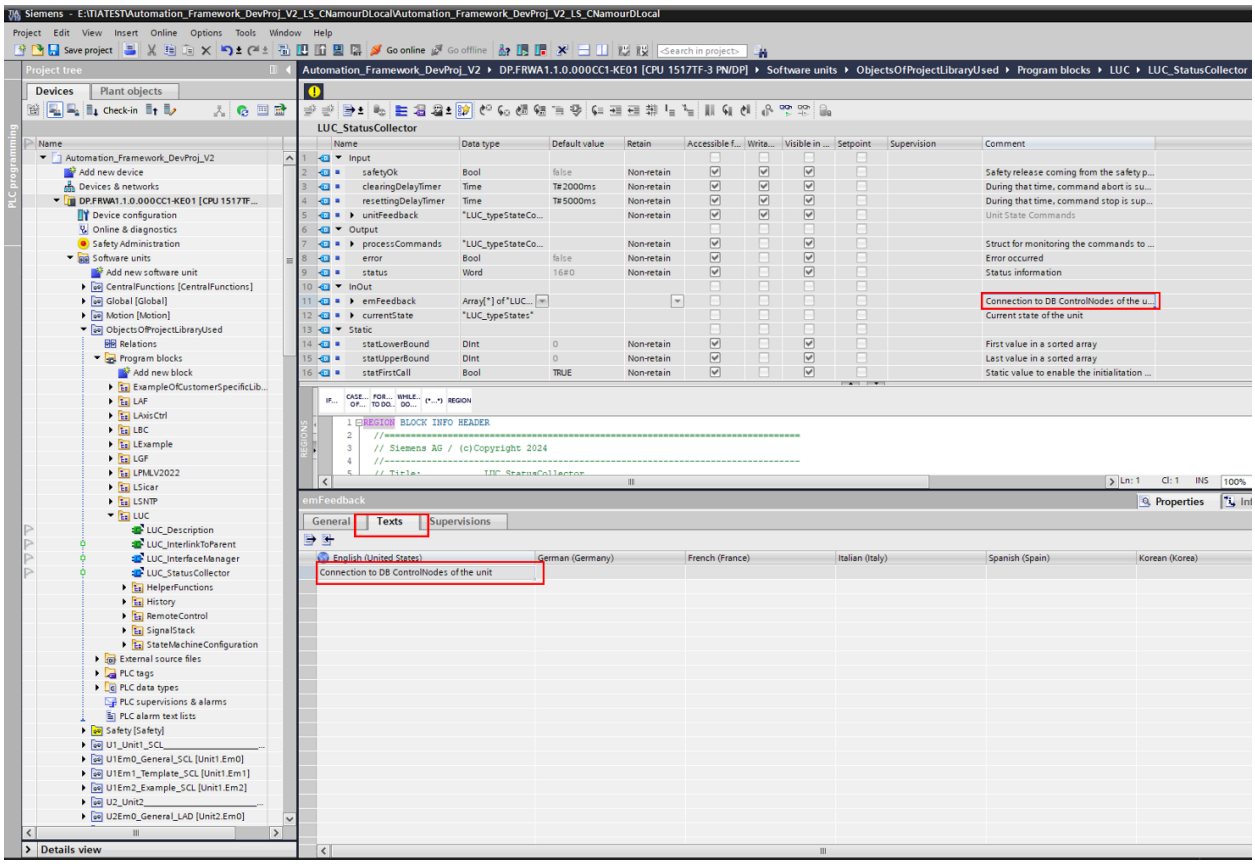


Figure 2-6: Manage and edit text and translations in the block editor easily.

- **NF005 Rule : Use PascalCasing for objects**

  Identifiers for TIA Portal objects, including blocks, Software Units, technology objects, libraries, PLC tag tables, PLC alarm text lists, watch and force tables, traces, and measurements, are formatted using PascalCase.In AF project there is a deviation in the PascalCasing rules by identifying an object with several capital letters in a row.

  The rule needs to be broken to ensure a clear and understandable identifier for the object. The reasons for breaking the rule are to maintain clarity when using widely recognized abbreviations like LAD, KOP, and FUP, and to introduce new abbreviations, such as LAF for LibraryAutomationFramework, ensuring they remain easily identifiable. For example, according to the rule, the block "LAF_SequenceManagerLAD" should be written as "Laf_SequenceManagerLad," but this creates a confusing identifier. Following the rule strictly would result in less readable identifiers.

Programming Guideline

Compliance with the following programming rules is strongly recommended:

- Do not use bit memory/markers, define global data block (DB) instead.
- Disable "evaluate ENO" for blocks to improve performance within LAD/FBD.
- Add supervisions directly inside the object (FC, FB, UDT) to benefit from the type instance concept.
- Exchange variables only via the block interfaces (In, Out, InOut) to ensure reusability of the blocks.
- Prefer CASE instruction before IF.. ELSIF..ELSE or loops where possible.
- Use user-defined data types (UDT) instead of STRUCT datatype.

NOTE	TIA Portal supports a maximum of 252 structures (STRUCT datatype) within one data block.

3. AF Guidelines

This document describes how to use the Automation Framework. Further guidelines used by the Automation Framework are listed in chapter Siemens Guidelines.

4. Library handling and updates

This chapter describes how to use a TIA Portal library the most efficient way. Preliminary actions are introduced to prepare for library updates during the project phase without affecting your custom code. The AF Library and all Types in the AF project are delivered "write-protected" to prevent changes on the original elements and to ensure updateability.

4.1. The library concept

There are two concepts supported for TIA Portal Library usage: Types and Master Copies.

NOTE For more information about libraries, please see the Guideline on Library Handling in TIA Portal.

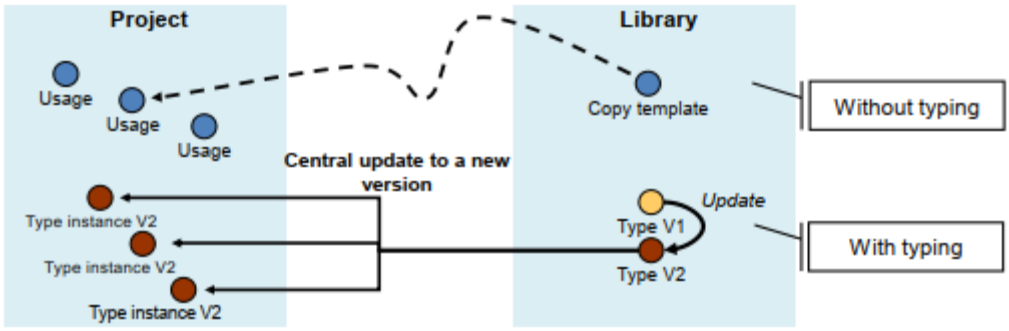


Figure 4-1: Differences using library copies vs library types.

Using Types

When storing an object as Type in the library any instance of this object keeps the relation to their origin library Type. Types support versioning and enable changes centrally within the library object. The changes on a library type can be updated throughout the entire project and all its instances. With this procedure only a single point of change is necessary.

Using Master Copies

When storing an object as Master Copy any instance is only a copy of this TIA Portal library object. The copies do neither keep the relation to their origin nor is versioning or central editing supported.

For this reason, the recommendation is to use library types where it is possible.

4.2. How to fix library inconsistencies

When working with nested library types, changes to the origin type may affect its dependent types. The TIA Portal library status will inform you about existing inconsistencies. It is important to resolve these inconsistencies before running a project update from a Global library.

NOTE It is recommended to begin with "Right click type" > "Fix inconsistencies" and, depending on the available options "Use default version" or "Advanced edit type" to prioritize the default version. More detailed explanation can be found in Guideline on Library Handling in TIA Portal.

https://support.industry.siemens.com/cs/ww/en/view/109747503

Use of Library management

Sometimes the following user interactions are necessary to resolve library inconsistencies. Right click the Types folder in your Project library and choose "Library management" from the context menu. Inconsistencies are highlighted in yellow.

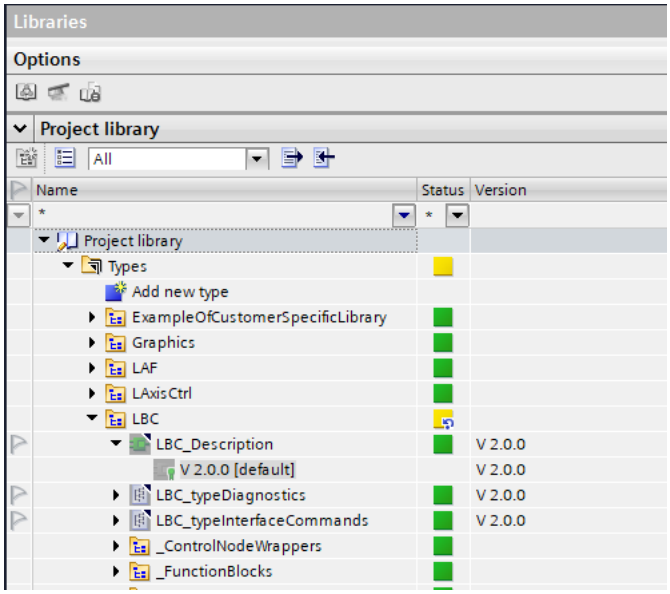


Figure 4-2: Library management.

The following notes introduces one example of library inconsistency and how to resolve them:

Use case with two inconsistencies at the same time deadlock auto-resolve

- The latest UDT version (V0.0.5) is not the default version.
- The current default version (V0.0.4) does not use ONLY default types.

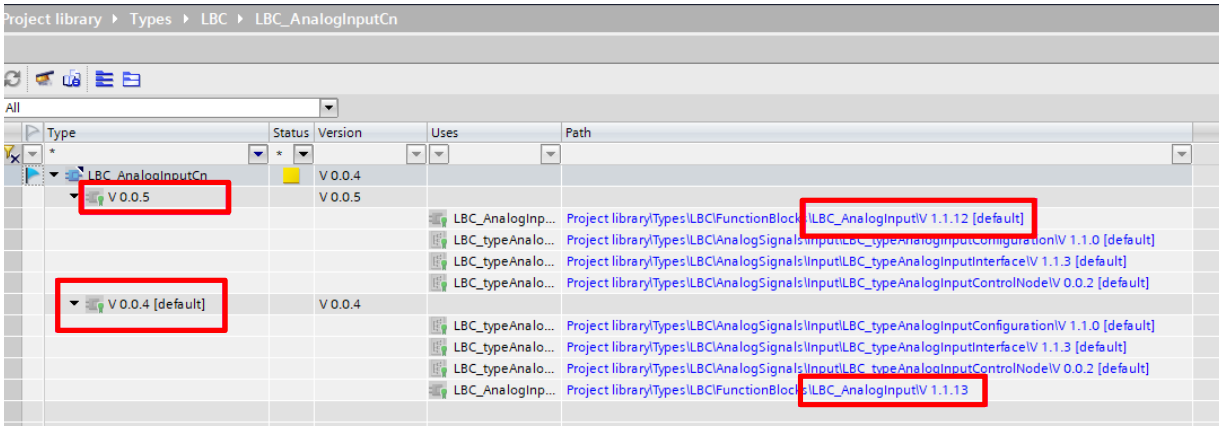


Figure 4-3: Library management.

The displayed information helps understanding the current inconsistencies:

- Which Types are used and its folders as clickable hyperlink to open the target.
- The Version of each Type used.
- Which version are in status [default].

Resolution: Update the interface of an HMI type to the default (latest) version.

- Open the HMI Type - Right click on the type and choose "Edit type".
- Go to Tag interface and check which UDT version is used.

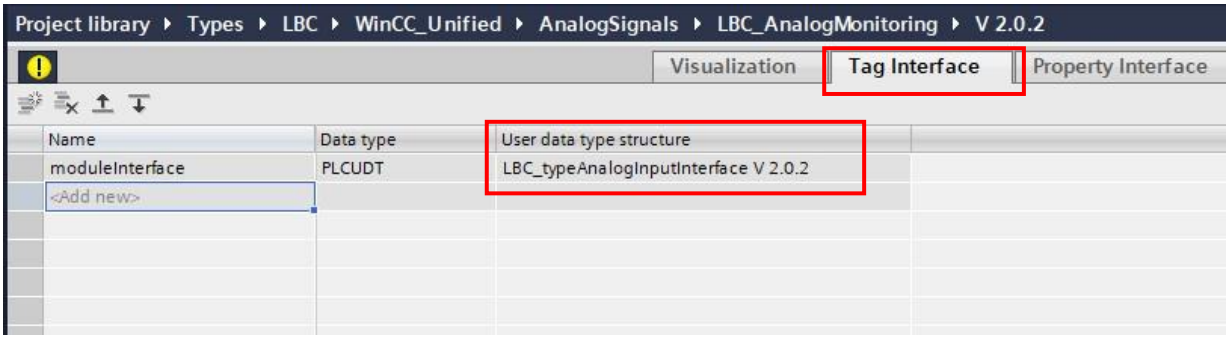


Figure 4-4: Tag Interface.

- Check in the Library Management which is the default version.

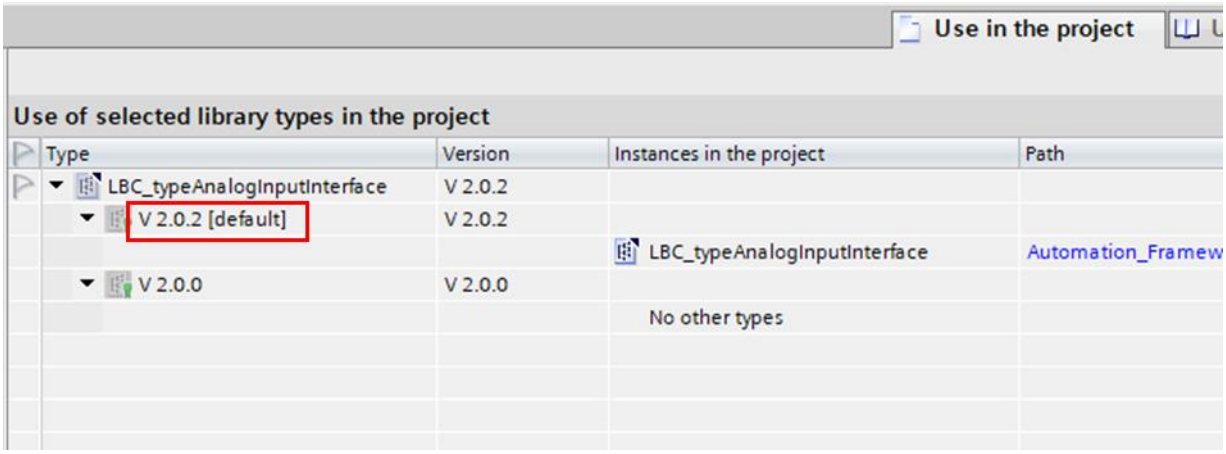


Figure 4-5: Library management.

- Connect the default version to the UDT in the Tag Interface of the HMI Type.

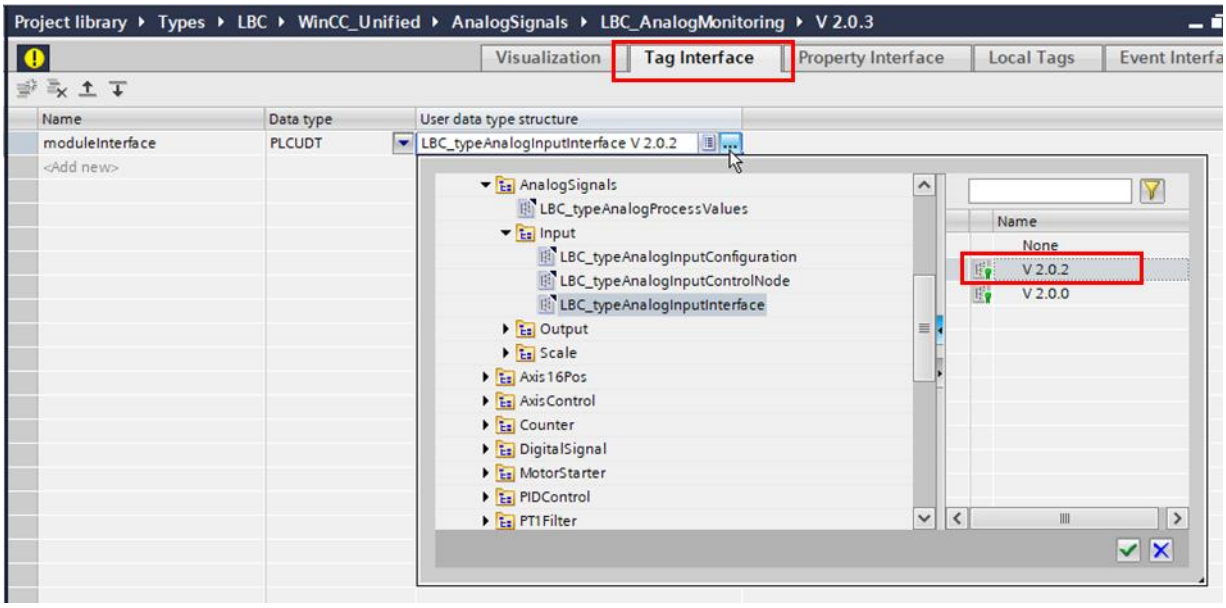


Figure 4-6: Update interface UDT.

- Now you can release the HMI type as new default version.
- Please always set both checkboxes to stay consistent.

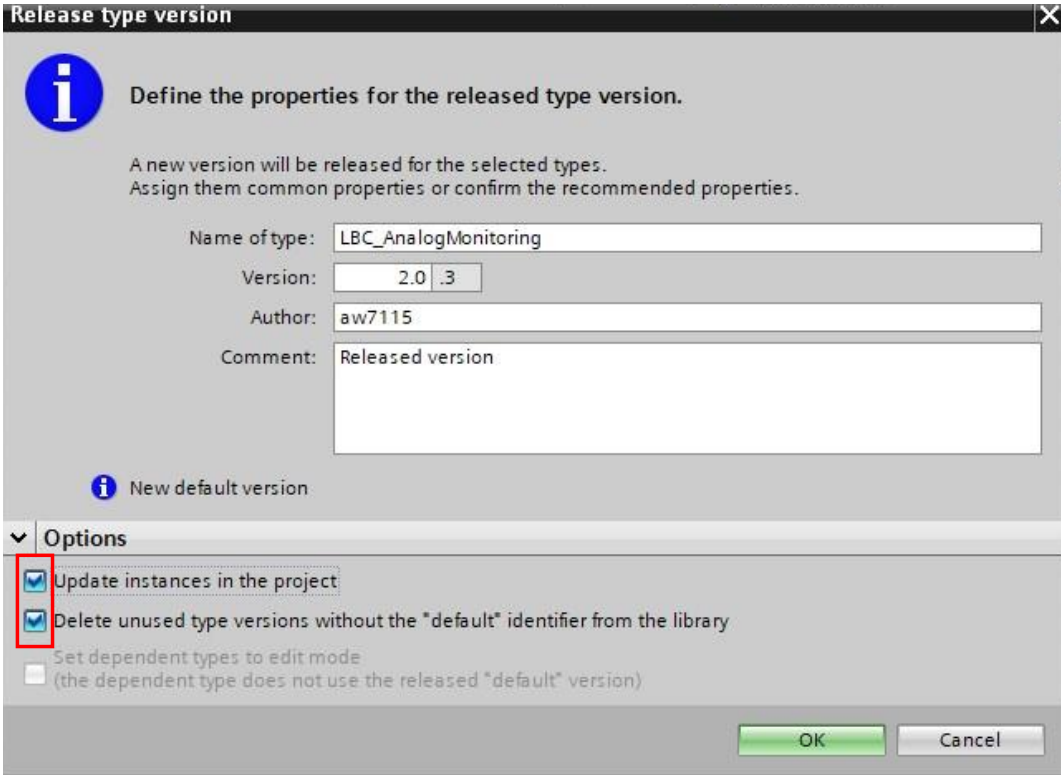


Figure 4-7: Release type.

NOTE If other library inconsistencies occur, they need to be fixed in a similar way.

4.3. How to use a provided library

The AF comes with a Global library where most of the provided objects, not all, are of typed objects. The user is highly encouraged to reuse and adapt the provided features to his needs but to ensure updateability the following information is important to understand.

Delivered Types

Use the type objects like they are delivered, do not modify them.

- Your modification will be overwritten with the next library update (newer version) of this type.
- The next library update may not work automatically concerning version conflicts.

Modify Delivered Types / Create own Types (within AF library content)

All AF library types are protected which is like a write protection.

These blocks cannot be duplicated nor modified. If you see the need to create your own derived type from such a block you can download the unprotected version of the specific library within the SIMATIC online support page – see chapter Used Libraries to find the links. After the download, please follow the next steps in the following chapter How to update the project with Global Libraries.

Use case - Modify Delivered Types / Create own Types (in general)

You receive a TIA Portal library and like to modify a type (or block) e.g. adapt the interface, along your specific requirements.

The recommended way is to create a duplicate of the type, you would like to modify in the library. The duplicate type is now a new type (new type-id) which can be modified along your needs.

Follow this procedure:

- Duplicate the original type from the library into the PLC.
- Select the type-object in the library.
- Right-click and choose "duplicate type".

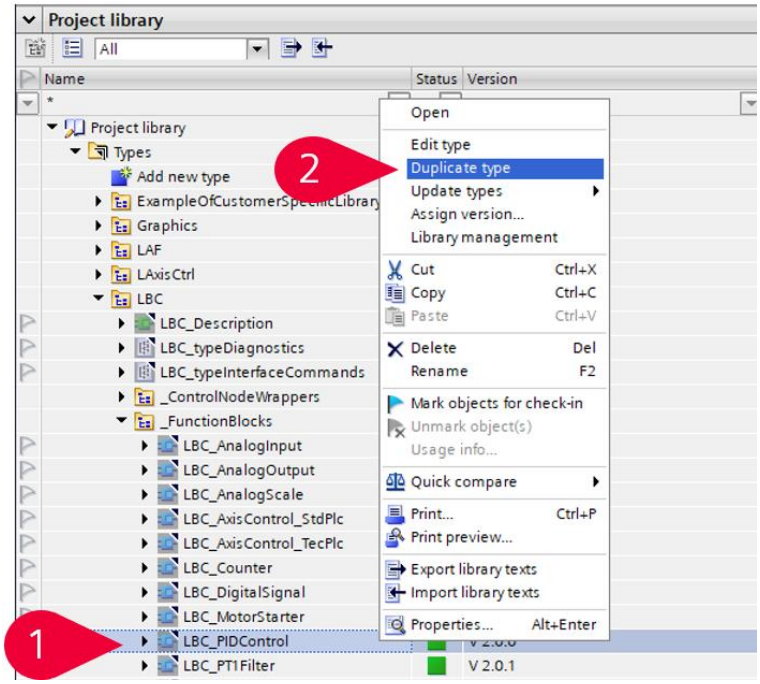


Figure 4-8: Duplicate a type.

- Save the Type - It is recommend to fill all the properties of the new type.

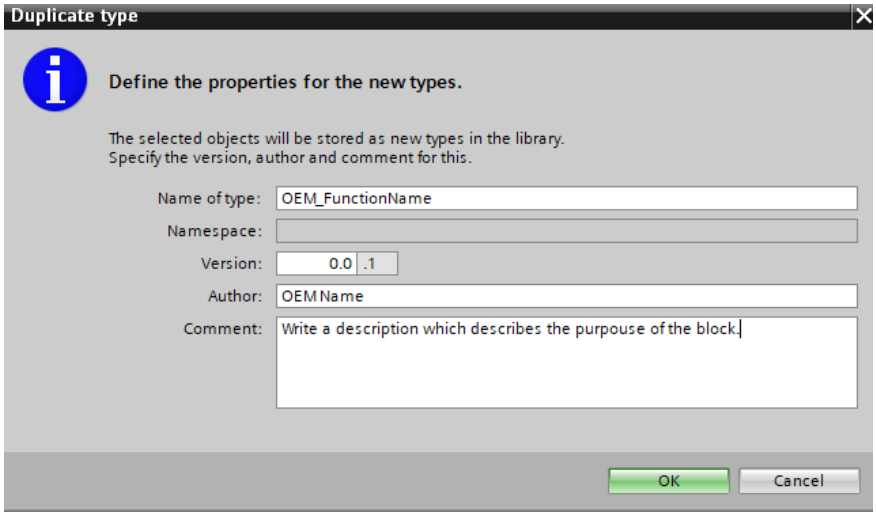


Figure 4-9: Save duplicate type.

- Now you can modify your new type along your needs, without interference the update chain of the original type.

NOTE Now, you are responsible to keep your type aligned with bugfixes or improvements of the original type, so keep an eye to the upcoming updates of the provided library.

Use case – Use your own type in the project

If you already used the original type in your project and now like to use your new duplicated type, you need to change the call (PLC or HMI) from the original type to your own type manually.

Use case – You already modified an original type

If you modified an original type and you receive a new version of the Global library as an update.

- Before you start the update, create a duplicate of the modified types as described above.
- Now you can run the library update without interference to your modification.

Versioning

If you release a new (own) type or version the "Release type version" box will appear.

Do not change the "default version". The new created version is and should be always the default version.

- Check always (if available) "Update instance in project".
- Check always "Delete unused versions without the "default" identifier from the library". Thus prevent to use the wrong versions.
- Check always (if available) "Set dependency types to edit mode". Thus prevents the library from inconsistencies.

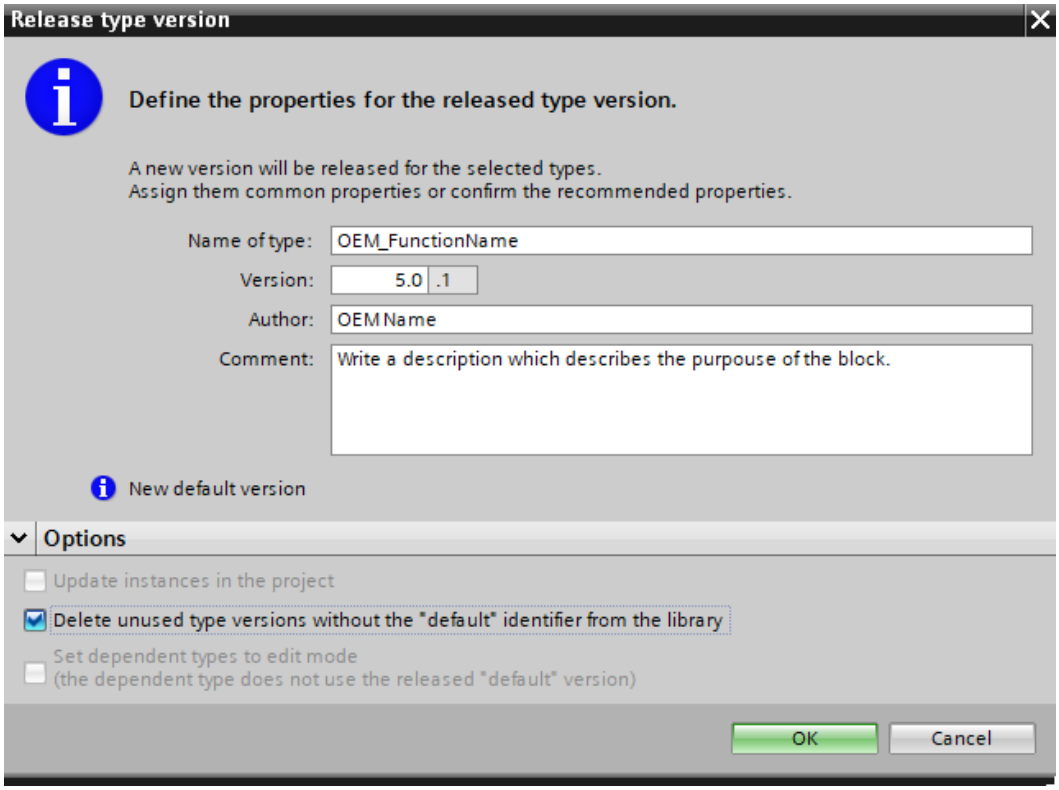


Figure 4-10: Release all types, non-default versions get deleted.

- You can use the Release All command to release all types in edit at ones.

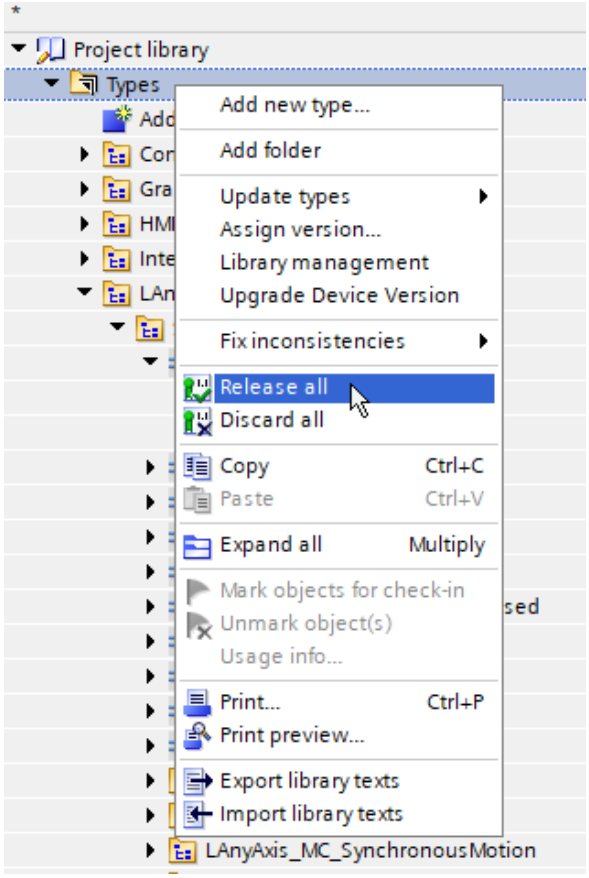


Figure 4-11: Release All.

If you consider these tips, the whole library update procedure will be a small effort and not very time consuming for you.

4.4. How to update the project with Global Libraries

Global libraries provide an efficient possibility to transport new versions of typed objects from a central development project to the machine projects.

NOTE Save your current Project as a backup before you do the next steps.

Navigate to global libraries section

- Go to Libraries section.
- Click on the "Open Global library" symbol.

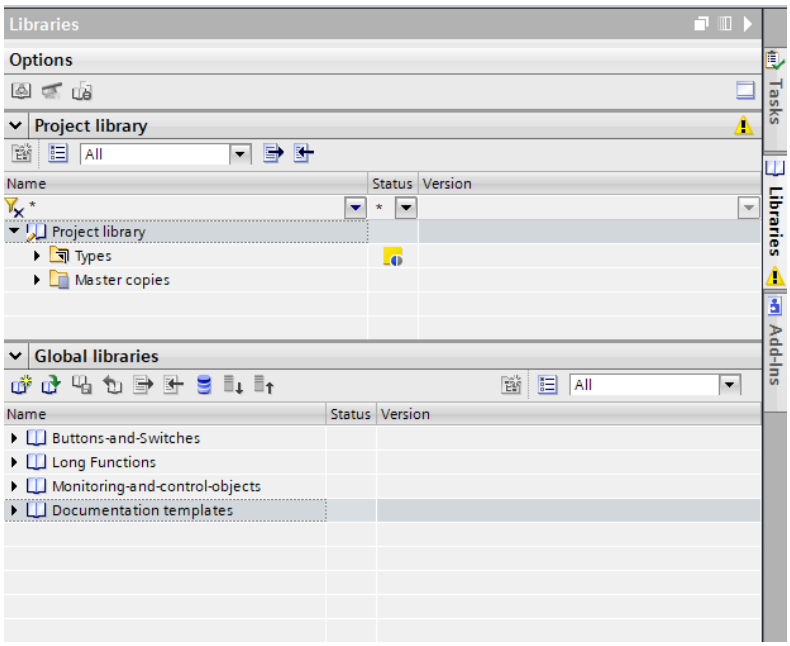


Figure 4-12: Open a Global library.

Select and open the Global AF Library

- Select "Compressed libraries" as file type.
- Choose the library that you want to use for the update.
- Open the library – it is write-protected permanently.

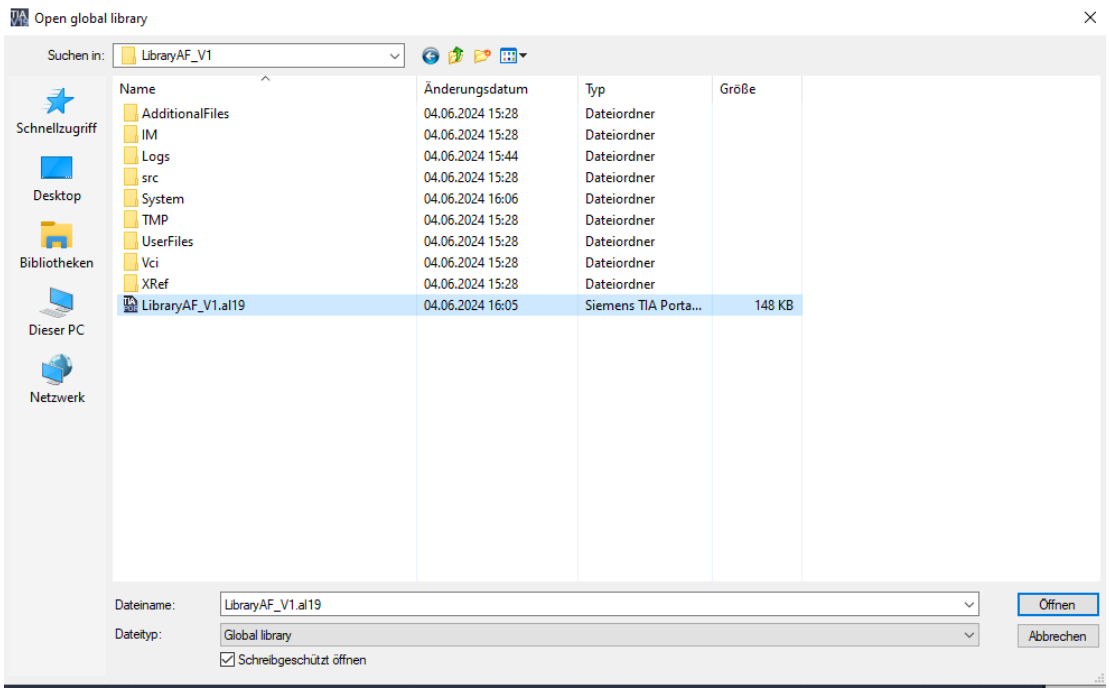


Figure 4-13: Open Global library.

Update the Project library with new AF Library items.

- Right click on the "Global Libraries".
- Select the "Update types" option.
- Select "Library", so that all the types and instances of your Project will be updated.

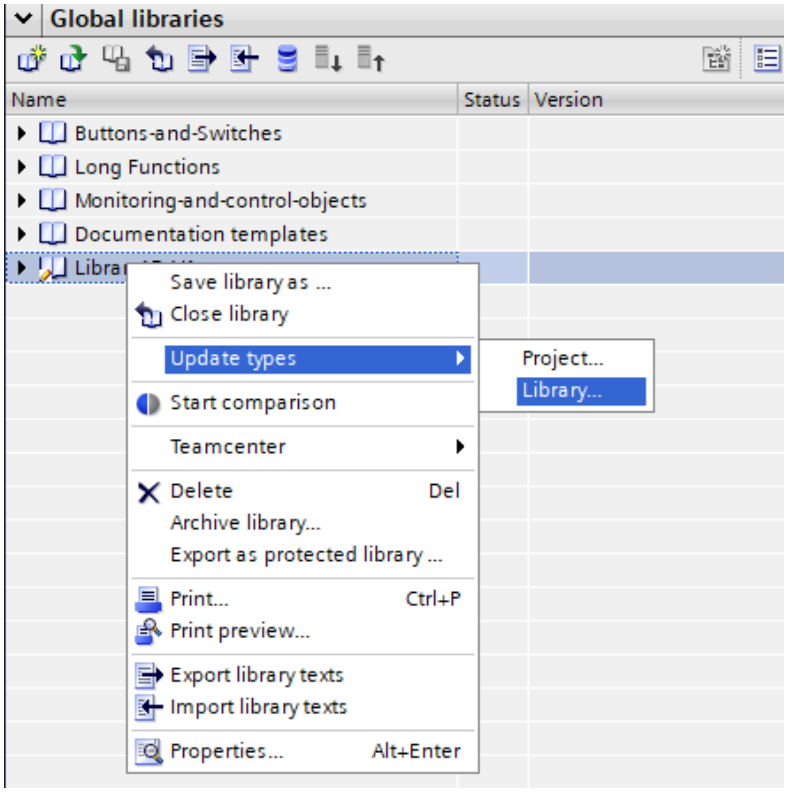


Figure 4-14: Update types from Global library.

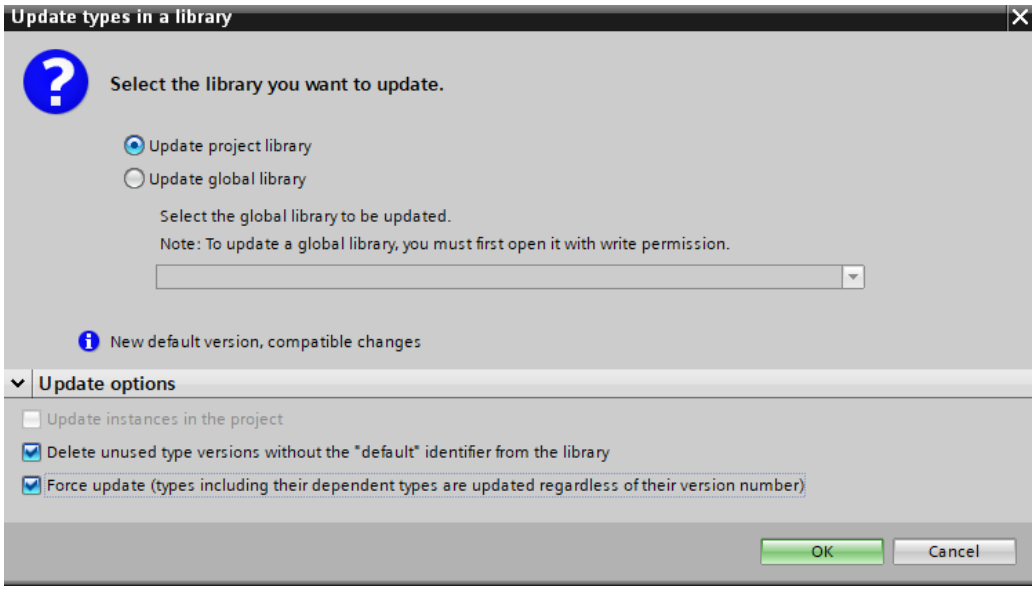


Figure 4-15: Update types in Project library.

- Activate "Delete unused type versions without the "default" identifier from the Library" to clean the Library.
- Activate "Force update (types including their dependent types are updated regardless of their version number)".

Choose devices to update

- Select all the devices in your project which should be updated (1).
- Activate "Delete unused type versions without the "default" identifier from the Library" to clean the Library(2).
- Activate "Force update (types including their dependent types are updated regardless of their version number)" (2).

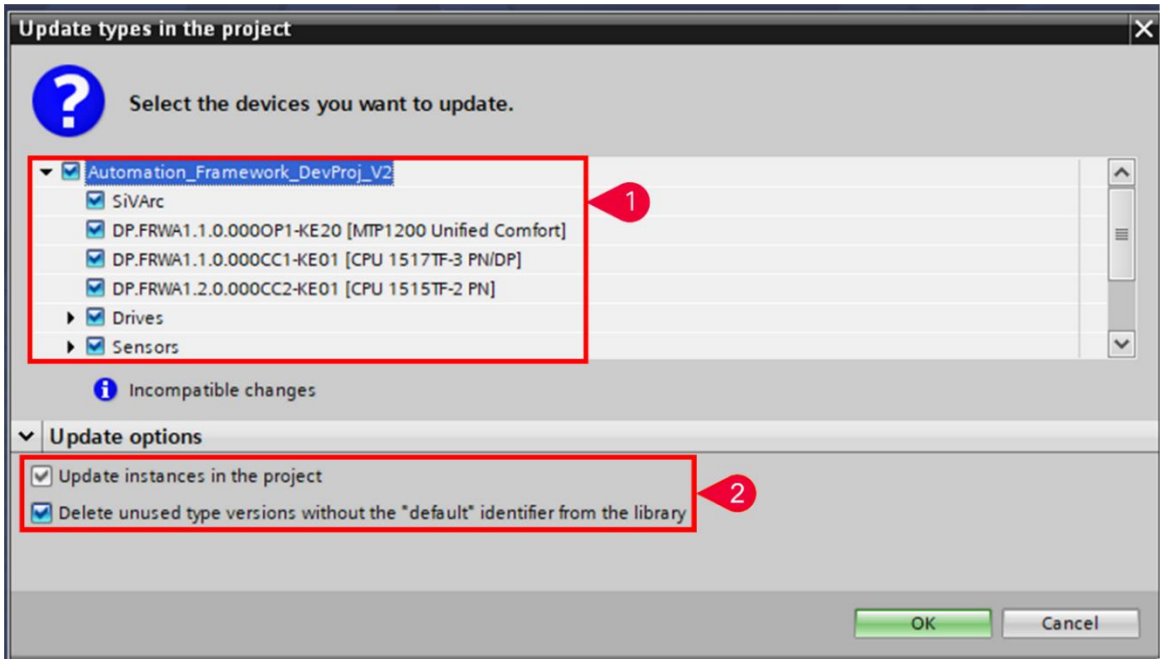


Figure 4-16: Update types in project devices.

4.5. How to update Master copies

The master copies need to be integrated via copy and paste or drag and drop into the Project library.

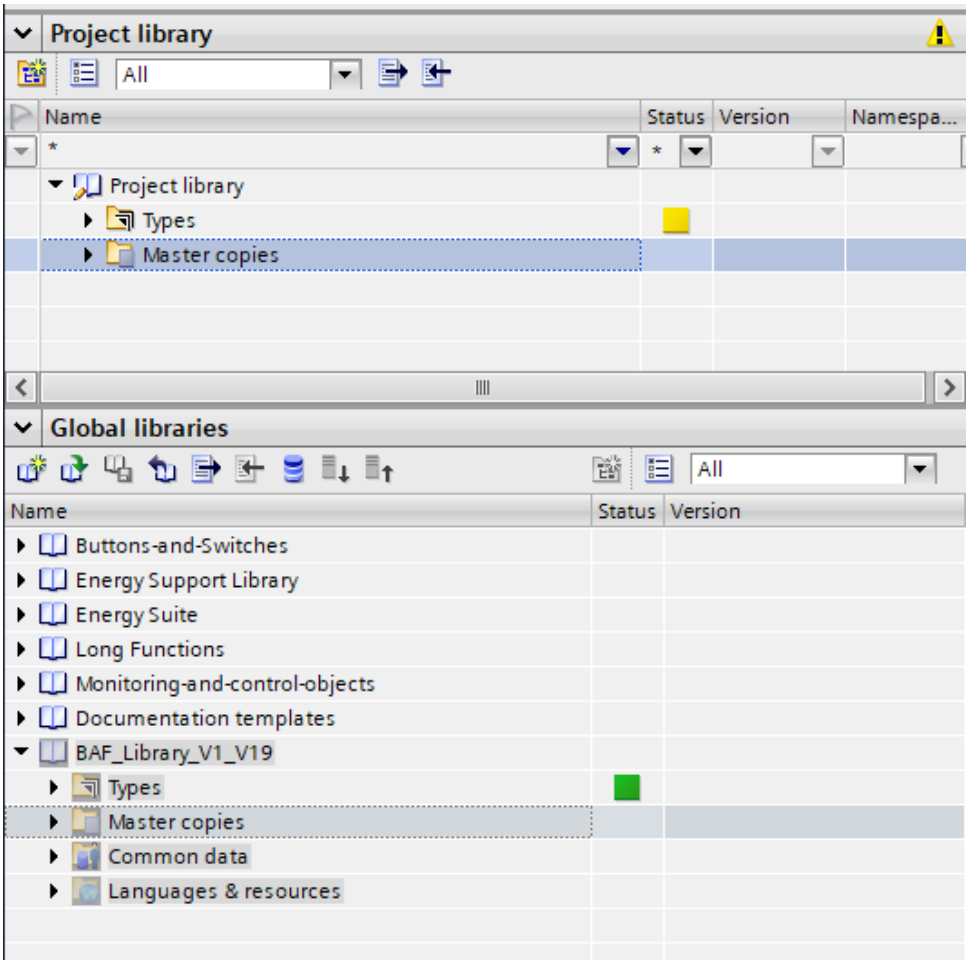


Figure 4-17: Import Master copies.

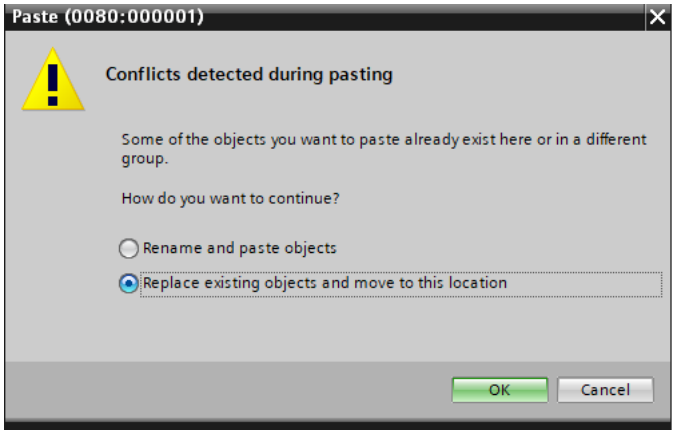


Figure 4-18: Overwrite existing items.

If this message appears, choose "Replace existing objects and move to this location".

Now the Project library is updated with the Master copies from the Global library.

Best practice if you already used a master copy and modified it in your project:

To avoid losing own implementation (only if you modified something) in your project by implementing the new master copies, the following procedure can be used as best practice:

- Drag and drop the new master copy to the target device location and choose option "Rename and paste objects".

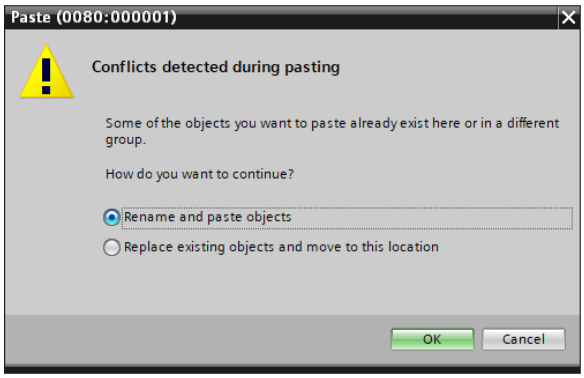


Figure 4-19: Rename and paste objects.

The result will look like this:

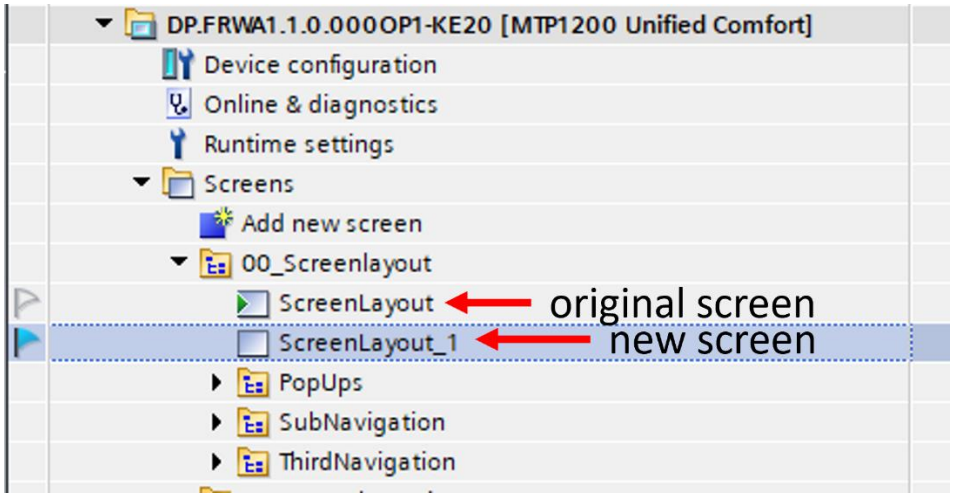


Figure 4-20: Renamed object.

Now you can open both screens in the split screen view and transfer the objects to your screen along your needs.

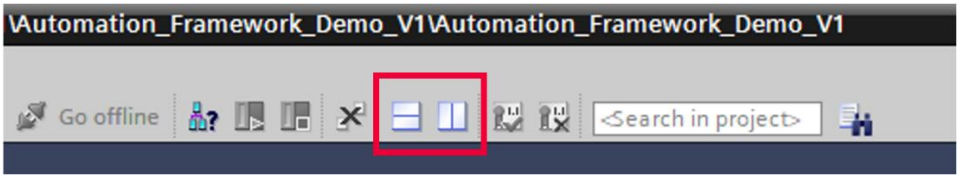


Figure 4-21: Split screen view.

5. Hardware configuration

The AF project comes with a pre-configured application example. In this chapter the selected HW configuration is introduced.

5.1. PROFINET IO with RT / IRT configuration

PROFINET IO is a scalable real-time communication system for fast Ethernet. There are two performance levels available, PROFINET RT for real-time-critical process data and PROFINET IRT for high-accuracy and isochronous processes.

IRT communication can be combined with RT communication on the same network. You can use IRT communication in one of the two IO systems. Use IRT either in a higher-level or in a lower-level IO system.

The AF project introduces an IRT configuration with two S210 and an RT configuration with IOs, HMI, and two G120.

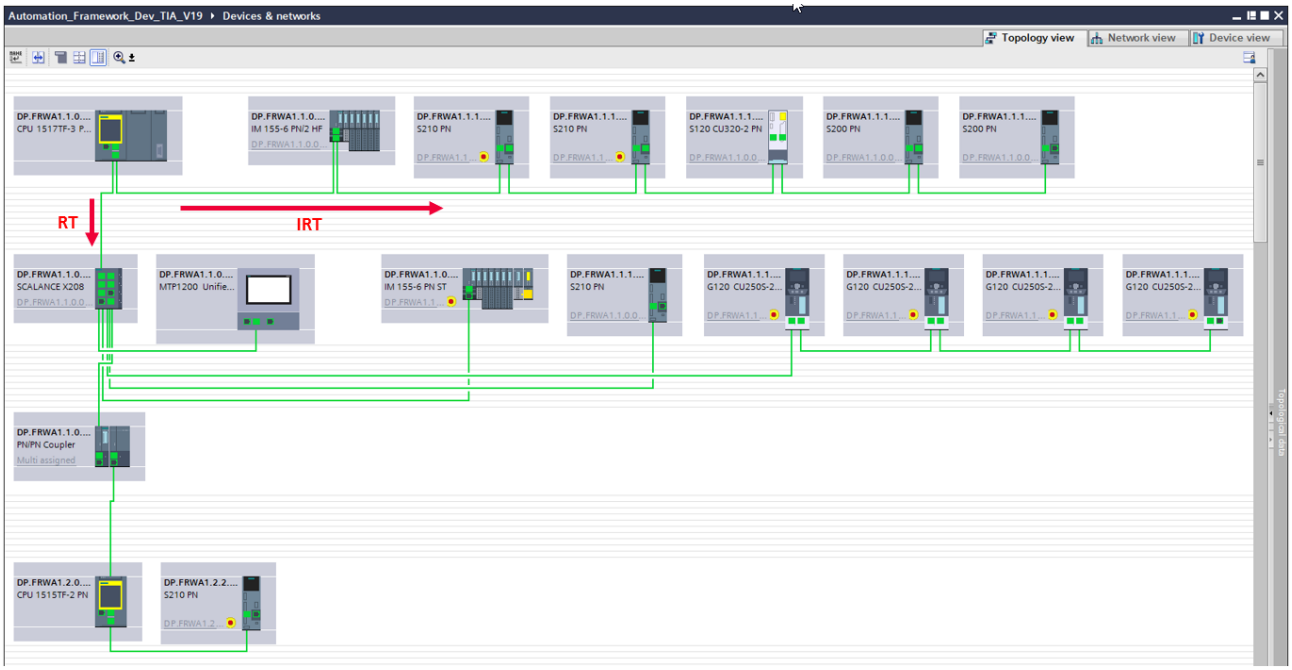


Figure 5-1: Topology-view with RT and IRT configuration.

With activated IRT, there is the option "Dynamic Servo Control" (DSC) on the drive configuration. DSC has the advantage of outsourcing parts of the motion control close loop from the PLC onto the drive. This heavily reduces the CPU load and enables highspeed position control executed in the drive HW (see highlighted in the next figure).

NOTE

By default, IRT support up to 64 network devices. If more than 64 IRT devices are needed it can be realized with PROFINET DFP (dynamic frame packing) and the software controller S7-1508S T/TF.

The following SIMATIC/SINAMICS components support DFP (as of 06/2024):

- SIMATIC S7-1508S T/TF
- SIMATIC ET 200SP (HS)
- SIMATIC PN/PN Coupler
- SINAMICS S200
- SINAMICS S210 (6SL5310...)

Further information regarding DFP can be found in the function manual "PROFINET with STEP 7".

SIOS-ID: 49948856

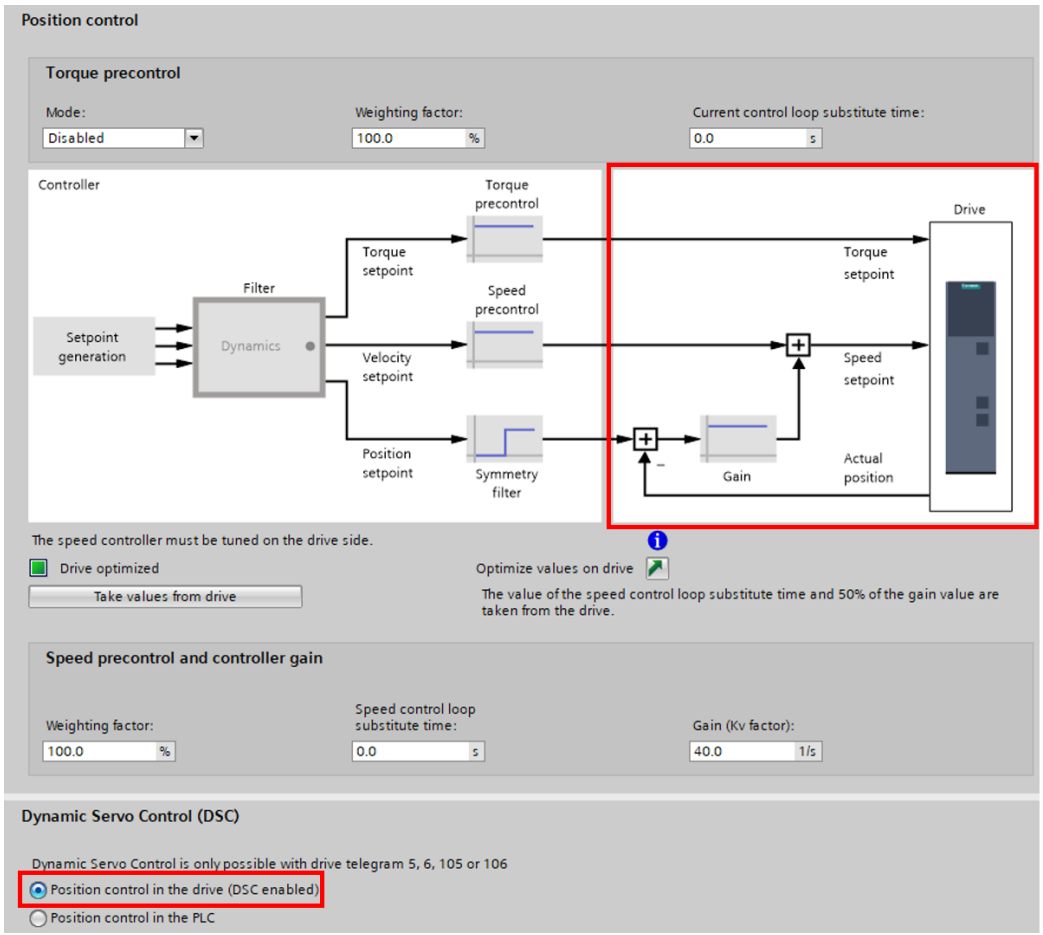


Figure 5-2: DSC enabled with PROFINET IRT configuration.

5.2. PROFINET Topology

Topology configuration is a prerequisite for IRT. The programmer must connect the specific ports between devices, exactly as it is connected in the real machine.

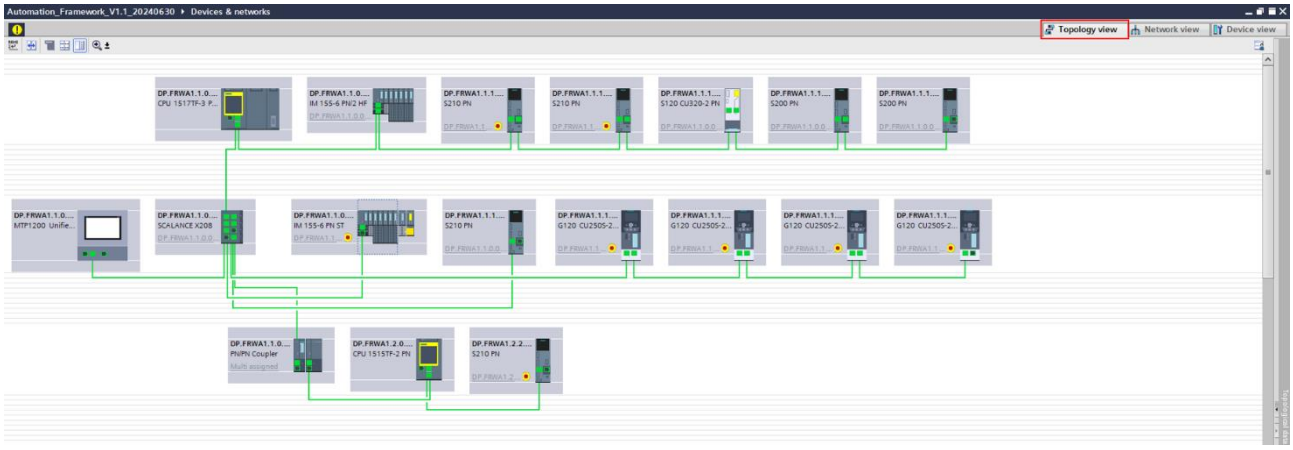


Figure 5-3: Topology view.

The topology information is used to optimized data transmission, bandwidth allocation, and communication frames which guarantees best PROFINET performance.

NOTE More information about PROFINET and all features can be found here https://support.industry.siemens.com/cs/ww/en/view/49948856.

5.3. Time synchronization and time setting

The system diagnostics with alarm logs benefit when the log timestamps from the subsystems are time synchronized with the PLC date-time. Sometimes the PLC date-time itself is synchronized with some higher-level system using NTP network protocol. The following chapter will introduce the necessary settings to enable a cascaded, system wide time synchronization using the LSNTP library.

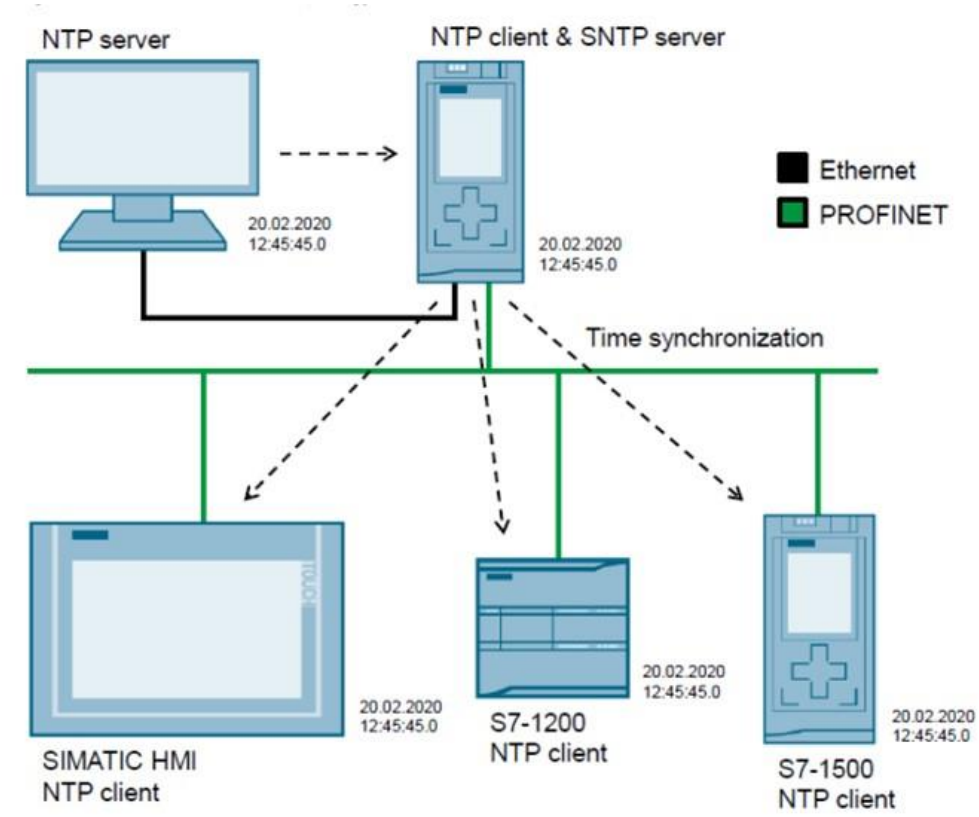


Figure 5-4: Time synchronization network overview.

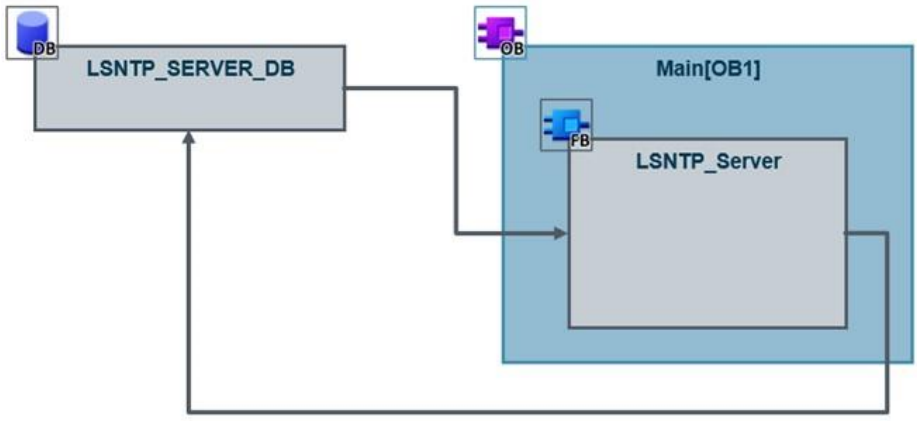


Figure 5-5: Call structure.

Technical concept

The use of a SIMATIC S7 CPU as an SNTP server enables flexible and simple synchronization of systems and subsystems, for example, to obtain meaningful timestamps for error messages and logging data system-wide. The library LSNTP provides a function block that performs the following functions:

- Receiving and evaluating an NTP telegram from an (S)NTP Server.
- Creating and sending an NTP telegram to the client for time synchronization.

The inaccuracy of the SNTP server is less than 10 ms.

In case the higher level of NTP, where CPU as a client is not used, CPU time can be set from "Settings – Date and Time" screen. This sends the time and date through the variables in Settings.setLocalPlcTime to the FC "LAF_SetPlcDateAndTime" in Central Functions – 14_Settings.

NOTE The use of an external NTP-Server is optional.
NTP client function in variable frequency drives, is currently supported only by S120 and S210.

How to use (see Software Unit CentralFunctions)

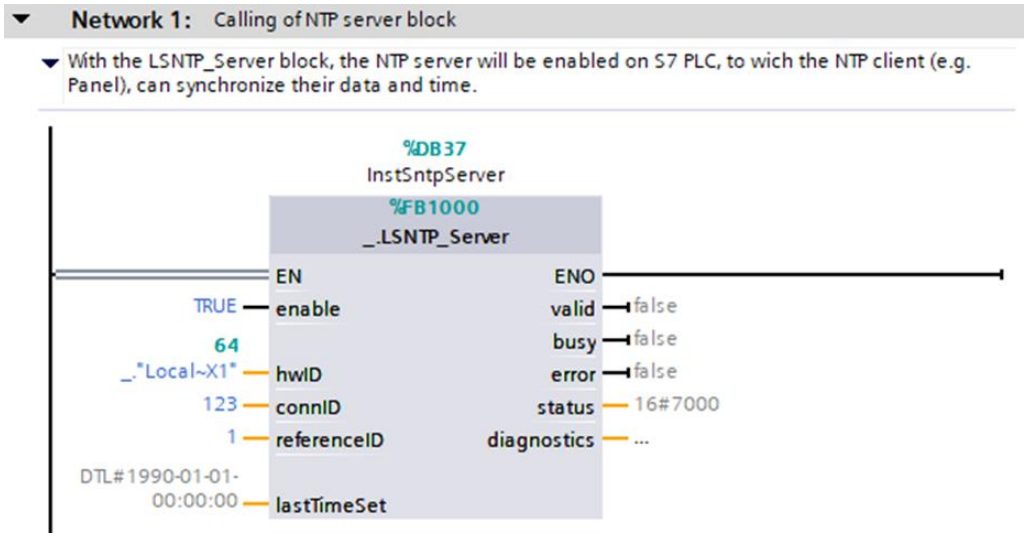


Figure 5-6: LSNTP_Server block activates the NTP server on the primary PLC, called in OB1 – Main.

- All the NTP clients to which the IP address of the NTP server (Primary PLC) is added synchronize to the date and time of the NTP server.
- To configure a unified comfort panel as NTP client the following setting must be configured in TIA Portal. To finish the configuration a download is needed.

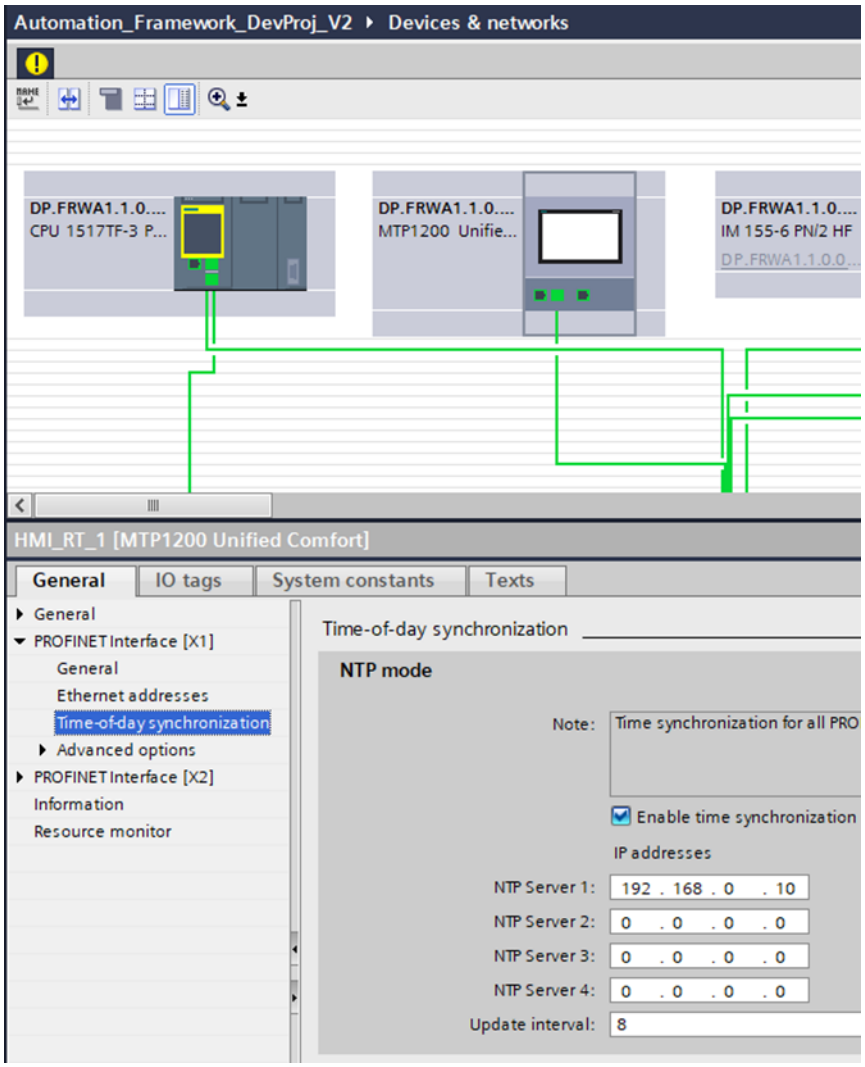


Figure 5-7: HMI settings in TIA Portal: IP address on which the LSNTP_Server block is initialized.

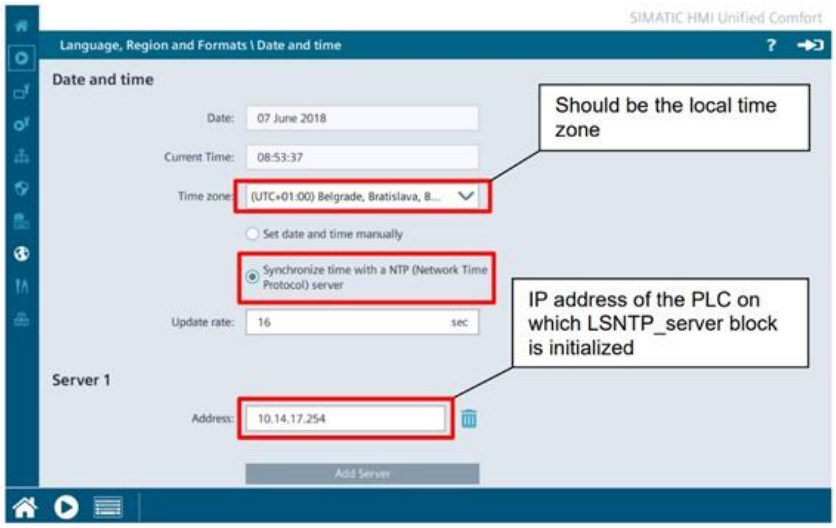


Figure 5-8: HMI setting directly on the panel.

- To enable the NTP client of each PLC the following settings must be configured in TIA Portal. To finish the configuration a download to the PLC is needed.

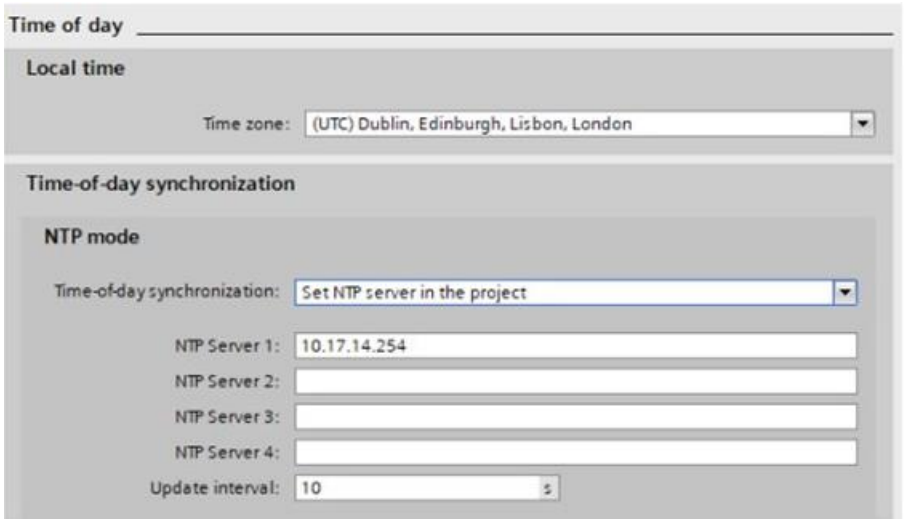


Figure 5-9: Configuration of NTP server.

- In case the PLC time is not synchronized via an external NTP server, the PLC time can be set manually via the AF HMI.

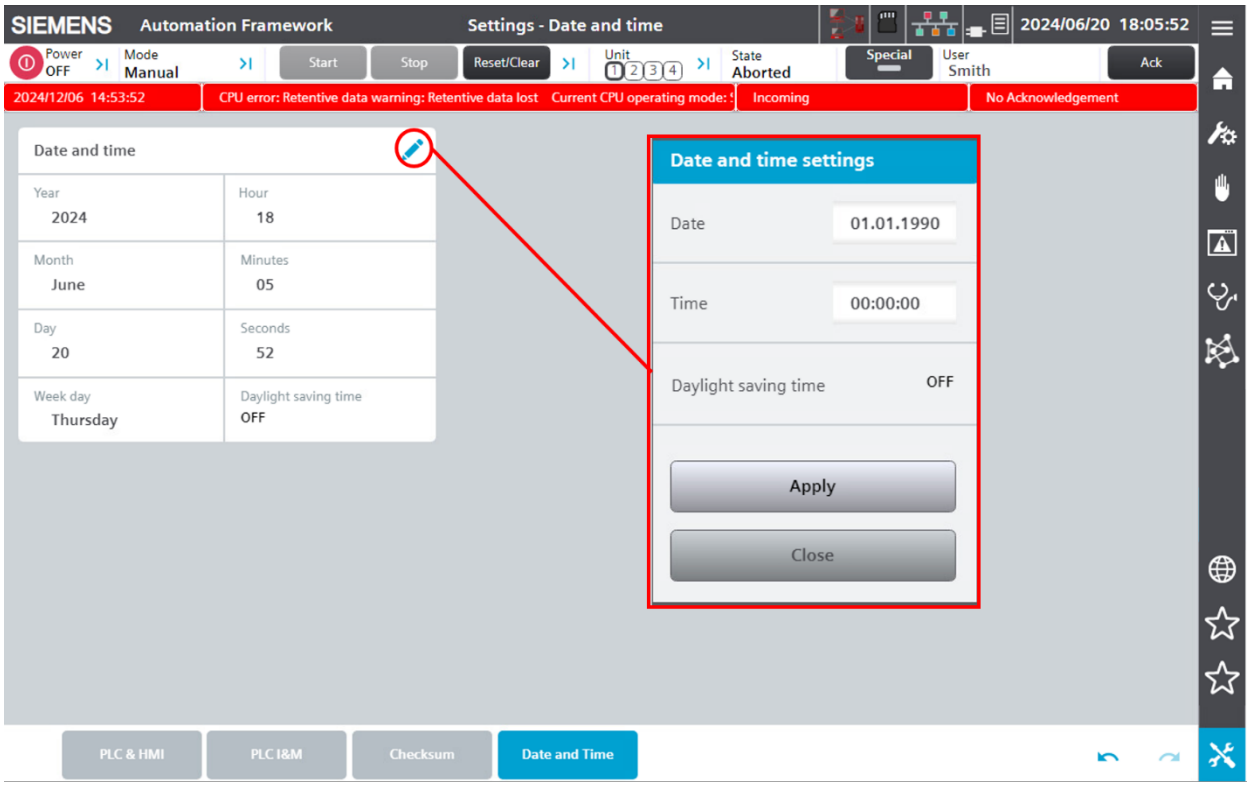


Figure 5-10: Manual setting PLC time and date via the AF Settings screen.

6. PLC Software Architecture

The description of the PLC software architecture contains information about the structure of the software, but also information about the communication between program parts.

6.1. ISA-88 design

ISA-88 introduction

Industrial processes can be classified as continuous, discrete or batch manufacturing processes. Continuous processes are classified having a continuous outflow, for example production of paper or steel. Discrete processes have a discrete output, for example the production of a car.

A batch process is a process that leads to the production of finite quantities of material by subjecting quantities of input materials to an ordered set of processing activities over a finite period using one or more pieces of equipment.

There was a need to provide a standard for managing batch processes because of the following reasons:

- No universal model for batch control.
- Difficult for plant operators to communicate processing requirements.
- Integration of solutions from different vendors is difficult.
- Batch control solutions are difficult to configure.

Therefore, the ISA (International Society of Automation) decided to provide a standard which describes a method to structure batch processes. It defines terminology and models that makes design and operation of batch plants easier and uniform. This standard is named ISA-88, whereby only the first part of the standard (ANSI/ISA–88.00.01–2010), called "Models and Terminology" is considered in this document.

ISA-88 is more than a standard for software, equipment, or procedures. It is a design philosophy. Understanding ISA-88 will help to better design processes and manufacture products.

How does ISA-88 help?

- Modularity allows easier global replication and better return on investment.
- Isolates equipment from recipes.
- Design concepts make validation easier.
- ISA-88-aware solutions help track product and process data.
- Gathering requirements from customers and conveying requirements to vendors is easier.
- Provides guidelines on how to recover from abnormal events.

NOTE For more details about Batch Processes please go to the linked document.
https://support.industry.siemens.com/cs/ww/en/view/109784331

Concept

The ISA-88 is an important foundation for the modular programming structure of production machines. The AF project introduces an implementation example of an ISA-88 conform physical model from a unit down to control modules (CM).

The following figure shows such hierarchical implementation.

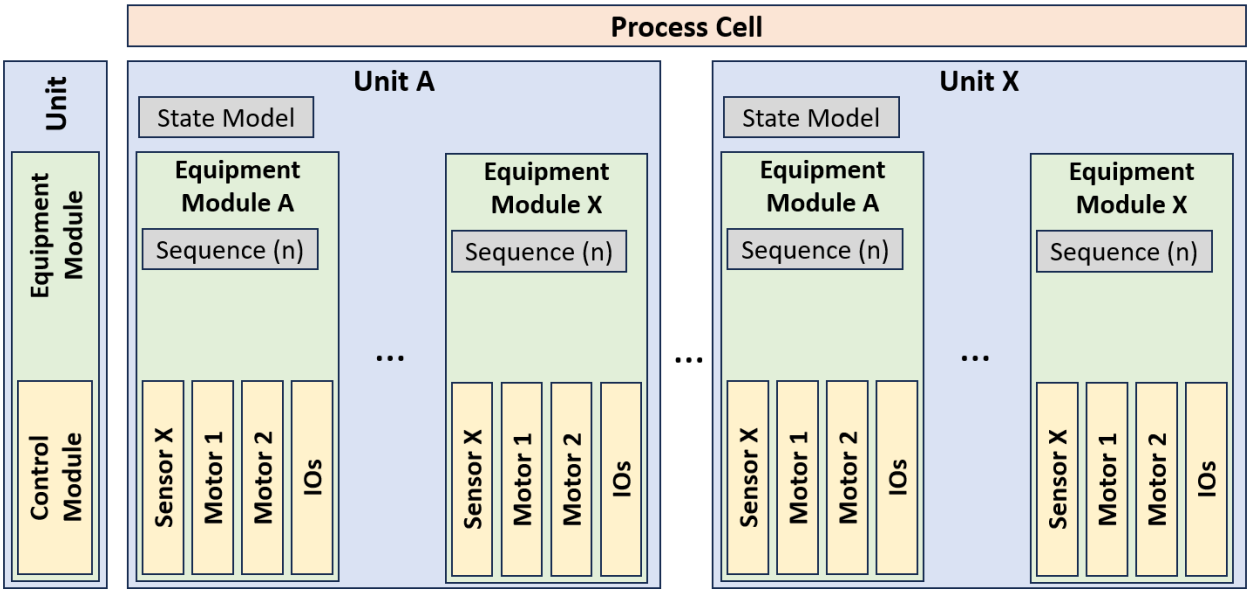


Figure 6-1: ISA-88 model.

Unit classification

A process cell contains several units. The units represent a physical processing equipment in production, for example a reactor. The unit contains equipment modules (EMs) which contain control modules (CMs). A unit is an independent grouping of equipment usually centered around a major piece of processing equipment, such as a mixing tank or reactor, and it can perform one or more major processing activities — such as fermenting, harvest or purifying.

Usually, the modules that make up the unit cannot be manipulated or controlled by equipment outside of its own unit. Any equipment that is considered common or manipulated by multiple units should be defined as shared equipment, which can be temporarily acquired by a unit to carry out specific tasks.

An important consideration when defining batch units, is that a unit can only operate a single batch at the same time. The batch may exist in multiple units, but a unit can hold only one batch at any given time.

Defining the boundaries for units is important to guarantee the flexibility of the plant, reusability of the modules, and consistent reporting over multiple plants. The Process flow diagram (PFD) can be used to initially identify the units. Once the P&ID details are known, the unit can be further refined. In case of modular integration of intelligent unit operations, the unit and its ISA-88 structuring are defined by the OEM supplier.

Control Module (CM) classification

Once the unit boundary is defined, the lower-level modules of the equipment model can be identified.

It's easiest to start with the control modules (CM). These are the lowest level grouping of equipment in the physical model that can carry out basic control (state-oriented or regulatory control) and typically have a direct relationship to the type of instrument/device that it is being monitored and/or controlled. Examples are pumps, motors, valves, but also PID control loops. The CMs are typically directly connected to IO signals. CM also contains exception handling related to equipment, product and operator protection. The CMs are easily identified on the P&ID charts, as shown in the following figure. The control modules can be controlled via the HMI by the operator in manual mode, or by the superior CM's or EM's in auto mode.

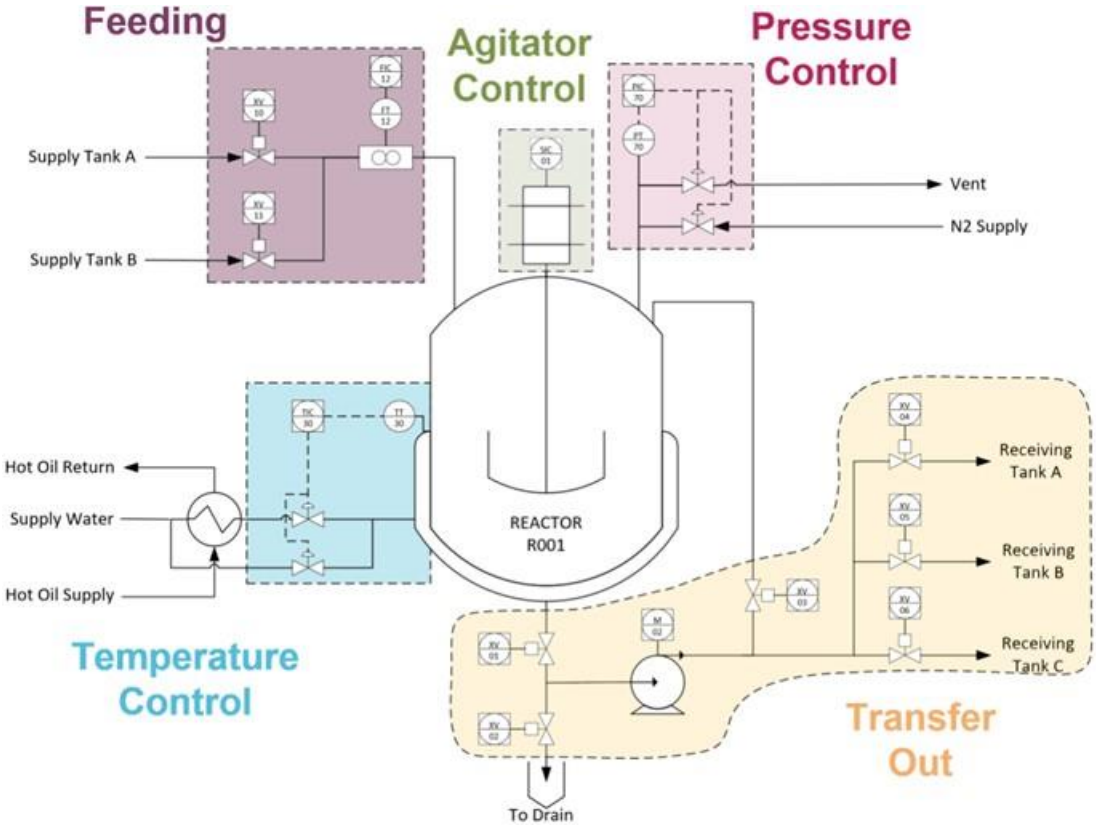


Figure 6-2: Example P&ID from process unit "Reactor" with its EM's (highlighted) and CM's.

Equipment Module (EM) classification

An EM is a functional group of control modules that can carry out a limited number of specific minor processing activities such as dosing, charging or temperature control. An EM is typically centered around a piece of process equipment such as vessel outlet/inlet manifold, process heater etc. The separation of a unit into functional EM should already reflect the process actions and recipe phases from the process and procedural control model. Introducing a new product on the same equipment should be possible without altering the base sequences of the EM's. The EM's of our example reactor are shown in the figure above: feeding, temperature control, pressure control, agitator control and transfer out.

According to ISA-88, it is possible for an EM to contain other subordinate EMs, although this is recommended to be restricted to areas where there are no phases or batch requirements.

An EM is usually a sequence in which all the subordinate CM's are monitored and controlled. The EM also contains exception handling logic that will handle failure of subordinate CM and process failure. Usually, the exception logic is labelled as held state or held logic and drives the EM to a failsafe state. The EM itself does not have mandatory state logic defined in the ISA-88. An EM is usually recipe-aware and it therefore has a state logic that follows the state logic of the recipe phase.

As the sequence of the EM reflects the process action, it is common to enable and disable control module related alarms and process interlocks or change alarm limits throughout the sequence. This must be considered within the design of such command sequences.

Arbitration of control modules shared between EMs within the same unit will be performed at the EM level.

Arbitration of EMs shared in the same unit or between units is performed at the phase level. Each phase will acquire the equipment modules needed to perform a specific process action. It is also possible to acquire EMs on another unit in the same way.

When partitioning the EMs within a unit, the following criteria should be applied:

- All CMs in the EM must support a similar processing purpose and be self-contained.
- The EM must be reusable, so that a type-instance concept can be used.
- The module has to be designed so that it can perform all of the available processing functions, see above EM must have the ability to operate on its own without other EM interactions.
- Ability to minimize the effects of process exceptions by containing them within the EM; ideally an exception will initially only affect one EM before being propagate to other EM's or even Units.

Like the CMs, the EMs can be controlled by the operator in manual mode, or by the phases in automatic mode.

It's important to note that the CMs and EMs should ideally follow the modularization approach where functionality is contained within modules that can be programmed and tested as standalone with a defined interface towards other modules. To increase reusability, within TIA Portal support for type-instance is made for the control and EMs. Similar EMs and CMs can be grouped in a type. This type can be tested and then all modules will be instantiated from this type.

By creating these standard ISA-88 batch elements, tremendous benefits can be achieved:

- savings in software designing and engineering
- faster testing and validation during the initial process of application implementation
- increased maintenance and flexibility - By means of the pre-tested standard types, the expansion of the installation can be realized in a faster way, with less (re-)validation effort.

Realization in the Automation Framework

The basic implementation of an ISA-88 unit is shown in the following figure. The logic of one AF unit is programmed within a dedicated Software Unit. Each EM also has a dedicated Software Unit. Each unit has one EM0_General which is reserved for unit wide functionality like preconditions, interfacing to circuit breakers or signal stacks or pneumatic system. The EM0_General usually contains non-complex logic and no sequences. Additional EMs on the other hand are process related. There is always at least one. The maximum number of EM is limited by the PLC memory size.

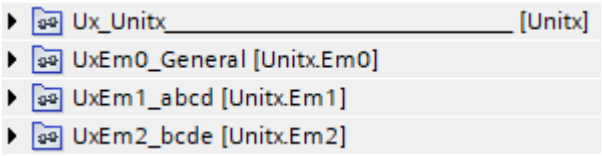


Figure 6-3: ISA-88 unit example in the Automation Framework.

The naming in the Software Units above are placeholders and must be adjusted depending on the real application. It is recommendable to name the Software units according to their functionality, not to their location, as they should be enabled to be cloned and reused in different applications.

6.2. State Machine

In the automation framework, each unit contains its own state machine, which can be customized according to specific requirements.

State Model

The mode and state completely define the current behavior and condition of a unit. Depending on the mode and state, the equipment and control modules execute their respective tasks. In a state model, states are arranged in an ordered fashion that is consistent with the specified operation of the machine. The mode state manager determines the current mode and state of the unit depending on operator commands and feedback from the process.

NOTE

PackML is an automation standard by the OMAC and adopted by ISA as TR88.00.02 that makes it easier to transfer and retrieve consistent machine data. The primary goals of PackML are to encourage a common "look and feel" across a plant floor, and to enable and encourage industry innovation.

OMAC's PackML group is recognized globally and consists of control vendors, OEM's, system integrators, universities, and end users, which collaborate on definitions to be consistent with the ISA-88 standards and the technology and changing needs of most of the automated machinery.

In the AF, the OMAC PackML state machine is the default state machine used. This state machine can be flexible reconfigured to comply with other state machines. Other state machine libraries could be used as well. However, that would imply major changes in the current AF implementation and shall be avoided if possible.

The unit's state machine it always in a defined mode and state. Mode and state are represented by two DINT variables. Various predefined commands trigger a mode or state transition. Such commands can be triggered from an operator (e.g. via HMI or command buttons), from external systems in the line (e.g. infeed station, line controller), or by the process itself (e.g. material low or stop due to internal error). The unit's mode state manager is responsible evaluating all these different commands and determining the currently active mode and state of the machine.

The following modes, states and commands are supported by the OMAC PackML state machine used in the AF:

Modes

0	Invalid
1	Production
2	Maintenance
3	Manual
4 .. 15	User definable

States

0	Undefined
1	Clearing
2	Stopped
3	Starting
4	Idle
5	Suspended
6	Execute
7	Stopping
8	Aborting
9	Aborted
10	Holding
11	Held
12	Unholding
13	Suspending
14	Unsuspending
15	Resetting
16	Completing
17	Completed

Commands

0	Undefined
1	Reset
2	Start
3	Stop
4	Hold
5	Unhold
6	Suspend
7	Unsuspend
8	Abort
9	Clear
10	Complete
11	StateComplete

Figure 6-4: Modes, states and commands that can be used in the AF.

The Library of Unit Control (LUC) provides the data types "LUC_typeModes", "LUC_typeStates" and "LUC_typeStateCommands" to represent the modes, states and commands. These data types are used inside and between the units and its EMs.

The following figure shows the PackML base state model. It represents the complete set of defined states, state commands, and state transitions. For each unit mode it is possible to define any set or subset of these base states depending on the application (see Figure 6-6).

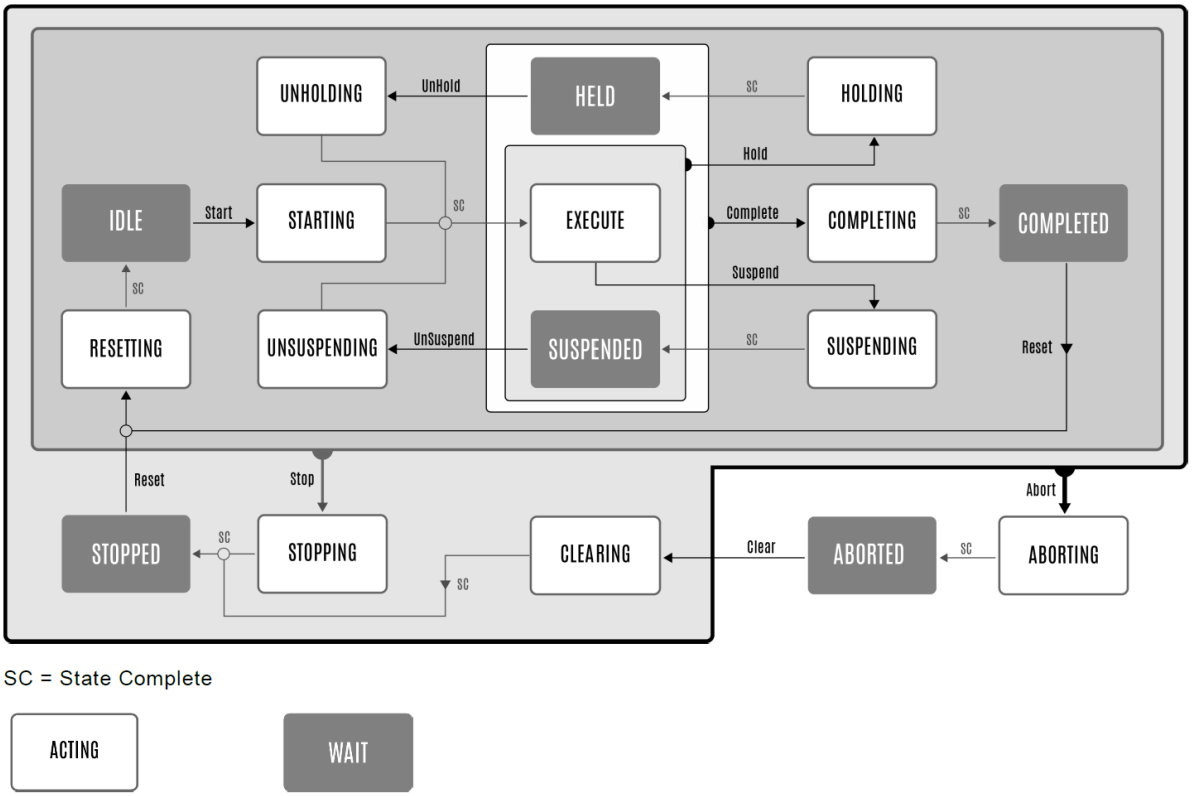


Figure 6-5: Base state model.

Configuration

The state machine can be configured. Up to 16 modes can be enabled. For each mode it can be configured which states and transitions are enabled.
With that, the state machine is highly customizable. Details are presented in the chapter about units.

Unit Modes

A unit can be operated in different modes, for example in Production, in Maintenance, in Manual, in Clean, in Jog Mode, etc. In each mode the unit can have different number of states and commands. Therefore, the state machine may look different for each mode (see Figure 6-6).

In the AF the following OMAC PackML standard modes are preconfigured:

- **Production** (mandatory):
  This is the production routine mode. When this mode is selected, the machine executes relevant logic in response to commands which are mainly coming from external systems or entered directly by the operator.

- **Maintenance** (optional):
  This mode allows authorized personnel to run the unit / machine independently from other systems. This mode would typically be used for troubleshooting, machine trials, or testing operational improvements.

- **Manual** (optional):
  This mode allows the direct control of single machine modules. It may be used for commissioning of individual components, fault diagnostics of unplanned technical intervention, etc.

Different or additional modes can be configured (up to 16 modes in total).

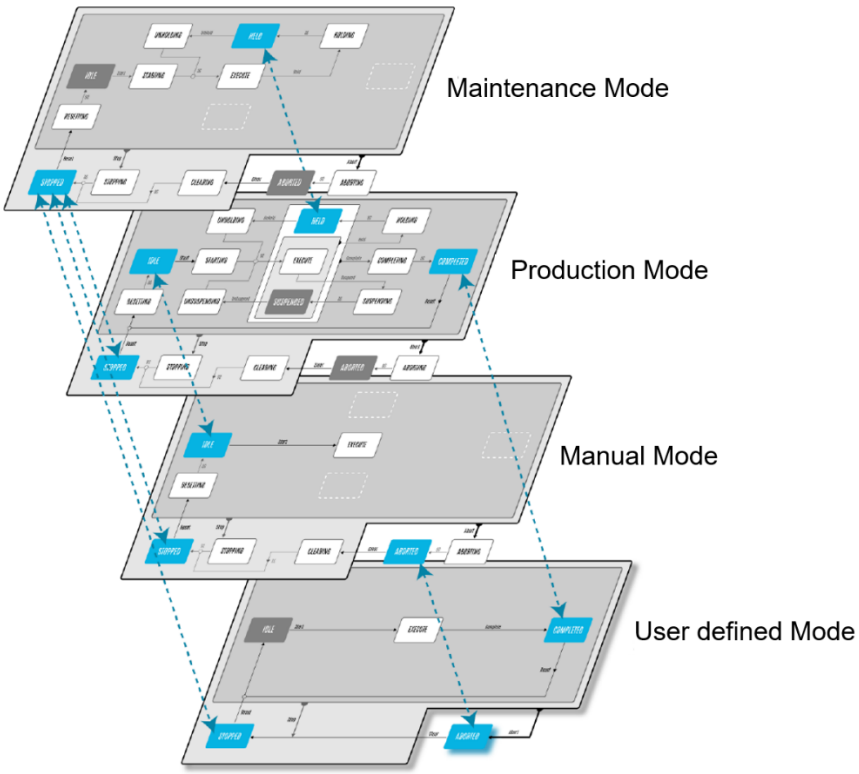


Figure 6-6: Example of different modes with the corresponding configured states and allowed mode changes.

Unit States

The different states can be grouped into two types:

- **Acting state:** A state that represents some processing activity. It implies the single or repeated execution of processing steps in a logical order, for a finite time or until a specific condition has been reached. In ISA-88.00.01 these are referred to as transient states, generally those states ending in "ING" like "STARTING" or "RESETTING".

- **Wait state:** A state used to identify that a machine has reached a defined condition. Once the state is reached the machine remains in this state until some transitioning is triggered via internal or external commands.

Definition of the possible states and description of what tasks typically should be executed in them:

- **Clearing:** Initiated by a state command to clear faults that may have occurred when Aborting, and are present in the Aborted state before proceeding to a Stopped state.

- **Stopped:** The machine is powered and stationary after completing the Stopping state. All communications with other systems are functioning (if applicable). A Reset command will cause an exit from Stopped to the Resetting state.

- **Resetting:** This state is the result of a Reset command from the Stopped or Complete state. Faults and stop causes are reset. Resetting will typically cause safety devices to be energized and place the machine in the Idle state where it will wait for a Start command. No hazardous motion should happen in this state.

- **Idle:** This is the state which indicates that Resetting is complete. The machine will maintain the conditions which were achieved during the Resetting state, and perform operations required when the machine is in Idle.

- **Starting:** The machine completes the steps needed to start. This state is entered as a result of a Starting command (local or remote). Following this command the machine will begin to "execute".

- **Execute:** Once the machine is processing materials it is in the Execute state. Different machine modes will result in specific types of Execute activities. For example, if the machine is in the "Production" mode, the Execute will result in products being produced, while in "Clean Out" mode the Execute state refers to the action of cleaning the machine.

- **Stopping:** This state is entered in response to a Stop command. While in this state the machine executes the logic which brings it to a controlled stop as reflected by the Stopped state. Normal Starting of the machine cannot be initiated unless Resetting had taken place.

- **Holding:** This state is used when internal (inside this unit/machine and not from another machine on the production line) machine conditions do not allow the machine to continue producing, that is, the machine leaves the Execute or Suspended states due to internal conditions. This is typically used for routine machine conditions that require minor operator servicing to continue production. This state can be initiated automatically or by an operator and can be easily recovered from. An example of this would be a machine that requires an operator to periodically refill a glue dispenser or carton magazine and due to the machine design, these operations cannot be performed while the machine is running. Since these types of tasks are normal production operations, it is not desirable to go through aborting or stopping sequences, and because these functions are integral to the machine they are not considered to be "external." While in the Holding state, the machine is typically brought to a controlled stop and then transitions to Held upon state complete. To be able to restart production correctly after the Held state, all relevant process set points and return status of the procedures at the time of receiving the Hold command must be saved in the machine controller when executing the Holding procedure.

- **Held:** Refer to Holding for when this state is used. In this state the machine shall not produce product. It will either stop running or continue to dry cycle. A transition to the Unholding state will occur when internal machine conditions change or an Unhold command is initiated by an operator.

- **Unholding:** Refer to Holding for when this state is used. A machine will typically enter into UNHOLDING automatically when internal conditions, material levels, for example, return to an acceptable level. If an operator is required to perform minor servicing to replenish materials or make adjustments, then the Unhold command may be initiated by the operator.

- **Suspending:** This state shall be used when external (outside this unit machine but usually on the same integrated production line) process conditions do not allow the machine to continue producing, that is, the machine leaves Execute due to upstream or downstream conditions on the line. This is typically due to a blocked or starved event. This condition may be detected by a local machine sensor or based on a supervisory system external command. While in the Suspending state, the machine is typically brought to a controlled stop and then transitions to Suspended upon state complete. To be able to restart production correctly after the Suspended state, all relevant process setpoints and return status of the procedures at the time of receiving the Suspend command must be saved in the machine controller when executing the Suspending procedure.

- **Suspended:** Refer to Suspending for when this state is used. In this state the machine shall not produce product. It will either stop running or continue to cycle without producing until external process conditions return to normal, at which time, the Suspended state will transition to the Unsuspending state, typically without any operator intervention.

- **Unsuspending:** Refer to Suspending for when this state is used. This state is a result of process conditions returning to normal. The Unsuspending state initiates any required actions or sequences necessary to transition the machine from Suspended back to Execute. To be able to restart production correctly after the Suspended state, all relevant process set-points and return status of the procedures at the time of receiving the Suspend command must be saved in the machine controller when executing the Suspending procedure.

- **Completing:** This state is an automatic response from the Execute state. Normal operation has run to completion, i.e. processing of material at the infeed will stop.

- **Completed:** The machine has finished the Completing state and is now waiting for a Reset command before transitioning to the Resetting state.

- **Aborting:** The Aborting state can be entered at any time in response to the Abort command or on the occurrence of a machine fault. The aborting logic will bring the machine to a rapid safe stop.

- **Aborted:** The machine maintains status information relevant to the Abort condition. The machine can only exit the Aborted state after an explicit Clear command, subsequently to manual intervention to correct and reset the detected machine faults.

NOTE For more information about the state machine model, which is used in this template, refer to: https://www.omac.org/packml

SIMATIC OMAC PackML V2022 Mode&State Management and Machine Data Interface \5\

6.3. Program structure

Use of Software Units

The Automation Framework uses Software Units to structure the PLC program. A Software Unit is a program unit that can be compiled individually and load independently of other program units. Therefore, Software Units bring a lot of benefits:

- Clearly defined interfaces and data handling.
- Independent compilation and download of software parts.
- Namespaces can be defined, easy use of naming conventions.
- Named value data types (NVTs) can be used.
- Copy&Paste of software without need to rename objects.

Content

The AF project comes with predefined Software Units for system wide functionality like CentralFunctions, Global, Motion, Safety and blocks from the Project library that are used in the program.

Additionally, it contains four example units with EMs coded in the different programming languages LAD, Graph and SCL.

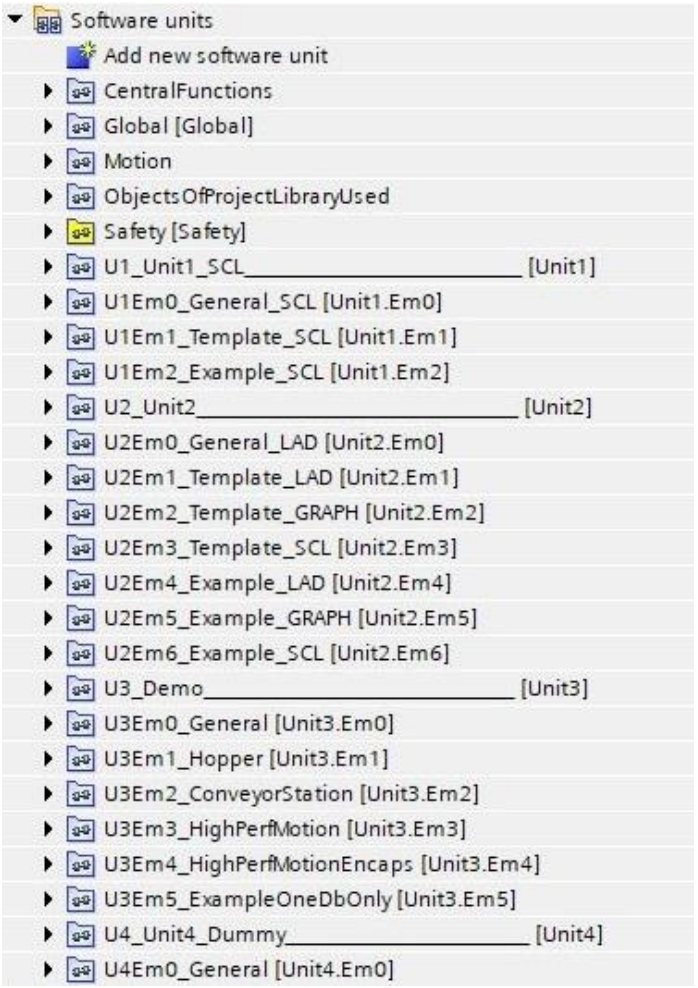


Figure 6-7: PLC program structure.

In the AF project there are two EM Types

- **Em_Template**: Contain only the mandatory blocks and calls. They are the base to start own EMs. One template for each programming language.

- **Em_Example**: Contain a basic example of an EM. Each Em_Example contains four CMs and implements one sequence example. All Em_Examples are designed to execute the same use case such that the user can easy compare the different implemtation across the different programming languages.

Namespaces

Most Software Units in the AF project have namespaces assigned. All objects within the respective Software Units can have the same namespace as the Software Unit. With namespaces it is possible to clone entire Software Units without naming conflicts. The namespace and the name of the program element together result in a unique designation with which the program element can be identified within the CPU. This makes it possible to use the same name for different program elements if they are in different namespaces.

Three Software Units in the AF project have no namespace assigned because they shall remain unique in the entire PLC program. The Safety Software Unit must be unique due to TIA Portal engineering constrains:

- CentralFunctions
- Motion
- ObjectsOfProjectLibraryUsed
- Safety

Software Unit overview and relations

By default, programs within different Software Units are isolated and cannot interact with each other. However, in certain projects, interaction between Software Units may be necessary, either through data exchange via DBs or by invoking function blocks (FBs) or function calls (FCs). To facilitate this, a user must establish a "Relation" with the Software Unit to gain access to its elements, such as FBs, FCs, databases, tag tables, or user-defined types (UDTs). These elements become accessible once they have been made public. Relations should be set up with target objects (TOs) as required, except in the case of the AF project's Motion Software Unit, which includes the TIA Portal motion OBs.

The following figure shows the relations of Software Units and published blocks in the AF.

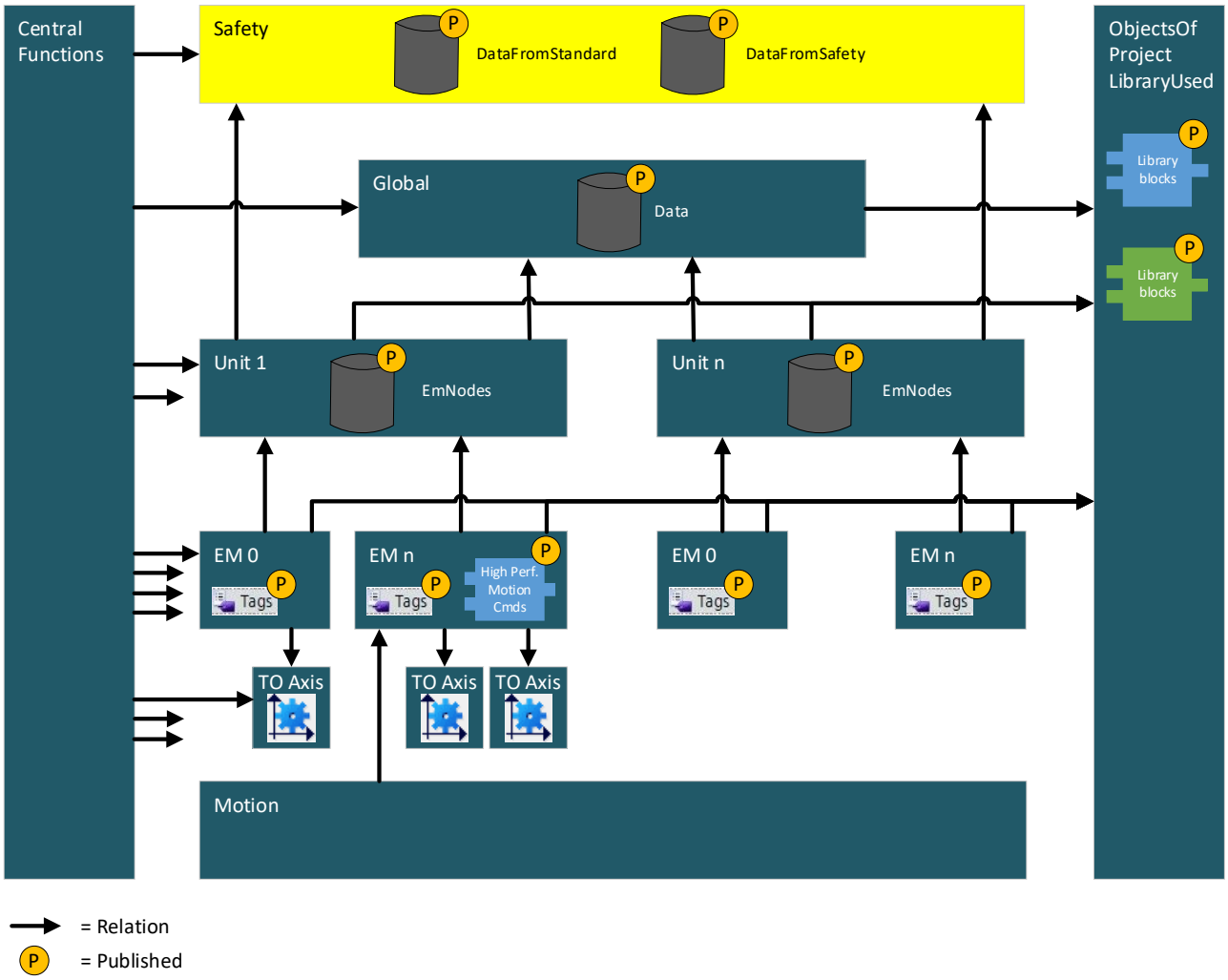


Figure 6-8: Relations between the Software Units.

Software Unit description

- **CentralFunctions:** Includes sytem wide functions and diagnostics. Relations to all other Software Units are possible. No other Software Unit however must have a realtion to CentralFunctions. All data that need to be shared with CentralFunctions by other Software Unit needs be provided by the respcetive Software Unit by a DB that CentralFunction accesses. Otherwise an unallowed loop of relations would exist.
- **Global:** Includes global data of all units and contains no logic, it is a purely passive Software Unit. It has a relation to ObjectsOfProjectLibraryUsed as it uses PLC data types from the libraries.
- **Motion:** Includes the Motion OBs. These execute the motion technology objects like axes or cam outputs. Each Motion OB can only exist once in the entire PLC, therefore all axes across all units are executed at the same time. The Software Unit Motion does not need relations to the TOs, this is handled PLC internally.

Motion commands can be called in two different ways:

- In standard EM cycle (OB "Main" of EM). With that, a few motion servo cycles might pass until the new motion command has been called by OB "Main" and applied to the axis.
- In a motion OB (e.g. OB "MC_PreInterpolator"). With that, new commands are called in the servo cycle and new commands to the axis are applied immediately in the next servo cycle. This might be necessary for applications that require highest reaction times for new position commands. More information can be found in the chapter Motion and a best practice implemenmtation is realized in the Software Unit HighPerformanceMotion (see chapter EM Demo Implementation – High Performance Motion).

A relation from the Motion Software Unit to an EM is only needed if motion commands in the EM must be executed within some servo cycle. For all other, non-time critical motion commands such relation is not needed.

- **ObjectsOfProjectLibraryUsed:** Local instances of the libraries that are used in the PLC program. It has no relations to other Software Units. It only contains blocks that are used by others.

- **Safety:** There is only one single Safety Software Unit allowed in a PLC. All safety functionality of the PLC must be programmed within this Software Unit. The Safety sofware unit has no relation to other Software Units to be completely independant. It doesn't actively write information somewhere else, only provides information and receives commands from other Software Units.

NOTE The safety Software Unit cannot be created by the user but is created by TIA Portal automatically if a new PLC is added and the according setting is set before a new PLC is added.

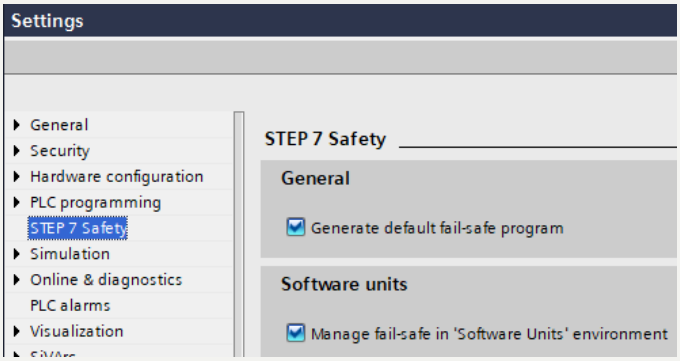


Figure 6-9 TIA Portal setting for manage the safety program in a Software Unit.

- **Unit:** Each ISA-88 unit is realized in a dedicated Software Unit. They shall have relations to:
  - Global: To store data and read cross unit commands
  - Safety: To monitor safety status and trigger safety commands
  - ObjectsOfProjectLibraryUsed: To use elements from library

- **Equipment Modules:** Each EM shall be realized in dedicated Software Unit. These shall have relations to:
  - It's unit: To monitor the status of the unit and write status feedback
  - ObjectsOfProjectLibraryUsed: To use elements from library
  - TOs: only if needed because TO Axes cannot be embedded inside a Software Unit. EMs that execute MC commands need to have access the TOs via the relation.

Programming languages

The AF project offers various Unit and EM templates and examples implemented in different programming languages. The user can choose which implementation to copy from for his application according to his programming preferences.

Software Unit	Description
U1_Unit1_SCL	Programmed completely in SCL
U1Em0_General_SCL	Programmed completely in SCL
U1Em1_Template_SCL	Programmed completely in SCL
U1Em2_Example_SCL	Programmed completely in SCL
U2_Unit2	Programmed in LAD
U2Em0_General	Programmed in LAD
U2Em1_Template_LAD	General logic programmed in LAD, sequencer programmed in LAD
U2Em2_Template_GRAPH	General logic programmed in LAD, sequencer programmed in GRAPH
U2Em3_Template_SCL	General logic programmed in LAD, sequencer programmed in SCL
U2Em4_Example_LAD	General logic programmed in LAD, sequencer programmed in LAD
U2Em4_Example_GRAPH	General logic programmed in LAD, sequencer programmed in GRAPH
U2Em4_Example_SCL	General logic programmed in LAD, sequencer programmed in SCL
U3_Demo	Example of a unit programmed in LAD
U3Em0_General	Example of a General EM programmed in LAD
U3Em1_Hopper	Example of the realization of a technical module
U3Em2_ConveyorStation	Example of an EM realized with GRAPH
U3Em3_HighPerfMotion	Example of how to split motion calls in standard cycle and motion cycle
U3Em4_HighPerfMotionEncaps	Example of a fully encapsulated EM. Data handover is only done the can be instantiated. Data handover by interface.
U3Em5_ExampleOneDbOnly	Example of replacing the three DB "HmiInterface", DB "ControlNodes" and DB "Config" by one DB Data
U4_Unit4_Dummy	Template unit in LAD
U4Em0_General	Template of a General EM programmed in LAD

Table 6-1: Overview of used programming languages

6.4. Program execution

In this chapter the default OB "Main" call structure and interface handling inside the Units and EMs will be introduced. The goal is to ensure maximum modularity of the programm structure such that Software Units can easily be cloned.

Every Software Unit will have a dedicated OB "Main" except for the Global and ObjectsOfProjectLibraryUsed. The AF follows the following call structure:

- Within a Unit the OB "Main" calls the FB "CallUnit" and passes all external data from connected Software Units and its public data via the FB interface.
- Within an EM the OB "Main" calls the FB "CallEquipment" and passes all external data from connected Software Units and its public data via the FB interface.

Any programming logic will be executed inside the FB "CallUnit" / FB "CallEquipment". This way the programming logic is seperated from the dependency to the other external Software Units. Only the relation between the Software Units have to be updated when copy and pasting an entrie Software Unit. Names and data access inside the Software Unit remain the same. Due to the use of namespaces, even names don't change.

The following schematic shows and example with two Software Units. SWUnit_2 has a releation to SWUnit_1. With that, it can access the data of the published DB of SWUnit_1. The data of SWUnit_1.PublishedDB is connected to an InOut interface of FB "CallUnit" in SWUnit_2. Code inside both FBs CallUnit can now access shared data internally without direct address dependencies. More information can be found in the chapter Software design principles.

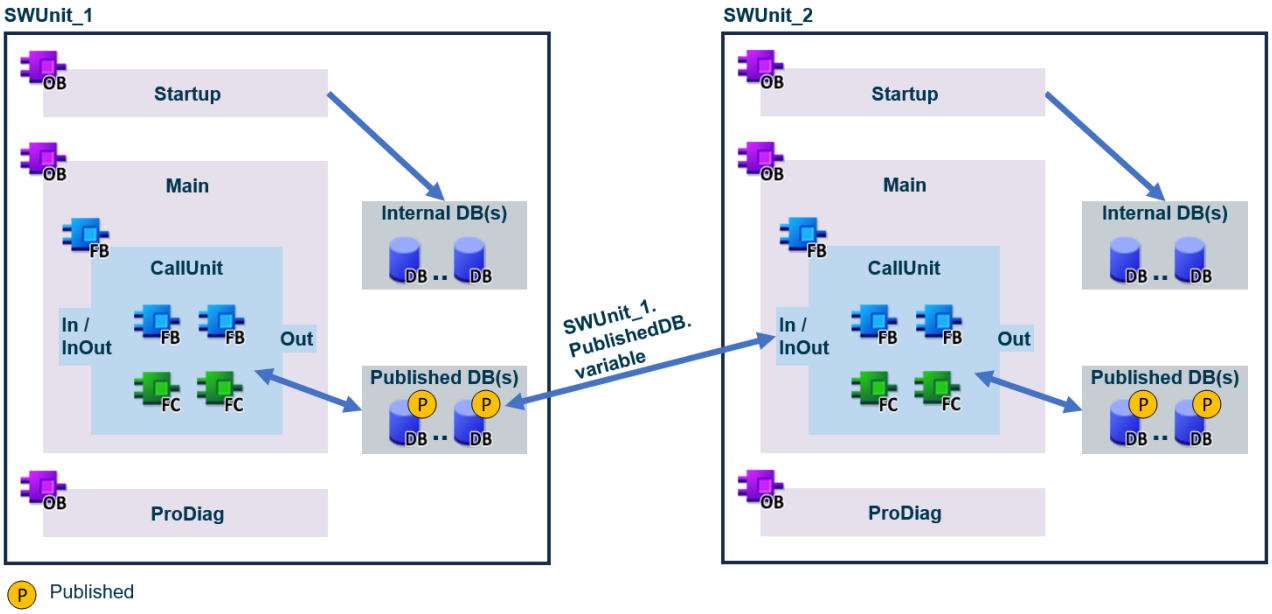


Figure 6-10: Relation between two Software Units and data access internally and externally.

The following figure shows the call of FB "CallUnit" in OB "Main" of a unit. It can be seen, that data from outside the own Software Unit, e.g. from "Global" or "Safety" is handed over via the interface of FB "CallUnit". Data within the own Software Unit is not handed over via the block interface, but handeled inside the block directly by using direct data block access. That brings the advantage that a cross-reference is possible and that the user can jump to definition by one click.

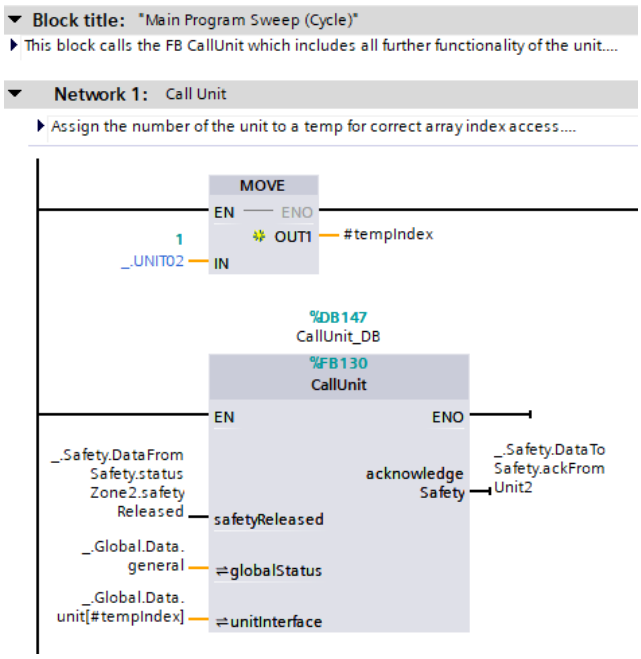


Figure 6-11: Example of an OB Main of a unit.

NOTE The Instance Data block of the CallUnit is not named according to the programming style guide (that would be InstCallUnit), but the default name is kept (CallUnit_DB). This is because of a ProDiag rule when using the feature "initial value acquisition" in an instance. More details can be found in the TIA Portal help within chapter "Acquiring initial values" for S7-1500.

Each Software Unit usually contains the following blocks:

- **OB Startup:** These OBs are called once at the very beginning after STOP – RUN transition of the PLC. General assigments, configuration setting or default value definitions are usually programmed here. After the startup routine has been completed, the operating system reads in the process image input and starts the cyclic program.
- **OB Main:** Main task of the PLC. It runs in a cyclic loop, only interrupted by other high-prior OBs. It calls the user programmed logic.
- **OB ProDiag:** Also executed in a cycle like OB "Main". It calls FB "ProDiag", which evaluates all configured ProDiag supervisions and triggers alarms.
- **Internal DBs:** Internal DBs could be e.g. a DB "Config" for configuration data. These DBs contain data that is limited to the own Software Unit and not to be shared with the rest of the PLC.
- **Published DBs:** These data blocks are for data that is to be shared with other Software Units. They are used as links between Software Units or as global data collections/hubs.
- **User defined FB/FCs:** User defined blocks for individual logic can be programmed.
- **Call of library blocks:** Each Software Unit has a relation to ObjectsOfProjectLibrary and can instantiate blocks from there.

6.5. PLC start and cyclic call sequence

The execution sequence of the user program depends on the initial event (PLC start or cyclic operation), the OB priority, and the OB number. Software Units have no impact on the order of the calls.

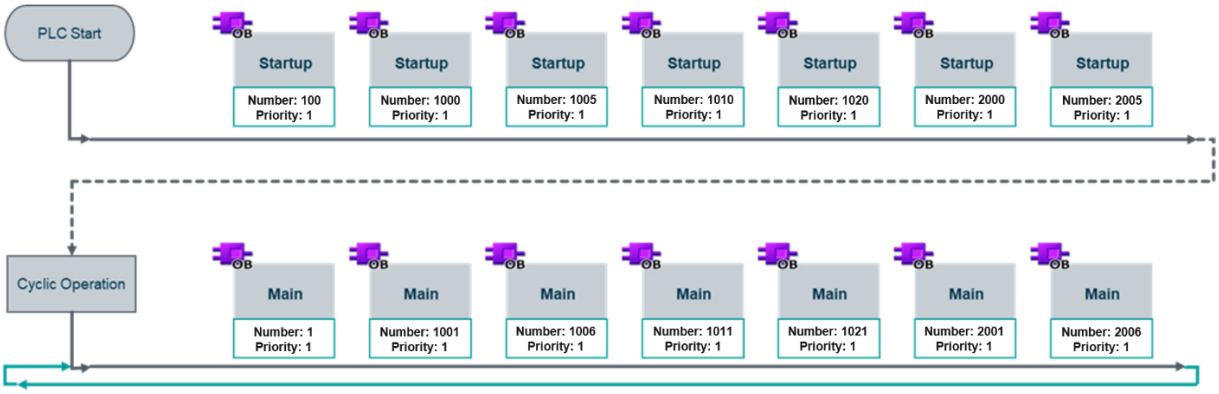


Figure 6-12: Call hierarchy of startup and cyclic OBs.

The essential basis of the user program is the cyclic program execution. When all program cycle OBs, the OB "Main" are executed, the PLC's operating system restarts the execution from the beginning in the same sequence. These OB "Main" have the lowest priority 1. Cyclic programs can and usually will be interrupted at any time by events of another events class having a higher priority. With multiple program cycle OBs, they are called one after the other in the order of their OB numbers. The program cycle OB with the lowest OB number is called first.

The following events start the cyclic program:

- End of PLC Startup processing.
- End of the processing of the previous run of the cyclic program.

After the cyclic program is complete, the operating system updates the process images as follows:

- It writes the values from the process image output to the output modules.
- It reads the inputs at the input modules and transfers these to the process image input.

OB numbering convention

In the AF, each Software Unit with user code has its own OB "Main", OB "Startup" and OB "ProDiag". To ensure the correct call hierarchy, the following numbering convention is used:

OB number = uxxo where

u: Unit number from 1-9

xx: EM number from 0-99

o: 0 = Startup OB, 1 = Main OB, 2 = ProDiag OB, 3/4=empty, 5=StartupEM0, 6=MainEM0, 7=ProDiagEM0, 8/9=empty

Examples:

OB Number = 2001 : Unit 2, Main OB

OB Number = 3022 : Unit 3, EM 2, ProDiag OB

Unit Nr.	Unit Startup	Unit Main	Unit ProDiag	EM 0 Startup	EM 0 Main	EM 0 ProDiag	EM 1 Startup	EM 1 Main	EM 1 ProDiag
1	1000	1001	1002	1005	1006	1007	1010	1011	1012
2	2000	2001	2002	2005	2006	2007	2010	2011	2012
3	3000	3001	3002	3005	3006	3007	3010	3011	3012
4	4000	4001	4002	4005	4006	4007	4010	4011	4012

Table 6-2: Extract of OB numbering

Additional OBs not related to Unit or EMs in the AF project have the following OB numbers:

Name	OB	#
CentralFunctions	Startup	0100
CentralFunctions	Main	0001
CentralFunctions	ProDiag	0250
Safety	ProDiag	0251
Global	ProDiag	0252

Table 6-3: Higher level function OB numbering

With this OB number convention the program will be executed in the following sequence during cyclic operation:

1. Central Functions OB Main
2. Central Functions OB ProDiag
3. Safety OB ProDiag
4. Global OB ProDiag
5. Unit 1 OB Main
6. Unit 1 OB ProDiag
7. Unit 1, EM 0 OB Main
8. Unit 1, EM 0 OB ProDiag
9. Unit 1, EM 1 OB Main
10. Unit 1, EM 1 OB ProDiag
11. …
12. Unit 2 OB Main
13. Unit 2 OB ProDiag
14. Unit 2, EM 0 OB Main
15. Unit 2, EM 0 OB ProDiag
16. Unit 2, EM 1 OB Main
17. Unit 2, EM 1 OB ProDiag
18. …

OBs having higher priorities (> 1) will interrupt the sequence of the cyclic program execution, e.g. OB Safety, OB "MC_Servo" or interrupt OBs.

6.6. Command and data flow

This chapter introduced the processing of unit control commands (Start, Stop, Abort, etc.) and the command propagation through the AF project and user code.

The Unit command flow propagation

The Automation Framework provides the architecture to control several units via several HMIs. The unit command flow therefore begins in an assigned HMI and the commands are forwarded to the central data storage DB "Data" in the Global Software Unit via the "LAF_HmiHeaderManager".

Each unit then evaluates incoming control commands, from HMI or externally via OPC UA, against the current unit's state and mode. The incoming commands usually trigger a transition to a new unit state or unit mode but not all commands are permitted at all times. The unit's current state and mode is then published in the DB "EmNodes" and DB "Data".

Each EM reads the status from it's unit and executes its programming logic accordingly. Each EM must feedback it's state to the unit. This can trigger a unit state change e.g. due to a failure inside an EM or sub-CM or hardware pushbutton inside some CM. The feedback from the EMs will be written into the DB "EmNodes" of the unit. The EM feedback is mandatory unless the isLinked flag in the feedback is false. Un-linked EMs can be operated independently from the unit state e.g. via em.commands.directModeControl and em.commands.directStateControl during comissioning or service. How exactly the linking of the EM works is described in more detail in chapter Interlink.

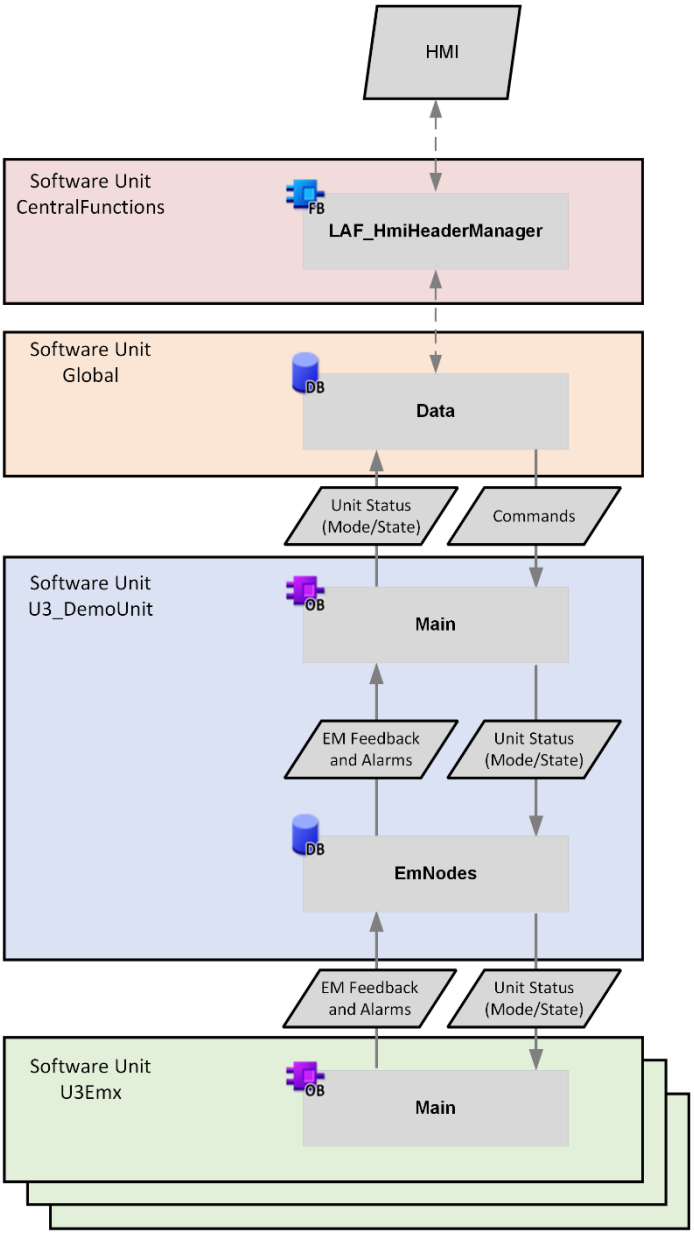


Figure 6-13: Command and data flow.

HMI interfaces

The AF HMI interacts with the Units via the Global and CentralFunctions Software Units. The "CallGlobal" block is called within the OB "Main" of the CentralFunctions Software Unit to handle the signals of the DB "Data" within the Global Software Unit. From this point on, data access is bi-directional to the "LUC_InterfaceManager" of the units.

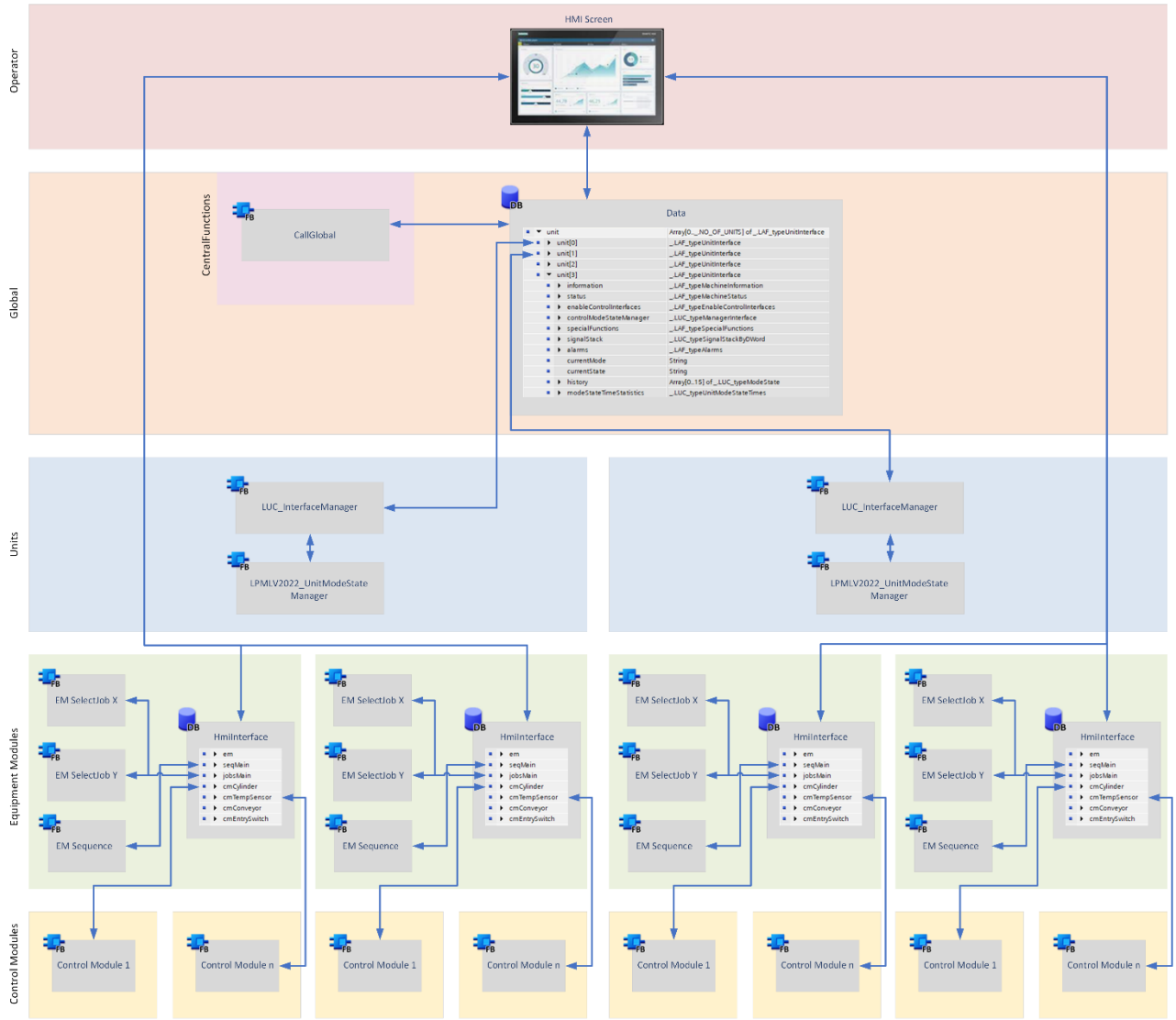


Figure 6-14: Use of DB Interface and HMI connection.

Additionally, each EM contains an DB "HmiInterface" with all the data structures needed for visualization and operation of the EM logic and its sub-CMs. Examples for HMI operations are manually controlling the EM sequence execution and/or faceplate for CMs.

6.7. Software design principles

6.7.1. Passing data via block interface vs. direct global access

Programming schema

In PLC software design, two implementation schemata are common:

- Connect global data (inputs, outputs, data blocks) in the highest level of the block hierarchy and only use the block interfaces to pass signals to lower levels
- Connect global data (inputs, outputs, data blocks) also in lower levels directly from the source without using the block interfaces

Both schemata have pros and cons:

- Schema 1 (using block interfaces):

Advantages	Disadvantages
Better reusability of functions and function blocks, as they are fully encapsulated.	Limited functionality of cross-references.
Can be fully versioned.	Lower comfort when drag-and-dropping tags from one block to another (e.g. control node variables into sequences).
-	Usability for programmers during debugging and commissioning is decreased

Table 6-4: Pros and cons of using exclusively block interfaces to handover data

- Schema 2 (using direct access to input and data blocks):

Advantages	Disadvantages
Easy jump to exact data source possible, via "Go to definition" tooltip. No chasing through interfaces necessary.	Reduced reusability as tag calls inside the block might need to be reconnected when a block is copied and called again with other data.
Convenient drag-and drop of tags, as separate blocks can be opened in spitted editors next to each other.	Increased risk of overwriting variables as they can be written easily from different locations.

Table 6-5: Pros and cons of reading and writing to global data directly from each block.

In summary, schema 1 enables a higher standardization. Automatic project generation as well as highly reusable programs can be better realized that way. It is considered it is a higher level of good programming. However, the usability for programmers during debugging and commissioning is decreased.
Schema 2 on the other side is a more basic programming style, easier to understand and debug, but higher effort the adjust for new programs.

The use of Software Units simplifies implementation schema 2, because it is now possible to copy entire Software Units without having to reconnect all internal variables.

Realization in the AF

The user can program using the AF libraries following the one or the other schema. The AF project was programmed according to schema 2 to keep the AF framework offering as general and simple as possible. Especially the possibility to use the cross-reference functionality in TIA Portal dramatically increases the usability for the programmer.

We already introduced the schema 1 in the chapter Program execution, for Units. The example of schema 1 within an EM implementation is shown in Unit3.EM4 (U3Em4_HighPerfMotionEncaps).

In the following it is described how the modules are programmed within the AF. Data coming from other Software Units are passed via the block interface of FB "CallEquipment". By connecting the data source of the other Software Unit only at the highest level, the Software Unit is fully encapsulated to another Software Unit. Data coming from blocks inside the same Software Unit, is accessed directly.

The following figure the FB "CallEquipment" is connected to data from another Software Unit. This data is passed into FB "CallEquipment" through an InOut interface. The logic inside FB "CallEquipment" can use that data referring to the formal parameters of the interface.

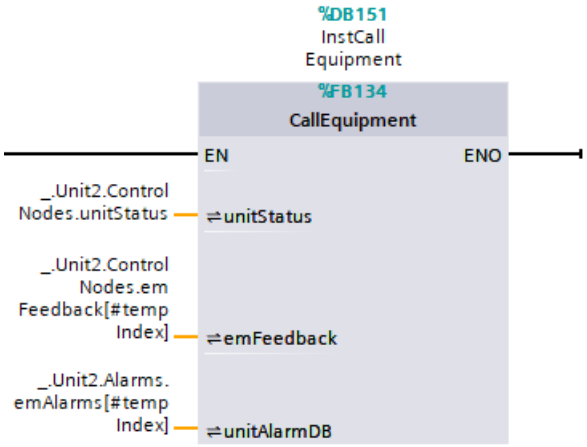


Figure 6-15: Call of FB "CallEquipment" in an OB "Main" of a Software Unit.

The following figure shows an FC inside FB "CallEquipment" with both schemata. The variables "#emFeedback.isLinked", and "#unitStatus" are read/write through the interface of FB "CallEquipment". The InOut Parameters "hmiInterface" and "localInterface" however are connected to local DBs inside the EM but outside the FB "CallEquipment", so it accesses these DBs directly without routing through the block interface. It is directly pointing to the DBs "HmiInterface" and "ControlNodes" in the EM and here to the data structure em within the data blocks. This direct variable link allows the user to jump to the program block directly via tooltip "Go to definition".

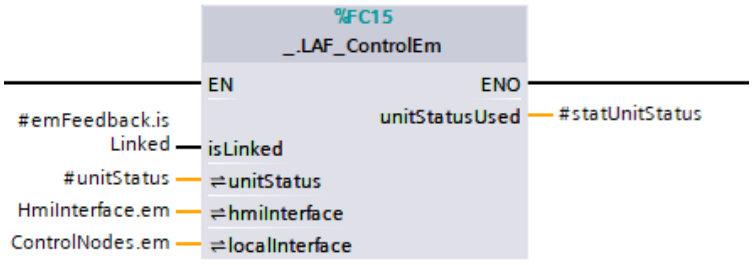


Figure 6-16: Call of FC "LAF_ControlEm" in a FB "CallEquipment".

6.7.2. PLC – HMI handshake

In the HMI development there are two ways to manage button press events between the HMI and the PLC:

- Set/Reset schema: HMI sets a PLC command bit, the PLC processes the command, finally PLC resets the command bit
- OnPressed schema: Only as long as the HMI command button is pressed the PLC command is set

Both architectures have pros and cons:

Set/Reset schema

Advantages	Disadvantages
It is guaranteed that the PLC receives and processes the command.	More complex programming, since the PLC must reset the command, independently of the result of the command execution
Better user experience for the operator: It does not matter how long an operator presses the button or how long the PLC needs, to execute the command, the command will be executed eventually	Increased risk ending in a deadlock, e.g. if HMI set a command, but PLC didn't reset due to any unforeseen circumstance (e.g. interrupt, failure, PLC stop/start, etc.). If the same operator command is set, it will have no effect, since no reset action of the command has been done.

Table 6-6: Pros and cons of Set/Reset schema

OnPressed schema:

Advantages	Disadvantages
Simple to set up.	Exists possibility that the operator presses the button and nothing happens because he has not pressed the button long enough for being detected by next PLC update cycle.
No risk of ending in a deadlock. If operator presses again, command will be sent again.	If multiple commands come in parallel, there is a higher risk of uncoordinated program execution.

Table 6-7: Pros and cons of OnPressed schema

In summary, Set/Reset schema enables a higher integration and standardization. It is considered as the most advanced programming technique. However, it is more complex and more scenarios must be considered to ensure a reliable operation in all circumstances.

Realization in the AF

The AF can be programmed in both architectures. However, the AF itself is realized with "OnPressed" schema, as the main goal is to provide a general and easy to understand framework. The easy set up and the low coding complexity increases the usability for the programmer.

7. HMI Software Architecture

The AF project provides a modular and customizable HMI Template for WinCC Unified to reduce programming efforts. The following chapter will introduce the HMI layout, reusable elements like HMI header, page navigation, and plug and play application examples.

7.1. Screen Layout

A screen layout refers to the arrangement and design of various visual elements and components on a graphical user interface (GUI) screen. This layout determines how the different process screens are positioned and organized on the HMI.

Screen layouts play a crucial role in enhancing the usability and effectiveness of the HMI.

This layout (starting screen) provides an intuitive and user-friendly interface, allowing operators and users to easily access and interact with the relevant information and controls. It involves considerations such as visual hierarchy, grouping related elements, optimizing space utilization, and ensuring a consistent and aesthetically pleasing design.

Use case

In a manufacturing facility, operators need a clear and user-friendly interface to monitor various parameters, control equipment, and respond to alarms promptly. The screen layout plays a fundamental role in achieving this goal.

The AF screen layout represents the starting screen of the HMI and is split into six sections.

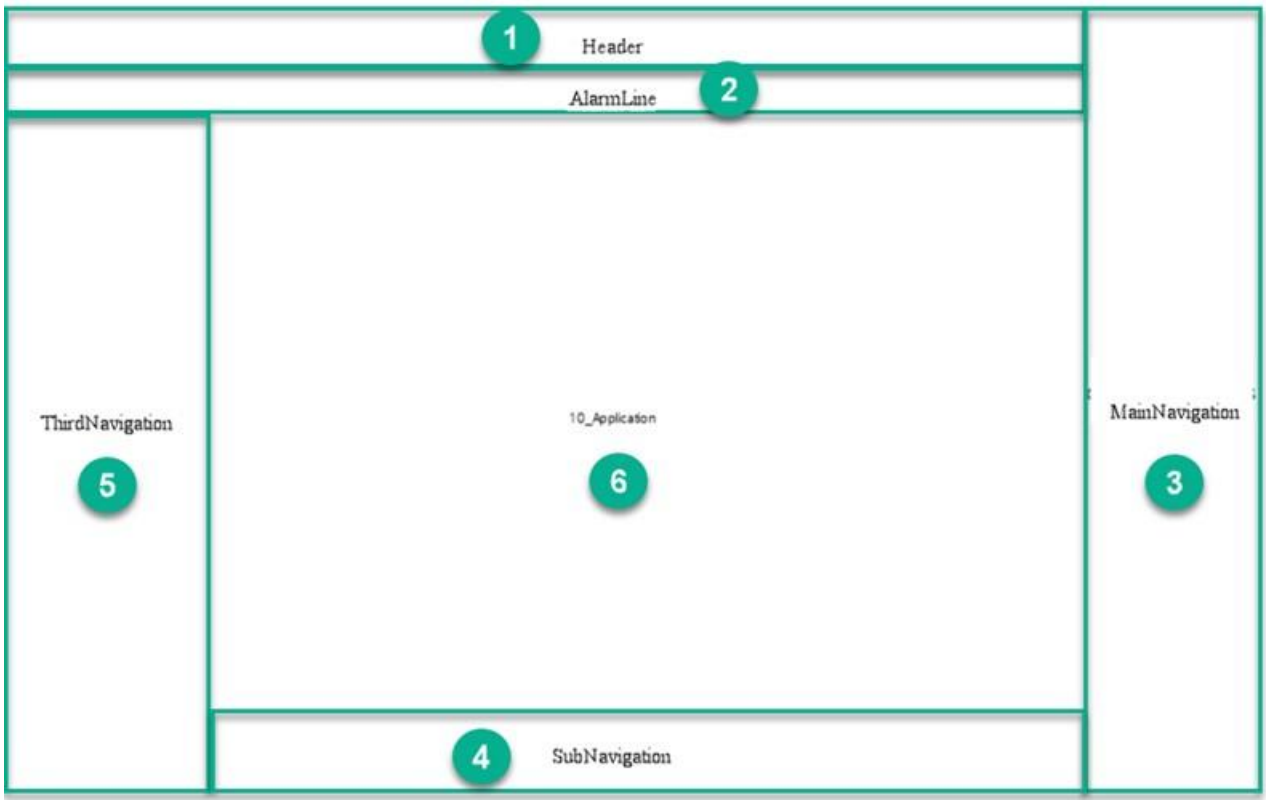


Figure 7-1: Screen layout.

Section	Name	Description
1	Header	The header is always visible, and has the most important information, like PLC-name, current screenname, memory-card information, PN-connection, date and time and also PLC-Checksum.
2	AlarmLine	The AlarmLine shows the latest active alarm.
3	MainNavigation	The MainNavigation has 11 navigation areas, which will help the user to go to specific areas like: Overview, State Model, Manual, Messages, Shift model, Production Data, Energy, CM info, Diagnostics, Web, Settings, and User operation. More information on the topic Navigation can be found in section Navigation in chapter 7.3.
4	SubNavigation	The SubNavigation offers an extra layer of navigation on the bottom of the screen, which can be customized individually.
5	ThirdNavigation	The ThirdNavigation offers an extra layer of navigation on the left side, which can be customized individually.
6	Application	The Application section is the main part of the screen. Here will be your application menus, I/Os, graphics, and control tags for your specific application.

Table 7-1: Description screen layout

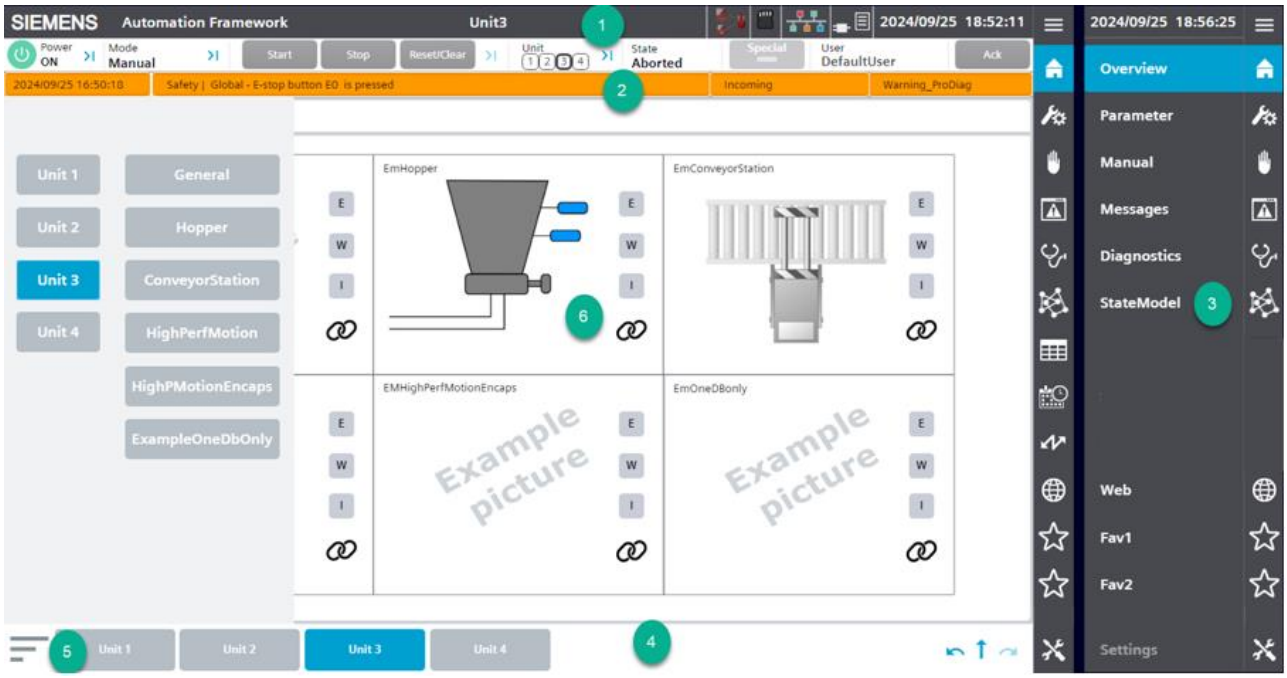


Figure 7-2: Screen layout of AF.

The sections four, five, and six are implemented in dedicated WinCC Unified pages which are loaded when the starting screen is loaded. The other sections are implemented into the screen layout and are programmed within the starting screen.

7.2. Header

The header displays the most important information: power status, current state and current mode etc., according to the selected units. It also allows the operator to control the selected units.

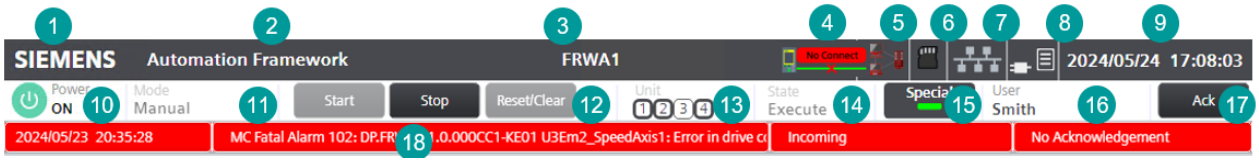
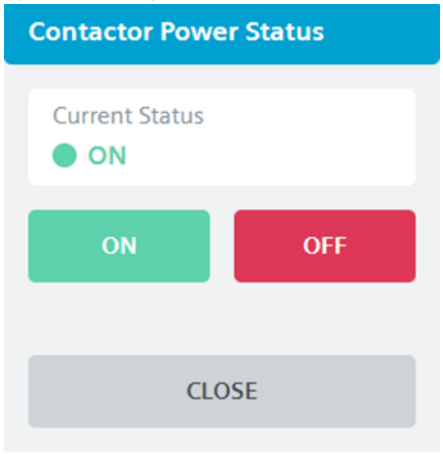
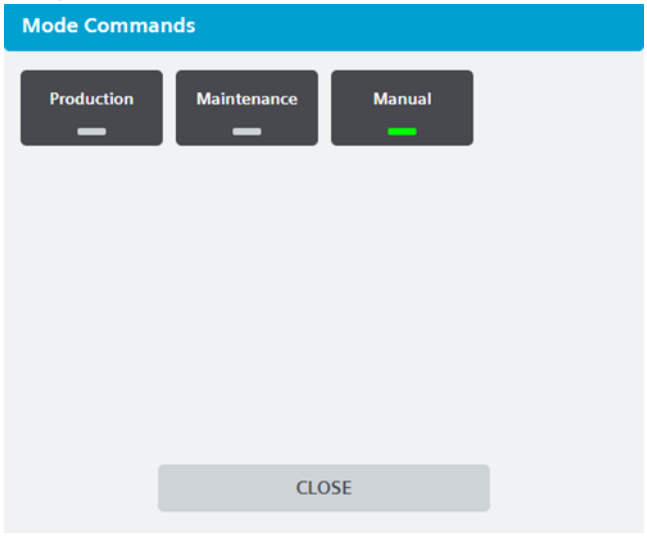
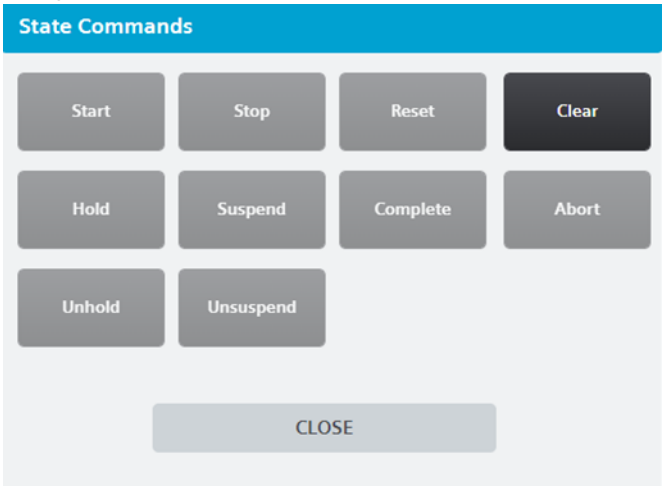
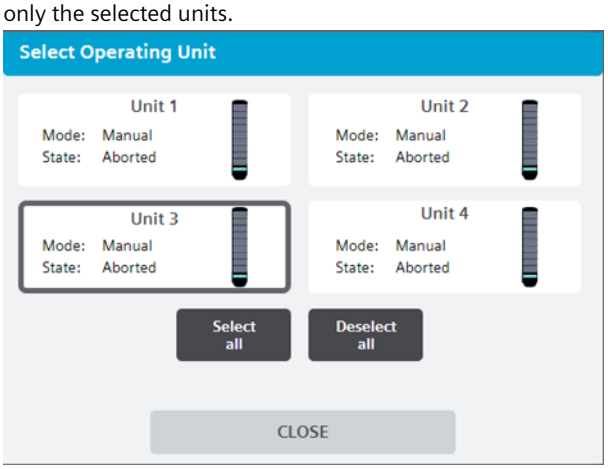


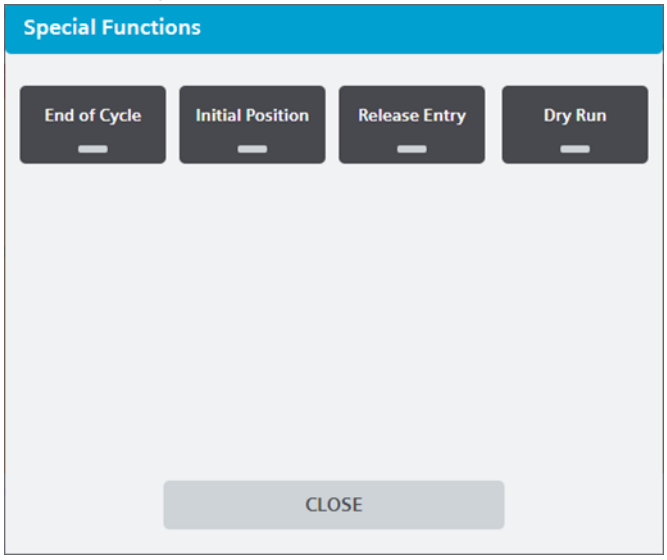
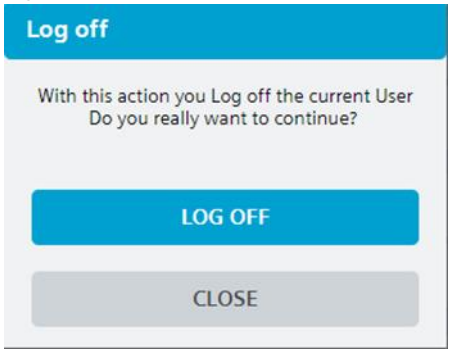
Figure 7-3: HMI layout.

The following visual elements are in the "header":

Element	Name	Description
1	Company name	Placeholder for a dedicated company name.
2	PLC name	The PLC name relates to a PLC variable and displays the content of the PLC's Identification & Maintenance/Plant designation data.
3	Current screen name	Displays the name of the current screen using a script called "Show Title".
4	HMI-PLC connection status / PLC operation mode	It is used in the AF project to visualize the PLC and connection status: PLC Stop No connection between PLC and HMI For further details see chapter Communication.
6	SIMATIC memory card status	Status information about the SIMATIC memory card used, e.g. memory size and allocated memory space. Previously used up portion of the service life (lifetime) and set proportion of the lifetime of the memory card.
7	PROFINET diagnostic	Summarize the diagnostics of the connected PROFINET devices.
8	CRC Check	Standard PLC blocks, safety PLC blocks and PLC text lists have a checksum. If a difference is detected by comparing the previous checksums with the current checksums, the dynamic SVG visually illustrates this.
9	Current Date & Time	The time settings described in more detail in the chapter. Settings

Element	Name	Description
10	Power On/Off	This command can be used to e.g. centrally turn on/off the main power circuit breaker for the actuators. This exact behavior is customizable. With the on/off commands, a single bit in the PLC is set to true/false, which can be connected by the PLC programmer as required.  Figure 7-4: Power On/Off
11	Mode and Mode commands	The current mode of the selected unit is shown. If multiple units are selected and if they are in different mode, "Various" is displayed. The mode can be changed in the pop-up screen by the operator. The mode commands only affect the selected units. The green indicators inside the buttons show, if at least one selected unit is in that mode. The buttons have three properties: Visibility, activation and operator control allowed. The FB "HmiHeaderManager" in the PLC determines the values for the properties, depending on the current mode, state and state machine configuration.  Figure 7-5: Mode Commands

Element	Name	Description
12	State command	The user can give state commands to the units by using this pop up. The commands only affect the selected units. FB "HmiHeaderManager" inside the PLC determines if the commands can be selected. This depends on the current mode, state and state machine configuration.  Figure 7-6: State Commands
13	Select Unit	The current mode and state of the unit is displayed here. Additionally, the user can select individual or all units. The buttons in the HMI header affect only the selected units.  Figure 7-7: Operating Unit
14	State	The current state of the selected unit is shown. If multiple units are selected and if they are in different states, "Various" is displayed.

Element	Name	Description
15	Special functions	Additional requests to the machine operation can be triggered here by the operator. The special functions can be customized. The text of the buttons can be adjusted in WinCC by adjusting the text list. The visibility and operator control in configured in the DB "HMI" in the PLC.  Figure 7-8: Special Functions
16	Current logged in user	Click on user toggle the log in / log off dialog, depending on the current use login state.  Figure 7-9: Log off.
17	Acknowledge	Button to acknowledges all pending alarms.

Element	Name	Description
18	AlarmLine	The last generated alarm is displayed in the Alarm Line. By selecting the AlarmLine, the AlarmScreen is displayed (see global scripts AlarmLine)

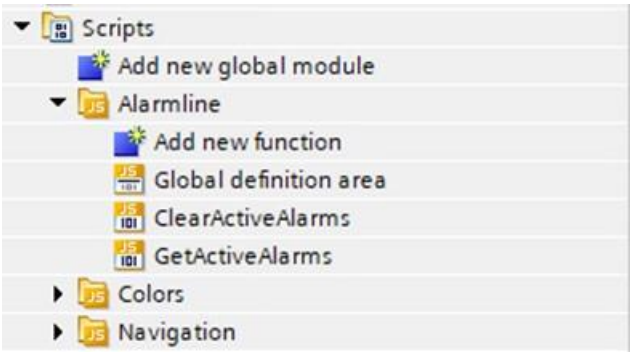
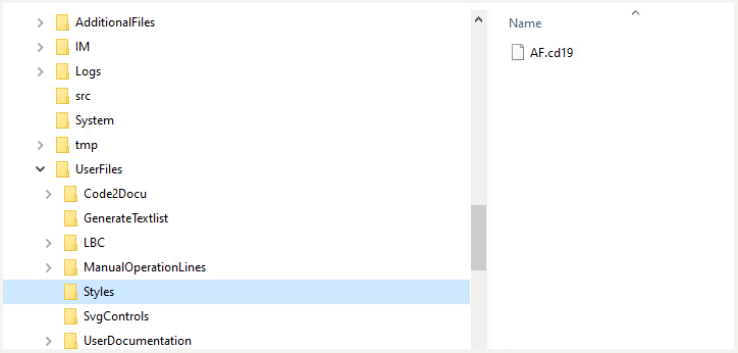


Figure 7-10: Global Scripts – Alarms.

Table 7-2: Elements in the header.

NOTICE

When using the LAF by itself in a new project, the date AF.cd19 in the User Files → Styles Folder needs to be copied in the new project to have the full functionality in the HMI.



7.3. Navigation

Monitoring parameters, controlling actuators, reading out machine data are only a few points that a plant operator can read/operate on the HMI. A standardized basic navigation concept is the foundation for user-friendly and efficient operation. Therefore, a navigation concept with the help of navigation levels is created to enable a uniform visual layout.

The navigation is divided into three navigation levels (Figure 7-1).

- AF screen section 3: MainNavigation -> "00_ScreenLayout/ScreenLayout"
- AF screen section 4: SubNavigation -> "00_ScreenLayout/SubNavigation"
- AF screen section 5: ThirdNavigation -> "00_ScreenLayout/ThirdNavigation"

The main navigation lists the different main points. Since main items can be subdivided into several subitems, the sub-navigation should be integrated for this function. If a further subdivision of the subitems is required, it can be implemented with the help of the "ThirdNavigation". Visibility of the sub- and third navigation is flexible while main navigation is permanently visible.

7.3.1. Main navigation

The main navigation consists of two parts: The icon bar, which is permanently visible, as well as a textual view which can be switched on via the Burger menu. Both views are linked to key default HMI features on the AF HMI project.

Figure 7-11: Main navigation.

Technical description

These AF main navigation is pre-configured for various key features with its name and graphic accordingly. These values and graphics can be customized through ES, as well as, the main screens displayed for each central function (every single time that the operator navigate through the screens using the main navigation). The screens displayed in Section 6 using the main navigation are pre-configured on ES. Event will be executed every single time that the operator presses on these graphics and text boxes to change the screen on "swContent "(Screen Window – Section 6).

In case, the screen of a certain central function should be changed, the event of this graphic and text box must be modified in ES.

Main navigation is located on "00_Screenlayout/ScreenLayout" in ES. The navigation icons are located on Layer 7, the navigation textl are on Layer 6 (following picture).

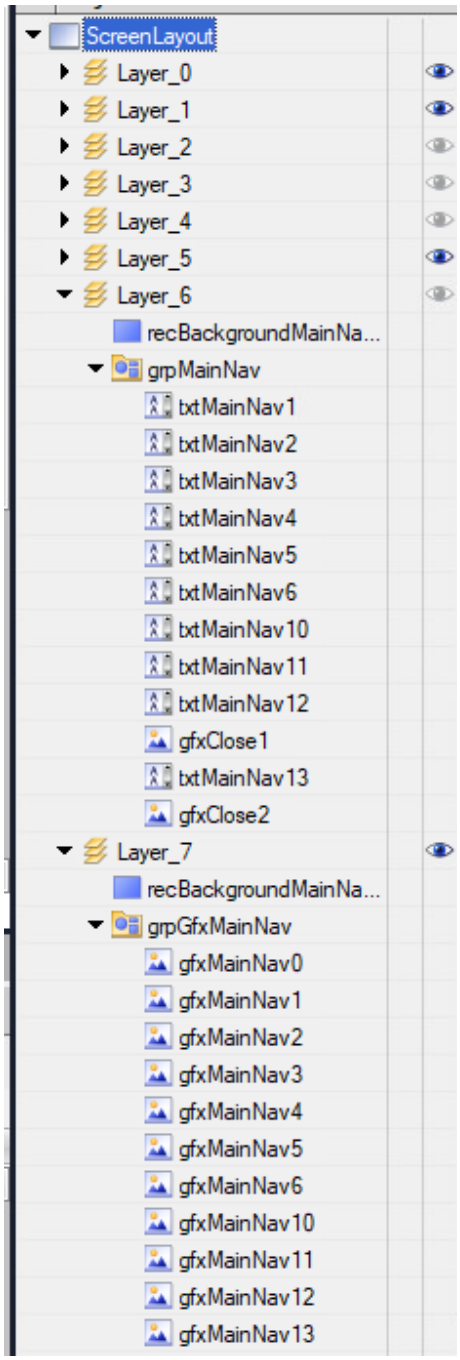


Figure 7-12: Main Navigation – Screenlayout Layer 6 & Layer 7.

7.3.2. Sub navigation

Sub-navigation can be used to extend each main navigation element where necessary. The sub-navigation, allows the operator to navigate through the multiple screens related to the same central function (main navigation). Every single central function contains a related sub-navigation for this purpose.

Figure 7-13: Sub navigation – Diagnostics example.

Technical description

Sub-navigation involves two different scenarios: central functions and Units. The sub-navigations for central functions are pre-configured, where their values and events are set according to the screens displayed, and related to every central function on the main navigation.
On the other hand, the sub-navigations for Units and EMs are dynamically configured based on the project configuration, since the number of Units and EMs can differ from one project to another. In these cases, the sub-navigation of each of those is dynamic during RT execution.

That means, that based on PLC constants and HMI Text Lists, the values can switch from one Unit to another and from one EM to another. Therefore, the AF only includes one Unit sub-navigation and one EM sub-navigation to display the corresponding values.

The "SubEm" navigation screen is dynamized by PLC constants linked to every Unit Instance DB (CallUnit_DB), which is connected to every UnitXEmNum HMI tag (Unit1EmNum, Unit2EmNum, Unit3EmNum and Unit4EmNum) to display dynamically the number of EMs in the sub-navigation depending on every Unit. These HMI tags are stored in the following HMI Tag tables: U1, U2, U3 and U4. These values are configured in the PLC constants, inside every Software Unit of every Unit, as it shown in the following picture:

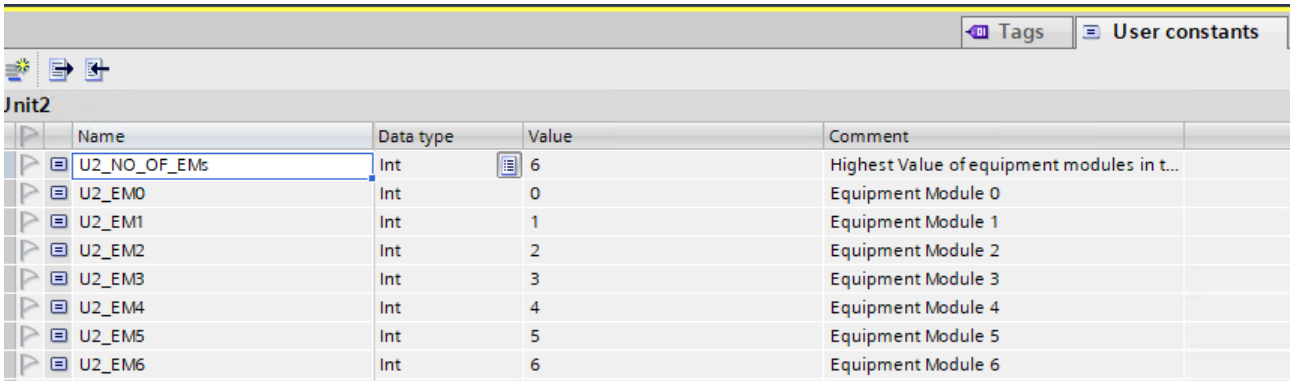


Figure 7-14 PLC Constants – Unit 2.

The values (displayed name), are configured dynamically as well as based on some HMI Textlists (U1Em, U2Em, U3Em and U4Em).

The values of "SubUnit" navigation screen are dynamized on a similar way based on another HMI Textlists (Units).

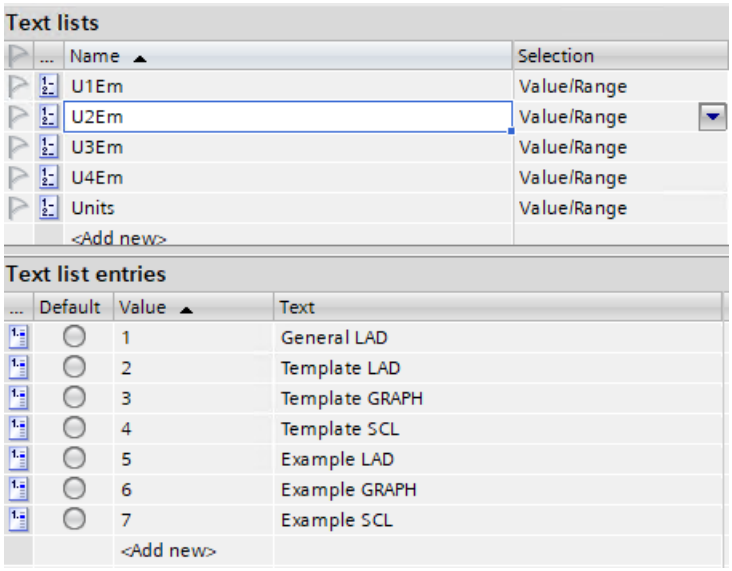


Figure 7-15 HMI Textlists to dynamize SubUnit and SubEMs Screens.

7.3.3. Third navigation

A third level navigation on the left of the AF screen layout is intended to select application specific content to be displayed in section 6 ("Content Screen"). The elements in this third navigation follow the same concept as the sub-navigation. Therefore, it can be set also dynamically. In addition, both navigations are synchronized while navigating among Units and EMs. That means, the operator can navigate through the screens using sub-navigation, third-navigation or the graphical navigation displayed in section 6 and all these parts are aligned and are displaying the same view.

Third Navigation can contain multiple levels on the screen cases, such as Home, Manual Operation, Motorstarter, Drives, IdentRFID and IdentMv screens. This is shown on Figure 7-13. Other screen cases only contain one level.

These levels are the following:

- General level or Unit level
- EM level
- CM level

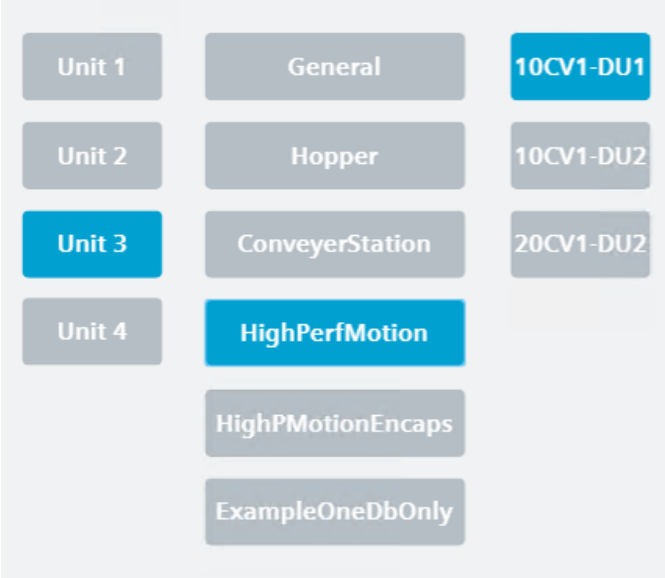


Figure 7-16: Third navigation.

7.3.4. Unit navigation

The Unit navigation is created to visualize and operate among the multiple units available on the system. This Unit navigation can be performed on multiple ways depending on the user preferences. The available options are the following:

- Application section: Graphical navigation where an overview of all units is displayed.
- Sub navigation: Regular sub navigation to navigate through units.
- Third navigation: Regular third navigation to navigate through units.

Application section – Unit Navigation

The Application section can be used to navigate through units. Operator can click on the desired unit to display the specific EMs of the selected Unit. The navigation through the EMs and CMs is available using the application section as well as it can be seen on the following example pictures:

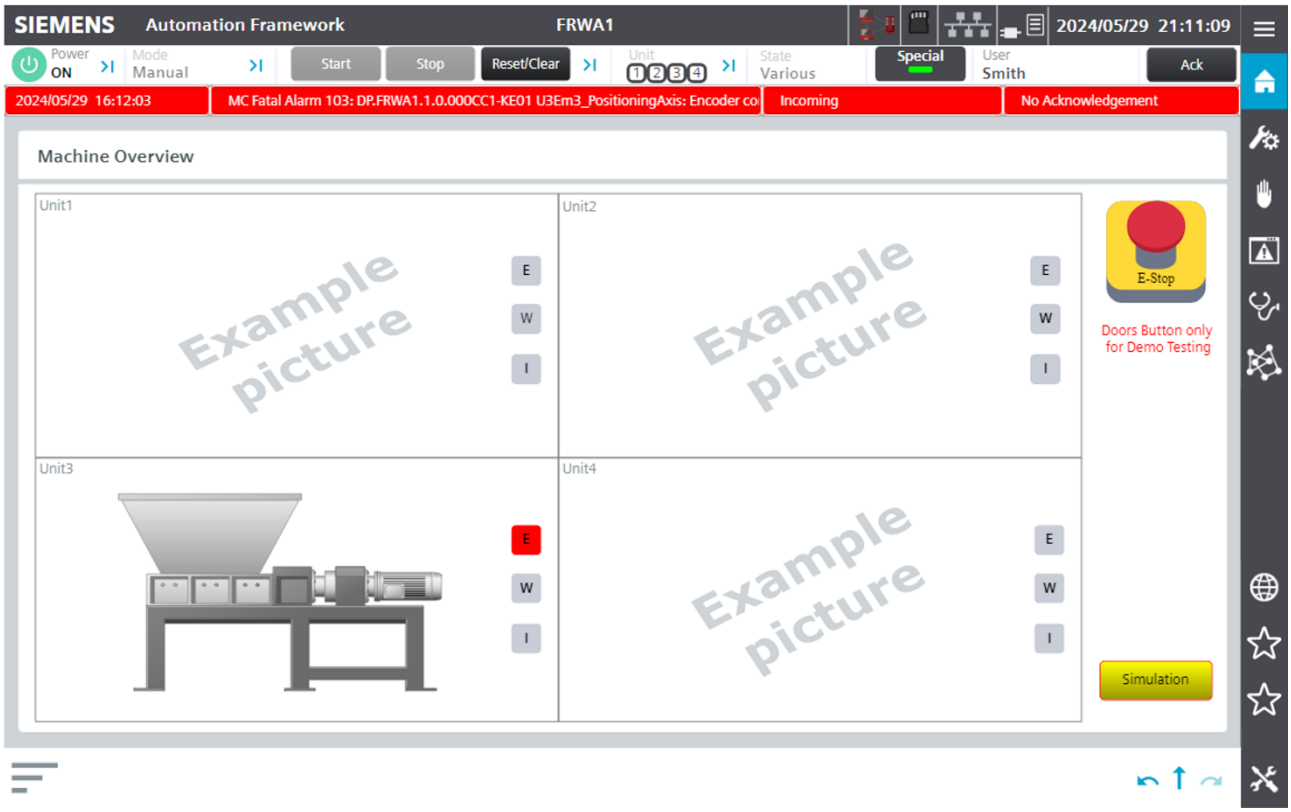


Figure 7-17: Unit Navigation – application section.

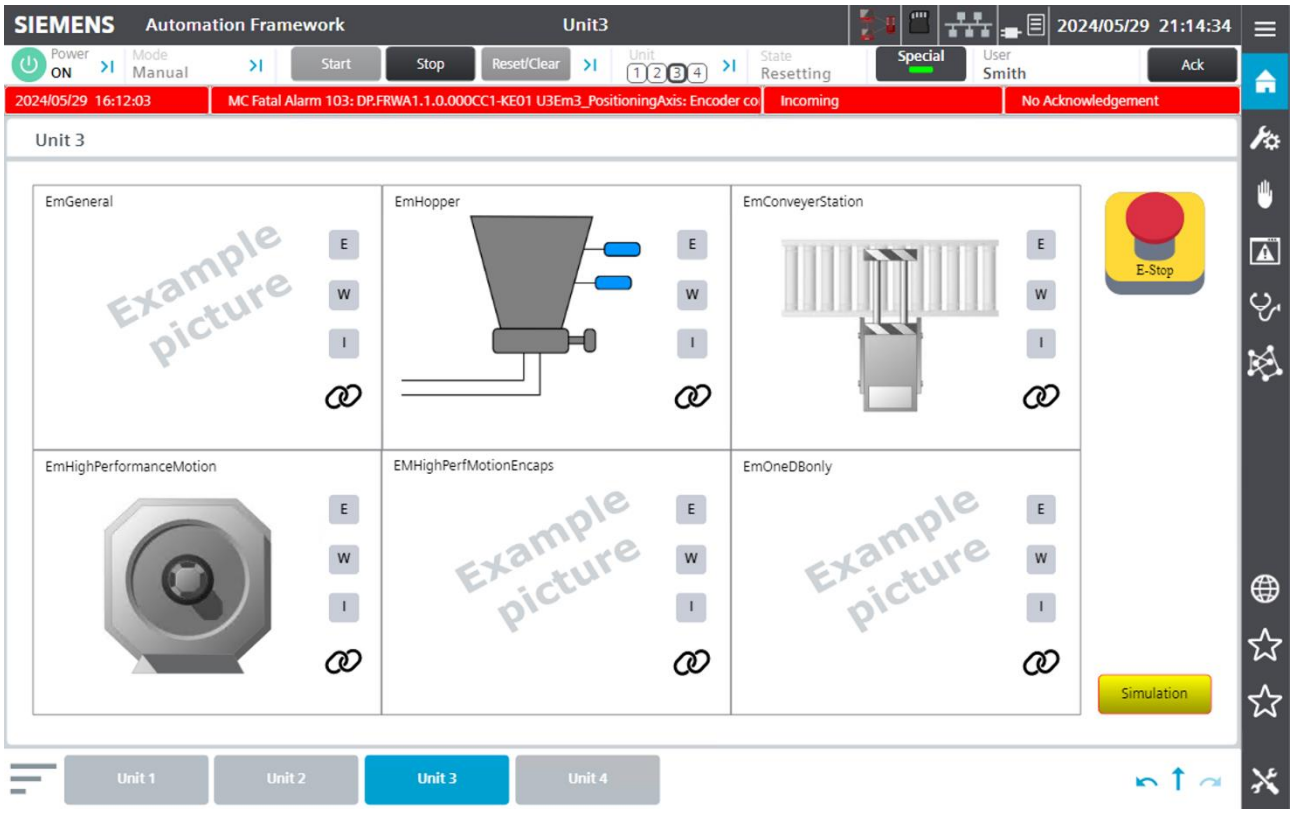


Figure 7-18: EM – application section.

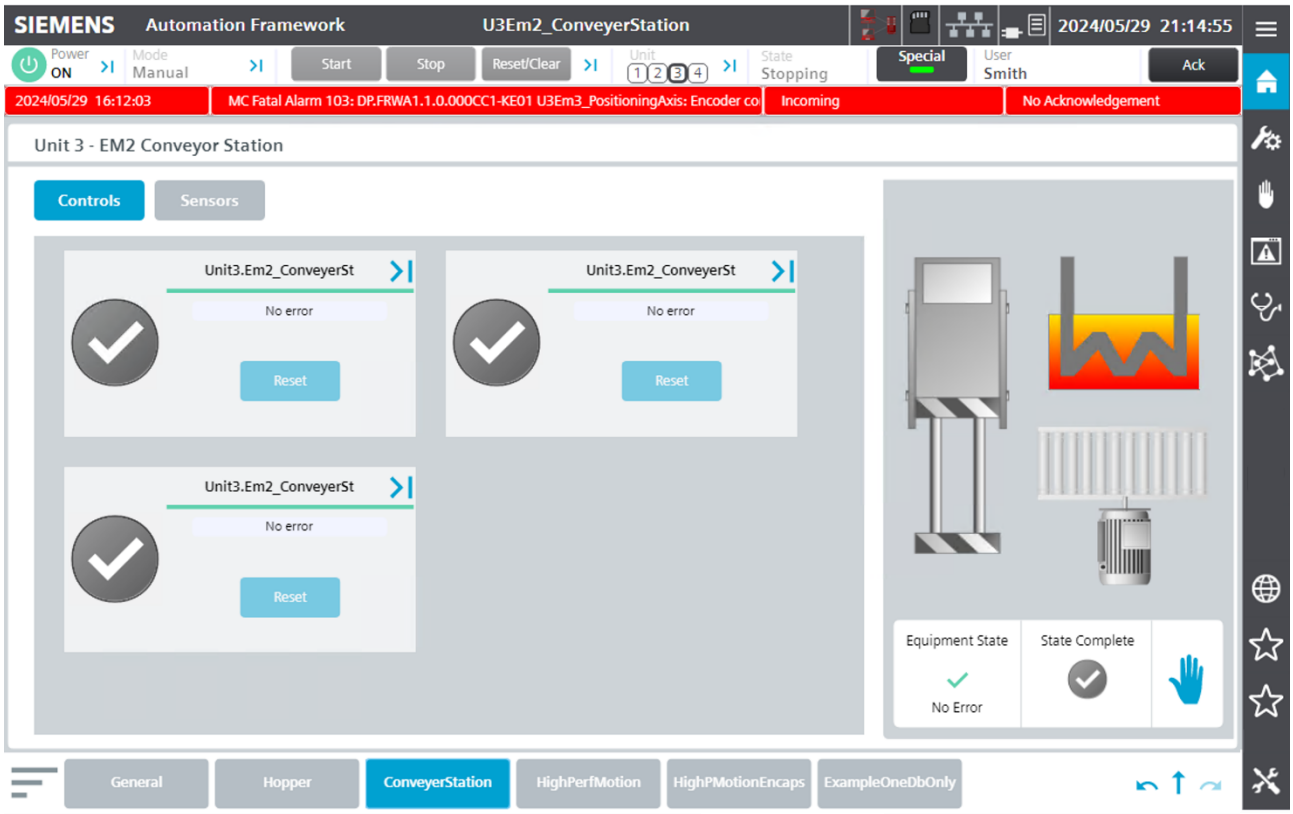


Figure 7-19: Control Module – application section.

SubNavigation & ThirdNavigation – Unit Navigation

The SubNavigtion and ThirdNavigation can be used to navigate through EMs and CMs in the same way. The Operator can click on the desired button to display the unit. The change from one unit to another can be easily executed by clicking on the multiple buttons available. The navigation through the units and the EMs and CMs are configured following the same concept. The name of those displayed buttons on RT are dynamically generated based on HMI Text Lists (Units, U1Em, U2Em, U3Em and U4Em) and PLC Constants where the system gets this information and dynamize those values on RT. The following pictures show how these navigations are displayed on the HMI panel:

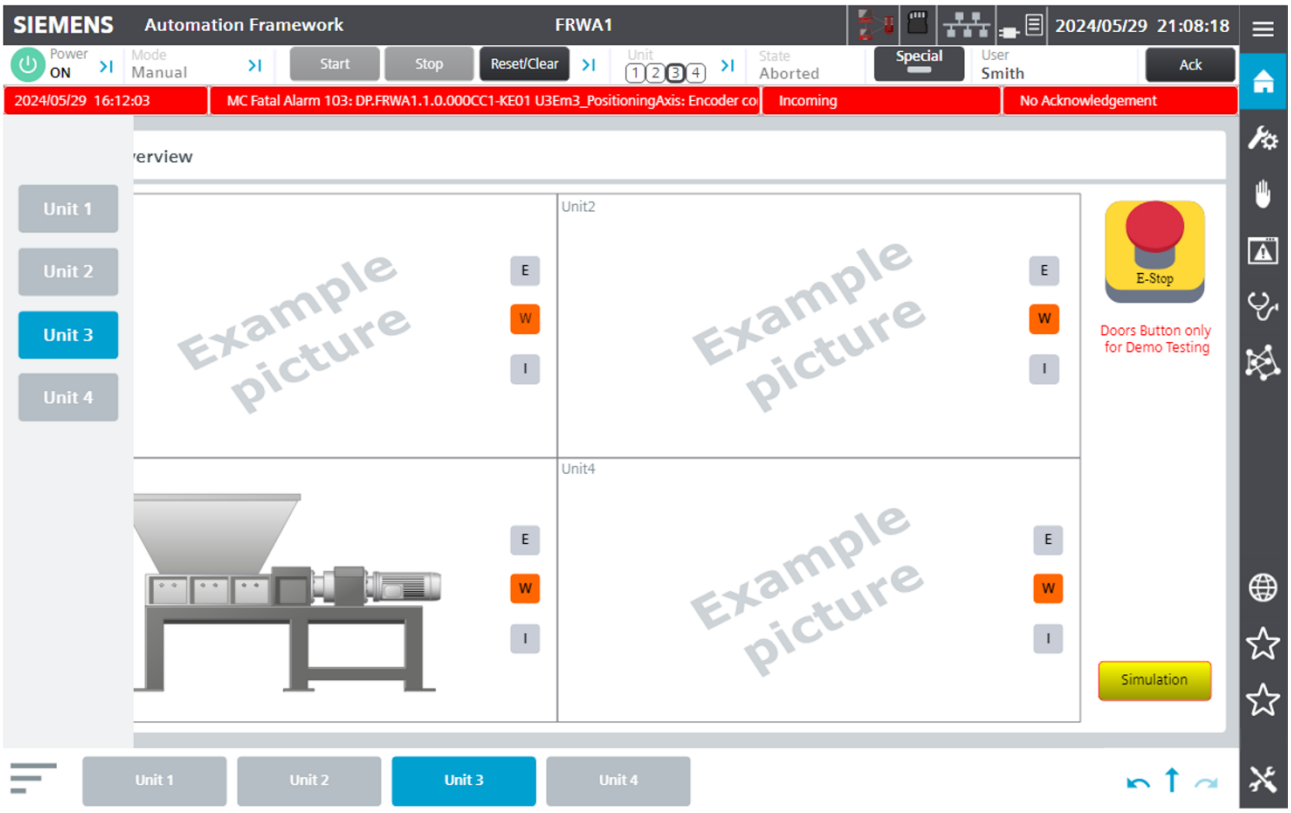


Figure 7-20: Unit SubNavigation and Unit ThirdNavigation.

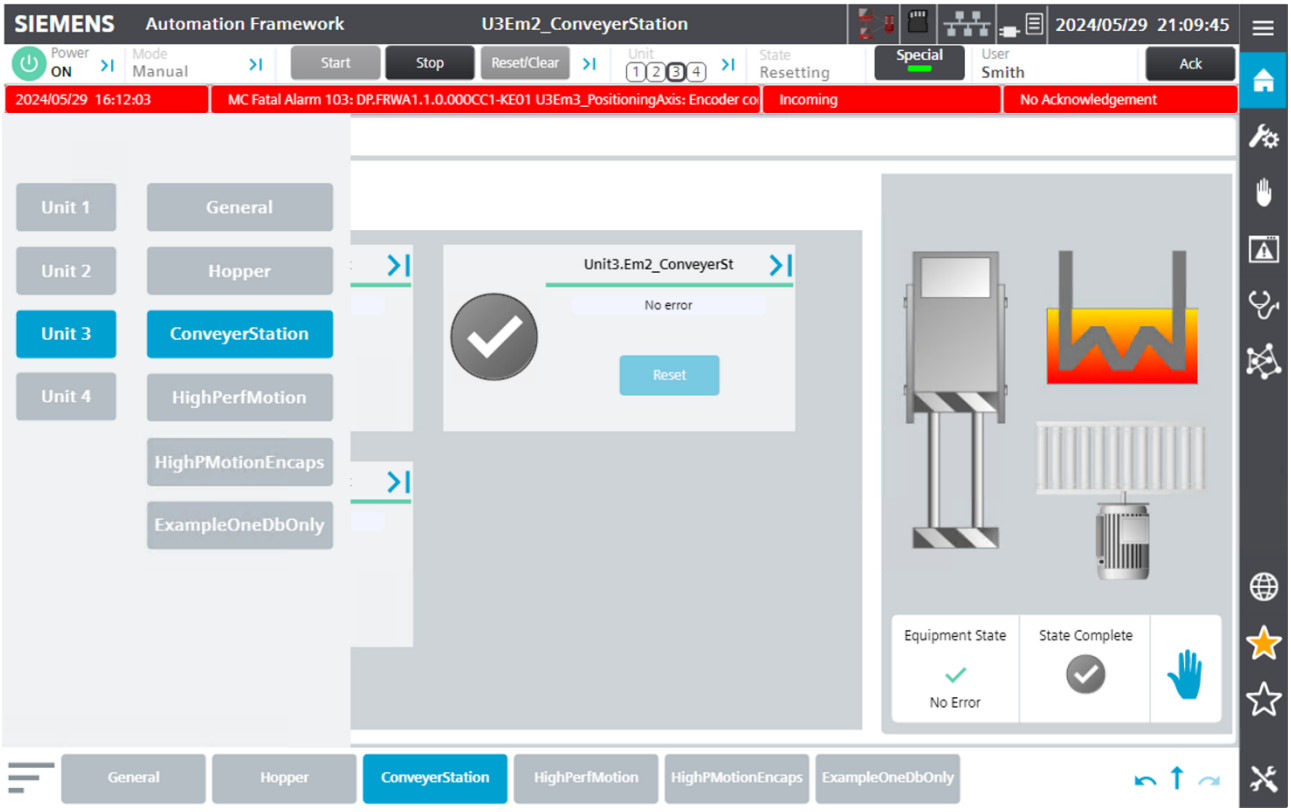


Figure 7-21: EM SubNavigation and EM ThirdNavigation.

7.3.5. General Navigation Commands

HMI navigation includes three additional commands to navigate through the screens. The operator can switch from one screen to another, using the following commands: Previous, Up and Next.

- Previous: It will displayed the previous screen the operaton has been visualizing.
- Up: It only works on Unit, EM and CM screens. It will displayed the upper level (e.g from CM to EM or from EM to Unit).
- Next: It will displayed the next screen. The operator should press previous first.

Figure 7-22 HMI General Navigation Commands – Previous, Up and Next.

7.3.6. HMI project tree navigation

The project tree navigation is based on the multiple types of navigation explained in the previous sections. Screens and HMI tags follow the same structure. This structure is shown in the following picture:

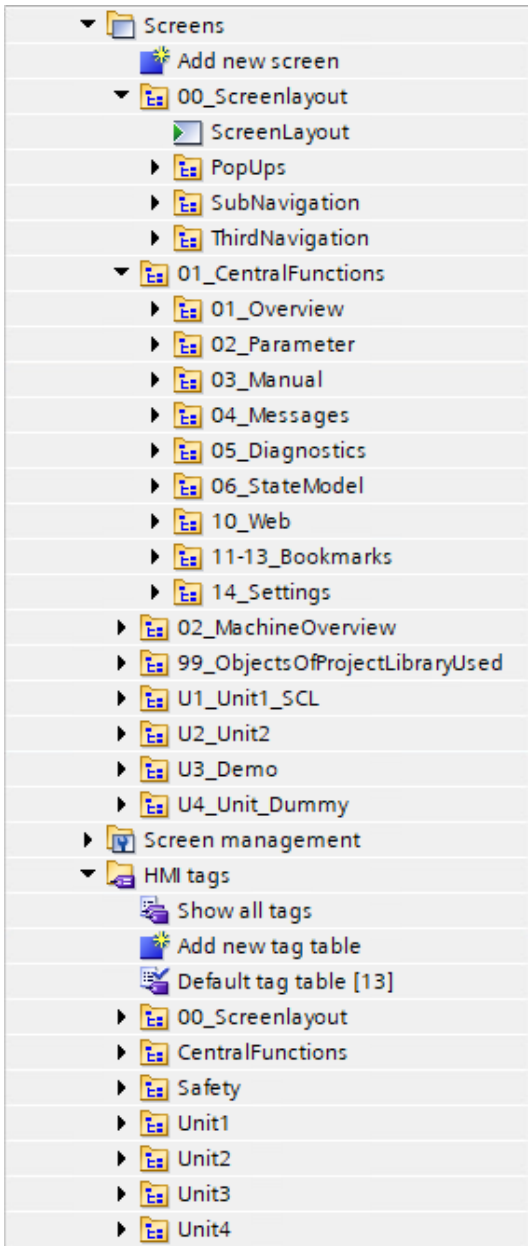


Figure 7-23: Project Tree structure – Screens and HMI tags.

7.4. Machine Overview

The machine overview screen is the highest level of the machine visualization on the HMI. Since the AF project supports multiple units, the highest screen therefore is the following overview screen. Additional elements like indicators for each unit about currently active error (e), warning (w), and information (i), etc. were added.

The ISA-88 machine structure is represented in a tree-architecture of the screens. By clicking on a unit, the screen will jump one lever lower to display all equipment modules of that unit.

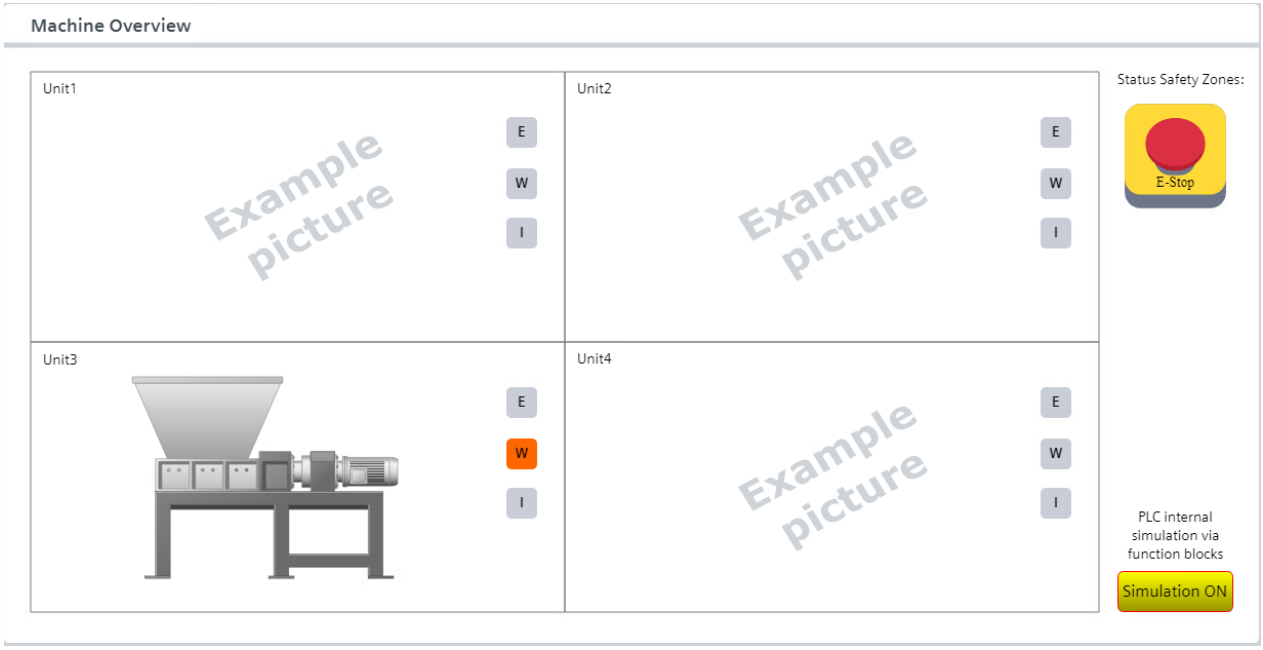


Figure 7-24: Example machine overview

7.5. Operation lock mechanism

In this case, there is more than one HMI to control the machine, and it is only allowed to control the machine from one of the HMIs at the same time. This can be achieved with the use of the Operation Lock function.

Concept

Basically, all HMIs will start with the Lock-Screen.

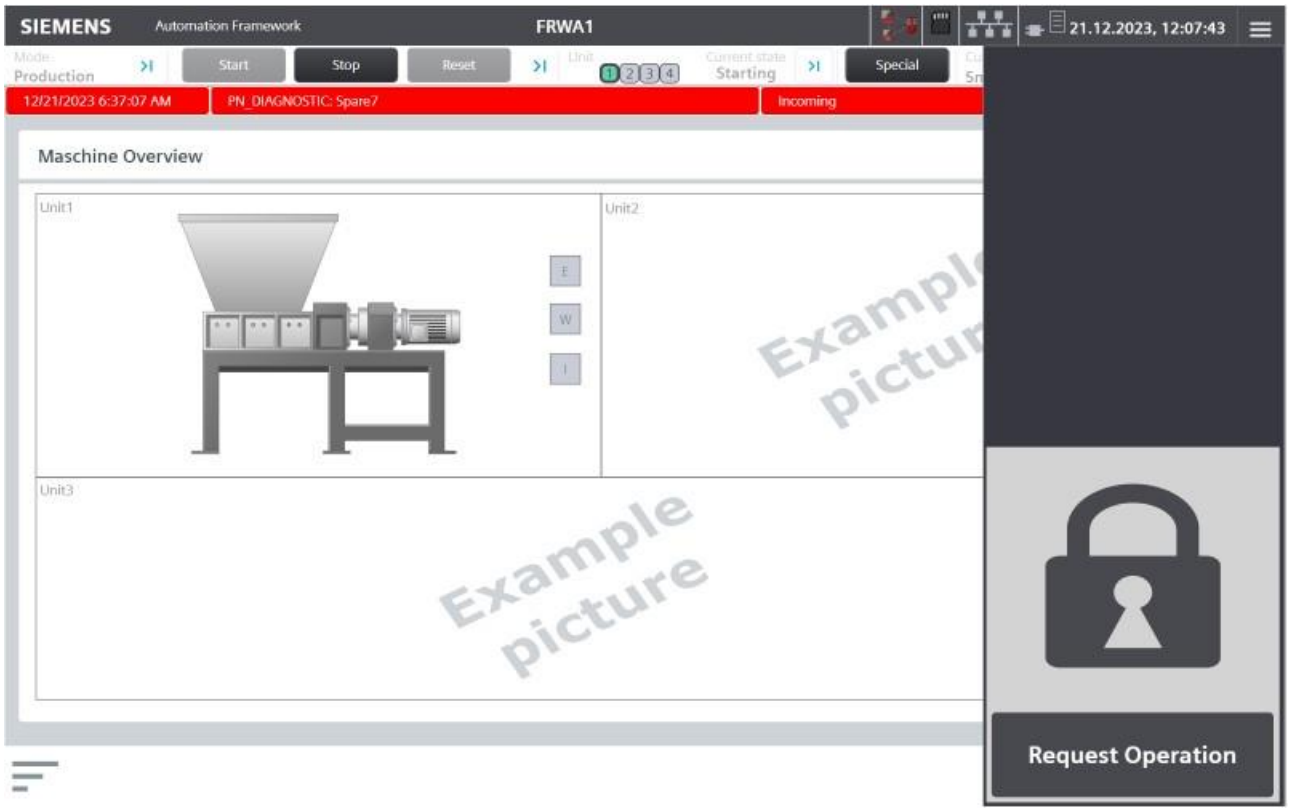


Figure 7-25: Operation lock.

When the machine is powered on, all HMI devices connected to this machine will show the general Operation Lock sidebar window. An operator needs to press the "Request Operation" button to control the machine.

The Operation Lock on an HMI indicates, that "no" or "another" station is operating the machine and the general operation right of the machine can be requested.

If an operator presses the "Request Operation" button on another HMI, the operator in control gets interrupted immediately and loses its right to control the machine. This default behavior can be customized.

Technical description

In the HMI screen "00_ScreenLayout" on the highest priority (31st) is the overlay screen called "swLock" which covers the complete screen area. When the Request Operation button is pressed, the following variables of the local HMI are copied from:

- @ServerMachineName to 00_Screenlayout\Hmi_operationInterlock\commands\"ServerMachineName"
- @LocalMachineName to 00_Screenlayout \Hmi_operationInterlock\commands\"LocalMachineName"

The swLock screen is visibility controlled by script, which is triggered by the PLC tags (Hmi.operatorInterlock.commands.serverMachineName and Hmi.operatorInterlock.commands.localMachineName) connected to HMI tags based on @ServerMachineName and @LocalMachineName from HMI Panels. It means when the operator presses the request operation button, the HMI changes these tags with its values and the access is granted to this specific HMI device. In this moment, the other HMI devices are being locked (since the ServerMachineName and LocalMachineName do no longer match to the PLC central values).

During the engineering or commissioning phase it is possible to disable the swLock function. To do so, the PLC variable "Hmi.operatorInterlock.commands.serverMachineName" must be overwriten with the WString "OFF"

NOTE The recommendation is to do it only manually via TIA Portal online. Do not change the start value.

operationInterlock	AF_typeOperationI...	Module interface for external systems
commands	AF_typeOperationI...	Module related commands from external systems
reqMaintenanceLock	Bool	TRUE: Request maintenance lock from HMI
reqMaintenanceReset	Bool	TRUE: Request reset of maintenance lock from HMI
serverMachineName	WString	@ServerMachineName tag of the panel currently in Operation
localMachineName	WString	

Figure 7-26: OperationInterlock data structure in the DB Hmi in Software Unit CentralFunctions.

8. Central Functions

Central Functions exist in the PLC and the HMI. In the PLC it is represented by a Software Unit, in the HMI by a folder. The folder structure and folder numbering inside "Global" is aligned between the PLC and the HMI, to easily find related folders. This chapter of the documentation explains both the PLC part and HMI part.

Central Functions contain system wide functionality which affect the entire system or are hierarchically above the unit level.

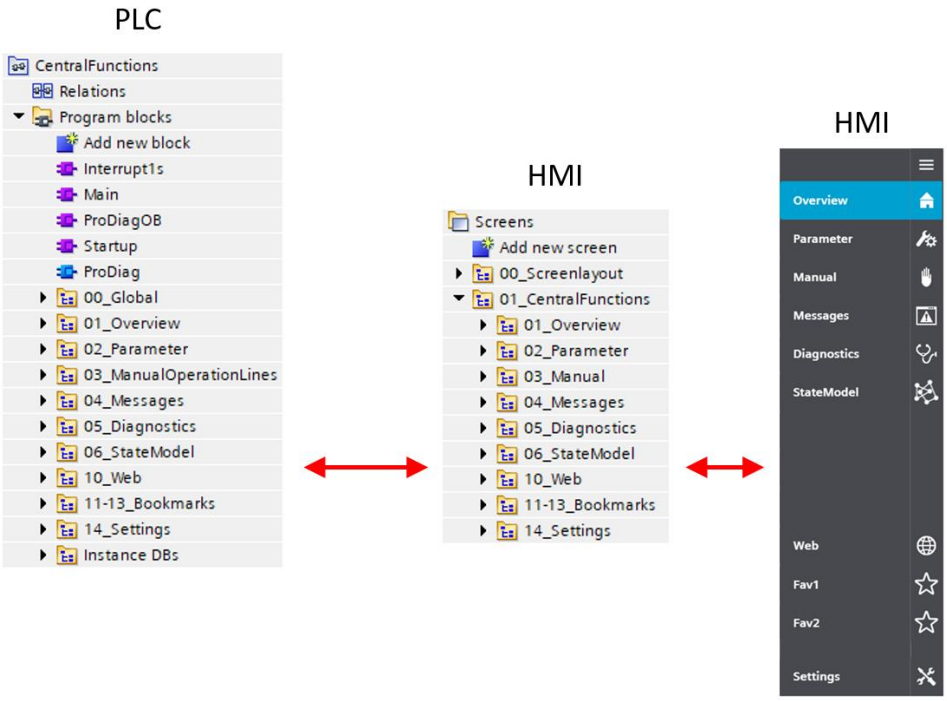


Figure 8-1: Overview of "Central Functions" in the PLC, HMI and side bar of the HMI.

8.1. Global

In the PLC, "Global" contains system functionality, HMI handling and simulation data.
In detail these are:

- Global Acknowledge
- Global Power On/Off
- Connection to Main Header on the HMI
- Local PLC date and time
- Clock bit generation
- Time synchronisation with NTP

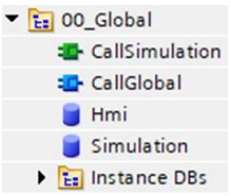


Figure 8-2: Content of folder "Global" in the PLC

In the HMI, there is no folder "Global" in "Central Functions", as the PLC functionality is related to the HMI main header in the screen "ScreenLayout".

Figure 8-3: HMI main header

The HMI header displays the most global information: Power status, current mode and state according to the selected units. It also allows the operator to send commands to switch the power status, mode and to send commands. Additionally, special functions could be selected as well as a global acknowledge command. The command integration was introduced in chapter Command and data flow.

The main header is connected to the DB "Hmi". FB "HmiHeaderManager" handles the data flow between the HMI and the unit. It can handle multiple panels and multiple units. Depending on the selected units in the HMI, the command given by the operator by the buttons are distributed to the respective units. Vice versa, depending on the selected units in the HMI, the button appearance and value displays (mode, state) are aggregated in the PLC from the units and send to the HMI. FB "HmiHeaderManager" handles all that. It coordinates multiple HMIs with multiple units.

FC "CallSimulation" automatically detects if a SIMIT simulation is connected and sets different bits accordingly.

8.2. Overview

The overview folder in the HMI contains pop-ups that are used in the machine overview screen. The AF provides two pop-ups:

- InterlinkControl : Enables the user to link and unlink and equipment modul from the unit.
- InterlinkControl_NonUnlinkable : Information box that the equipment module cannot be unlinked.

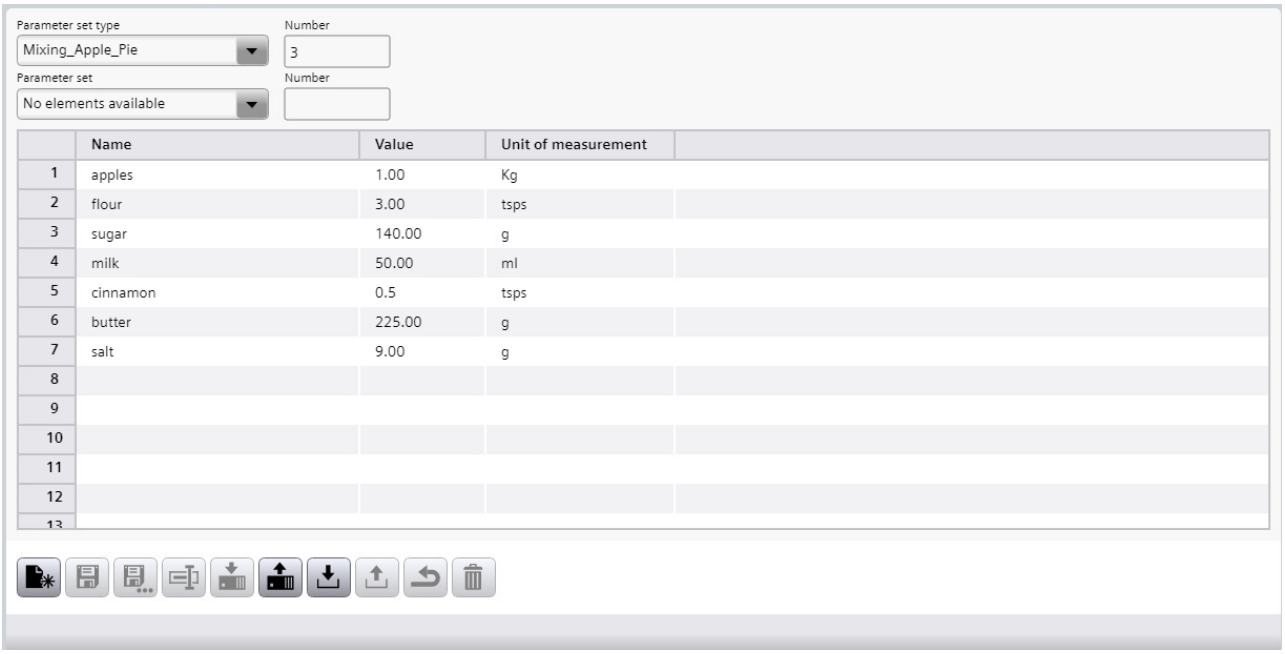
In the PLC, the functionality related to these pop-ups is handled within the FB "CallEquipment" of the equipment modules.

8.3. Parameter

In the parameter screens various use cases of configurations, e.g. commission settings, recipes, etc. can be implemented. In the AF project the screen uses the WinCC Unified "Parameter set control" object.

The parameter control is a comprehensive function for the control of parameter sets for configuration engineers, operators and recipe creators. The parameter control brings you the following benefits:

- You can apply the structure of a user data type for one or more parameter set types.
- You can change the structure of one or more parameter set types automatically via a new user data type version.
- You can exchange a large number of parameters manually or automatically between HMI device and PLC to set up a machine for a production.
- You can create parameter sets simply and uniformly for products to be manufactured in the works during engineering or during ongoing operation.
- By structuring the associated parameters/setpoints, you can easily transfer parameters.



	Name	Value	Unit of measurement
1	apples	1.00	Kg
2	flour	3.00	tsps
3	sugar	140.00	g
4	milk	50.00	ml
5	cinnamon	0.5	tsps
6	butter	225.00	g
7	salt	9.00	g
8			
9			
10			
11			
12			
13			

Figure 8-4: Parameter set control.

For more details, please have a look into the Documentation for WinCC Unified - Creating and configuring parameter set control

https://support.industry.siemens.com/cs/ww/en/view/109821886

8.4. Manual

In the manual screens, various use cases for manual operating CMs can be implemented. For this, the AF introduced the Manual Operation Lines, a straightforward solution with high custom ability to control a CM with two buttons and display the information about end switches and/or values like current speed and position. An example of a manual operation line can be seen in the following image:

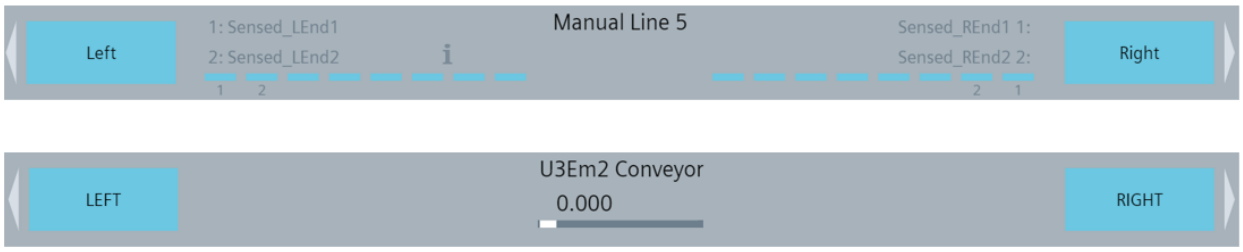


Figure 8-5: Example of Manual Operation Lines.

There are mainly two use cases for using the manual operation lines:

- Using the prepared generic screen "LAF_ManualOperation1_1280x800", here the user can get an overview of the CMs. This solution is simpler to use, when adding new CMs, since most of the HMI architecture is ready-to-use in the AF.
- Using the faceplates in a custom screen, this solution needs more work, since the prepared features need to be added manually to the screen to function properly.

8.4.1. Getting Started with Manual Operation Lines

This chapter introduces the necessary configuration elements for the Manual Operation Lines. A detailed step by step example can be found in the following chapter Framework Demo implementation.

Global DB for all manual operation lines

For internal usage of Manual Operation Lines a global DB named "LAF_InternalManual" with an Array[1..X] of "LAF_typeManHMIDisp", has to be created. A configuration is not necessary because the Automation Framework project works with one HMI (the array size determines this).

Equipment module DBs

Following the AF programing recommendation each EM should have a dedicated DB for each CM being operated with Manual Operation Lines. A corresponding template DB "ManualOperationControl_Template" can be found in the "Central Functions" Software Units under the folder "03_ManualOperationLines". The data structure contains three elements:

- The EM configuration "manualOperationEm" of type "LAF_typeManEm"
- The array "manualOperationEmArea" of type "Array[1..<X>] of LAF_typeManOperationArea" with X as the number of areas the Manual Operation Lines are being grouped
- The array "manualOperationCms" of type "Array[1..<Y>] of LAF_typeManOperation" with Y being the number of CMs in the EM

With the "manualOperationEm" only the "equipmentModuleNum" configuration needs to be set. The "equipmentModuleNum" has the identification values of the EMs. Since the EMs is part of a Unit, it is recommended to use the second and third digit from the right for the EM number and the fourth and fifth from the right for the Unit number. For example, the EM 2 in Unit 3 in the Area 5 has the value "3025". Since the datatype is <UInt> the maximum value should not exceed 64999 (Unit: 64, EM:99., Area:9)

For the "manualOperationEmArea" the only setting for each area in the array to be set is the "displayedLines" of type <string>. The string can be configured such that various combination of Manual Operation Lines in that area can be configured. For example, to display lines 1 to 6 use the format '1-6;'. To do sorting using commas, for example '6,5,1,8' to

display CMs in the order 6 -> 5 -> 1 -> 8. If there are more than 6 CMs for one area, there will be a page up and down in the "LAF_ManualOperation1_1280x800" screen.

For the "manualOperationCms" the only configuration is the number of CMs by setting the array size.

NOTE Similar to all configuration settings in the different Software Units it is recommended to set the Manual Operation Lines configurations described above during the Startup OB.

Textlists

The text lists have two different functionalities:

- The color configuration for the manual lines
- The area names for the EMs

The text lists for the colors have the format "LAF_manualColor<X>" with X being number for the selection in the Json files, like "LAF_manualColor1". The color text lists have a fixed format for the values being 1 – normal state, 2 - pressed, 3 - confirmed, 4 – deactivated and 5 - deactivated+pressed, and the color values are set in the Text with the hex code. The first two digits of the hexcode after "0x" are the transparency setting, being "ff" for 0% transparent and "00" for 100%, and the last 6 digits are the normal color hex code.

The text lists for the area names have the format "EquipmentAreas_<X>" with X being the equipment area code, like "EquipmentAreas_3020". This Text list is used for the screen "LAF_ManualOperation1_1280x800" so that the areas shown in the following image have the respective text depending on the EM:

Figure 8-6: Area names for each EM.

Json Files

A Json file is needed for each control module/manual operation line. Each Json file has the information about the name, value unit, amount of end switches, button colors, end switches colors and names for each end switch. The format of the Json file name is "<X>_<Y>_<Z>.json" with X as the EM number, Y as the line number and Z as the Language Code. An example of Json file name would be "3020_1_1033" for line 1 of EM 3020 with the language English (defined by the Locale ID standard as 1033).

The Json file needs to be added manually to the PC Runtime or panel and the path where it needs to be placed is configured in the HMI Tags as "manualOperationFilePath".

NOTE When using a Windows PC Runtime path strings must include additional escape characters like the following: "D:\\JsonFiles\\" (double backslash).

When using a Unified Panel, the path should be "/home/industrial/JsonFiles/" without the additional escape characters.

The Json files need to be transferred via USB-Stick to the Unified Panel. In the Panel the App "File Browser" can be open and the Json file folder can be copy and pasted to the respective path. To copy an item in the "File Browser" use the menu "Edit".

FC Calls

For the use of the Manual Operation Line there are two FCs mandatory:

- "LAF_manEmConfig" and
- "LAF_manOperationCm"

The following configuration is necessary for:

- LAF_manEmConfig

The three strcutures of the EM configuration DB ("manualOperationEm", "manualOperationEmArea", "manualOperationCms") introduced in the beginning of this chapter, must be connected to the FC "LAF_manEmConfig". Additionally, the DB "LAF_InternalManual", used for the screen Data must be connected to the FC to manage the "LAF_ManualOperation1_1280x800" screen. The following image shows an example of such configuration.

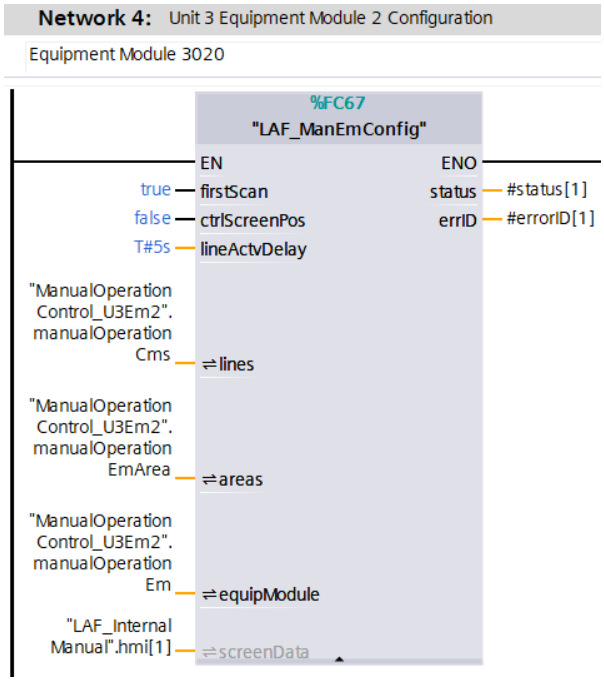


Figure 8-7: Block: "LAF_manEmConfig" configured for EM 2 of Unit3.

- LAF_manOperationCm

With the "LAF_manOperationCm" the variables used for the CM control are connected. Each CM must be connected to only one "LAF_manOperationCm" FC. All FC inputs are described in the block documentation but in the following examples for some basic functionality:

Input(s)	Description
"rightEndPosSens<X>" and "leftEndPosSen<Y>"	Feedback from the process about end switches being On or Off
"execRightCmd" and "execLeftCmd"	Control the CM's hardware like a physical button
"cnfrmExecRightCmd" and "cnfrmExecLeftCmd"	Feedback that the command is being executed
"expandable"	Open the screen windows for the specific control node. The screen that will be open needs to use the name "ControlScreen_<X>_<Y>", with X as the EM number and Y the line number.

...

Table 8-1: A selection of "LAF_manOperationCm" inputs

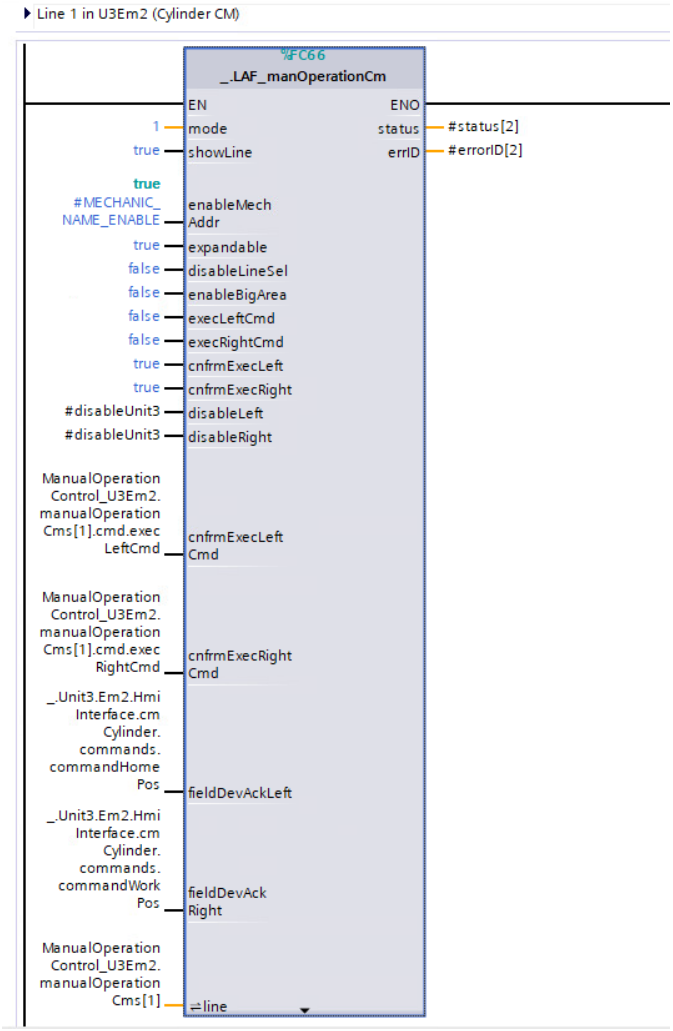


Figure 8-8: Configuration example of "LAF_manOperationCm" for Manual Operation Line1 in EM2 of Unit3.

8.4.2. Framework Demo implementation

The Manual Operation Lines are implemented in the AF project in the main navigation under manual operations.

In the AF project all library blocks and UDTs for the Manual Operation Lines have been added into the Software Unit "ObjectsOfProjectLibraryUsed", located inside the folders "Program Blocks -LAF - 10_Central - ManualOperationLine".

All the UDTs and blocks are published, so that they are accessible from the Software Unit where they are needed. As they are part of the main navigation bar, in the AF project, the Manual Operation Lines have been configured in the "CentralFunctions" Software Unit, in a folder called "03_ManualOperationLines" where the lines are called.

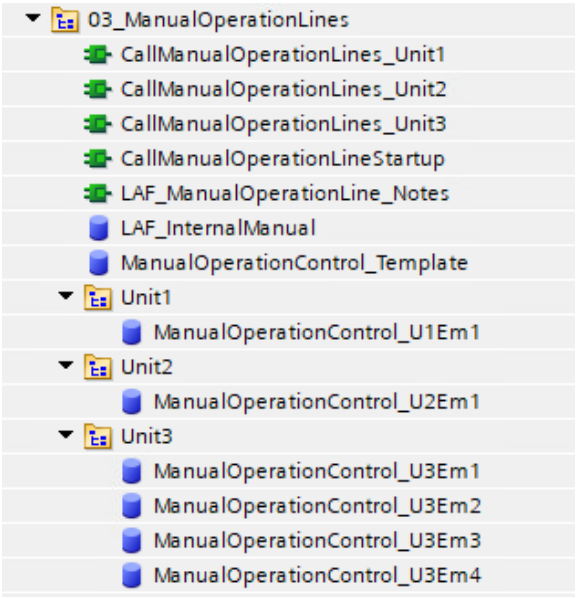


Figure 8-9: Blocks programmed for the Manual Operation Lines.

Base configuration

The "CallManualOperationLines_UX" FC is created to call the following blocks: the "LAF_manEmConfig" for each EM and "LAF_manOperationCm" blocks for each line that is required to be configured as explain in the previous Getting Started chapter.

The connections for the FCs in the "CallManualOperationLines_UX" are done through DBs. Each EM has a DB for the configuration, the areas configuration, and the CMs. Each ManualOperationControl DB has the Unit number and EM number as extension, and are located in folders with the name of the Unit they belong to. All of them are created based on the template DB "ManualOperationControl_Template". The following image shows how one of the DBs, for EM2 of Unit3 was configured. The changes from the templated are highlighted in the red:

- number of the EM with the correct nomenclature (3020).
- number of lines to be displayed for each area, here six lines in the first area.

ManualOperationControl_U3Em2

	Name	Data type	Start value
1	Static		
2	manualOperationEm	"LAF_typeManEm"	
3	equipmentModule...	UInt	3020
4	actvArea	UInt	0
5	pages	"LAF_typeManPageDetails"	
6	manualOperationEm...	Array[1..7] of "LAF_typeManOperationArea"	
7	manualOperation...	"LAF_typeManOperationArea"	
8	areaNum	UInt	0
9	displayedLines	String	'1-6;'
10	hmiPage	"LAF_typeManPageDetails"	
11	manualOperation...	"LAF_typeManOperationArea"	
12	manualOperation...	"LAF_typeManOperationArea"	
13	manualOperation...	"LAF_typeManOperationArea"	
14	manualOperation...	"LAF_typeManOperationArea"	
15	manualOperation...	"LAF_typeManOperationArea"	
16	manualOperation...	"LAF_typeManOperationArea"	
17	manualOperationCms	Array[1..10] of "LAF_typeManOperation"	

Figure 8-10: Manual operation DB configuration example U3 EM2.

The "CallManualOperationLineStartup" FC in the "CentralFunctions" Software Unit, is used to set the configuration of the Manual Operation Lines for the EMs, to ensure that the configuration will not be in blank after a startup.

Connecting the CMs

After the configuration of the DB, the connections are added to the blocks inside "CallManualOperationLines", connecting several of the inputs to the corresponding control node DB of the EMs. For example, in the case of Line1 of EM2, Unit3, the connections are the following:

- The inputs "cnfrmExecRightCmd" and "cnfrmExecLeftCmd" are used to get the feedback, that the "button is being pressed" and the PLC got this information.The simple implementation is to use "execRightCmd" and "execLeftCmd" from the interface, similar to "ManualOperationControl_U3Em2.manualOperationCms[2].cmd.execLeftCmd".
- The input "leftEndPosSens1" is connected to "_.Unit3.Em2.HmiInterface.cmCylinder.monitoring.inHomePos" which is the location of the parameter "inHomePos" from the cylinder example of the EM2 in the Unit3 that is being controlled through this specific Manual Operation Line.
- The inputs "fieldDevAckLeft" and "fieldDevAckRight" are the acknowledgement of execution from field device and it is connected to "_.Unit3.Em2.HmiInterface.cmCylinder.commands.commandHomePos" which comes from the control node of the cylinder when the command to go to home positition has been activated.

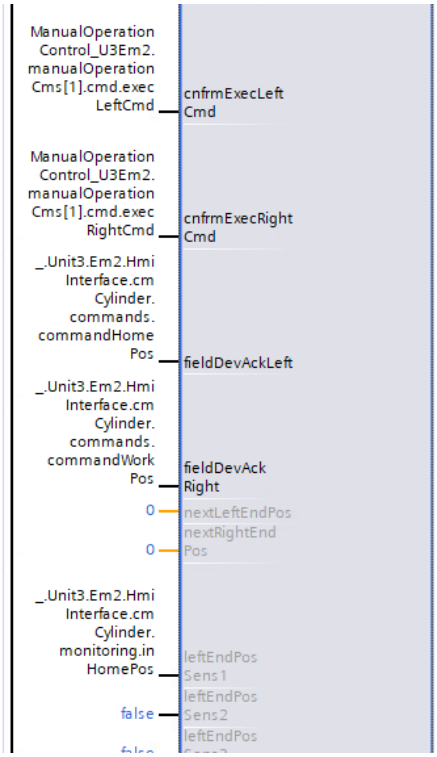


Figure 8-11: Closer look to inputs configured in "LAF_manOperationCm".

Once all the configuration blocks of the EM and the blocks for each line are set up in the "CallManualOperationLines", the implementation of the Manual Operation Lines in the PLC is completed.

HMI implementation

The HMI tags are configured trough the predefine table called "03_manualOperation", located in the "HMI tags – CentralFunctions" folder; establishing the connection with the corresponding PLC tag that will come from the DB "LAF_InternalManual and which will show the data from the HMI interface.

As it was mentioned in chapter 8.4.1. here in the HMI tag table is where the path for the json files needs to be specified. The tag to be modified is "manualOperationFilePath", and the path should be specified in the Properties>Values (see image 9-11). Since the project uses a Unified Panel, the path value would be "/home/industrial/JsonFiles/", but as shown in the image below, the initial path was set to "D:\\JsonFiles\\" for testing in the simulation. When downloading to a Panel, the loaded event of the "ScreenLayout" screen changes the json file path automatically to "/home/industrial/JsonFiles/".

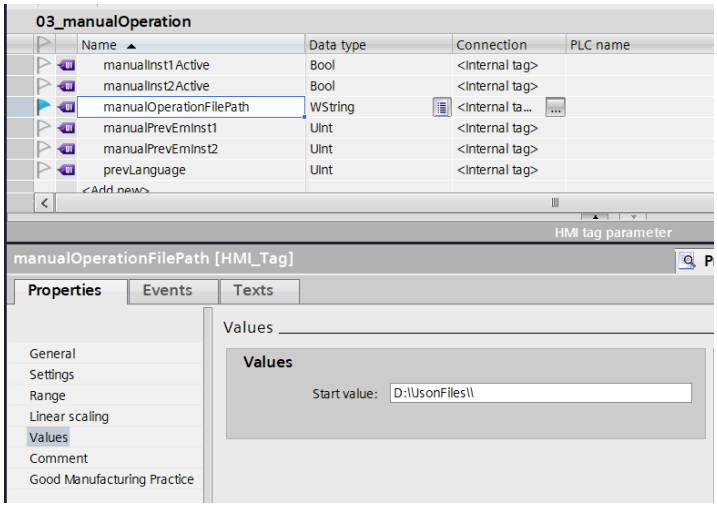


Figure 8-12: HMI Tag "manualOperationFilePath" configuration.

As next step, the text lists need to be added. There will be two different kinds of text lists that affect the use of the Manual Operation Lines in the AF project:

- "EquipmentAreas" text lists : need to be configured for each EM with the corresponding name. For example, in the "EquipmentAreas_1010" text list the areas are configured as in the image below:

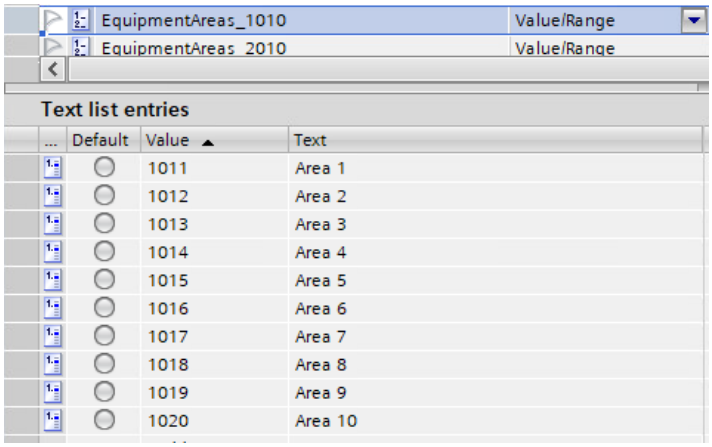
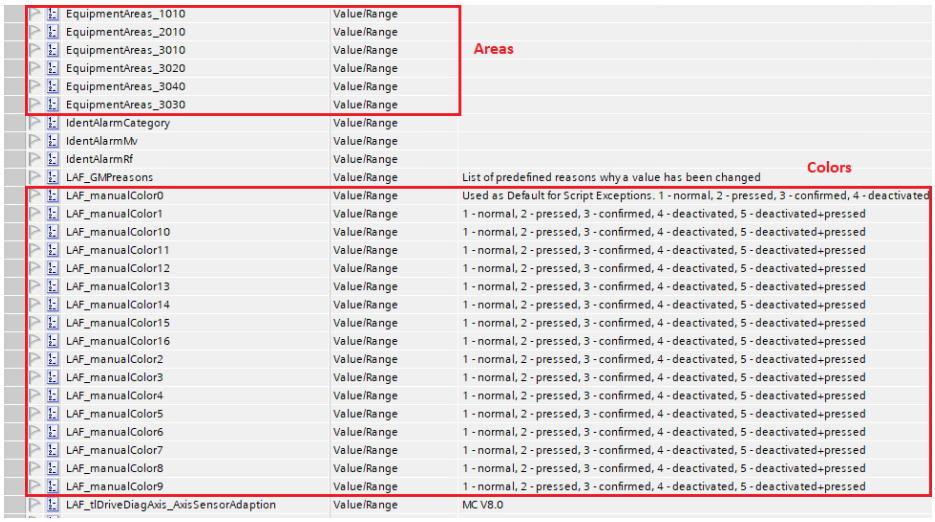


Figure 8-13: "EquipmentAreas_1010" text list.

- "LAF_manualColor" text lists : They are text lists from 0 to 16 which contain as text, the colors used in the Manual Operation Line faceplates as a hexadecimal value. These text lists are needed to transform the colors that are read from the json files.



EquipmentAreas_1010	Value/Range	
EquipmentAreas_2010	Value/Range	
EquipmentAreas_3010	Value/Range	
EquipmentAreas_3020	Value/Range	
EquipmentAreas_3040	Value/Range	
EquipmentAreas_3030	Value/Range	
IdentAlarmCategory	Value/Range	
IdentAlarmMv	Value/Range	
IdentAlarmRf	Value/Range	
LAF_GMPreasons	Value/Range	List of predefined reasons why a value has been changed
LAF_manualColor0	Value/Range	Used as Default for Script Exceptions. 1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated
LAF_manualColor1	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor10	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor11	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor12	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor13	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor14	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor15	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor16	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor2	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor3	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor4	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor5	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor6	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor7	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor8	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_manualColor9	Value/Range	1 - normal, 2 - pressed, 3 - confirmed, 4 - deactivated, 5 - deactivated+pressed
LAF_tlDriveDiagAxis_AxisSensorAdaption	Value/Range	MC V8.0

Figure 8-14: Manual Operation Lines text lists needed in the Framework Demo.

In the following step, the scripts for the Manual Operation Lines must be added to the HMI project. There are two different folders which contain multiple mandatory scripts for the lines to work:

- The first one is the "ManualOperation" folder which contains the main scripts used inside the screens

The second folder is called "ManualOperationAddrMode," which contains the scripts to display, when necessary, the other addressing modes such as electrical addressing and mechanical addressing.

NOTE **Addressing mode:**
If a manual operation line is touched in the middle, up to three different texts (symbolic, electrical or mechanical) can be displayed.

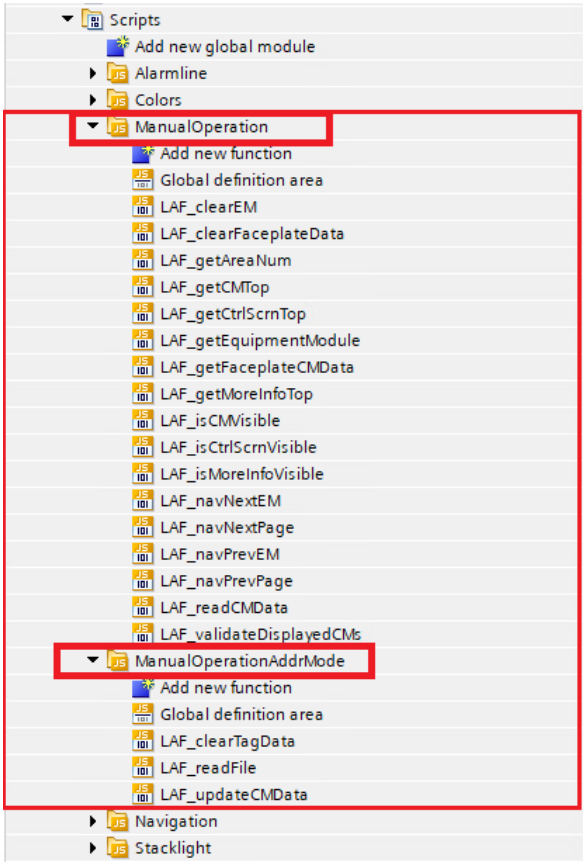


Figure 8-15: Manual Operation Lines scripts.

The screens for the Manual Operation Lines need to be created. In this case, they have already been created in the "03_Manual" inside the "01_CentralFunctions" folder. In the following step, the focus will be on setting up the "LAF_ManualOperation1_1280x800".

This screen contains the tabs for the different areas (which access to the corresponding text list by using a script to get the area number) and a main screen window that will open the "LAF_ActvInst1_1280x800" screen.

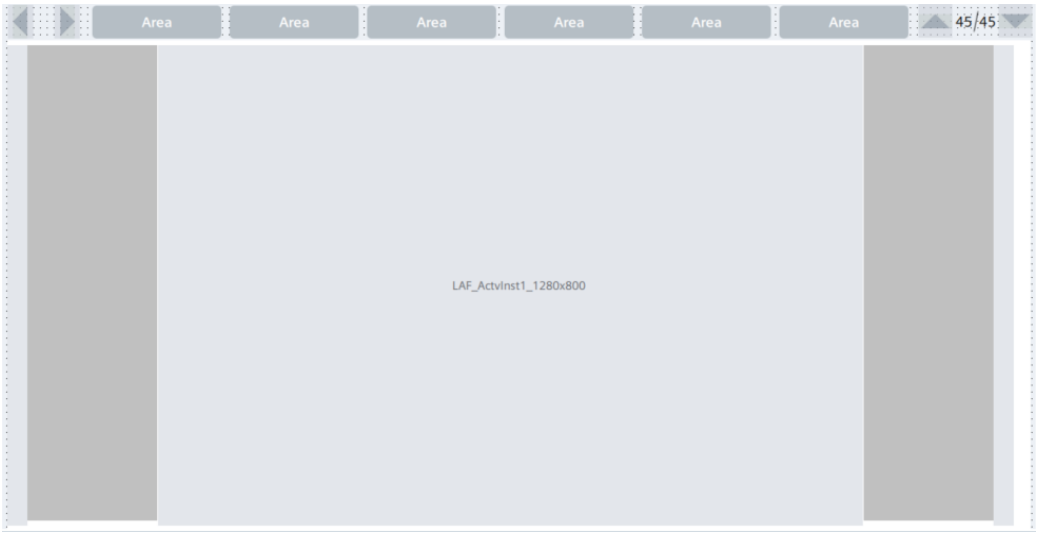


Figure 8-16: "LAF_ManualOperation1_1280x800" screen.

- "LAF_ActvInst1_1280x800" : This screen contains the faceplates for the Manual Operation Lines, in the position they should appear. Each one of the faceplates should be configured to connect through the interface to the HMI tags that will display the needed information in the manual line. In the next image an example is shown of the interface configuration for line0 (which would be the first one to appear in the screen and its connected in the HMI tag table to the line 1 of the DB in the PLC):

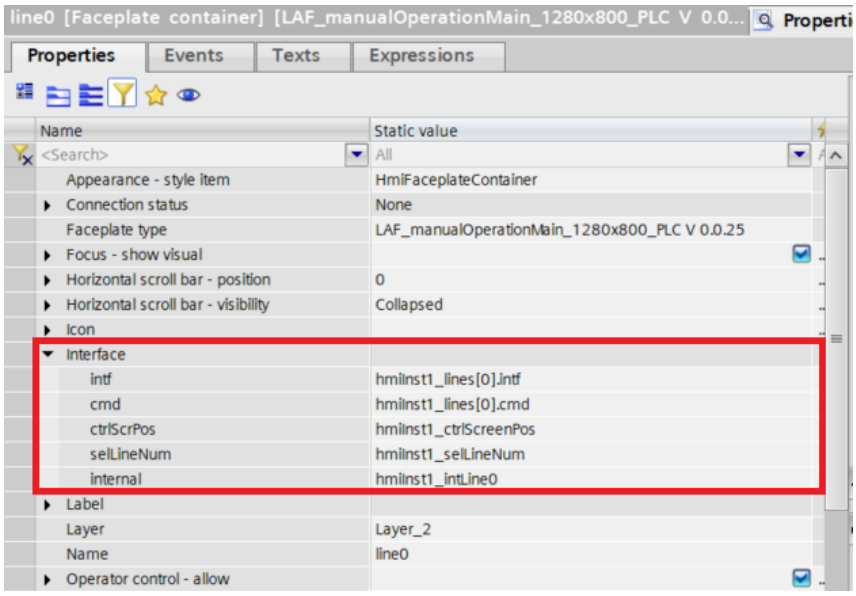


Figure 8-17: Interface configuration for faceplate of the Manual Operation Line 0.

In the last step the Json files are used to render the Manual Operation Lines. Without the files, the faceplates would appear completely blank in the screen. The mandatory information is the line name, the unit, the number of end switches, the button colors, the end switches colors and the end switch names.

The json files for each Manual Operation Line must be stored in the folder specified in the HMI tag "manualOperationFilePath", in the case of the Framework Demo is "D:\\JsonFiles\\". There, the json files must be named as "<X>_<Y>_<Z>.json" with X as the EM number, Y as the line number and Z as the Language Code. The structure of the json file must follow the example in the next image (the number of End Sensors can vary up to maximum of eight Right End Sensors and eight Left End Sensors):

Figure 8-18: Example of Json File structure for Line 1 of the EM2 in the Software Unit3 (3020_1_1033.json).

NOTE The TIA Portal project contains a *.json-file helper (Excel-Sheet) to define the json-files. For more details see folder TIAPortalProjectPath/UserFiles/ManualOperationLines.

8.5. Messages

The Message has three sub menus. The first two handle current alarm/messages viewer, the second opens the historical viewer while the third sub navigation open the viewer for audit trails.

8.5.1. Alarms

The current alarms and warnings can be shown via the Button "Messages" of the Main-Navigation. To view the Alarm History, use the appropriate Button in the Sub-Navigation. Messages are displayed with the WinCC Unified "Alarm Control".

With the button "Alarm History" you can display the archived messages. The system diagnosis is also displayed.

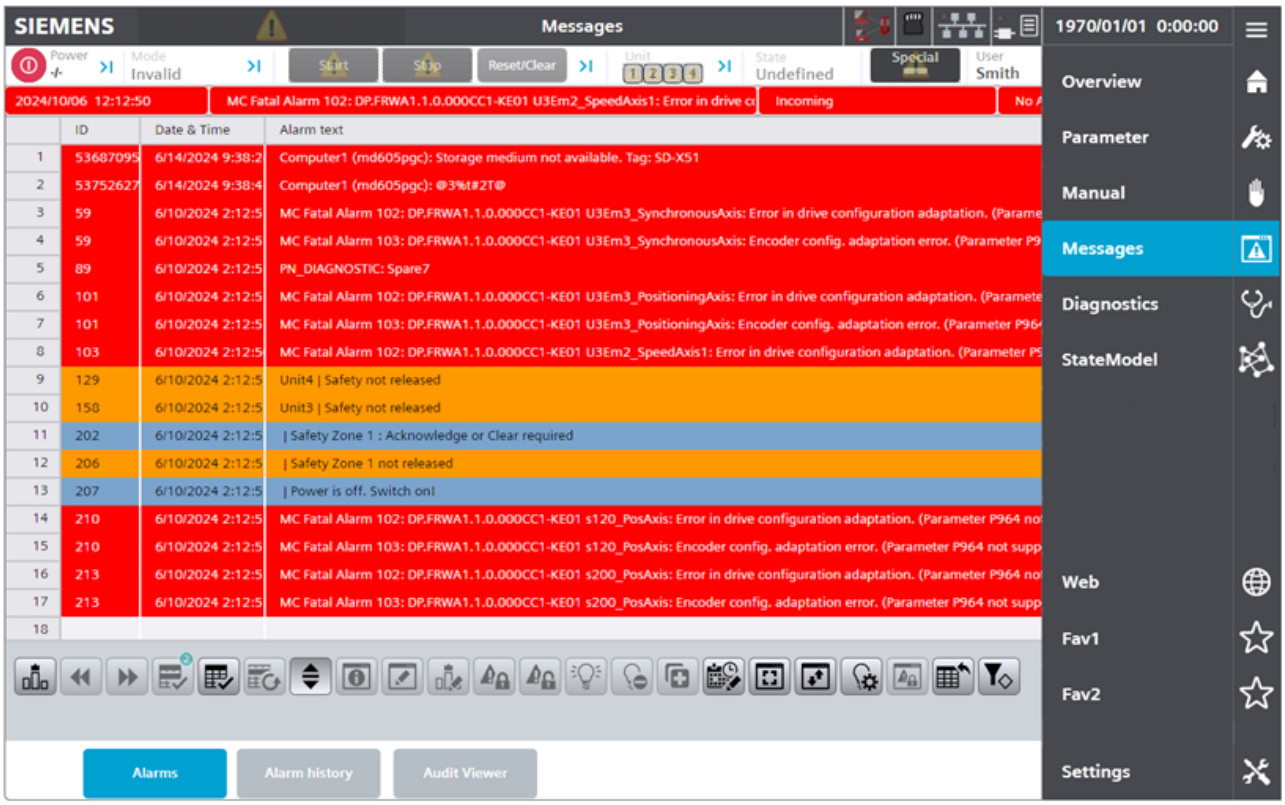


Figure 8-19: Alarms

8.5.2. Audit

Audit Trail is used when operator action must be traced e.g. to comply with regulations like FDA Guideline 21 CRF Part11 or EU regulation 178/2002.

Introduction

"Configuration conforms to GMP" means creating projects in accordance with "Good Manufacturing Practice". The requirements are set out in FDA rules 21 CFR Part 11. The FDA is the U.S. Food and Drug Administration. Also the Eudralex Volume 4, Appendix 1 of the EMA (European Medicines Agency) regulation 178/2002 applies.

GMP-relevant and Audit Trail

WinCC Unified offers the "Audit" option for implementing GMP compliance. Using the Audit option, the "Configuration conforms to GMP" function can be enabled.

Enable the "Configuration conforms to GMP" function directly in the Runtime settings of the HMI device. GMP relevant functionalities are then added to WinCC Unified.

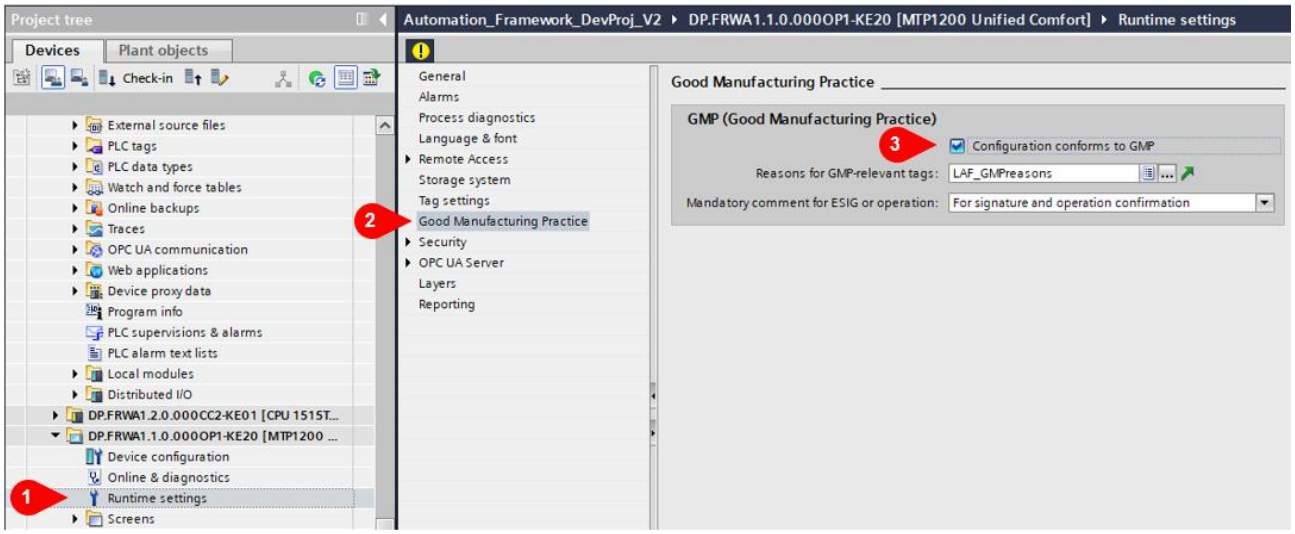


Figure 8-20: Enable Good Manufacturing Practice.

These functionalities are:

- Electronic signature
- Option to label tags as "GMP relevant"
- Suggestions for the comment with typical reasons for changes of GMP-relevant tags can be pre-defined by selecting text lists
- Automatic identification of significant changes in GMP-relevant tags
- Generation and storage of electronic records of the relevant changes - Audit Trail
- System function for recording relevant user actions - electronic recording
- Recording of manipulated log data with checksum
- Audit Trail record for printing logged changes

Execution or changes to labeled objects are saved in a special log, the "AuditTrail". GMP-compliant configuration means, HMI devices have electronic production data documentation functionalities.

Scope of logging:

The following operations are Audit-relevant, and they are automatically saved in the Audit Trail:

- Runtime sequence
  - Runtime start and runtime stop.
  - Project information: Version and project name of the configuration environment, device and current runtime configuration.
  - Failure of the voltage supply of an active Uninterruptible Power Supply (UPS).
  - Change values of GMP-relevant tags by the user.
- User administration
  - Logon and logoff of users.
  - Invalid logon attempts.
  - Import of user administration.
  - Changes of user administration.

- for GMP-relevant recipes:
  - Storing after changing and creating recipe data records.
  - Transfer of recipe data records to the PLC and from the PLC.
  - For recipe tags: Changing the setting for the synchronization of the tag values with the PLC ("offline"/"online").
- NotifyUserAction
  - Use the system function "InsertElectronicRecord" to record user actions that are not automatically recorded by the audit trail.
  - You can configure this system function for screen object events.

Audit Viewer

You can find the Audit Viewer in the TIA Portal Toolbox.

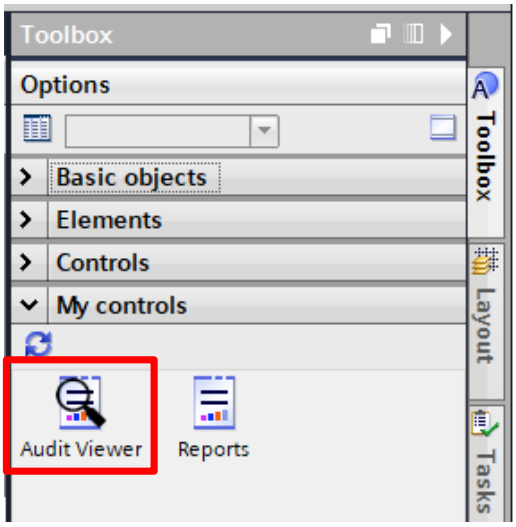


Figure 8-21 Toolbox – Audit Viewer.

The Audit Viewer is already implemented in the HMI under Messages (Main Navigation), Audit Viewer (Sub Navigation). To update the displayed content it is necessary to click the update button – there is no automatic update of the content.

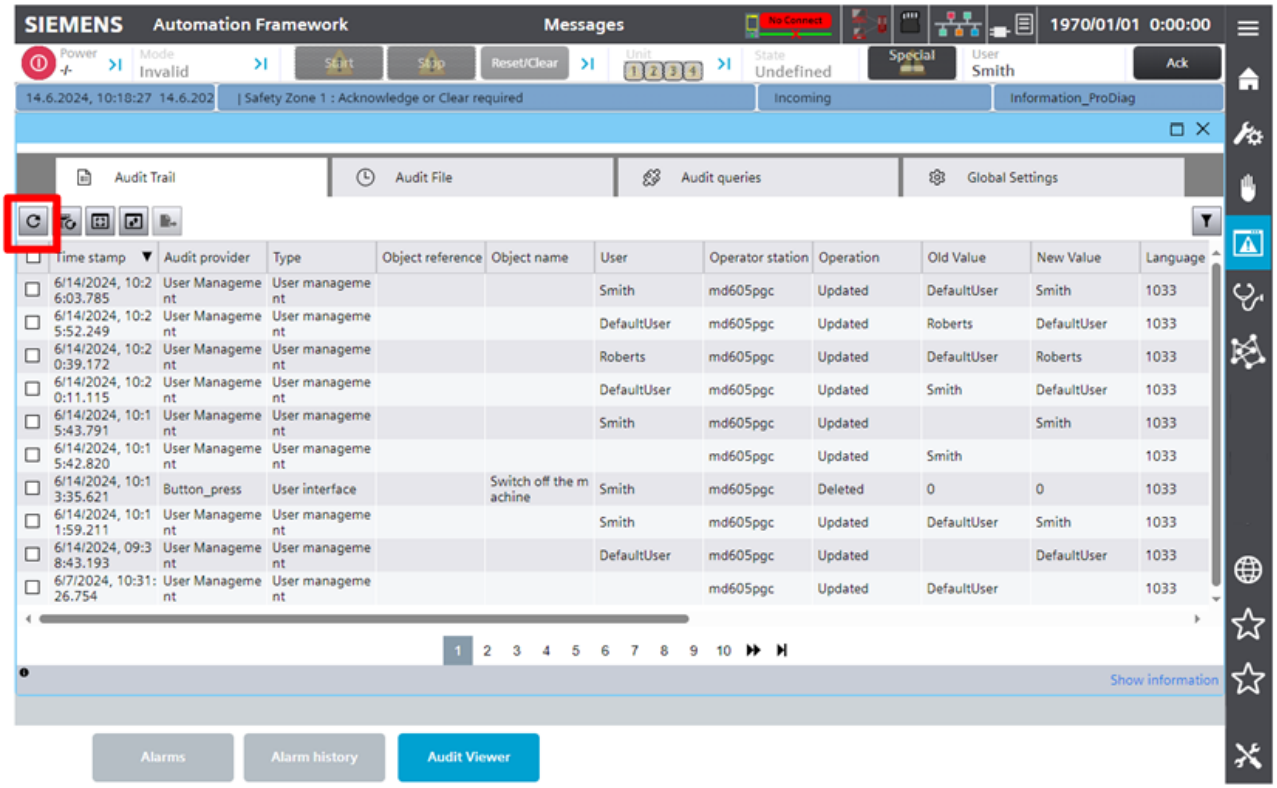


Figure 8-22 Audit Viewer.

Example implementation of "InsertElectronicRecord"

This system function is used to log user actions in the Audit Trail that are not automatically logged by GMP setting. You can also use this system function to require the user to enter an acknowledgment or an electronic signature and a comment for the operator action.

The "InsertElectronicRecord" function is implemented in Screens/00_ScreenLayout/PopUps/Template_PowerStatus with the PRESS event of the OFF button.

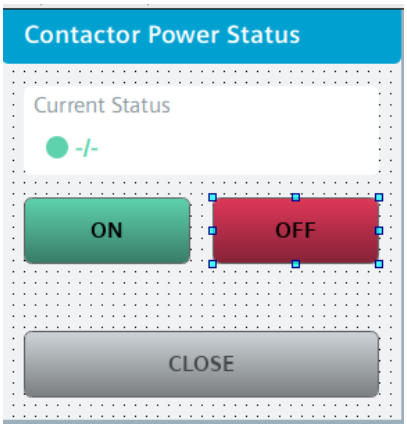


Figure 8-23: InsertElectronicRecord function.

For more details, please have a look into the Documentation for WinCC Unified (chapter 19.1):
https://support.industry.siemens.com/cs/ww/en/view/109828368.
In addition, also an application example "GMP-compliant Configuration with WinCC" is available:
https://support.industry.siemens.com/cs/ww/en/view/109744244.

8.6. Diagnostics

Various predefined, ready-to-use diagnostic screens, are being implemented in the AF. The following chapter introduces the different diagnostic implementations.

8.6.1. System diagnostics

The AF provides an integrated system diagnostic solution in the HMI, this allows the commissioning engineer, operator, maintenance, and service personal to get a fast diagnosis of several PLCs and the connected PROFINET IOSystems.

To show the "System diagnostic" screen navigate to "Diagnostics" (1) and "System diagnostics" (2) in the HMI. The WinCC Unified object "System diagnostics control" is used to show the diagnostic status of several PLCs using traffic light SVGs. The diagnostic status, contains the overall status of the relevant PLCs. The merged state is always the worst state of all PLCs. The "Diagnostics view" shows the diagnostic buffer of the selected PLC with the diagnostic event including event text and timestamp. Use the buttons (3) to update and select different diagnostic events as well as to show detailed information of the diagnostic event.

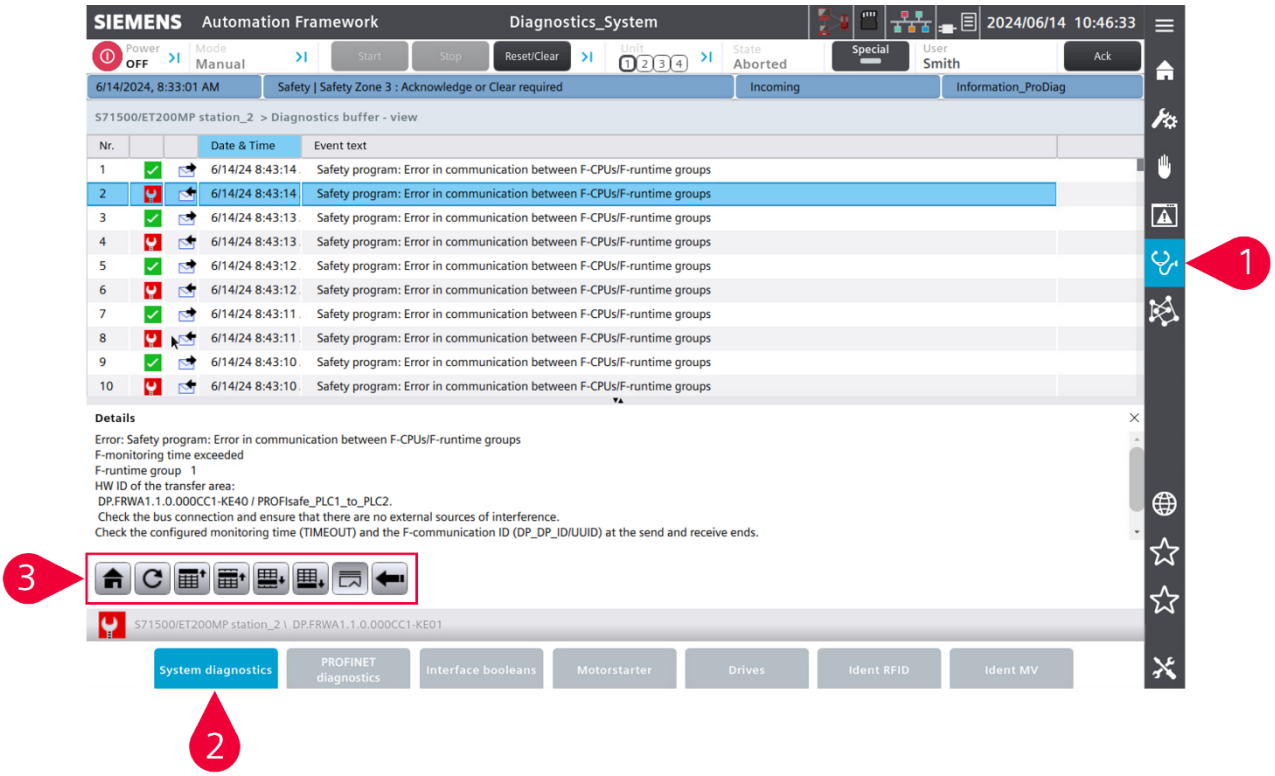


Figure 8-24: System diagnostics (Diagnostic view).

NOTE In terms of usability, it is recommended to navigate between several PLCs via the "Matrix view" of the "PROFINET diagnostics" (details see PROFINET diagnostics).

For detailed documentation regarding the WinCC control object "System diagnostics control", refer to the online documentation in TIA Portal or have a look in the WinCC Unified manual:
https://support.industry.siemens.com/cs/ww/en/view/109828368

8.6.2. PROFINET diagnostics

In addition, the AF provides a detailed status diagnostic information of the PLC and their lower-level hardware components of the PROFINET IOSystem in the HMI. To show the "PROFINET diagnostics" screen navigate to "Diagnostics" (1) and "PROFINET diagnostics" (2) in the HMI.

The functionality is implemented with the "Distributed IO view" of the WinCC Unified object "System diagnostics control". Each hardware component is displayed as a tile. Detailed information about the diagnostic status is retrieved by clicking on the relevant tile (3). Navigating through the individual hardware components is based on the network topology and is also possible by clicking on the tiles.

If only one PLC with a PROFINET IO-System is configured, the "PROFINET diagnostics" screen will show the "Distributed IO view" by default. Otherwise, the WinCC Unified object "System diagnostics control" changes to "Matrix view" during runtime. The "Matrix view" is also retrievable via the "Start page" button (4) which allows switching between several PLCs and PROFINET IO-Systems very user-friendly.

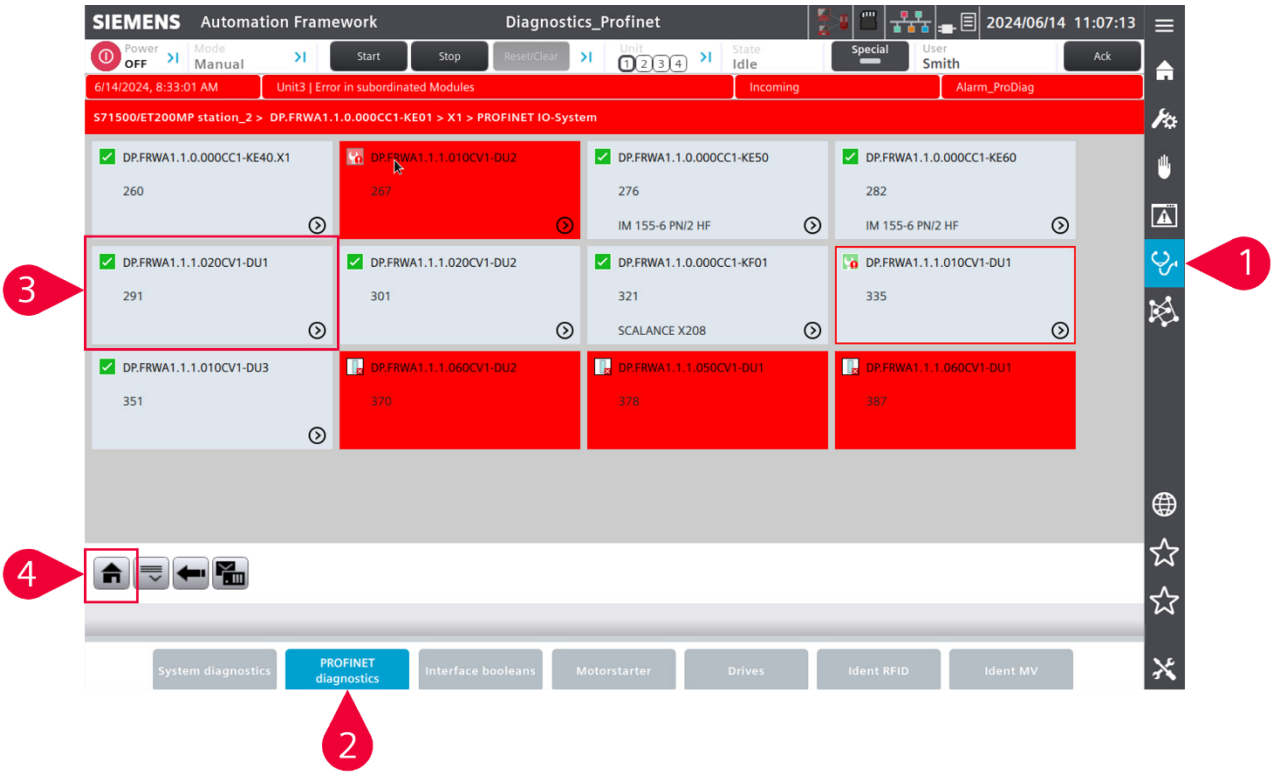


Figure 8-25: PROFINET diagnostics (Distributed IO view).

8.6.3. Interface Booleans

In the AF it is possible to monitor the status of various boolean process signals (e.g., such as a light barrier signal) for diagnostic purposes. These signals are selected in the engineering process, configured once in the PLC and then, during operation the selected signals can be displayed in the HMI.

For this, inside the "CallInterfaceSignals" Programm Blocks in the "05_Diagnostics" folder are the FBs "LSicar_InterfaceSignalsBool" where each monitoring signal must be connected. The FBs collect the signals for the HMI interface and they are displayed on the HMI.

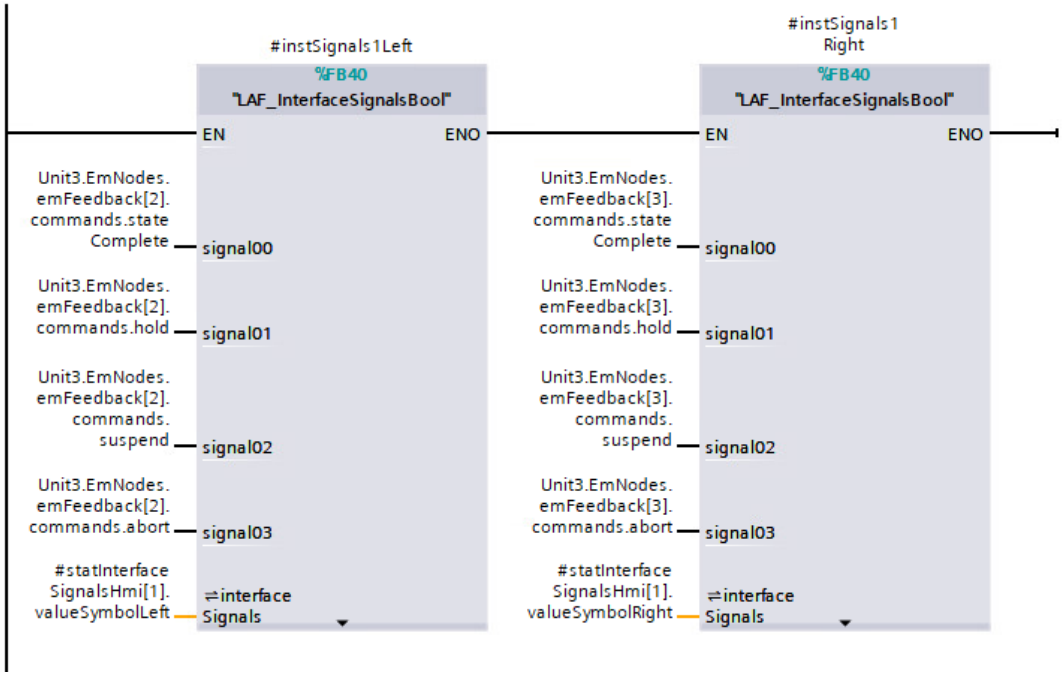


Figure 8-26: Interface Signals Call. Up to 16 signals can be transmitted by each block.

The displayed variables page is responsive to the selected page in the third navigation. There are 16 available pages with 32 variables in each page. These 32 variables are further divided into two sections (left and right section). In summary up to 512 Booleans can be configured for monitored.

A header text for each section can be defined using the pencil icon in the top left. This can be set via the HMI (but is then overwritten again after a PLC stop/start) or can be set with initial value in during engineering (remains after a PLC stop/start). The status of the Boolean is displayed via the icon where TRUE = green and FALSE = gray. In addition, the variable names are imported automatically from the "LAF_InterfaceSignalsBool" FBs by the HMI.

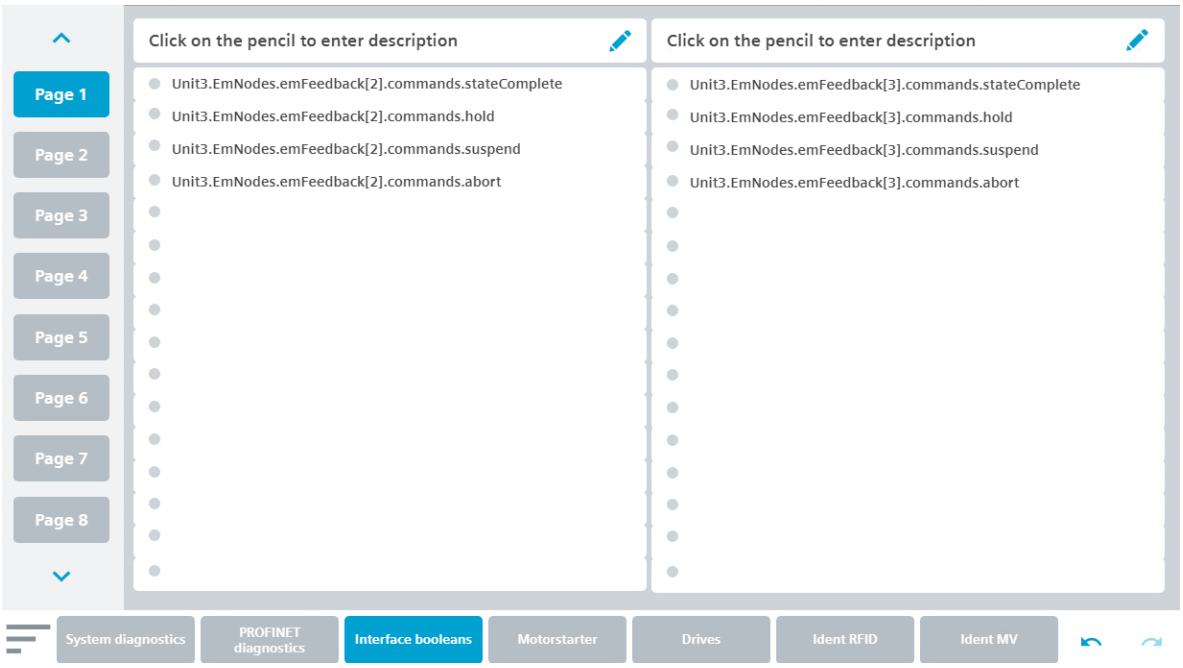


Figure 8-27: Diagnostic screen for the interface Booleans.

As an example, the feedback of the EMs "EM Hopper" & "EM HPM" to the Unit was defined for page 1 (Figure 8-27). In addition, the corresponding header is configured in the PLC code. The signals that are displayed on the HMI are to be defined in the "CallInterfaceSignals" block of the PLC. It can be found in the folder for the central functions as part of the diagnostics. The "CallInterfaceSignals" in turn is called in the "CallDiagnostics" block. In the "CallInterfaceSignals" block the "LAF_InterfaceSignalsBool" is called 32 times to read out the 512 signals.

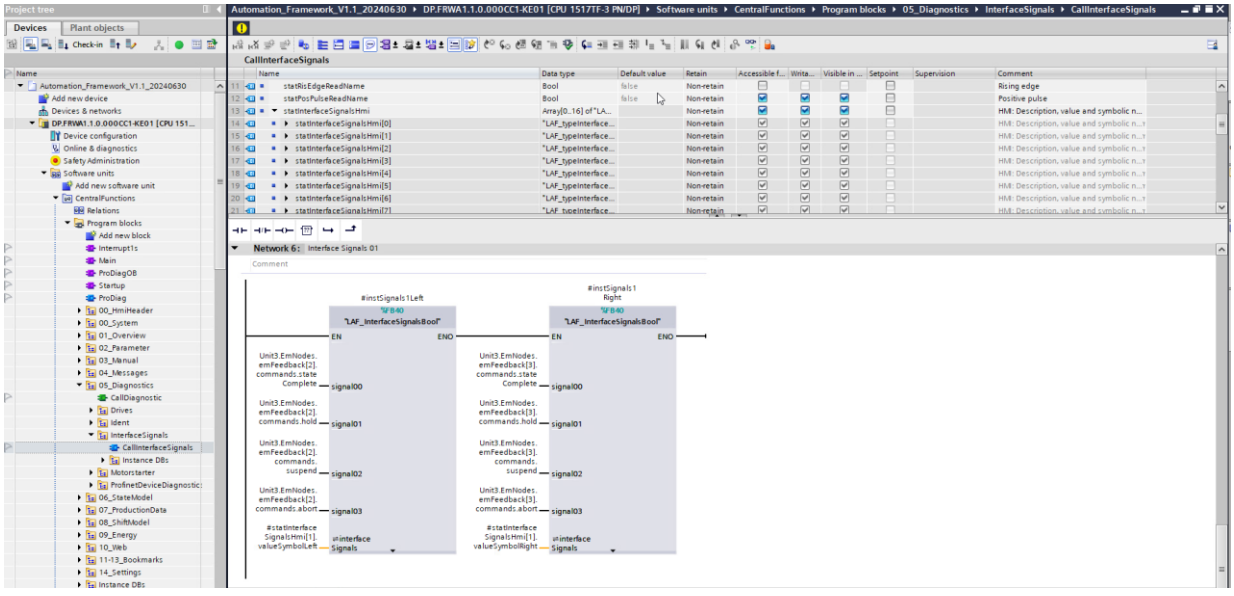
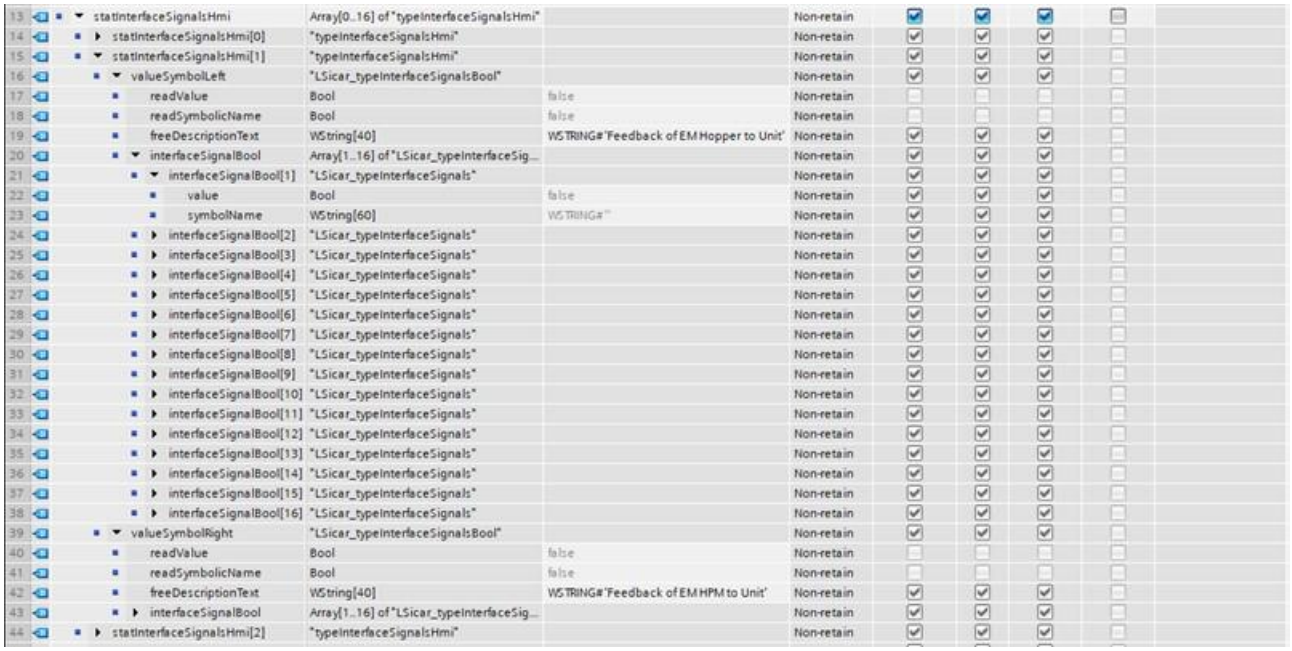


Figure 8-28: Call of the function blocks which can read out the value and name of the Boolean variables.

The structure of the function block is as follows: in networks 2 to 4, it is evaluated whether a page change has taken place on the HMI and which page of the HMI is currently loaded. This information is used to dynamically read the values and names of the variables.

The names of the variables are read only once when changing the page on the HMI. The values of the 32 variables on the current page are read continuously. However, only the values that are visible on the current page are being read. For this purpose, the interface of the FBs "interfaceSignals" is used. The types and FBs for reading the variables are sorted in the networks in the same way as the images on the HMI. Each page has its own network. In this network there are then two blocks – one for the left section of the monitoring display and one for the right section of the monitoring display. The data structure "statInterfaseSignalsHmi" is not only used for controlling the reading and writing of the variable names and values, but also to store the values and names as well as the head text. It is also used to exchange the data between HMI and PLC.



13	statInterfaceSignalsHmi	Array[0..16] of "typeInterfaceSignalsHmi"		Non-retain	☑	☑	☑	☐	
14	statInterfaceSignalsHmi[0]	"typeInterfaceSignalsHmi"		Non-retain	☑	☑	☑	☐	
15	statInterfaceSignalsHmi[1]	"typeInterfaceSignalsHmi"		Non-retain	☑	☑	☑	☐	
16	valueSymbolLeft	"LSicar_typeInterfaceSignalsBool"		Non-retain	☑	☑	☑	☐	
17	readValue	Bool	false	Non-retain	☐	☐	☐	☐	
18	readSymbolicName	Bool	false	Non-retain	☐	☐	☐	☐	
19	freeDescriptionText	WString[40]	WSTRING#'Feedback of EM Hopper to Unit'	Non-retain	☑	☑	☑	☐	
20	interfaceSignalBool	Array[1..16] of "LSicar_typeInterfaceSig...		Non-retain	☑	☑	☑	☐	
21	interfaceSignalBool[1]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
22	value	Bool	false	Non-retain	☑	☑	☑	☐	
23	symbolName	WString[60]	WSTRING#''	Non-retain	☑	☑	☑	☐	
24	interfaceSignalBool[2]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
25	interfaceSignalBool[3]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
26	interfaceSignalBool[4]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
27	interfaceSignalBool[5]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
28	interfaceSignalBool[6]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
29	interfaceSignalBool[7]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
30	interfaceSignalBool[8]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
31	interfaceSignalBool[9]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
32	interfaceSignalBool[10]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
33	interfaceSignalBool[11]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
34	interfaceSignalBool[12]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
35	interfaceSignalBool[13]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
36	interfaceSignalBool[14]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
37	interfaceSignalBool[15]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
38	interfaceSignalBool[16]	"LSicar_typeInterfaceSignals"		Non-retain	☑	☑	☑	☐	
39	valueSymbolRight	"LSicar_typeInterfaceSignalsBool"		Non-retain	☑	☑	☑	☐	
40	readValue	Bool	false	Non-retain	☐	☐	☐	☐	
41	readSymbolicName	Bool	false	Non-retain	☐	☐	☐	☐	
42	freeDescriptionText	WString[40]	WSTRING#'Feedback of EM HPM to Unit'	Non-retain	☑	☑	☑	☐	
43	interfaceSignalBool	Array[1..16] of "LSicar_typeInterfaceSig...		Non-retain	☑	☑	☑	☐	
44	statInterfaceSignalsHmi[2]	"typeInterfaceSignalsHmi"		Non-retain	☑	☑	☑	☐	

Figure 8-29: HMI interface for the interface Booleans.

The structure shown for the HMI (statInterfaceSignalHmi) is an array with 16 elements. Each element contains two elements of the data type "LAF_typeInterfaceSignalsBool". One for the left section of the page on the HMI and one for the right section. The individual elements are described by the blocks and the name and value are read out by the HMI. No additional engineering is required in the HMI, unless more than the configured tags are to be read, in which case the structure in the HMI must be adapted accordingly. Here you can see the variables for controlling the reading process of the names and values (readValue & readSymbolName), as well as the header (freeDescriptionText) and the array for the names and values themselves. Since the names of the tags are read out automatically, only the heading has to be set here.

More information about the Interface Booleans can be found in the following documentation:

Reference to "LAF_InterfaceSignalsBool"

8.6.4. Drive Diagnostics

The provided blocks and screens allow the commissioning engineer, operator, maintenance, and service personal a fast diagnosis of technology objects (Axis) and SINAMICS drives in the HMI. The drive diagnostics includes status and control information of TOs and SINAMICS drives, as well as messages for faults and warnings of the SINAMICS. Furthermore, diagnostic information about the positioning status of the basic positioner (EPOS) functionality for SINAMICS drives are available.

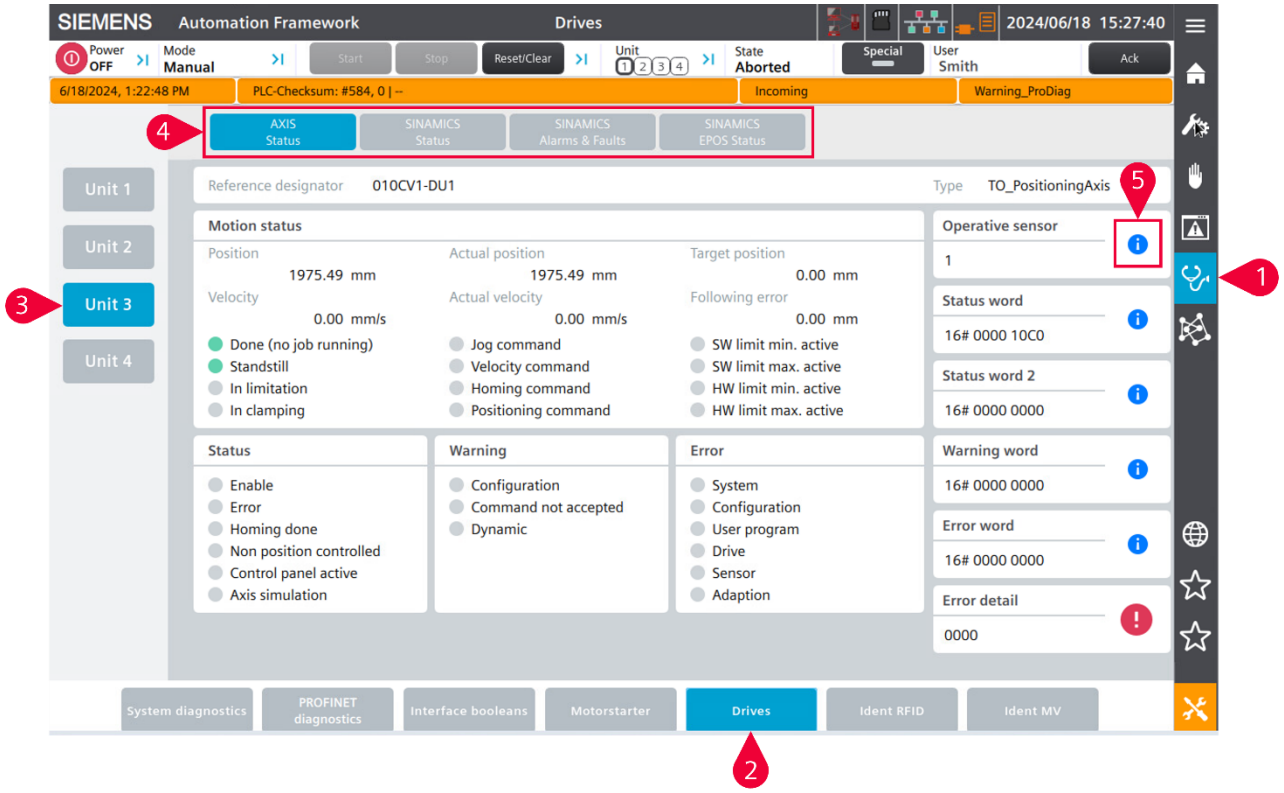


Figure 8-30: HMI navigation and drive diagnostics screen layout.

To show the drive diagnostics screens navigate to "Diagnostics" (1) and "Drives" (2). Then select the relevant drive via the third navigation (3). With the navigation bar (4) the essential information for each drive diagnostics are displayed in the main screen. Detailed status information is retrievable via pop-ups (5).

8.6.4.1. Overview

The drive diagnostics consists of three main parts. Data blocks are used to store configuration data and as interface to the HMI. Function blocks read out the diagnostic data and edit it to a visualization format. Screens show the diagnostic data in the HMI in a clear and structured layout. Each HMI screen is linked to a function block in the PLC. The following table provides an overview of all blocks and screens:

Block	Functionality	Screen
DB "DriveDiagConfig"	Configuration data array of all drive objects for drive diagnostics.	-
DB "DriveDiagInterfaceHmi"	HMI interface for drive diagnostics.	-

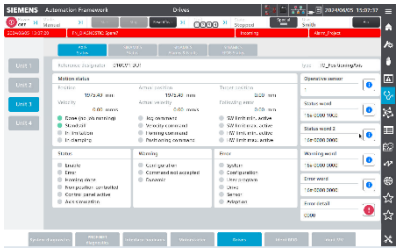
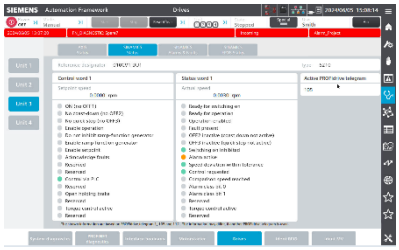
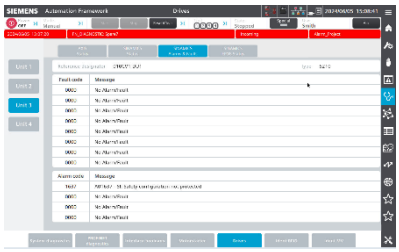
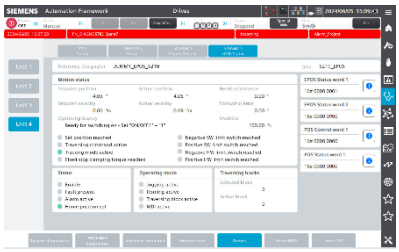
Block	Functionality	Screen
FC "LAF_SetDriveObjectConfig" (optional)	This block sets the configuration data of a drive object in the configuration data array depending on the parameterized index and input values during PLC startup.	-
FB "LAF_GetAxisStatus"	This block reads diagnostic information of an axis depending on the configured technology object and the selected drive object index in the HMI by the operator.	"AxisStatus" 
FB "LAF_GetSinaStatus"	This block reads PROFIdrive telegram status word 1 (ZSW1) and control word 1 (STW1) as well as setpoint and actual speed values from a SINAMICS drive via acyclic communication depending on the configured type of SINAMICS and the selected drive object index in the HMI by the operator.	"SinamicsStatus" 
FB "LAF_GetSinaAlarms"	This block reads faults and alarms of a SINAMICS drive via acyclic communication depending on the configured type of SINAMICS and the selected drive object index in the HMI by the operator.	"SinamicsAlarms" 
FB "LAF_GetSinaEpos"	This block reads EPOS positioning status information as well as setpoint and actual values from a SINAMICS drive via acyclic communication depending on the configured type of SINAMICS and the selected drive object index in the HMI by the operator.	"SinamicsEpos" 

Table 8-23: Drive diagnostics blocks and screens

8.6.4.2. Architecture

The architecture is explained using the "SinamicsEpos" screen and "LAF_GetSinaEpos" function block as an example.

The operator selects the relevant drive via the third navigation in the HMI. By doing this, the "DriveObjectIndex" is set. The "DriveObjectIndex" and the active "SinamicsEpos" screen information (1) is transferred to the PLC via the "DriveDiagInterfaceHMI" data block (2). The "LAF_GetSinaEpos" function block (3) is enabled by the active "SinamicsEpos" screen. This function block reads the configuration from the "DriveDiagConfig" data block (4) depending on the "DriveObjectIndex". A part of the configuration data is transferred directly to the HMI. The other part is used to establish an acyclic communication to read parameters from a SINAMICS drive. The received parameters are formatted and then transferred to the HMI (5) too.

The "LAF_SetDriveObjectConfig" function (A) sets the configuration for one drive diagnostic object in the "DriveDiagConfig" data block during PLC startup.

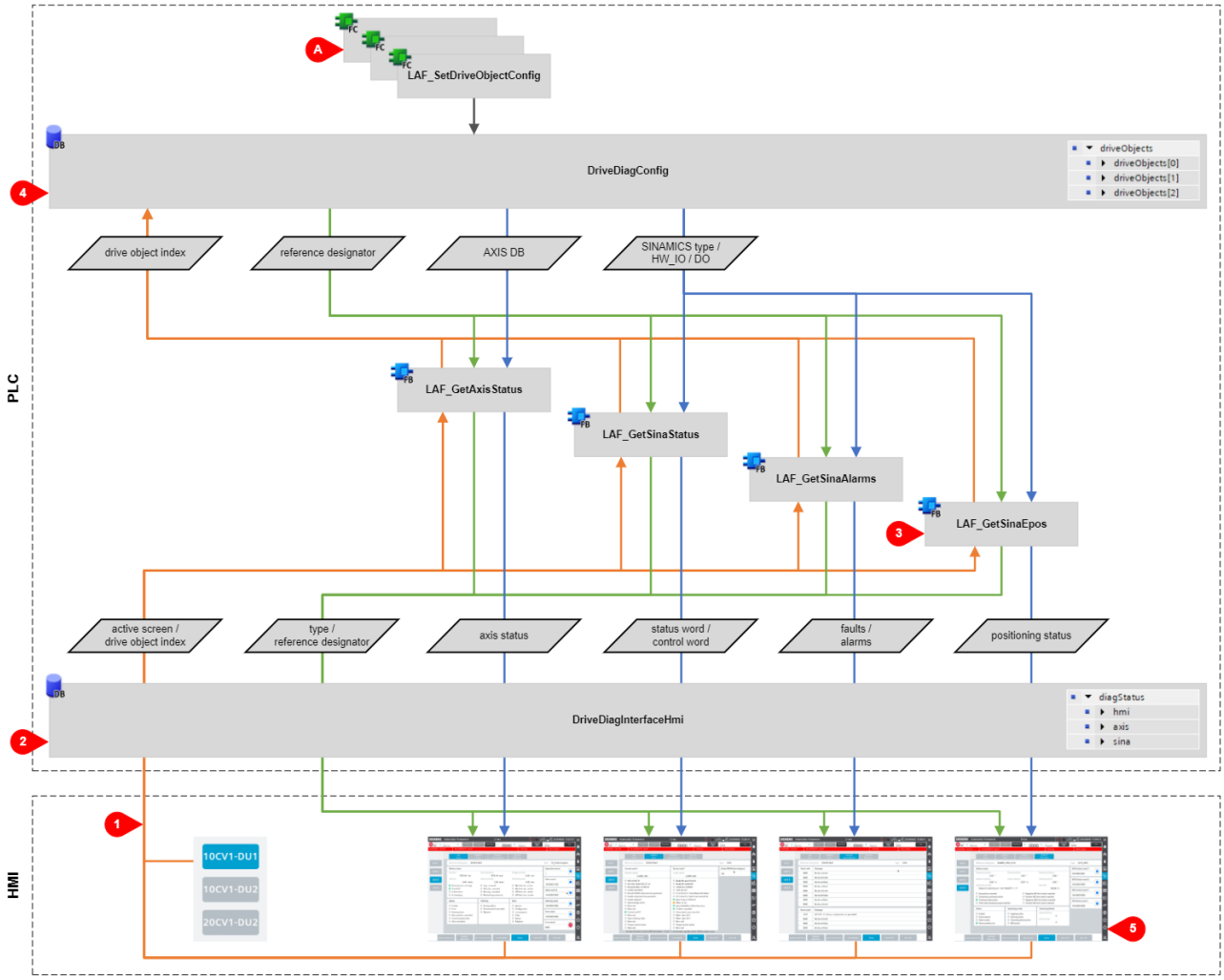


Figure 8-31: Big picture of the drive diagnostics architecture.

Configuration data

To make it as user friendly as possible for the application engineer, all drive diagnostics objects are configurated in one place. This configuration is stored in the "DriveDiagConfig". Each drive diagnostics object has its own array element.

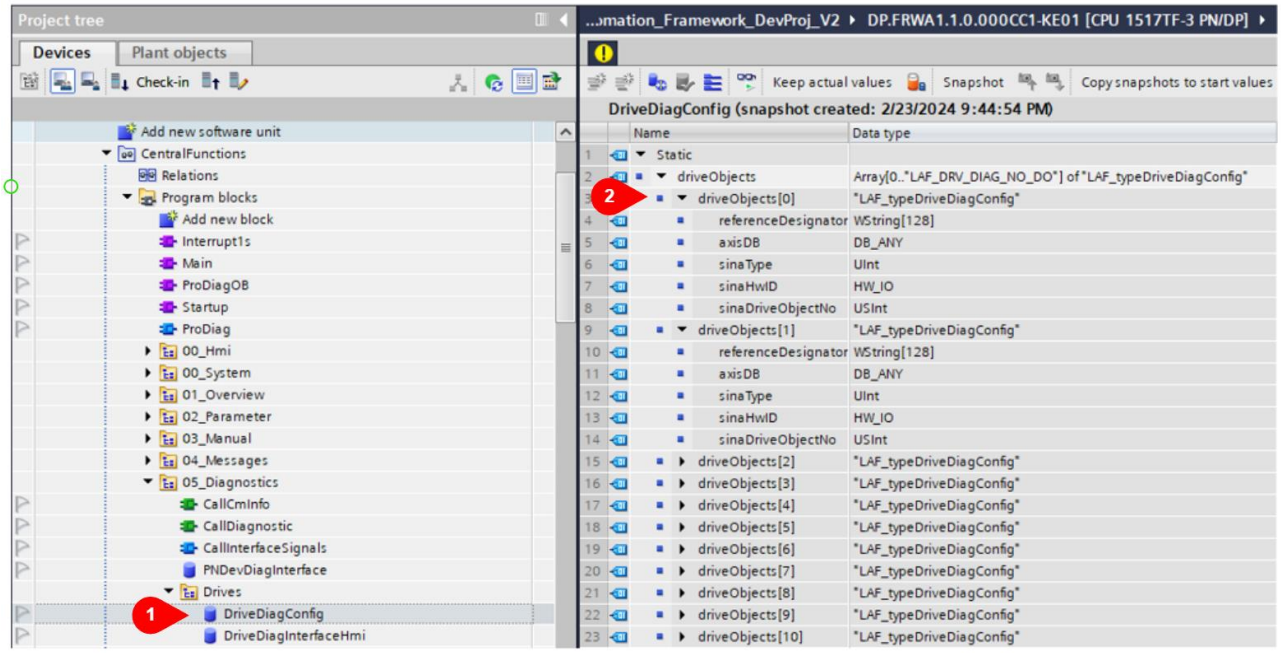


Figure 8-32: Structure of the configuration data block "DriveDiagConfig".

NOTE To reduce system load, it is recommended to set the configuration of each drive diagnostic object with the "LAF_SetDriveObjectConfig" function during PLC startup.

Each drive diagnostics object consists of the following tags:

Tag	Description
referenceDesignator	Name or reference designator of the drive diagnostic object that is shown in HMI.
axisDB	DB number of technology object. If no TO is used, set "axisDB = 0. The following types of TOs are supported: • TO_SpeedAxis • TO_PositioningAxis • TO_SynchronousAxis • TO_ExternalEncoder
sinaType	Type of SINAMICS drive. The type is required to read the correct parameters from the SINAMICS. The following types of SINAMICS are supported: • 000 = NO TYPE only TO diagnostic information • (mandatory for TO_ExternalEncoder, optional for other TOs) • 114 = SINAMICS G120 with TO or SinaSpeed • 124 = SINAMICS S120 with TO • 125 = SINAMICS S120 with EPOS • 220 = SINAMICS S200 with TO • 221 = SINAMICS S200 with EPOS • 222 = SINAMICS S210 with TO • 223 = SINAMICS S210 with EPOS

Tag	Description
sinaHwID	Hardware identification of the PROFINET IO device or PROFIBUS DP slave to address the SINAMICS drive. In this case, the hardware identification refers to the PROFIdrive telegram.
sinaDriveObjectNo	A PROFIdrive device consists of one or more functional objects according to the number of axes. Each of these objects represents the functionality of an axis and is referred to a drive object (DO). The drive object number for SINAMICS drives is as follows: • SINAMICS G120: single-axis, "sinaDriveObjectNo" = 1 • SINAMICS S120: multi-axis, see guide below • SINAMICS S200: single-axis, "sinaDriveObjectNo" = 1 • SINAMICS S210: single-aixs, "sinaDriveObjectNo" = 1

Table 8-45: Configuration tags for one drive diagnostics object

To access the drive object number, open the "Device configuration" (1) of the SINAMICS. Then the drive object number is shown in the "Device overview" (2) in collum "Drive object number" (3). This way works for every SINAMICS drive.

In addition, the drive object number of a SINAMICS S120 multi-axis system is accessible via the PROFINET interface (4) of the control unit (CU). Therefore, select the "Telegram configuration" in the properties (5). Then the drive object number is displayed in the "Item" collum (6) too.

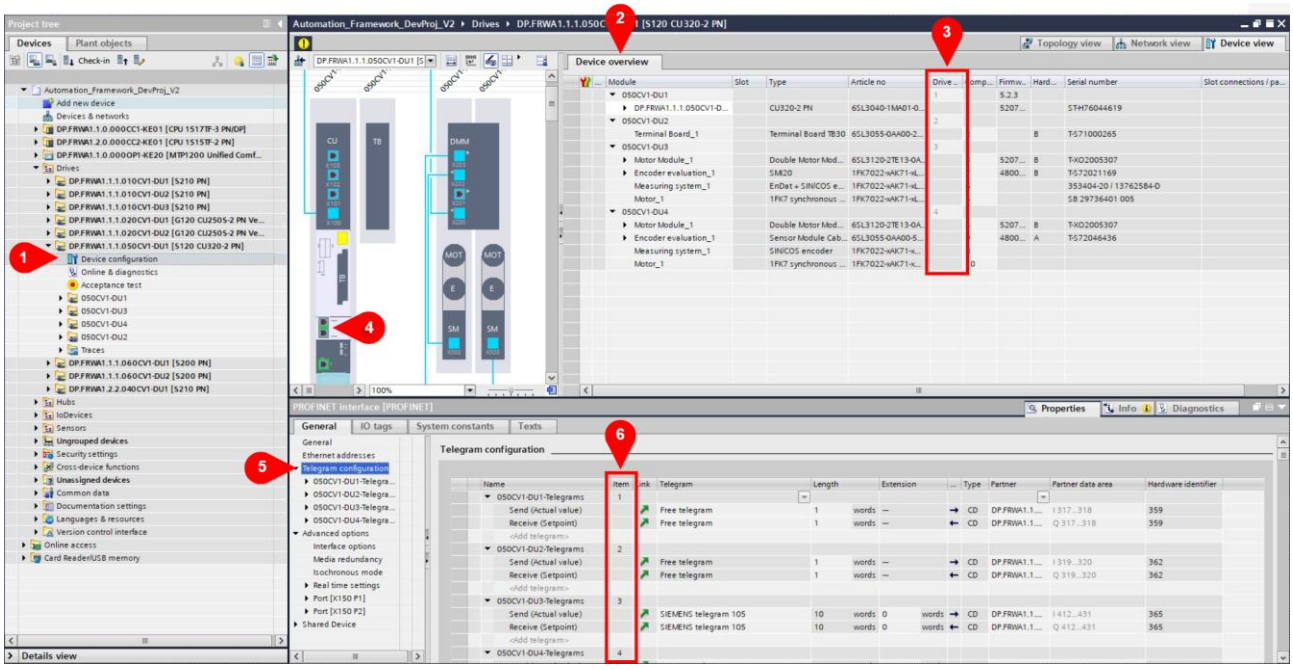


Figure 8-33: Access drive object number in SINAMICS S120 multi-axis system.

HMI interface

The interface between PLC and HMI is the "DriveDiagInterfaceHmi" data block (1). This block consists of three parts:

- HMI control tags as well as shared diagnostic data (2). These tags are used by all drive diagnostics blocks.
- Axis status information of the technology object (3) from the "LAF_GetAxisStatus" function block.
- Sinamics status information (4), which are filled by the "LAF_GetSinaStatus", "LAF_GetSinaAlarms" and "LAF_GetSinaEpos" function blocks.

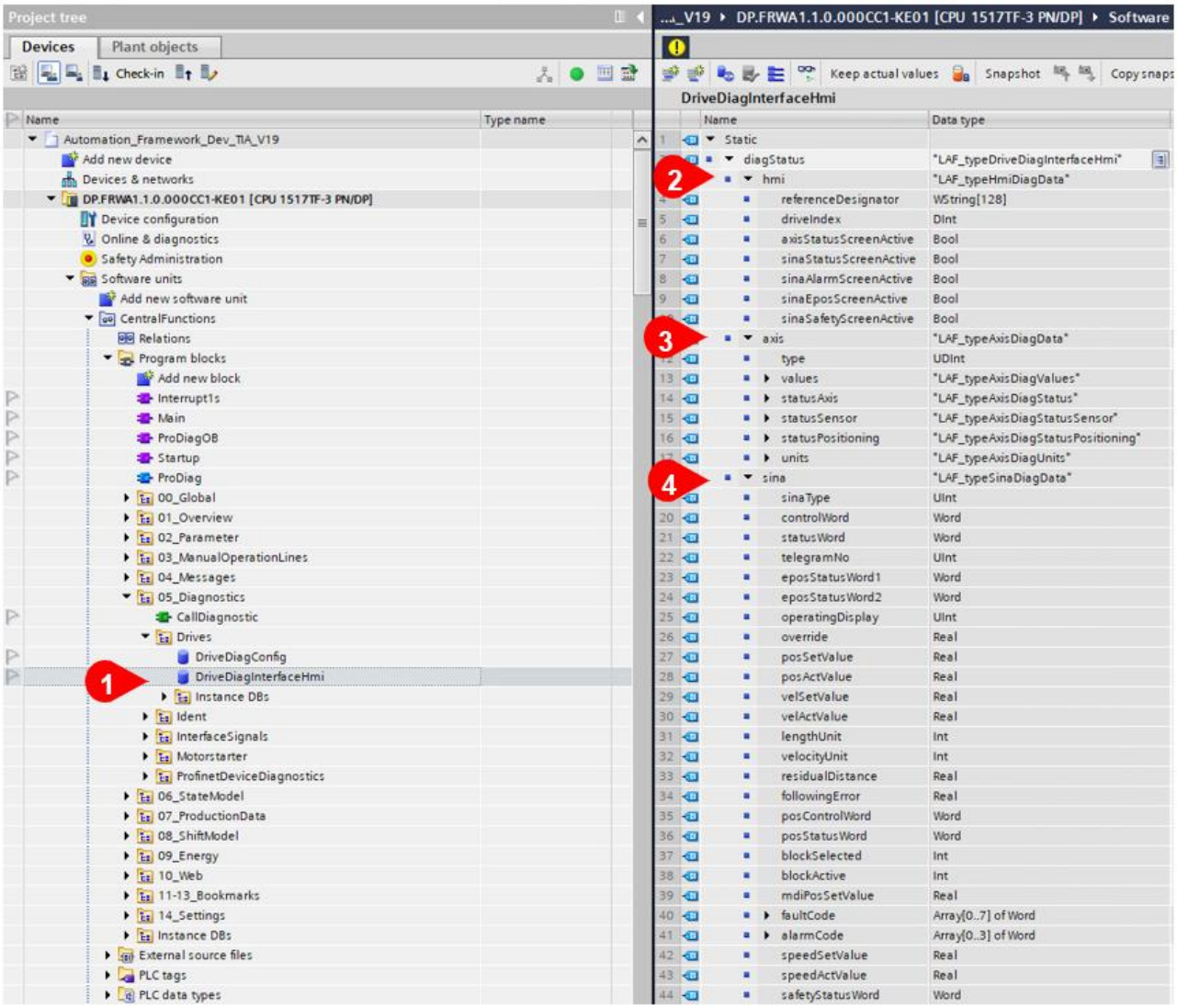


Figure 8-34: Structure of HMI interface data block "DriveDiagInterfaceHmi".

NOTE The "DriveDiagInterfaceHmi" data block as well as the connection to the HMI tags and screens objects are implemented in the preconfigured TIA Portal project of the AF.

PLC function blocks

The PLC functionality is implemented with the four function blocks "LAF_GetAxisStatus", "LAF_GetSinaStatus", "LAF_GetSinaAlarms" and "LAF_GetSinaEpos" which are called in the "CallCmInfo" function (1) and share the two data blocks "DriveDiagConfig" and "DriveDiagInterfaceHmi" (2).

To reduce system load, each function block in the PLC is enabled when the corresponding drive diagnostics screen in the HMI is displayed. It is possible to disable the drive diagnostics function blocks from the user program by assigning a PLC tag with value "FALSE" to the "enable" input (3). While this tag is "FALSE" no drive diagnostic data is read and transferred to the HMI.

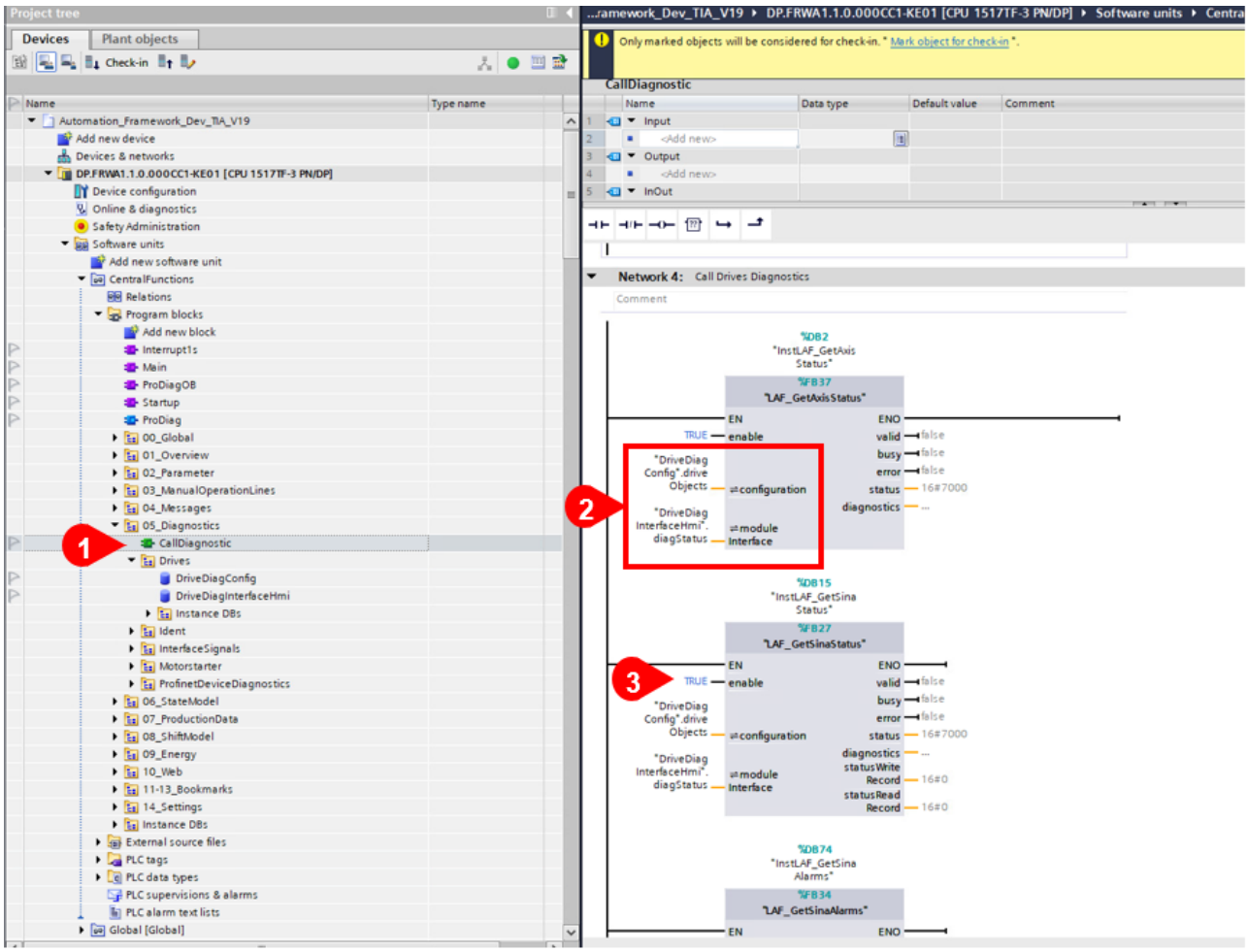


Figure 8-35: Function block calls for drive diagnostics.

NOTE The drive diagnostics function blocks are called in the "CallCmInfo" function and connected to the "DriveDiagConfig" and "DriveDiagInterfaceHmi" data blocks in the preconfigured TIA Portal project of the AF.

8.6.4.3. Engineering

During the engineering phase a few steps are must to be done to set up the drive diagnostic in the PLC and in the HMI. In order to make it as easy as possible for the engineer, the tasks are reduced to set the configuration in the "DriveDiagConfig" data block in the PLC program and to extend the third navigation in the HMI.

PLC engineering

First navigate to "LibraryObjects" tag table (1). Select the "User constants" (2) and set the value of the "LAF_DRV_DIAG_NO_DO" constant (3) to the number of technology objects and SINAMICS drives (= drive diagnostic objects) in the TIA Portal project. Then compile the "DriveDiagConfig" data block.

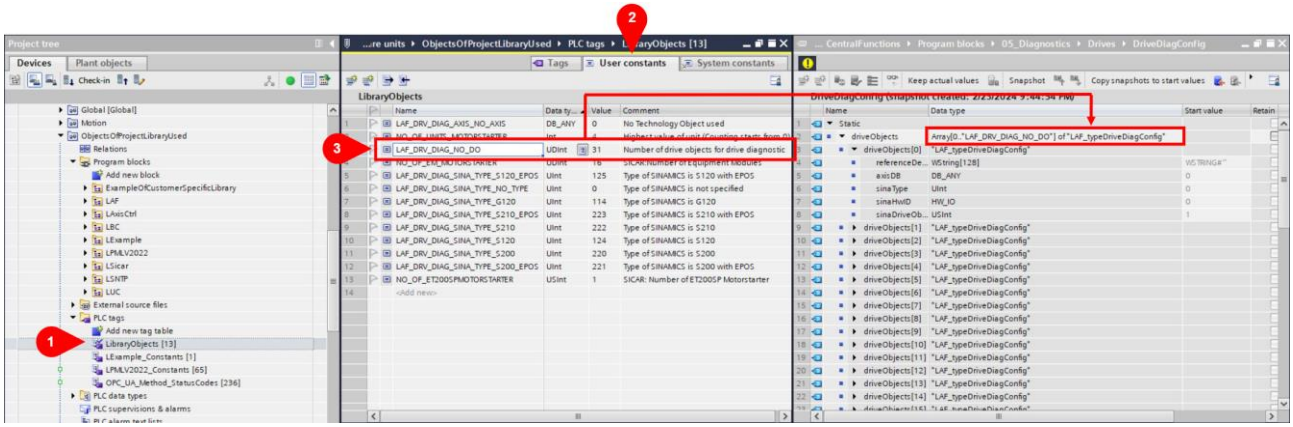


Figure 8-36: Set number of drive objects for drive diagnostics.

Example:

- 1x SINAMICS G120 with SinaSpeed
- 2x SINAMICS S210 with TO_PositioningAxis and TO_SynchronousAxis
- 1x SINAMICS S210 with EPOS

  ⇨ "LAF_DRV_DIAG_NO_DO" = 3

In this example "LAF_DRV_DIAG_NO_DO" means the array contains 4 elements "[0..3]".

Next create relations (1) between the Software Unit "CentralFunctions" and the technology objects (2).

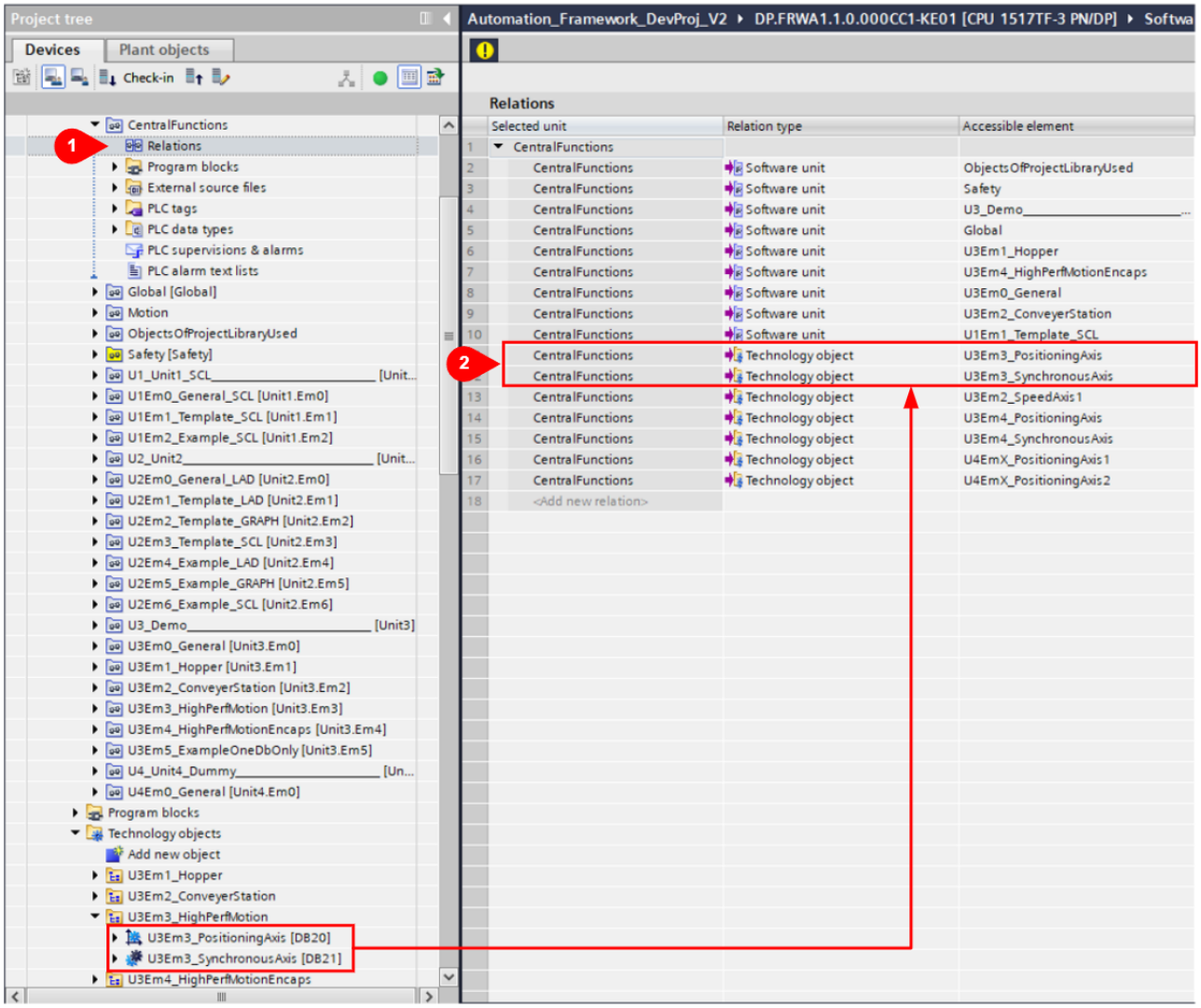


Figure 8-37: Creation of relations to technology objects.

Open OB "Startup" (1). Call the "LAF_SetDriveObjectConfig" function (2) for each array element of the "DriveDiagConfig" (3) and assign the array index (3) as well as the configuration array itself (4).

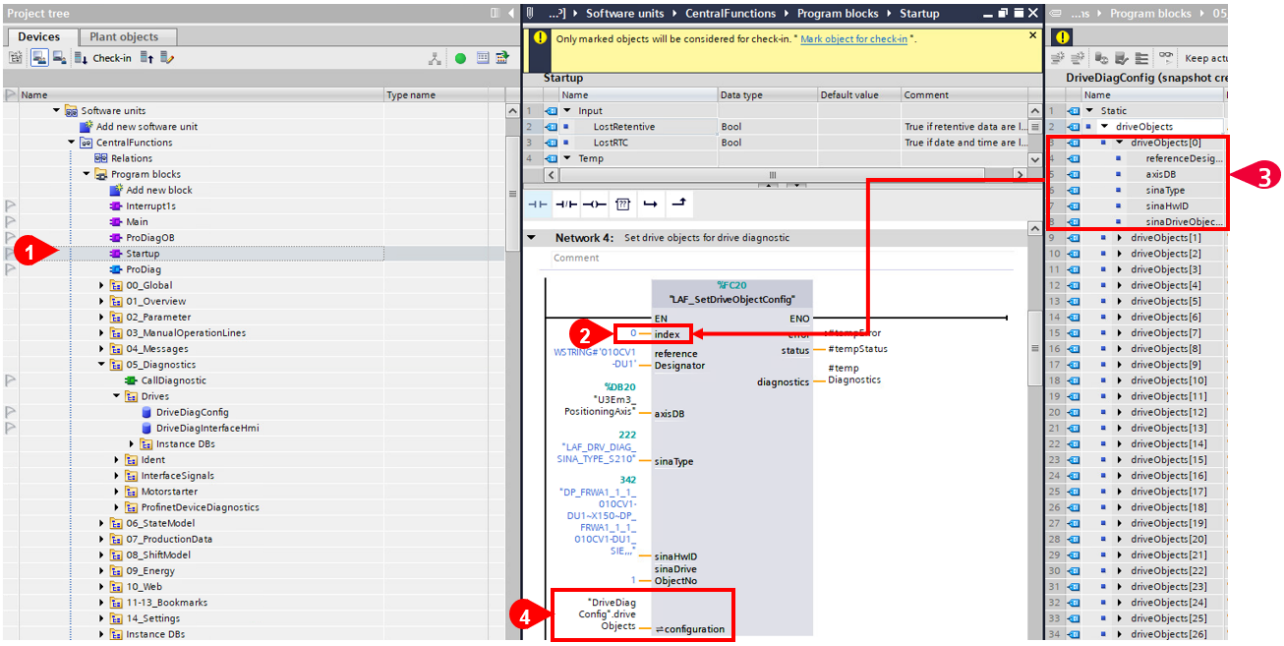


Figure 8-38: Call of "LAF_SetDriveObjectConfig" function in startup.

Next assign the reference designator of the drive (1) and the DB of the technology object (2) via drag and drop as well as the type of SINAMCS (3). For the type of SINAMICS predefined constants in the "LibraryObjects" tag table are available.

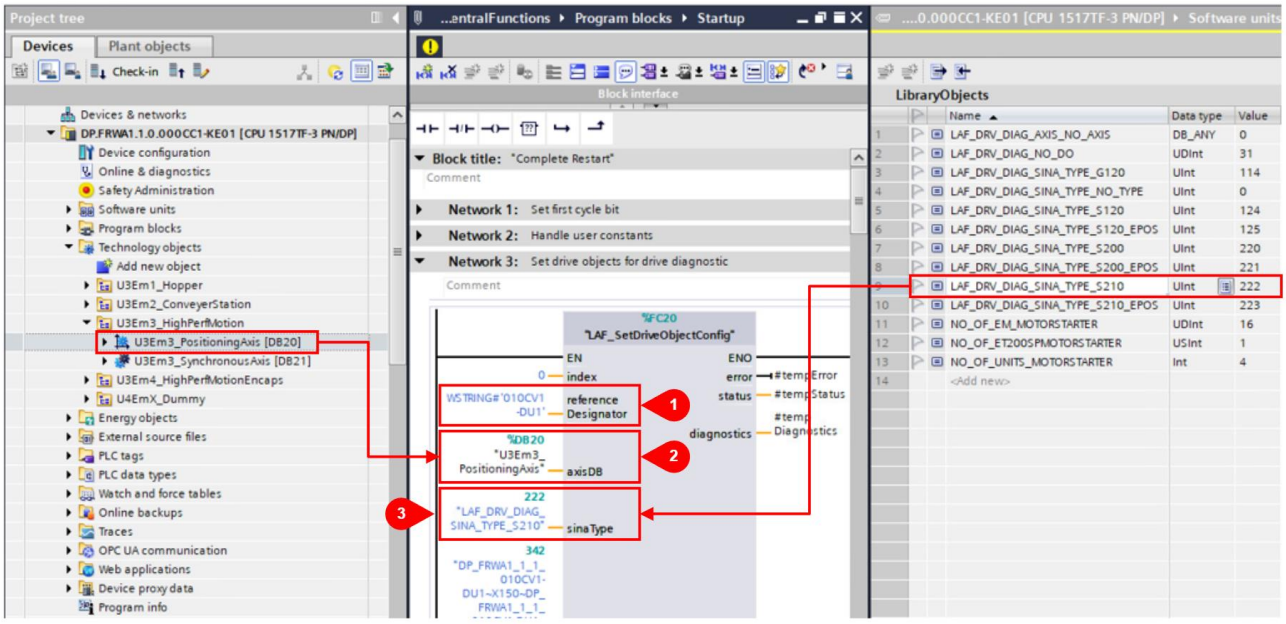


Figure 8-39: Assign reference designator, TO and type of SINAMICS to "LAF_SetDriveObjectConfig" function.

Then open the "Device configuration" (1) of the corresponding SINAMICS and select the PROFINET interface (2) and navigate to the "Telegram configuration" in the properties and select the "System constants" tab (3). Assign the system constant of the PROFIdrive telegram (4) to the "LAF_SetDriveObjectConfig" function via drag and drop.

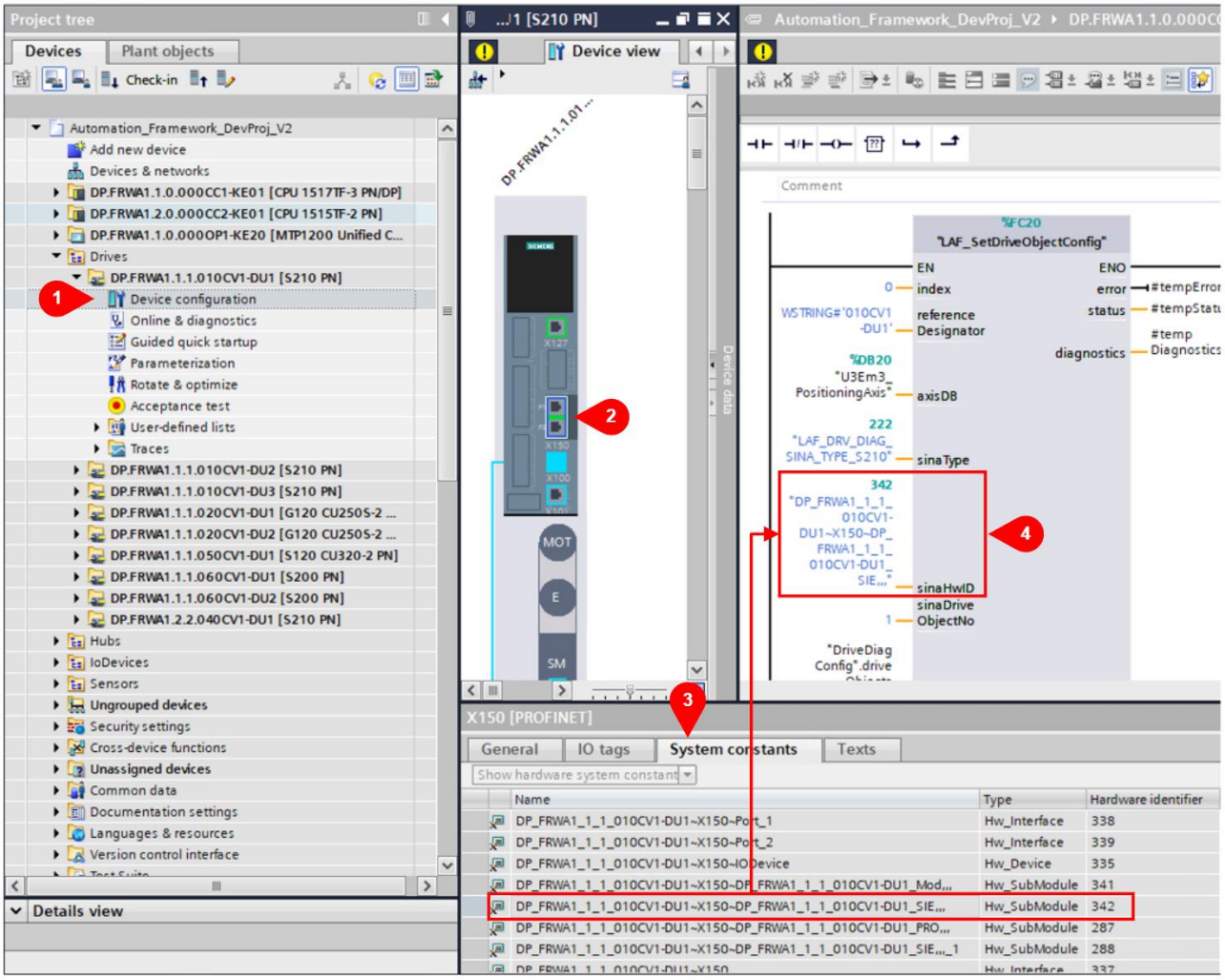


Figure 8-40: Assign system constant of the PROFIdrive telegram to "LAF_SetDriveObjectConfig" function.

NOTE The system constant must match the hardware identifier of the PROFIdrive telegram. Double check the hardware identifier in the "Telegram configuration" of SINAMICS drive.

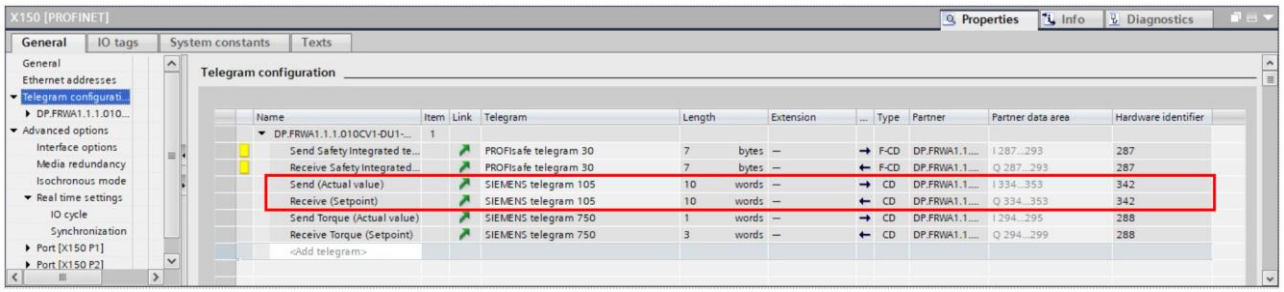


Figure 8-41: Access to the hardware identifier of the PROFIdrive telegram.

Finally assign the drive object number to the "LAF_SetDriveObjectConfig" function. Therefore, open the "Device configuration" (1) again and check the drive object number in the table of "Device overview" (2).

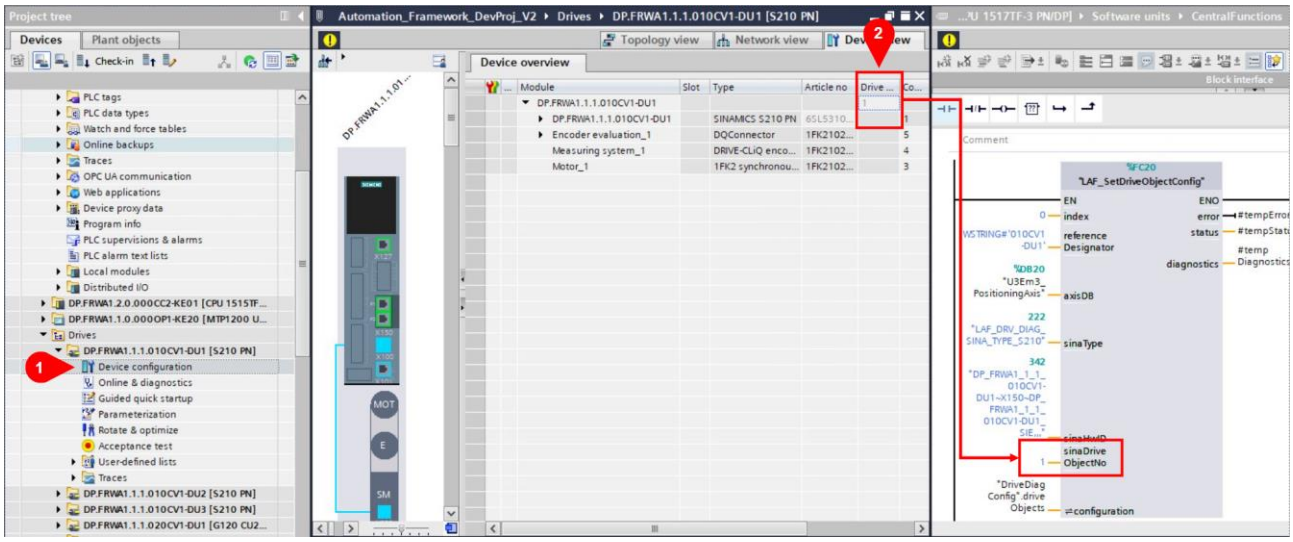


Figure 8-42: Assign drive object number to "LAF_SetDriveObjectConfig" function.

HMI engineering

The HMI screens for the drive diagnostics are preconfigured and dynamized depending on the configured technology object and SINAMICS drive. Therefore, the focus of the HMI engineering is reduced on expending the third navigation. Via the third navigation, the operator selects the relevant drive diagnostic object.

NOTE The third navigation consist of three layers for unit level, EM level and CM level. This chapter describes the tasks that are required to set up the drive diagnostics on the control module level. How to implement the navigation in general is explained in chapter Navigation.

For a faster HMI engineering workflow, the AF offers a preconfigured template screen for the control module level (1) with up to eight drives (2).

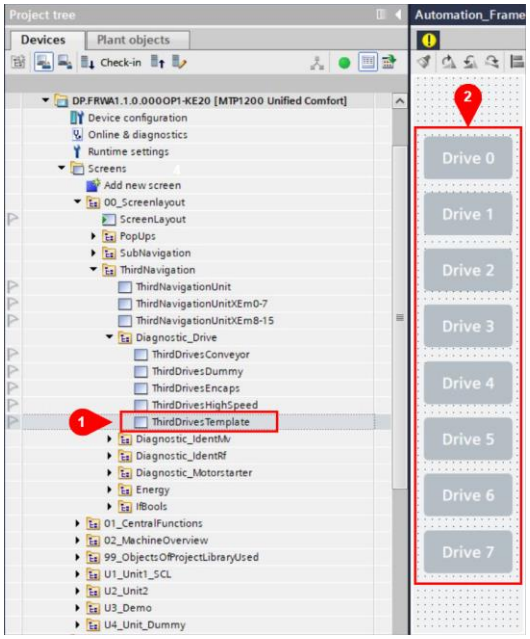


Figure 8-43: Template screen for third navigation of drive diagnostics.

Copy the template screen and rename it according to the name of the corresponding EM in the PLC. Then open the screen (1) and adjust the number of text boxes as well as the names or reference designator for the drive control modules (2).

Next set the value of the PLC tag "DriveObjectIndex" (3) from the "DriveDiagInterfaceHmi" data block to the relevant index of the array element (4) in the "DriveDiagConfig" data block which should be diagnosed. Also call the scripts "Navigation.HideThirdNavigationEM" and "Navigation.HideThirdNavigationCM" to hide the third navigation (5).

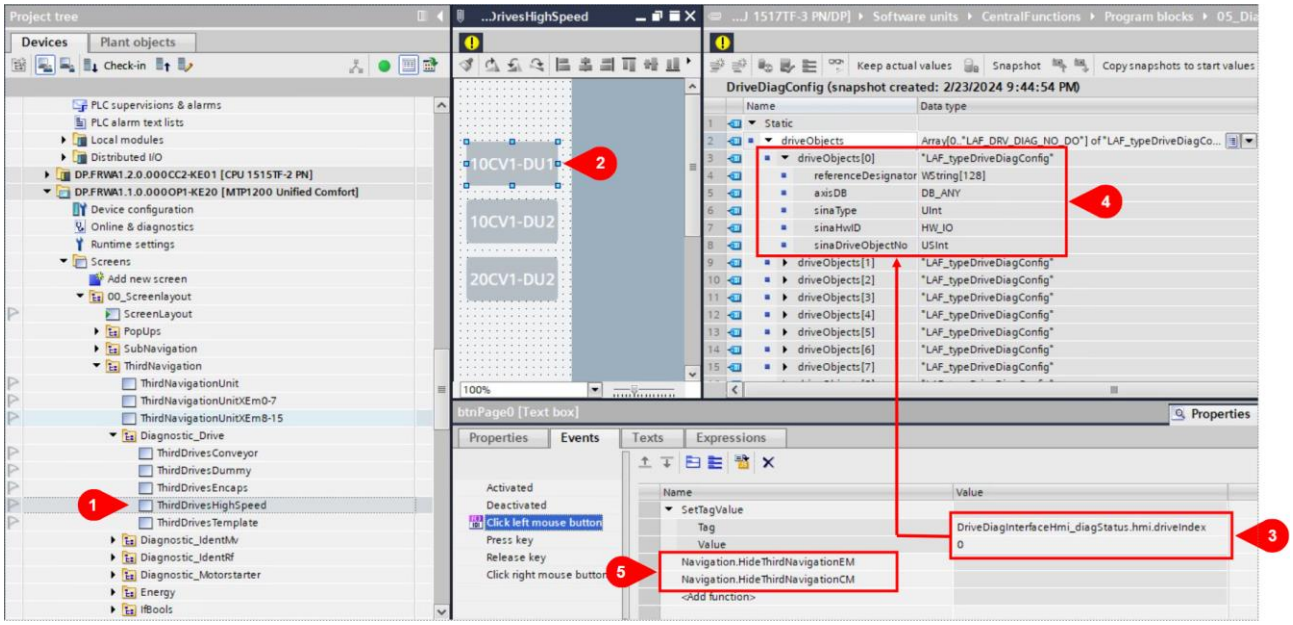


Figure 8-44: Events for drive diagnostics in third navigation.

NOTE The value of "DriveObjectIndex" must match the index of the array element in the "DriveDiagConfig" data block which should be diagnosed. Double check the value with the relevant array element.

Last navigate to "Properties" (1) and set the condition for the background color dynamization (2) to the same value as in the event for the "DriveObjectIndex" tag (3).

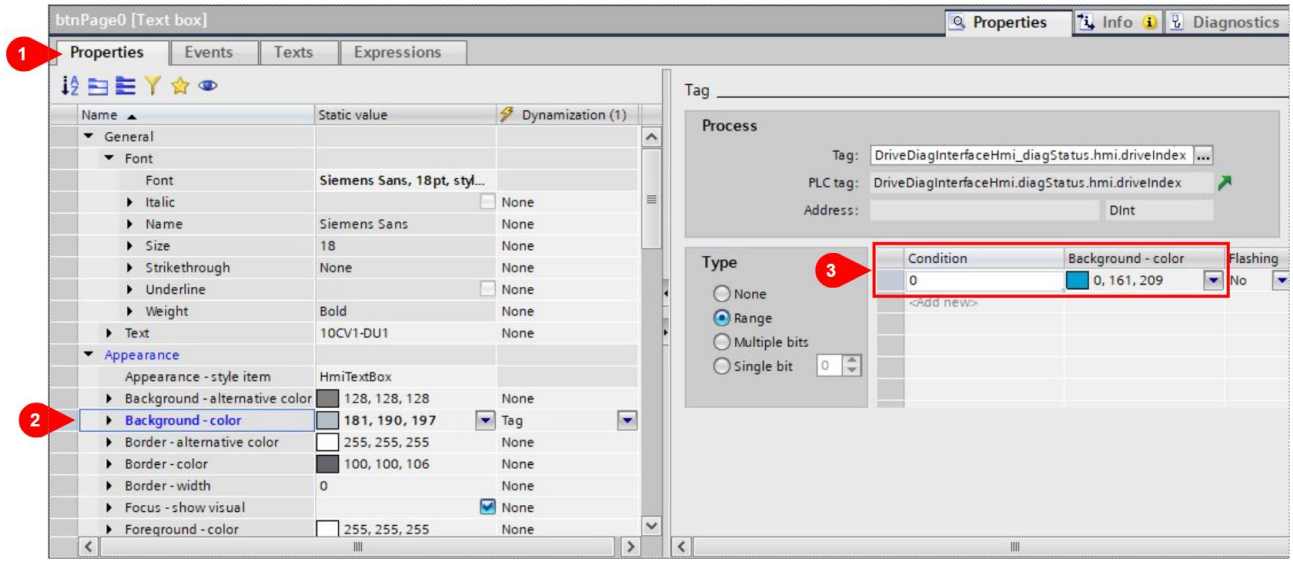


Figure 8-45: Dynamization for drive diagnostics in third navigation.

8.7. StateModel

In the HMI, this section contains screens to visualize the state model and the mode state history. More information can be found in the chapter about units as well as in the documentation of the LUC ("Library of Unit Control").

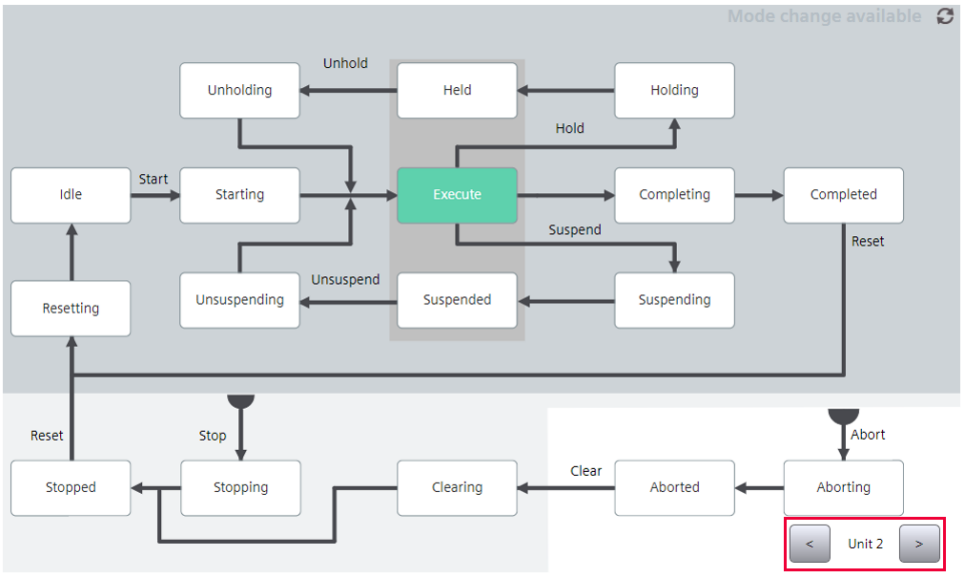


Figure 8-46: State model screen

In the lower right corner of the screen, the displayed unit can be changed.

8.8. Webserver

A defined webpage can be shown for example the embedded webserver of S7-1500 PLC (if enabled).

Authorized users can monitor and administer the CPU through a web server over a network. This allows for evaluations, diagnostics, and modifications to be done remotely. Monitoring and evaluation can be done without STEP 7, only requiring a web browser. It is important to take necessary precautions to protect the CPU from compromise, such as restricting network access and using firewalls.

It is also possible to insert own "User-defined pages" into the webserver. To create your own user page(s), you can use TIA Portal WinCC Unified as of version V19 or any HTML editor.

The web server can be reached via the configured IP address. In this example you can reach the webserver via following address: https://192.168.0.10.

Figure 8-47: Example of the web server view for the PLC S7-1500.

The rights for the web server can be assigned individually based on the UMAC settings. This setting can be made in the security settings. The rights can be set in the "Role" tab. More Information can be found in the Documentation for the Webserver.

NOTE Please note that the granular setting of the rights for the web server in the security settings is only possible with TIA Portal V19. If an older version is used, have a look into the documentation of the webserver - chapter 3.5 "User Management".

More information about the Webserver can be found in the following documentation:
https://support.industry.siemens.com/cs/ww/en/view/59193560

8.9. Bookmarks

This feature allows the user to save up to two screens to directly jump to from the main navigation.

The current screen can be saved by clicking one empty star. When this is done, the name of the screen will appear next to it, and the star color will be set to yellow. As shown in the following picture (1).

To delete the bookmark, click the cross that appeared when the bookmarked was created (2).

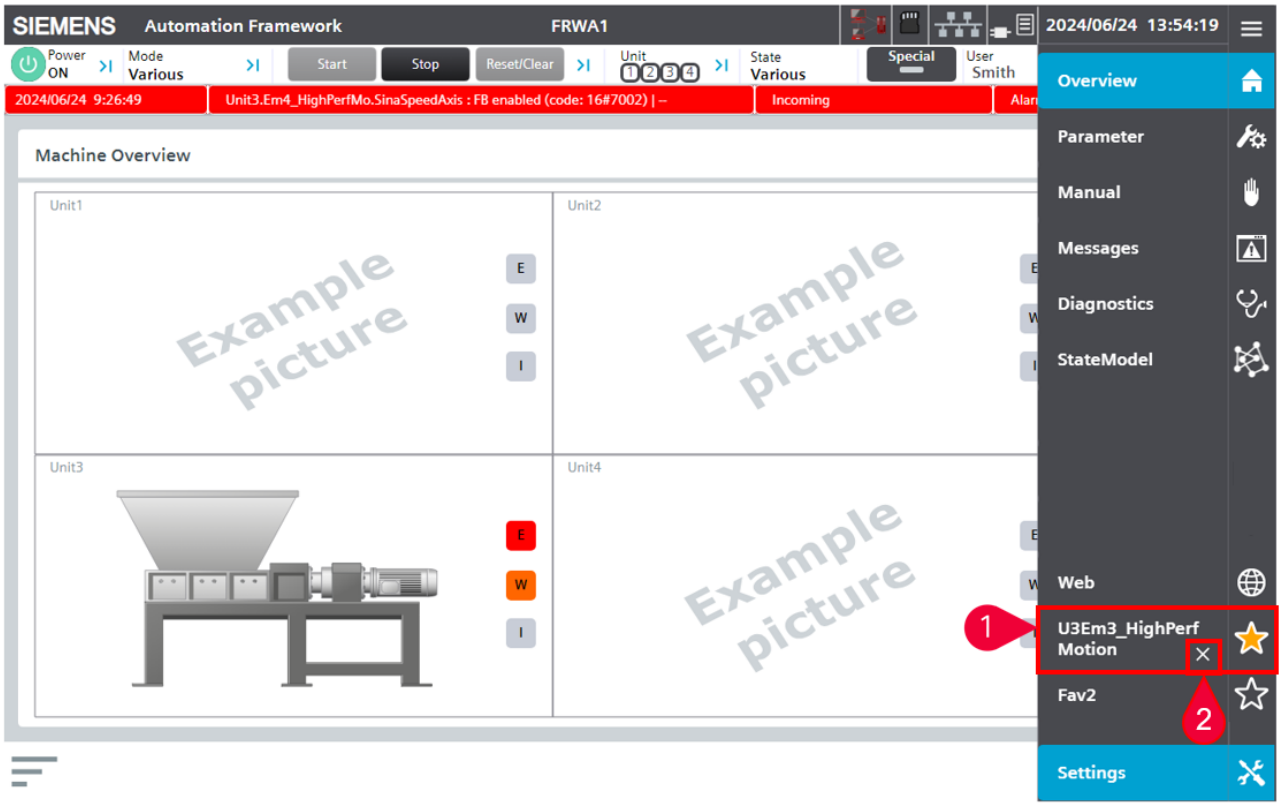


Figure 8-48: Bookmarks.

8.10. Settings

Settings are for general PLC information as well as specific TIA Portal project information. The following information is available in four dedicated screens:

- PLC & HMI
- PLC IM (Identification and Maintenance)
- Checksums
- Show and change current Date and Time

PLC & HMI

The PLC & HMI screen provides information about the PLC and safety cycle times (1, 2). The current as well as the minimum and maximum cycle time are displayed. A new reading of these parameters can be triggered via the button Reset (PLC/Safety) Timers.

The predefined parameters in the PLC as well as the set communication load can also be displayed.

An evaluation of the SIMATIC Memory Card state is also presented. (3)

More information about the Memory Card Diagnostic can be found by pressing shift + F1 in your TIA Portal Project on block "LAF_MemCardDiag"

Furthermore, the following WinCC Unified panel functions are implemented: (4)

- Stop WinCC Unified Runtime
- Open the control panel
- Update trigger
- Switching Language

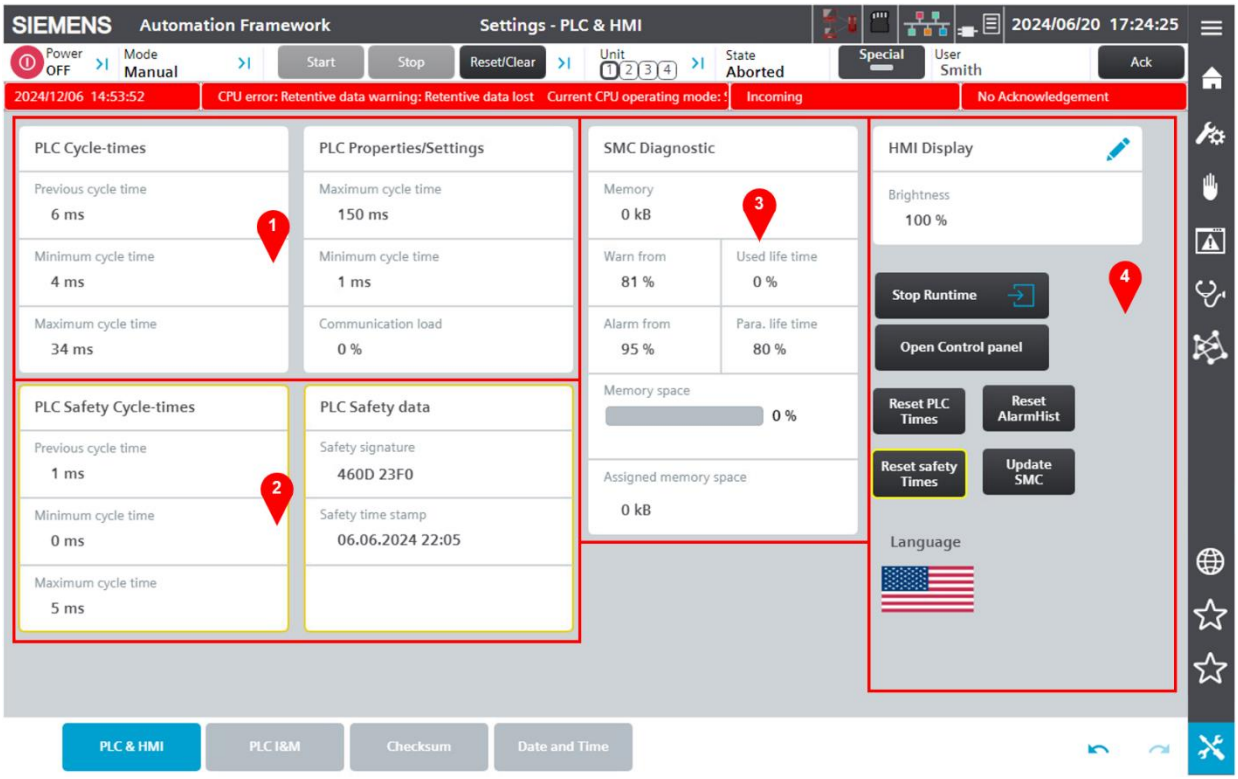


Figure 8-49: Settings PLC & HMI.

PLC I&M

The Identification and Maintenance Data as well as the CPU network addresses are read once after a CPU warm start and displayed in the screen below.

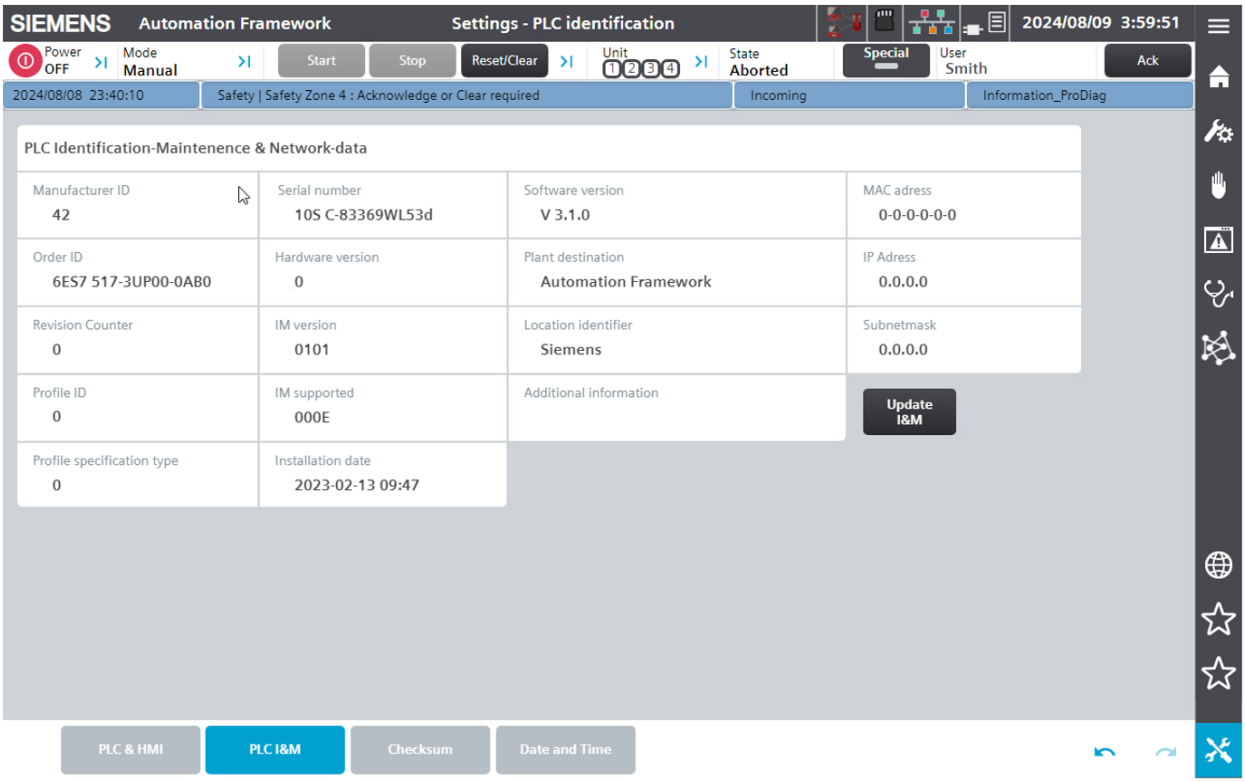


Figure 8-50: Settings PLC & HMI.

More information about the PLC Diagnostic can be found by pressing shift + F1 in your TIA Portal Project on block "LAF_PlcDiag"

Checksum

The block reads the checksums for PLC standard blocks (1), for PLC safety blocks (2) and for PLC text lists (3). The current checksums are compared with the previous values / old values. For each deviation a ProDiag warning is issued, and the colors of the WinCC objects in screen "Settings - Checksum" are animated.

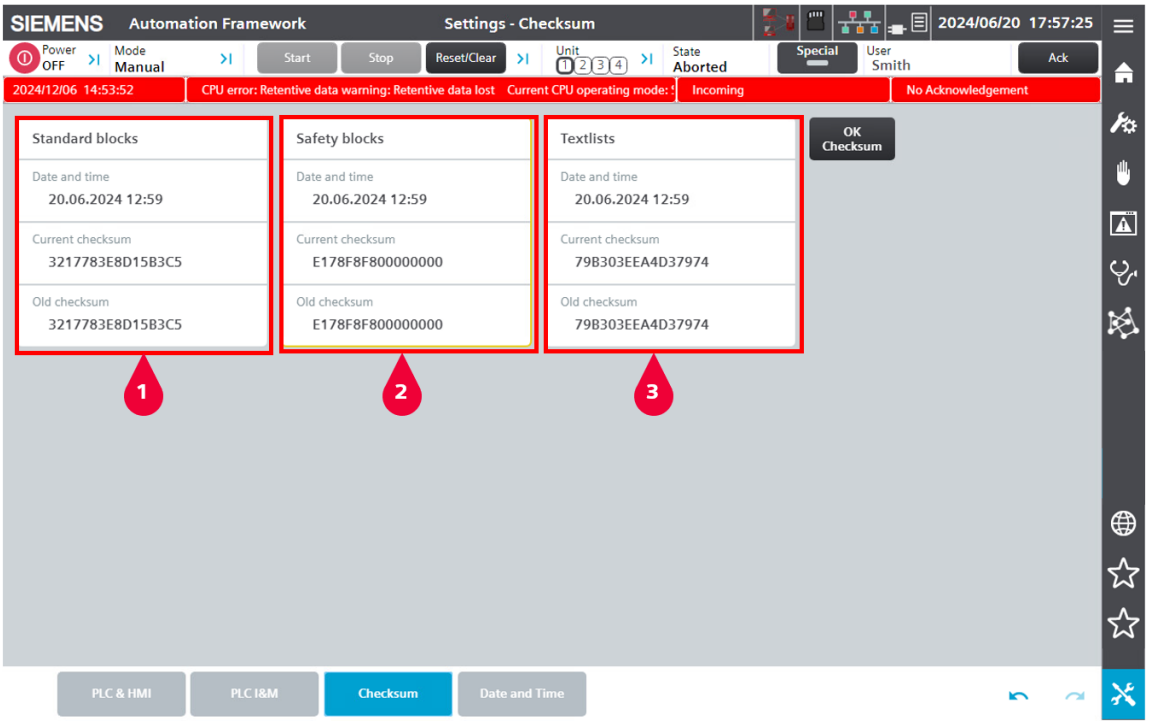


Figure 8-51: Settings Checksum.

More information about the Checksum can be found by pressing shift + F1 in your TIA Portal Project on block "LAF_Checksum"

Date and Time

The PLC time can be edited via the Screen Date and Time. This date and tine information is passed to the CentralFunctions → Programm blocks → 14_Settings → CallSettings.

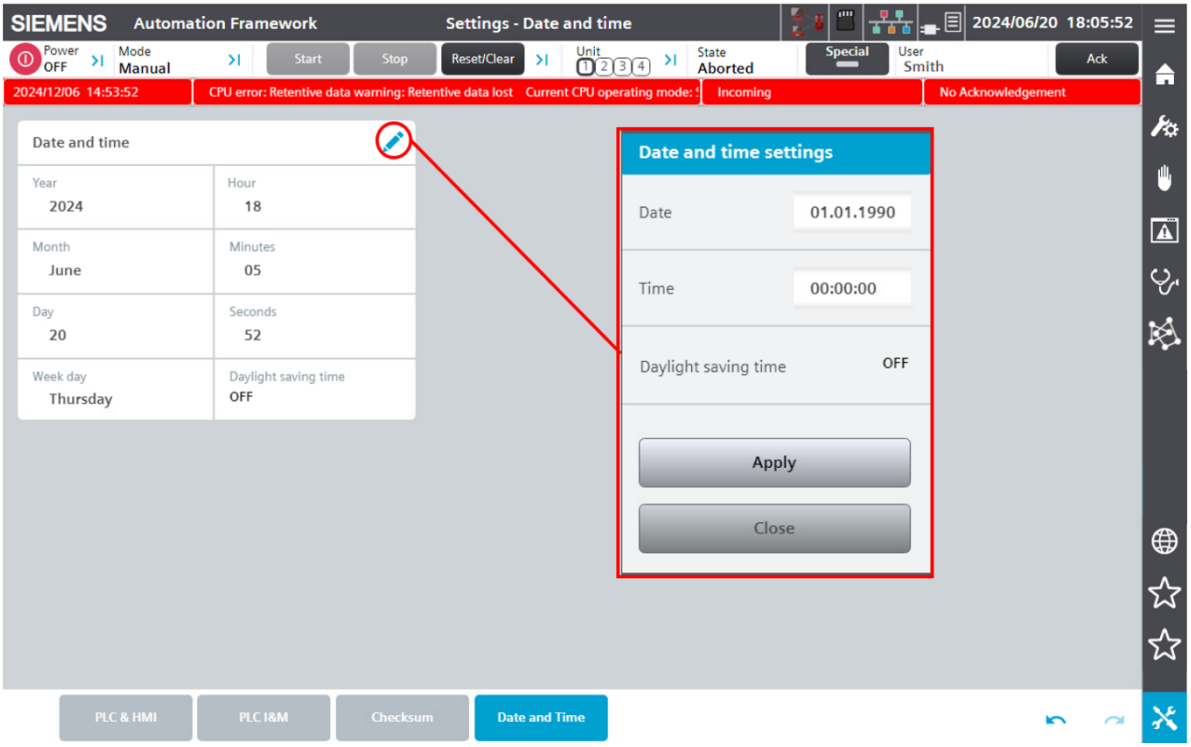


Figure 8-52: Settings Date and Time.

9. Units

This chapter focuses on the units. It describes the general structure, interfaces, commands, and data flow.

NOTE Keep in mind to distinguish between a unit according to ISA-88, and a Software Unit in TIA Portal. Units in this documentation always refer to ISA-88 units. Software units are explicitly expressed as Software Unit.

Four units are implemented in the AF project:

- Unit 1: Example implementation programmed with SCL.
- Unit 2: Example implementation programmed with LAD.
- Unit 3: Demo implementation demonstrating the handling of demo EMs. Programmend in LAD.
- Unit 4: Dummy unit that can be used as startup for setting up an own unit with underlaying EMs.

Figure 9-1: Software Units for the Unit and EM.

9.1. General Structure

Each Unit consists of several Software Units. The first Software Unit representing the Unit itself. Additional Software Units representing each equipment module that are assigned to the unit.

Main program structure

- The Unit level is handling the Safety evaluation and reaction based on the safety zone the Unit is assigned to
- Internal Alarms are processed and managed
- Handling of the feedback, interfaces and post processing of the State & Mode Manager and the State & Mode Manager itself, which will be further discussed in the following chapter.

Signal flow

The signal flow starts with the assessment of the current feedback obtained from the EMs, as well as the feedback generated by the own module and the safety status. This evaluation is carried out using the function block "LUC_StatusCollector". The outcome is the creation of a current process interface. This interface is then evaluated together with the external interfaces, such as the data from the HMI, i.e. commands actuated by the operator, or external signals such as those from an IT system (e.g. OPC UA) in the FB "LUC_InterfaceManager". The block generates a customized interface based on this information, which is compatible with the LPML block. It can request mode and state changes or transmit "StateComplete" messages. The "LPMLV2022_UnitModeStateManager" then evaluates the commands and sets the current mode and state, which is transferred to the DB "EmNodes" via the "LUC_SetUnitStatus" function. From this, the EMs in turn read the current unit status and execute their logic and set new feedback signals in this cycle, which are then evaluated again in the new cycle.

The overall signal flow of a unit can be seen in the following figure.

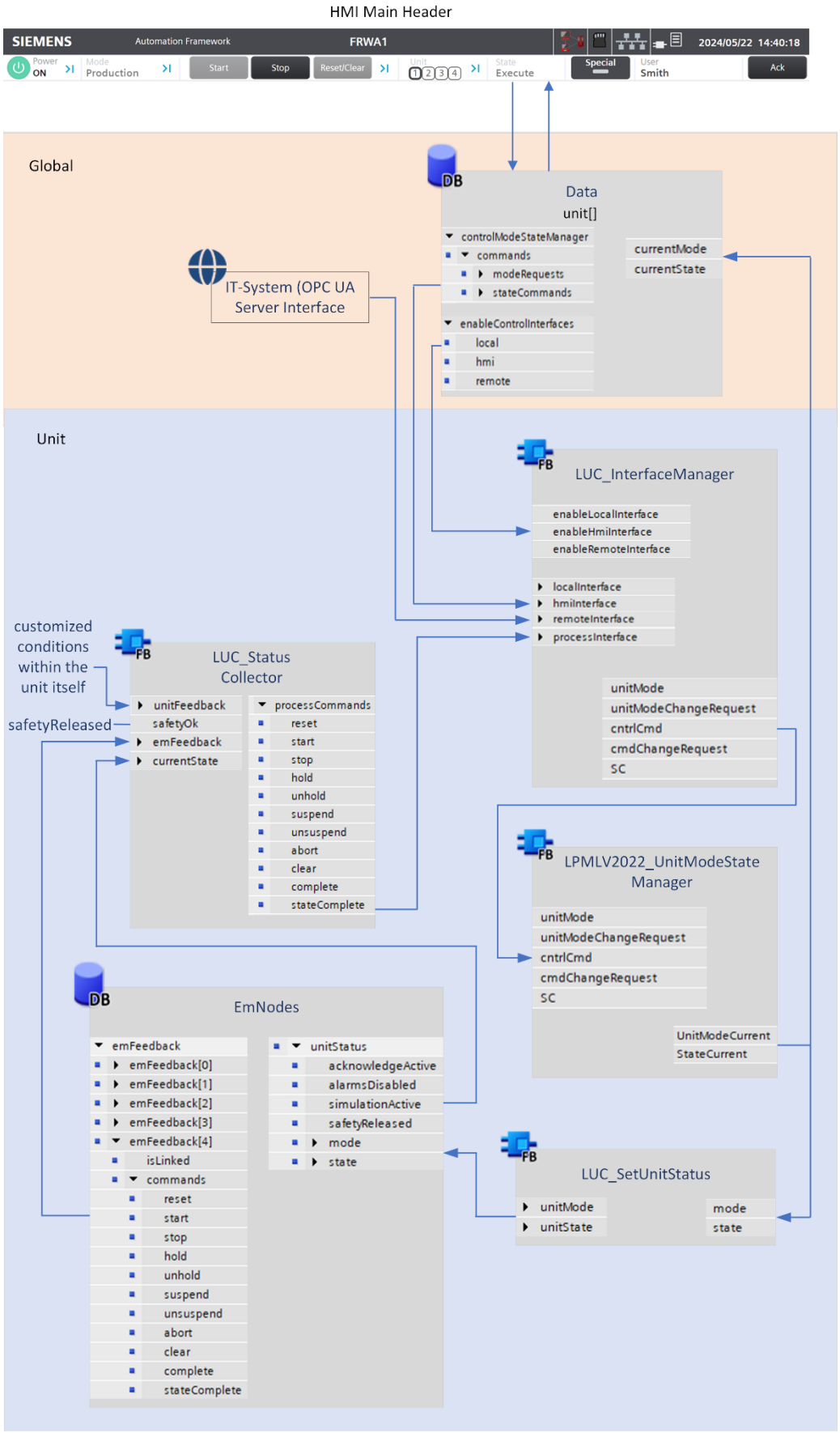


Figure 9-2: Signal flow of a unit.

9.2. Interfaces and blocks

The general program architecture of a unit is based on the description in the chapter Program structure. In order to specify the differences between the levels and the objects used, the following chapters describe in more detail the unit level and the different approaches of the units implemented within the AF.

Relations

The unit requires Relations to be configured to the following Software Units as shown in the figure.

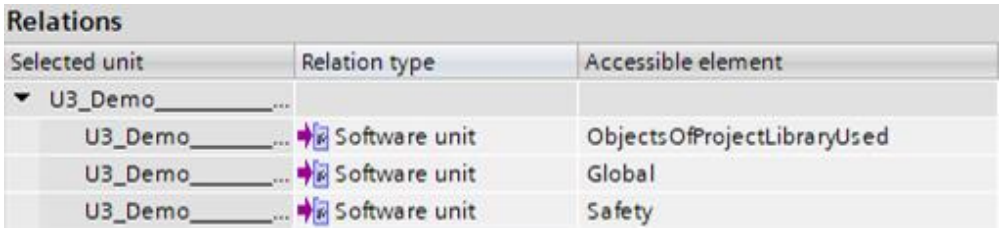


Figure 9-3: Relations of the unit program blocks.

A unit contains the following blocks:

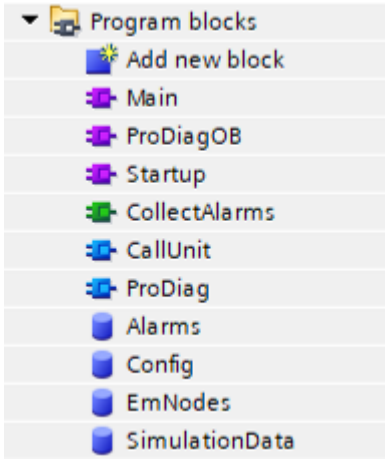


Figure 9-4: Program blocks of the unit.

Primarily the blocks are used from the LPMLV2022 (PackML library) for the mode state manager, the LUC (Library of Unit Control) for the supplementary signal handling within the unit and the LAF (Library of Automation Framework) to handle the alarm management within the respective unit. To gain a more detailed understanding of how special blocks operate at the unit level, it is necessary to investigate the specific implementation within a unit. This discussion will cover the additional functionality beyond the basic operations already described.

OB Startup (Unit)

Configuration of the State-Model:

The user can adjust the state machine to his requirements. This is optional, and only needs to be done if the user has a different state machine like the standard OMAC PackML state machine.

In OB "Startup", FB "LUC_StateModelConfig" transfers the user setting settings into the stateMode manager configuration which are used by the state mode manager during runtime.

DB Config (Unit)

In the unit DB "Config", the following configuration setting can be adjusted by the user:

- Reference Designators of the relative and parent level
- Signal stack behaviour
- Configuration of which OPC UA methods are enables to control the state machine remotely
- State model configuration based on a OMAC PackML state machine

DB EmNodes (Unit)

This DB contains the status of the unit, the feedback of the equipment modules, the relation Signals between the different EMs of a Unit and the current status of the signalStack.

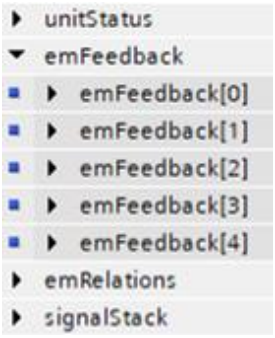


Figure 9-5 EmNodes Data block of the unit level.

In the demo unit of the AF (U3_Demo), the machine has five equipment modules that are assigned to the unit level. Em4 and Em5 are only for showing how different approaches can be implemented but are not functional in terms of the unit to EM behaviour.

Each equipment module writes its current status in the related array element of "emFeedback". This status is evaluated in the unit logic and influences the state machine. Similarly, the unit writes its current status into the "unitStatus". This status is evaluated in the logic of each equipment module, and they run their jobs/sequences depending on the current mode and state. The "emRelations" signals can be used to transfer information between the different EMs. This signal exchange is centralized in the unit's data block "EmNodes" to see what is currently being processed in the levels below. The Array of the "signalStack" can be used to connect hardware objects like a Stack Light within the Em0.

DB SimulationData

This global DB hosts values for simulating the TIA Portal project without the need of a hardware setup.

OB ProDiagOB / FC CollectAlarms / FB ProDiag

These blocks are used for the alarm concept implemented within the Automation framework. Further explanation how the blocks are used is provided within the chapter Diagnostic Concept.

FB CallUnit

The function block FB "CallUnit" performs the followings actions:

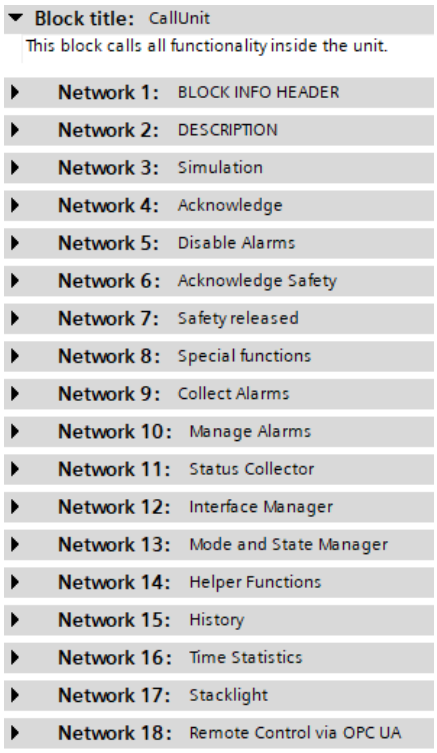


Figure 9-6 Content of FB "CallUnit".

Simulation / Acknowledge / Disable Alarms

If simulation, acknowledge or disable alarms is active on a global level, inherit this status is set active in this unit also. Information is written to DB "EmNodes", from where it is distributed to the equipment modules.

Acknowledge Safety / Safety released

If safety acknowledgement is required, the unit sends an acknowledge command to the safety program, if the state machine is in the corresponding state or if an acknowledgment was set on the global level. The unit evaluates the safety releases of the assigned safety zone and reacts accordingly. If a safety event occurred, it sends an abort command to the state mode manager.

Special functions

Special functions commands to the unit are copied to the DB "EmNodes", so that all EMs can inherit them an react accordingly.

Collect Alarms

Any event can be assigned to an alarm within the collect alarms function. This specific alarm can be monitored by ProDiag and mapped to a reaction for the unit (e.g. abort, stop, hold). More information on the process diagnostic within the AF is given in chapter Diagnostic Concept.

Manage Alarms

This block derives commands from the ProDiag supervision. The ProDiag supervisions contain categories which define the reaction of these supervisions (e.g. abort, stop, hold). These are handed over as commands to the status collector.

Status collector

Here the feedback is collected from the unit level (commands) and all underlaying equipment modules (commands, stateComplete) and merged. The block handover one summarized set of commands to the interface manager.

InterfaceManager

The different commands sources for the ModeState Manager are evaluated and summarized. This could be commands from the operator via HMI or IT, or state change request due to the process status evaluation.

Mode and State Manager

This block represents the mode/state model of the machine. It always outputs a current mode and state of the machine. It evaluates if a mode change request, state change request or stateComplete input is pending and updates the mode and state accordingly, if allowed.

Helper Functions

Further signal handling of the mode state manager to interact with the data structure of the unit level.

History / Time Statistics

This block updates the mode and state history. When a mode or state change occurs, the current mode and state are stored in the current data area of the history and the previous ones in ascending order. The runtime of the previous mode and state of the unit is then evaluated by the "LPMLV2022_UnitModeStateTimes" block within the time statistics network.

Stack Light

This network sets the signal stack outputs according to the configuration and the current mode and state.

Remote Control via OPC UA

The function block "LUC_RemoteCtrl_OpcUa" within this network is used to control the state machine via OPC UA method calls.

9.3. HMI Screen

To access the overview screen of a unit the operator can either select the unit from the main overview of the whole machine (1) or get there via the Navigation bar which will open if the icon in the left corner is pressed (2). For simulation purposes a simulation activation Button is added to the main overview screen. Additionally, the collective signal of the safety zones is displayed through the virtual E-Stop Button. If all safety zones are released, the button will be deactivated. However, if any of the zones is not released, the button will be shown as activated.

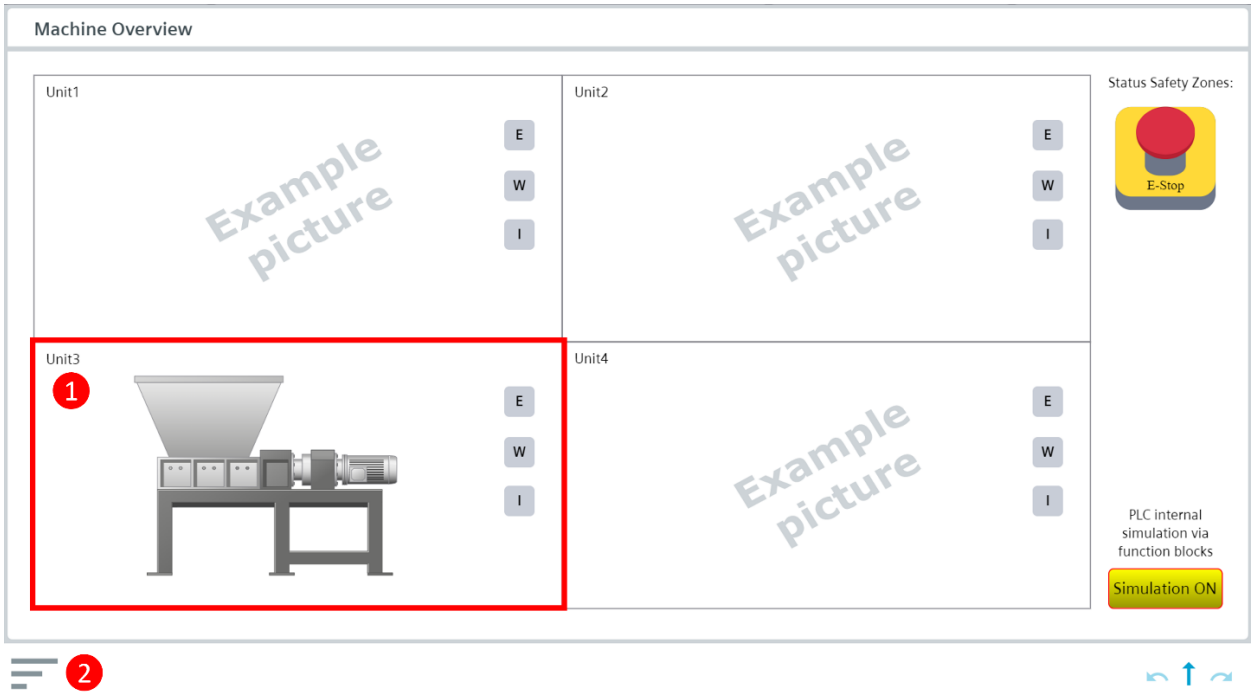


Figure 9-7: Overall machine view.

If the selection of a unit is made, the overview screen of the specific unit opens and shows the different equipment modules. The demo implementation of Unit 3 is shown in the Figure 9-8: Demo Unit 3 Overview Screen. . For customer projects, the specific machine schematic or screens can be added here. The operator can access the individual equipment screens by clicking on the equipment in the overview screen or also use the navigation bar which also include the EM level of the machine. The structure of the equipment module screens will be presented in chapter Hmi Screen.

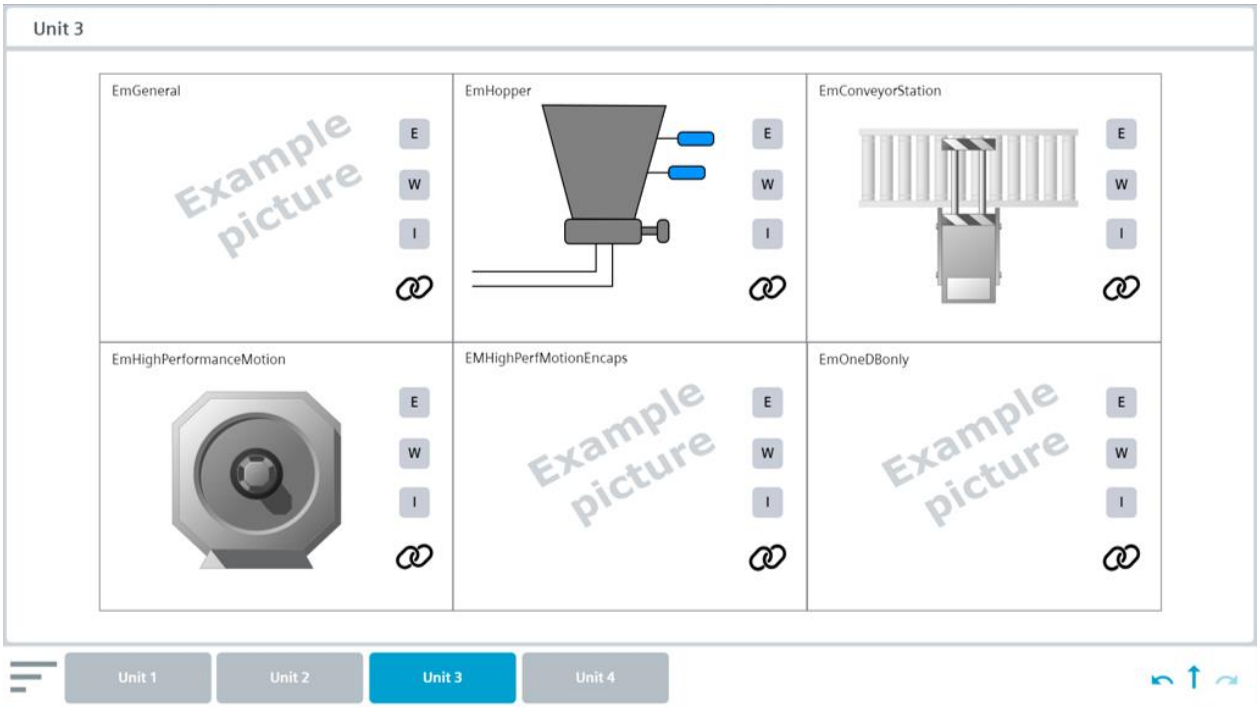


Figure 9-8: Demo Unit 3 Overview Screen.

9.4. Mode and State Manager

The AF uses the LPML library, which provides an implementation of an OMAC PackML V2022 compliant mode and state manager including supplementary blocks. Additionally, the Library of Unit Control (LUC) is used for further related mode operation tasks like command and status collection and interface management to different command sources.

The Mode and State manager is called in the FB "CallUnit" of each unit. It is realized with the FB "LPMLV2022_UnitModeStateManager" of the LPML library. For details about that block refer to the documentation of that library. The main functionality of the block is to evaluate the requested Modes and States of external Interfaces, if a Mode or State change could be done and provide the current Mode and State to other program parts.

Configuration

The Automation Framework is providing an easy way on implementing the needed configuration for the State Machine by adding another data structure based on the PLC data type "LUC_typeStateMachineConfiguration" to the DB "Config" of the unit. Therefore, there are two data structures inside this DB. One for the configuration of the user and one UDT which is used by the "LPMLV2022_UnitModeStateManager". The following parameters can be adapted to the own needs:

- Used modes
- Used states per each mode
- Allowed transitions between modes
- States that are allowed to transition to the hold or complete process

The state model configuration can be defined by the user in "enableModes", "enableStates", "enableModeTransitions", "enableHoldCommand" and "enableCompleteCommand". Each mode has its own configuration of states and transitions. The configuration of the hold and complete command is independent from the current mode and defines which state can transition to the hold or complete state.

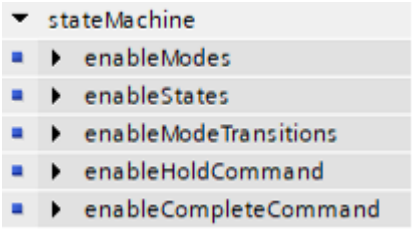


Figure 9-9: Configuration of the state machine in the DB Config of the unit.

To enable a new mode or disable old modes, the user can set the corresponding Bit within the "enableModes" variable. The next step would be to define the states that are existent within the different modes. For this purpose, the "enableStates" data structure is including every state as a Boolean variable for each mode. The "enableModeTransitions" is defining in which state the mode could be changed. This data is structured in the same way so every mode has the transition activation for every state. Additionally, by setting or not setting the bits, it can be configured whether the transition from a state to Hold or Complete is possible.

HMI Interface

The data of the Mode and State manager for the HMI is centrally stored in the DB "Data" of the Software Unit Global. Each unit that was projected is including the data structure shown in the figure below. This interface enables the state machine to be controlled via the HMI and the current status to be read out.

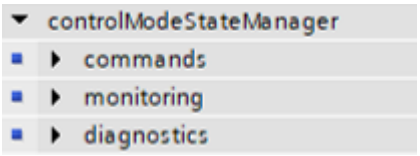


Figure 9-10: ModeStateManager Interface in DB "Data" in Software Unit Global.

History and statistics

A history of the previous states and mode is also stored for each unit within the DB "Data" in the UDT "history" of the specific unit. The last 15 steps of the state machine are provided. The FB "LUC_UpdateModeStateHistory" (called in FB "CallUnit") updates that history upon each mode or state change.

An time statistic of all mode and states is stored in the UDT "modeStateTimes". This provides a detailed overview of the times the state machine has stayed in the individual modes and states. The FB "LMPLV2022_UnitModeStateTimes" (called in FB "CallUnit") updates that statistic constantly.

Screens

The mode and state of the unit can be controlled through the HMI by utilizing the mode and state command buttons located on the main header (see Figure 9-11). Additionally, from the main navigation on the right site of the HMI, a specific state model screen can be opened. This screen provides an overview of the configured states of the selected mode and the current state. Another screen shows the history of the state machine (see Figure 9-12).

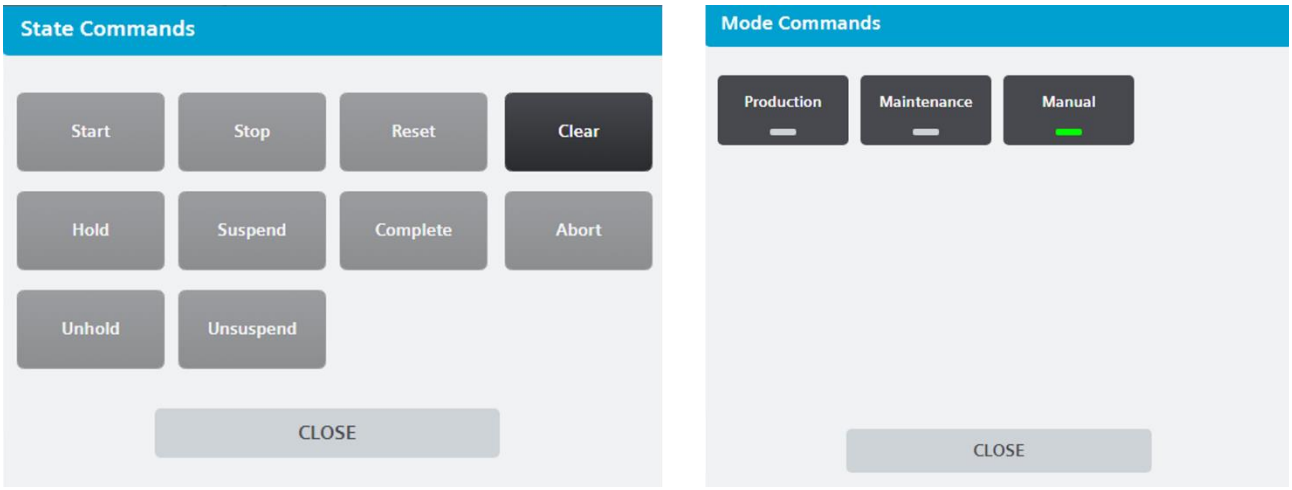


Figure 9-11: State Commands and Mode Commands pop-ups.

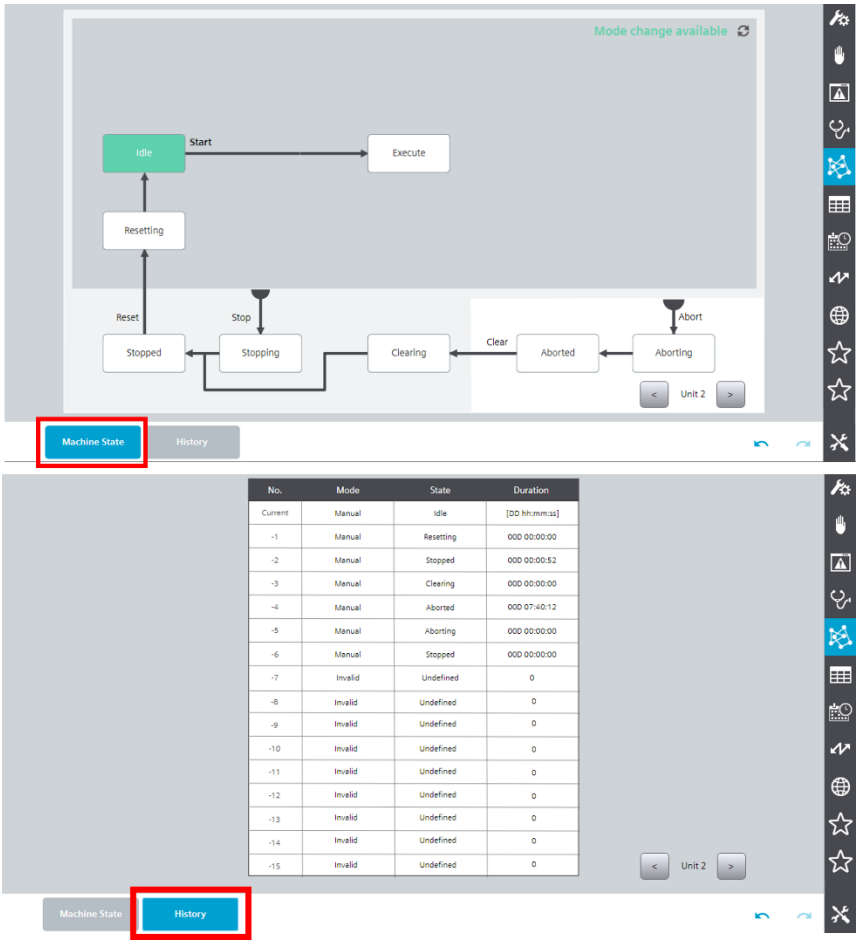


Figure 9-12: State model view and history.

NOTE The visualization of OMAC state model of the Siemens application example \5\ was adapted to comply with the AF Siemens style guide

9.5. Stack light

Each machine typically has at least one stack light that indicates the machine status to the equipment operators. The stack light in the AF supports the five standard stack light colors with the various illumination options (static or flashing) and sufficient reserves for additional user defined colors plus on/off signal for acoustic indicators.

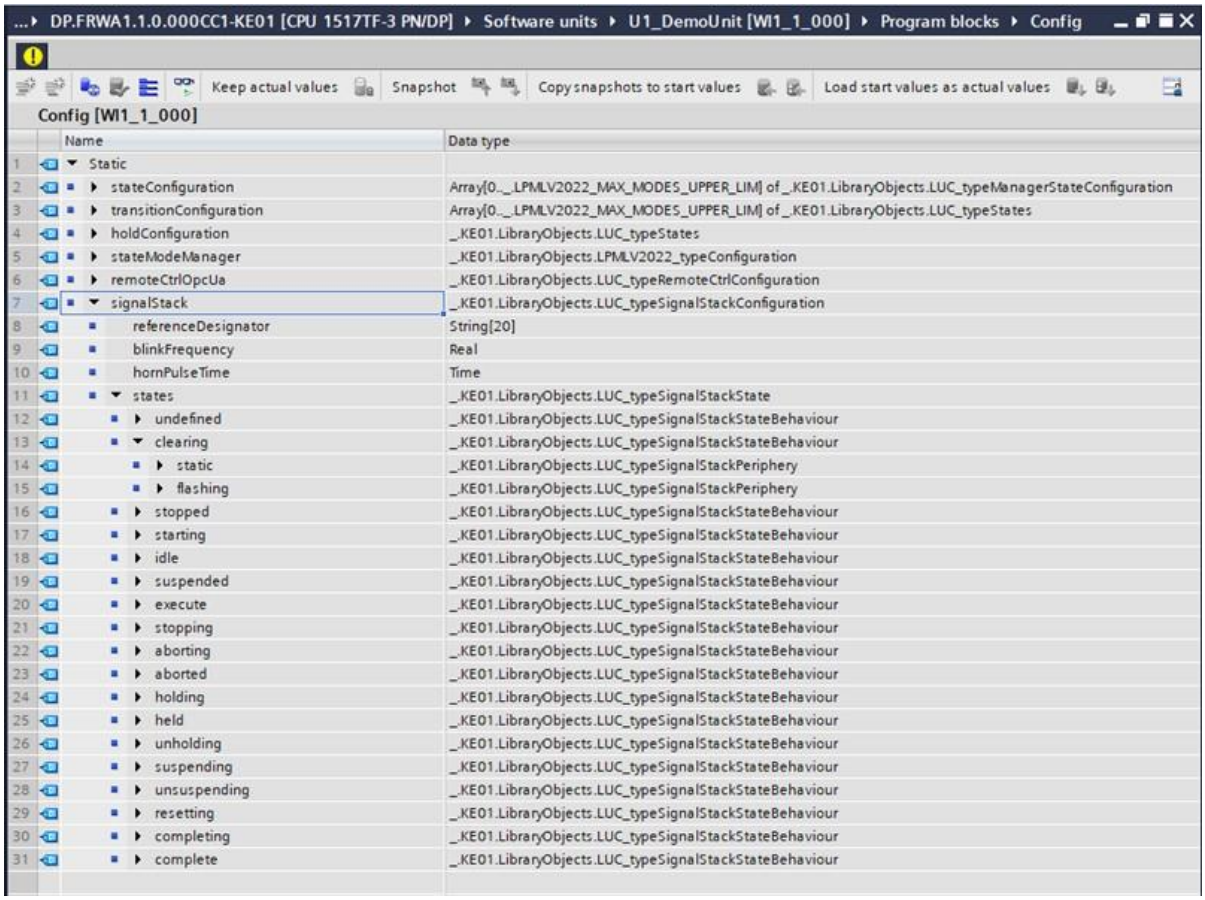
The following figure shows the implementation of a stack light mapped to the different states of the unit, implemented within the AF:

	Undefined	Clearing	Stopped	Starting	Idle	Suspended	Execute	Stopping	Aborting	Aborted	Holding	Held	Unholding	Suspending	Unsuspending	Resetting	Completing	Completed
Red Lamp Flashing	F	F							F	F								
Red Lamp Solid			S					S								S		S
Yellow Lamp Flashing														F				
Yellow Lamp Solid						S												
Blue Lamp Flashing											F							
Blue Lamp Solid												S						
Green Lamp Flashing					F													
Green Lamp Solid				S			S						S		S		S	
Horn Flashing																		
Horn Solid																		
White Lamp Flashing																		
White Lamp Solid																		
User1 Lamp Flashing																		
User1 Lamp Solid																		
User2 Lamp Flashing																		
User2 Lamp Solid																		

Figure 9-13: PackML State Model mapping to the stacklight.

The AF provides a preconfigured implementation of such unit stack light. The example was implemented according to IEC 60073 and matches the OMAC guideline Part 6: PackML User Interface – HMI. The stack lights are configured in the DB "Config" of the Unit, as shown in the Figure 9-14: Stack light in the DB "Config" of a unit.

NOTICE **Please have in mind that the stack light functionality has not been fully implemented in the AF project.**



	Name	Data type
1	Static	
2	stateConfiguration	Array[0..._LPMLV2022_MAX_MODES_UPPER_LIM] of _KE01.LibraryObjects.LUC_typeManagerStateConfiguration
3	transitionConfiguration	Array[0..._LPMLV2022_MAX_MODES_UPPER_LIM] of _KE01.LibraryObjects.LUC_typeStates
4	holdConfiguration	_KE01.LibraryObjects.LUC_typeStates
5	stateModeManager	_KE01.LibraryObjects.LPMLV2022_typeConfiguration
6	remoteCtrlOpcUa	_KE01.LibraryObjects.LUC_typeRemoteCtrlConfiguration
7	signalStack	_KE01.LibraryObjects.LUC_typeSignalStackConfiguration
8	referenceDesignator	String[20]
9	blinkFrequency	Real
10	hornPulseTime	Time
11	states	_KE01.LibraryObjects.LUC_typeSignalStackState
12	undefined	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
13	clearing	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
14	static	_KE01.LibraryObjects.LUC_typeSignalStackPeriphery
15	flashing	_KE01.LibraryObjects.LUC_typeSignalStackPeriphery
16	stopped	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
17	starting	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
18	idle	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
19	suspended	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
20	execute	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
21	stopping	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
22	aborting	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
23	aborted	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
24	holding	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
25	held	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
26	unholding	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
27	suspending	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
28	unsuspending	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
29	resetting	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
30	completing	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
31	complete	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour

Figure 9-14: Stack light in the DB "Config" of a unit.

9.6. Demo Unit

In Unit 3 a demo implementation is demonstrating the handling of the EMs. The Unit has the following structure:

U3_Demo

Implementation of a unit programmed in LAD, including:

- Safety and power release handling
- Alarm Handling within the level itself and underlying modules
- Evaluation of the related EM statuses
- Processing the unit interfaces from HMI or remote control via OPC UA
- State mode manager including the OMAC Model

U3Em0_General

This is a demo implementation of an equipment module, focusing on:

- logic that is needed in the entire unit
- include general releases and system wide functionalities, such as the handling of a machine stacklight

U3Em1_Hopper

This is a demo implementation of an equipment module, demonstrating:

- how to use a technical module (combined use of multiple control modules)
- how to implement different jobs with a fully programmed LAD step sequence

U3Em2_ConveyorStation

This is a demo implementation of an equipment module, demonstrating:

- how to use S7-Graph as the main step sequence
- how to implement different jobs with the S7-Graph sequence

U3Em3_HighPerformanceMotion

This is a demo implementation of an equipment module, demonstrating:

- how to call high performant motion axes in the motion control cycle
- how the high performance motion programm is linked to the equipment module sequence

U3Em4_HighPerformanceMotionEncaps

This is a demo implementation of an equipment module, demonstrating:

- how to use the same high performant motion EM encapsulated (all of the internally used signals are passed as an interface to the equipment module call)

U3Em5_ExampleOneDbOnly

This is a demo implementation of an equipment module, demonstrating:

- how to store all the data of the equipment module within one DB. Therefore the ControlNodes, Config and HmiInterface values are all assigned to the objects of the EM inside this DB

9.7. How to add a new Unit

To add a new Unit, an existing Unit can be used to copy and paste into the Software Unit structure. As an example, another unit called U5_Unit5_NewUnit will be added.

- In the Software Unit Global, go to PLC tags and select the Global PLC tag table. Within User constants Tab, the following steps have to be taken

Global

	Name	Data type	Value	Comment
	UNIT05	Int	4	Unit5
	UNIT04	Int	3	Unit 4
	UNIT02	Int	1	Unit 2
	UNIT03	Int	2	Unit 3
	UNIT01	Int	0	Unit 1
	NO_OF_UNITS	Int	4	Highest value of unit (Counting starts from 0)

Figure 9-15: Global PLC Tag list.

- Increase NO_OF_UNITS by 1 so that is shows the highest value of the last unit within the whole machine.
- Add a new constant with the name of the additional unit and assign the next possible higher number.
- Copy an existing unit of the AF and paste it to the project by right clicking on the Software Units folder in the PLC project tree and select paste.
- In the new Software Unit, adjust the name and namespace as needed. Therefore right click on the new Software Unit and open the properties.
- After putting in the new name and namespace, click "Apply to all underlying elements" to apply the namespace to all objects within the Software Unit.

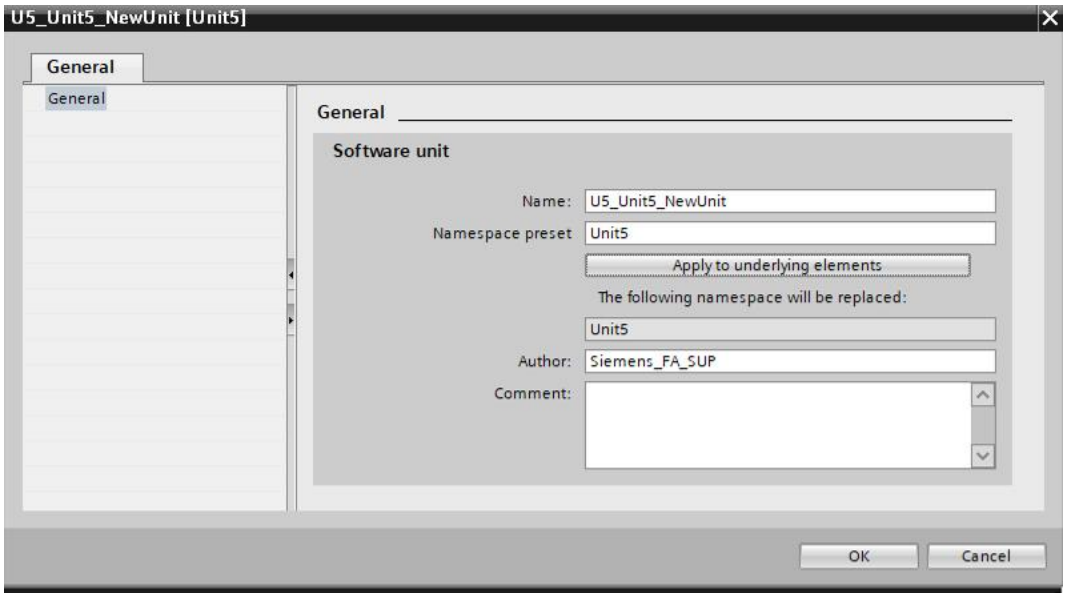


Figure 9-16: Properties of the new Software Unit.

- Open the OB "Main" of the new unit and assign the user constant that was created in the Global Software Unit at the Move-block that is called before the FB "CallUnit" is called
- Assing the right safety zone for the unit, by connecting the corresponding interface to the safetyReleased input and the acknowledgeSafety output.

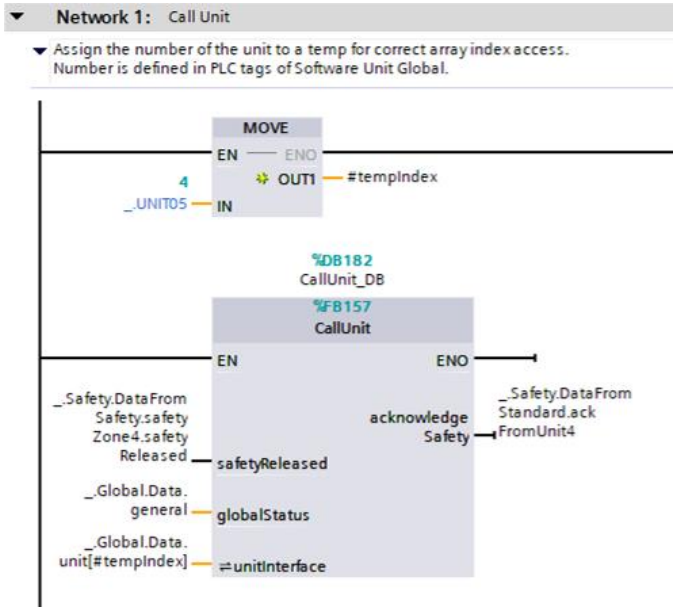


Figure 9-17: OB "Main" of the new Unit.

- Change the OB numbers of the OB "Main", "Startup" and "ProDiag" according to the OB numbering convention presented in chapter PLC start and cyclic call sequence.

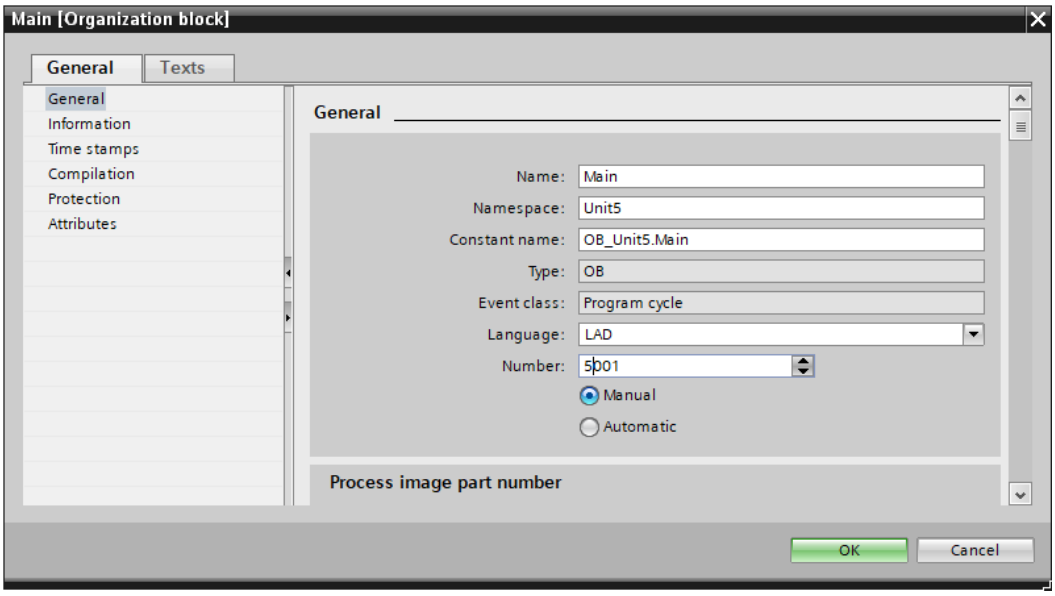


Figure 9-18: Properties of OB "Main".

- Renumber the rest of the blocks by compiling the PLC and select to solve the conflict automaticaly.
- The unit can then be configured as required.

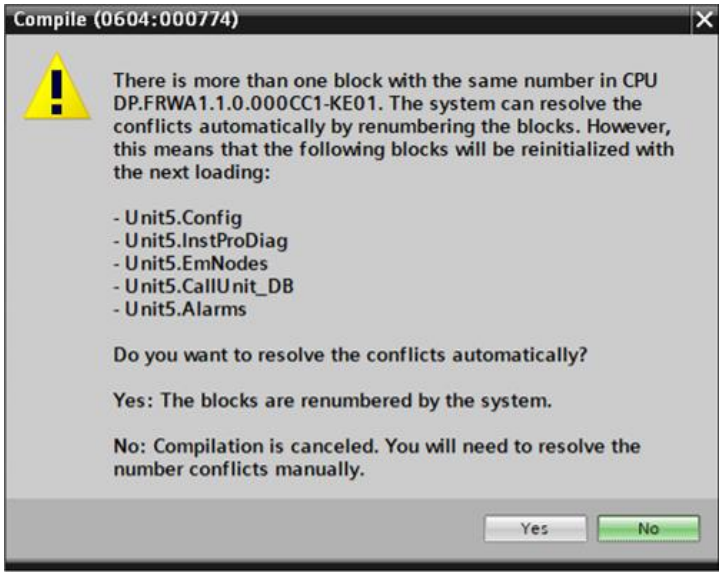


Figure 9-19: Prompt to solve block numbering.

9.8. How to remove a Unit on the HMI

For this example, the Unit4 will be deleted. Therefore, multiple screens and scripts involved should be adjusted and removed to be able to reduce the number of Units.

Firstly, the folder of the Unit should be identified:

- Location: Identify the Unit to be to deleted in the screen hierarchy. Observe its relationship with other Units and screens.

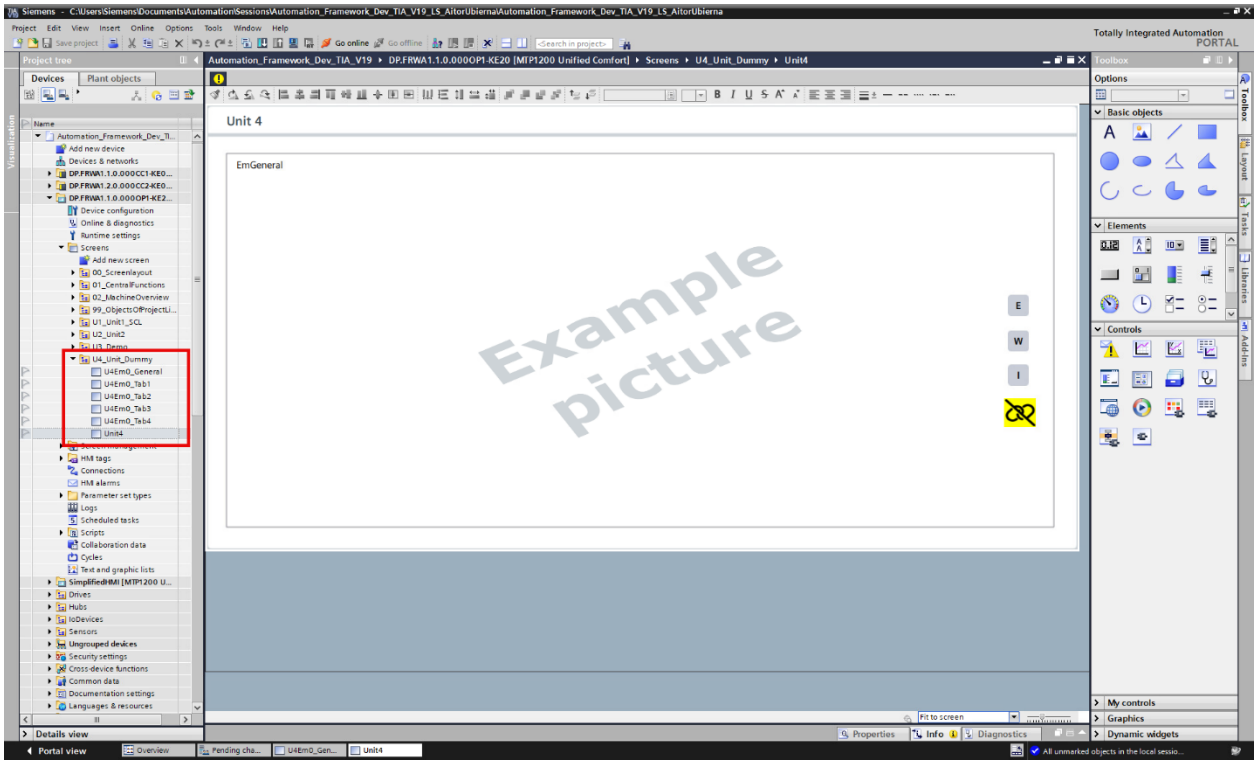


Figure 9-20: Identifying the screen folder and the hierarchy of the desired unit.

In addition, the Machine Overview screen should be adjusted to remove the group of the deleted Unit.

Elements on Other Screens:

- **MachineOverview**: Delete the dynamizations and graphical elements associated with the Unit on the MachineOverview screen:

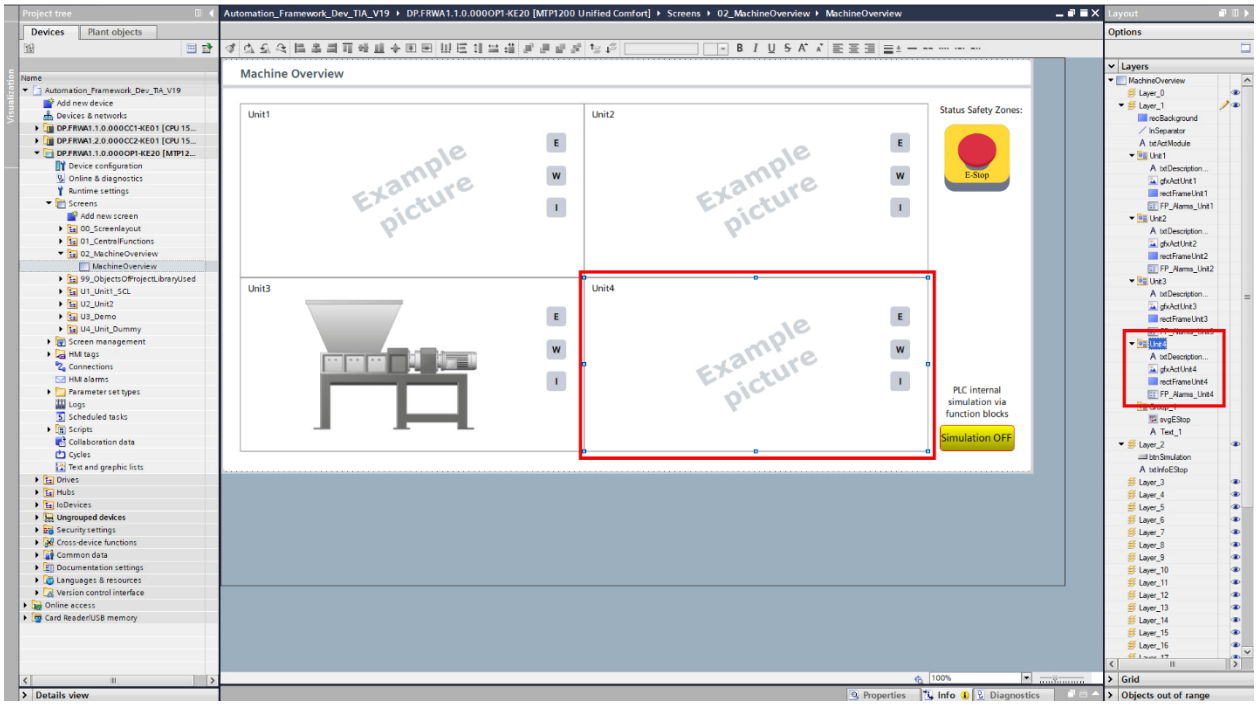


Figure 9-21: Old MachineOverview Screen.

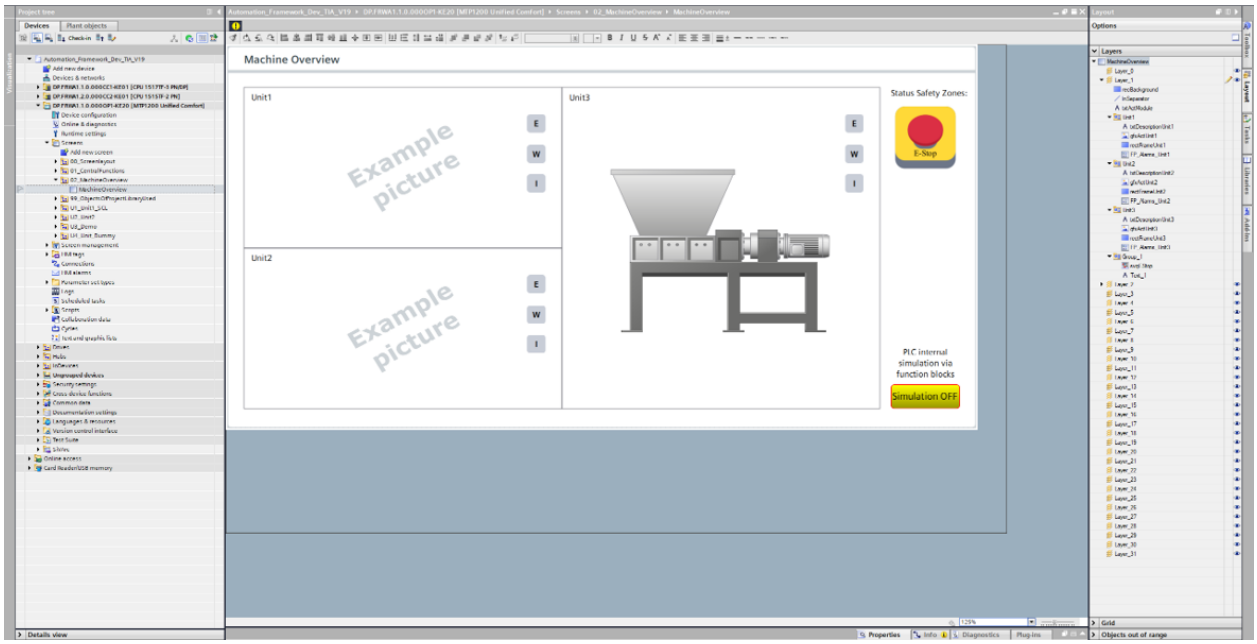


Figure 9-22: New MachineOverview Screen without Unit 4.

- **ScreenLayout**: Delete any graphical elements (buttons, images, etc.) that reference the deleted Unit on the ScreenLayout screen:
  - In the grpUnits, delete the txtUnit4 and adjust the buttons :

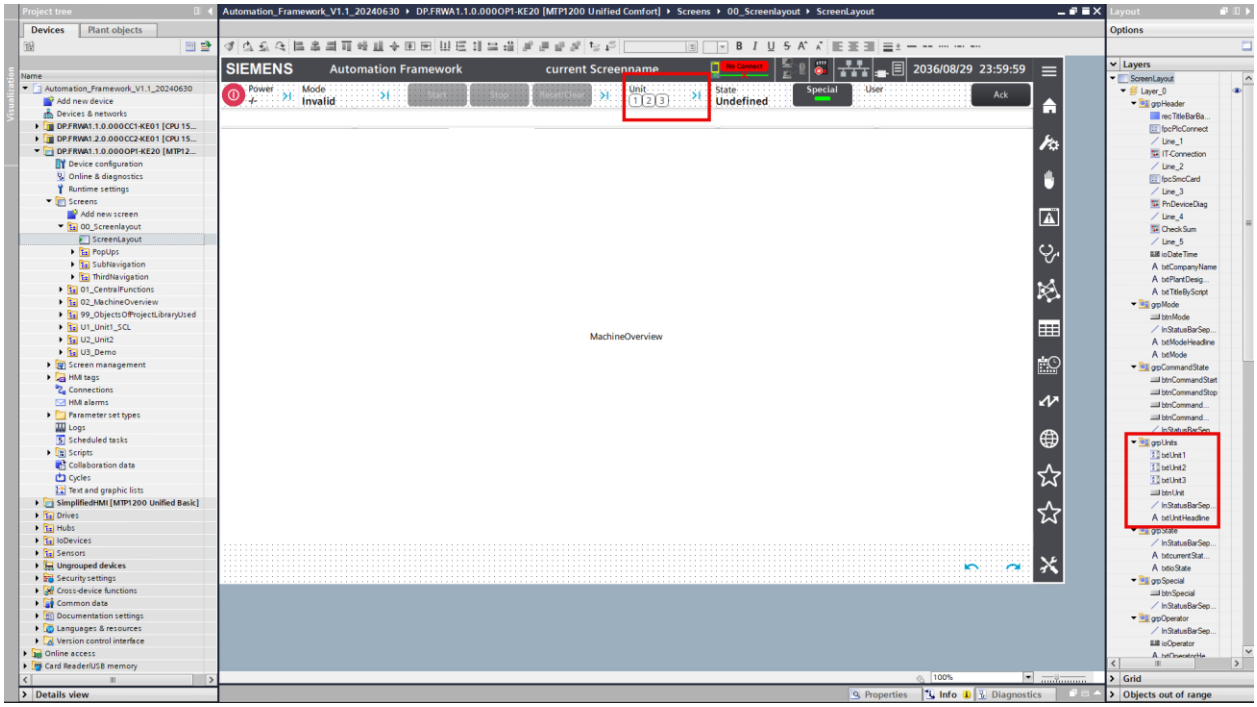


Figure 9-23: grpUnit adaption to delete Unit 4.

  - In the Template_SelectUnit, delete the Unit 4 and re-arrange the popup:

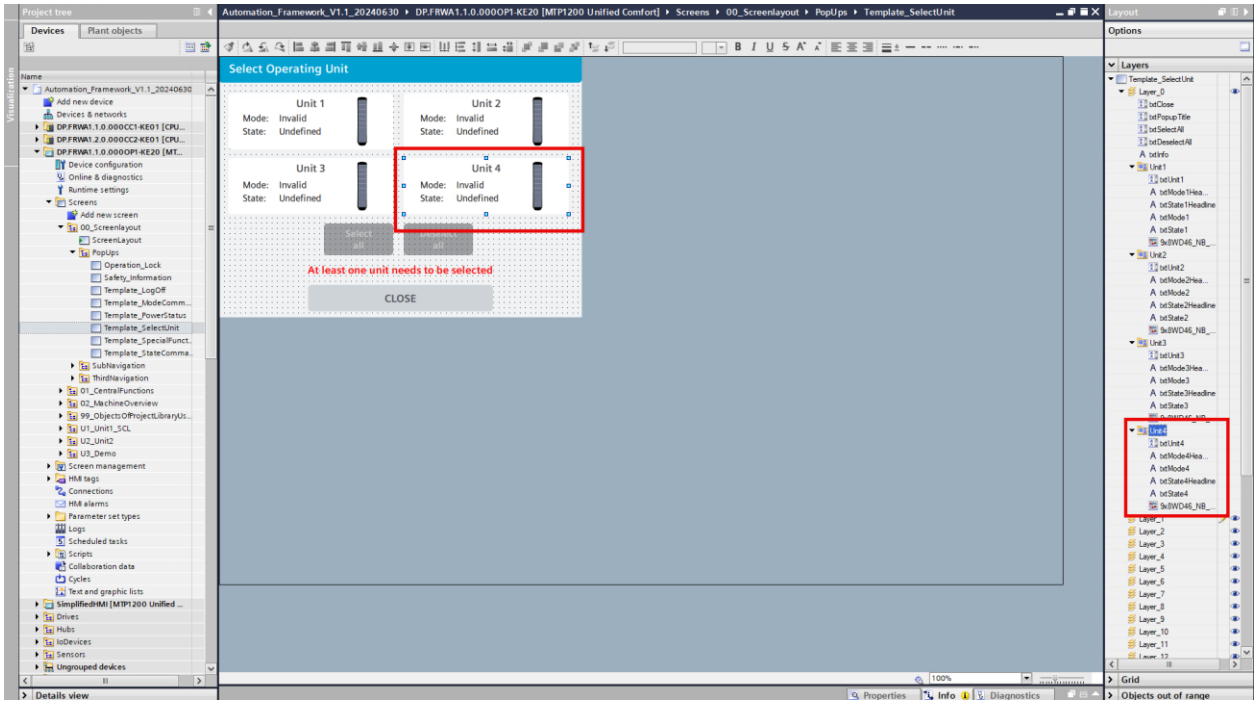


Figure 9-24: Old Template SelectUnit Screen.

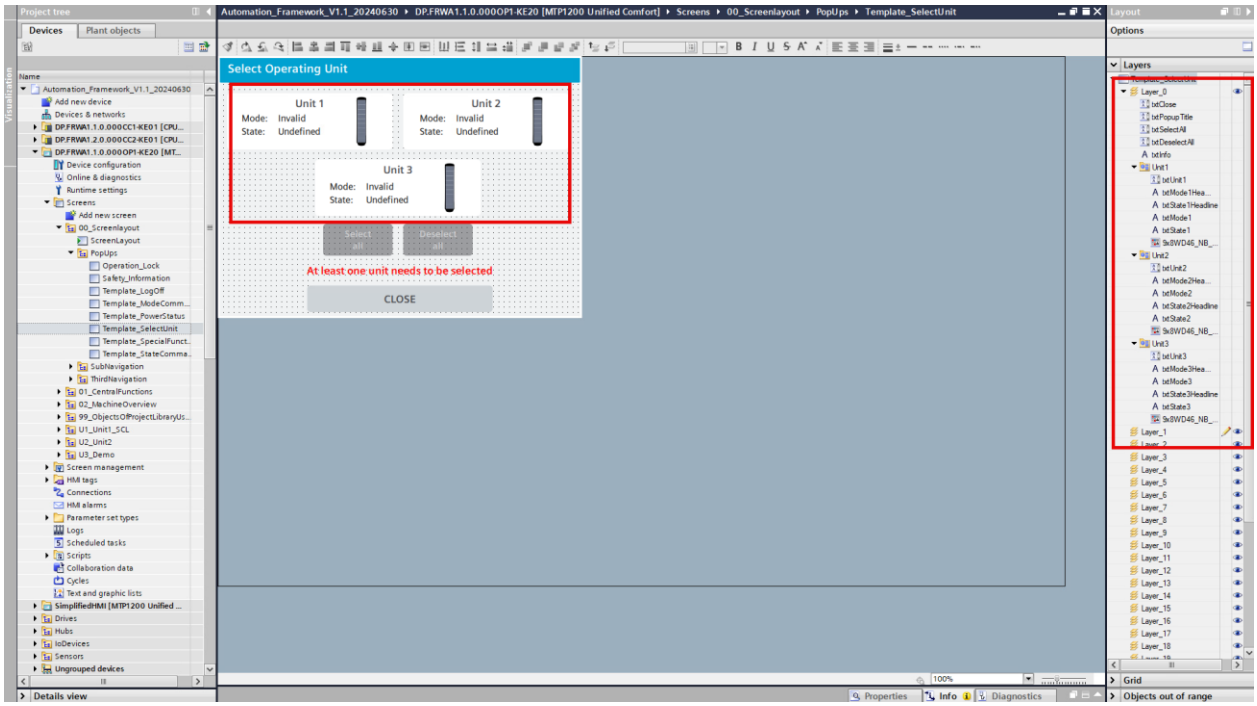


Figure 9-25: New Template SelectUnit Screen.

Then, the navigation should be adjusted to the new number of Units, removing the additional buttons.

- Navigation Buttons:
  - **SubNavigation:** Delete the corresponding button for the Unit on the SubNavigation screen.

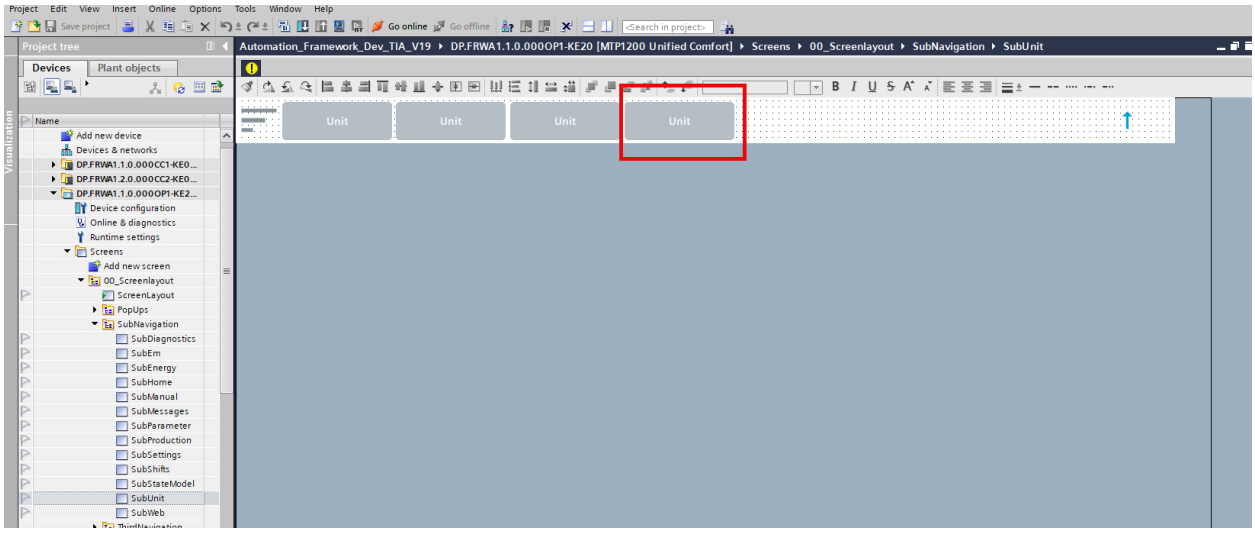


Figure 9-26: SubUnit Navigation adaption to delete Unit4.

In the other buttons of the units, in the script, delete the part of that Unit and re-arrange with the new number of units:

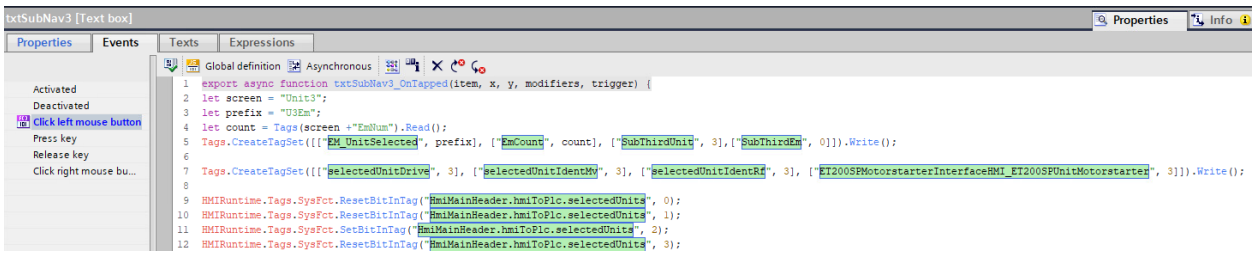


Figure 9-27: Old script of the Sub-navigation.

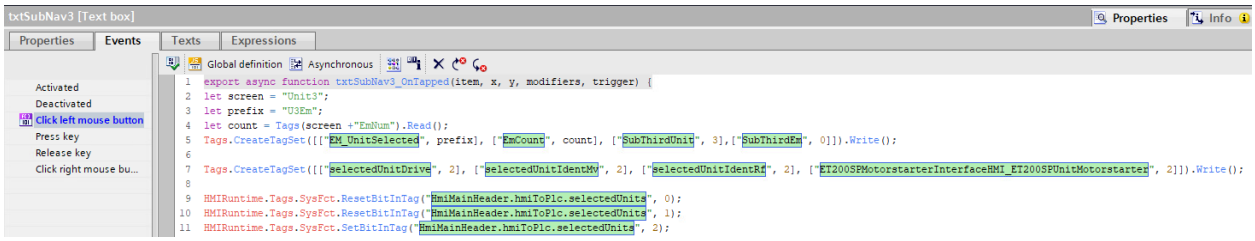


Figure 9-28: New Script of the Sub-navigation.

- **ThirdNavigation**: Delete the corresponding button for the Unit on the ThirdNavigationUnit screen.

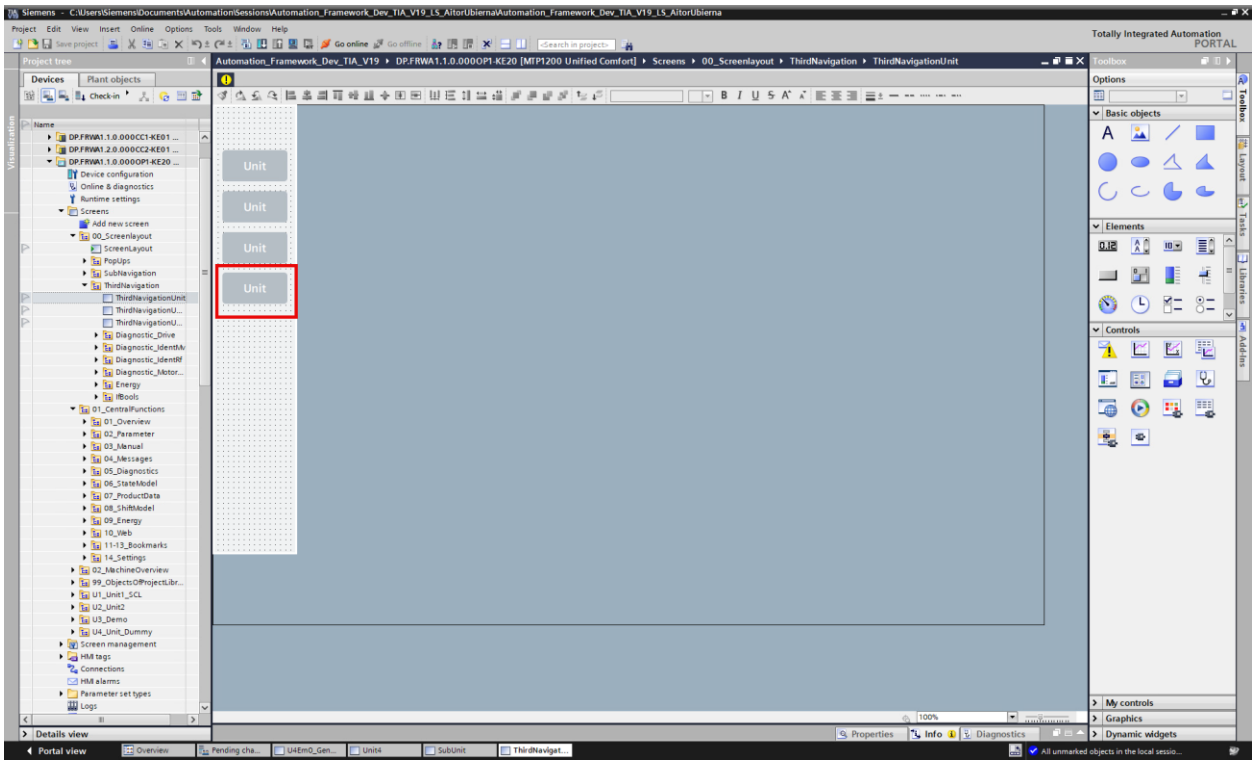


Figure 9-29: Select the unit that is going to be deleted.

In the other buttons of the units, in the script, delete the part of that Unit and re-arrange with the new number of units:

Old Script:

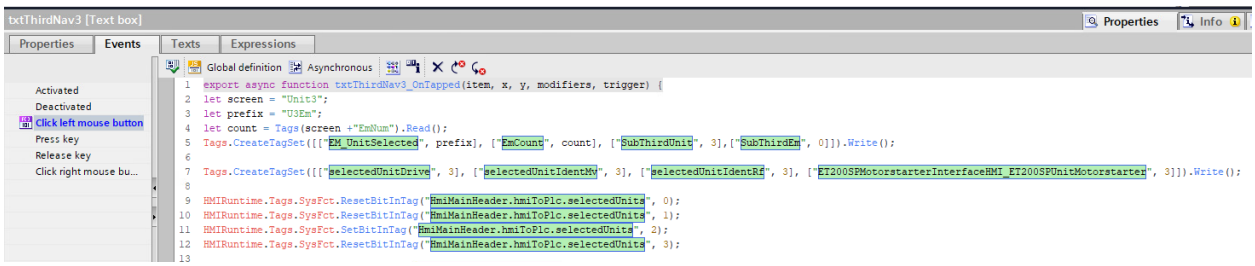


Figure 9-30: Old Script of the Third navigation.

New Script:

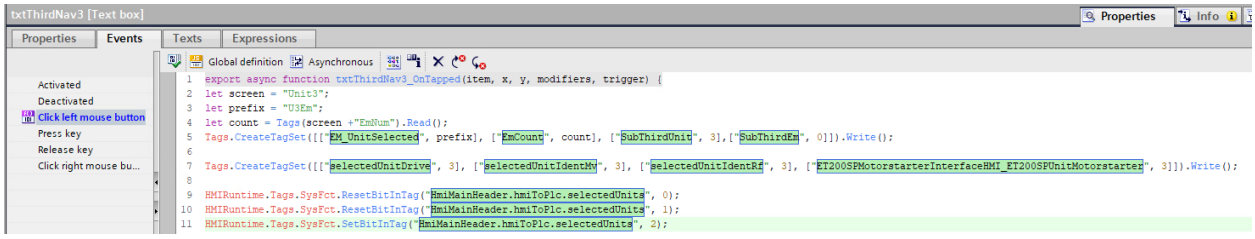


Figure 9-31: New Script of the Third navigation.

After adjusting the navigation, the textlist involved for the equipment modules should be removed (in this case called "U4Em") and the Unit name from the text list used for the Units (called "Units")

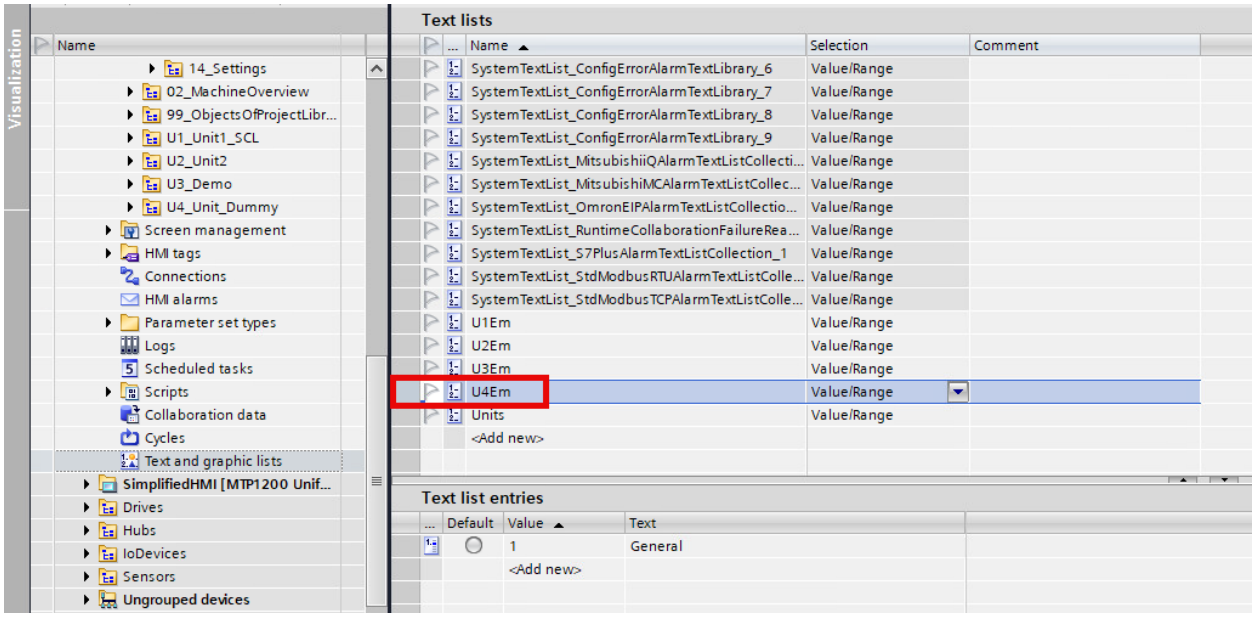


Figure 9-32: Location of the Text list that must be deleted.

- **Compilation:** Compile the project to detect and resolve any errors caused by the deletion.
- **Simulation:** Simulate the project to verify that the Unit has been deleted correctly and the rest of the functions work as expected.

9.9. How to Resize the HMI

For the resizing of the HMI to the desire panel size, you should follow the next steps:

1. Changing the Target Device
   a. Navigate to the device settings within your project.
   b. Select the desired device that matches the screen resolution or display size you intend to work with.

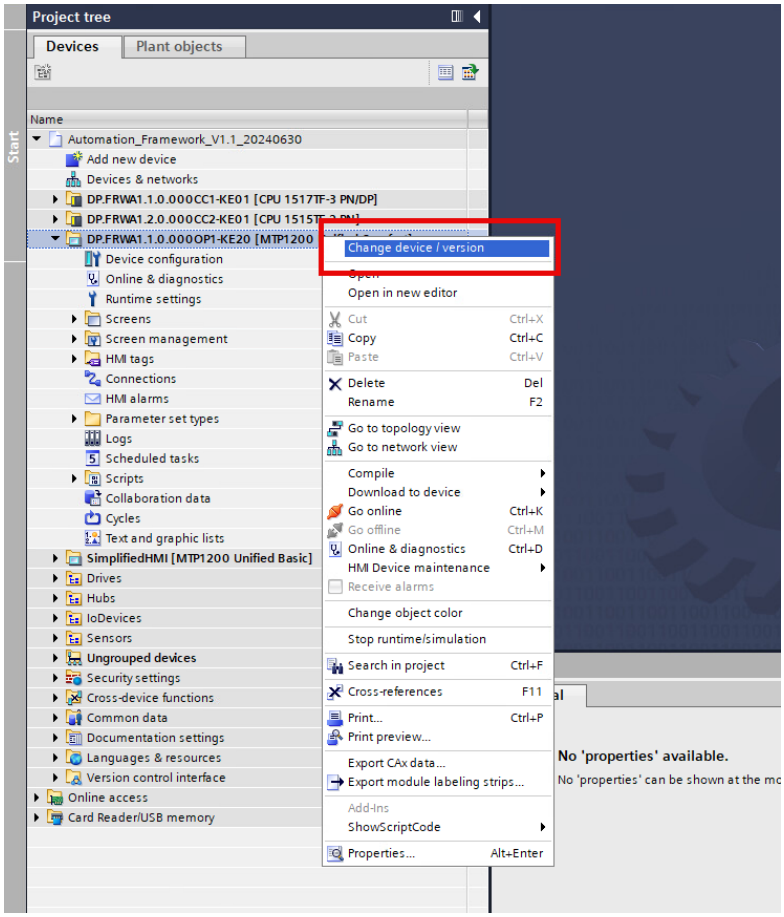


Figure 9-33: Selection of the "Change device/version" property of the HMI.

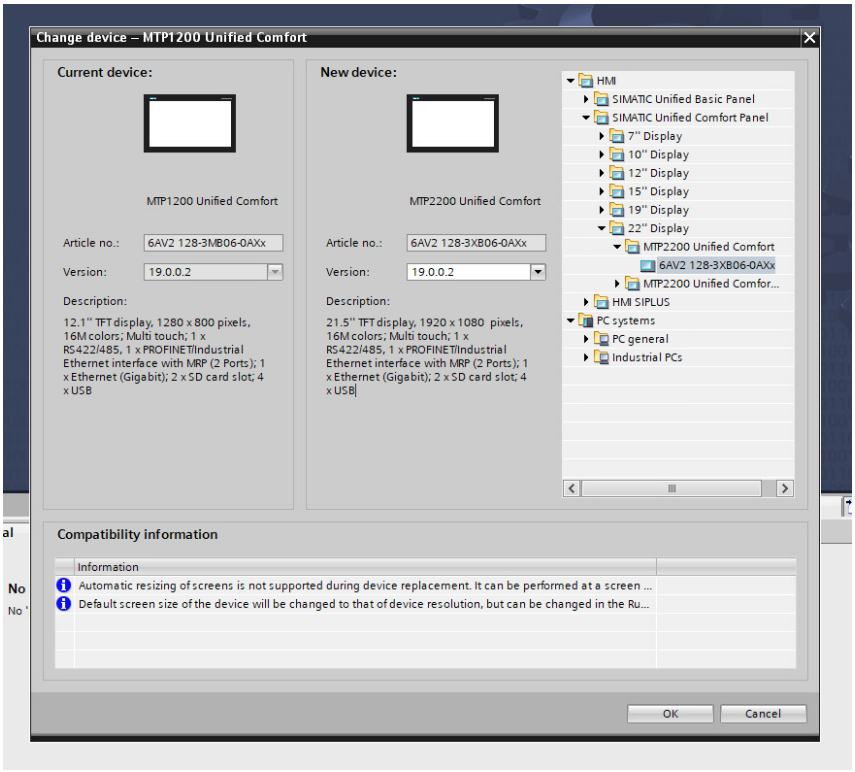


Figure 9-34: Selection of the desire device to make the resize.

2. **Resizing Screens to Match the Display**
   Once the target device is selected, you can proceed to resize the screens to fit this new display size. Follow these steps:

   a. In the project tree, locate the screens you wish to resize.

   b. Right-click on the screen(s) and select the **"Resize to Display"** option from the context menu:

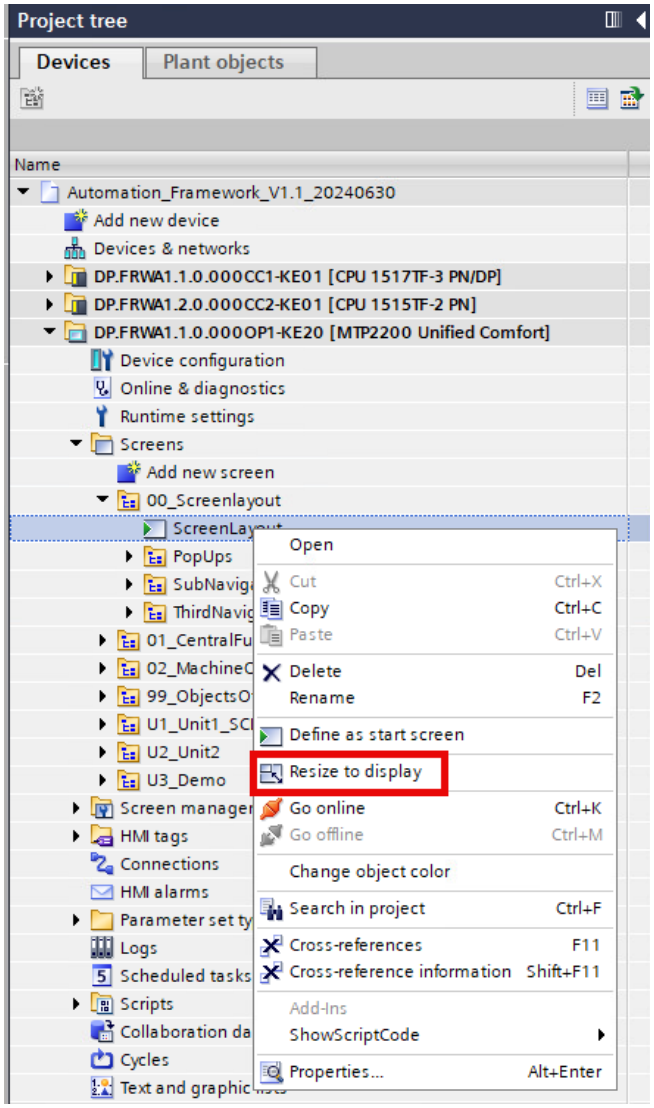


Figure 9-35: "Resize to display" property

This option automatically resizes the screen dimensions to match those of the selected device. Most objects on the screen will be resized appropriately, ensuring a consistent appearance across different resolutions.

3. **Adjusting Faceplate Containers**

   While the screen and most objects will resize correctly, certain elements, such as faceplates, may require additional adjustments:

   a. The containers within the faceplates are resized automatically; however, the faceplate itself may not adjust in size.

   b. To resize the contents within a faceplate, navigate to the properties panel of the faceplate.

   c. Locate the option labeled **"Fit Screen to Window"** and enable it. This will ensure that the contents of the faceplate scale correctly within the resized screen.

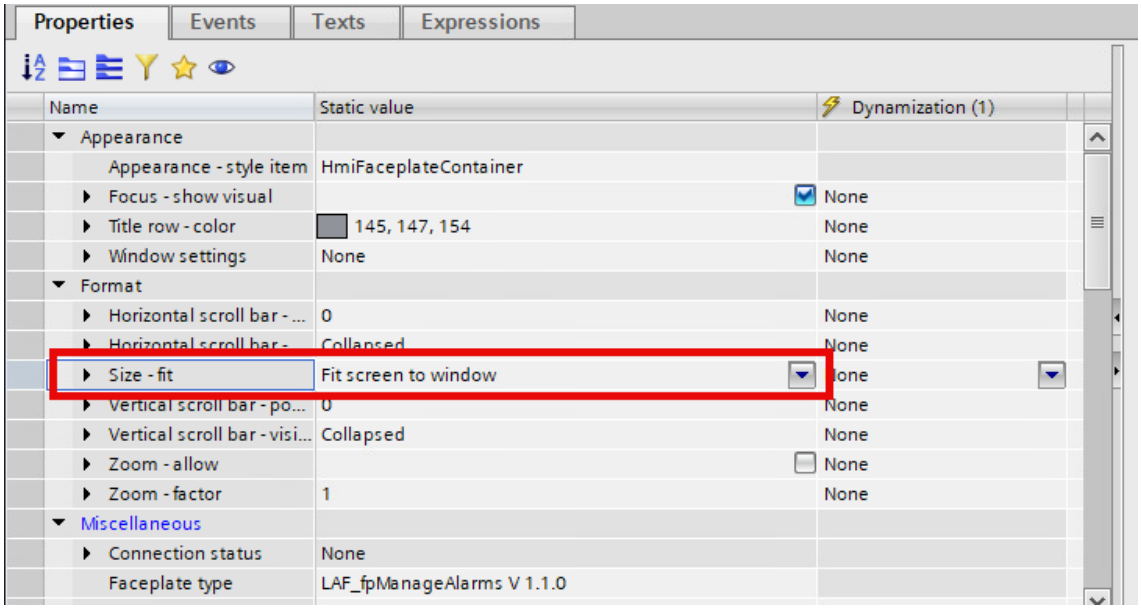


Figure 9-36: "Fit to screen window" property in the faceplate's properties.

4. **Managing Static Size Values**

   Some objects or elements may have static size values defined, which could prevent them from resizing automatically. A common example is a navigation bar with a fixed width:

   a. Identify any objects with static size values, such as width or height, particularly in navigational elements.

   b. Remove these static values to allow the object to resize dynamically according to the screen dimensions.

   c. Once the static values are removed, the objects should automatically adjust to fit the new screen size.

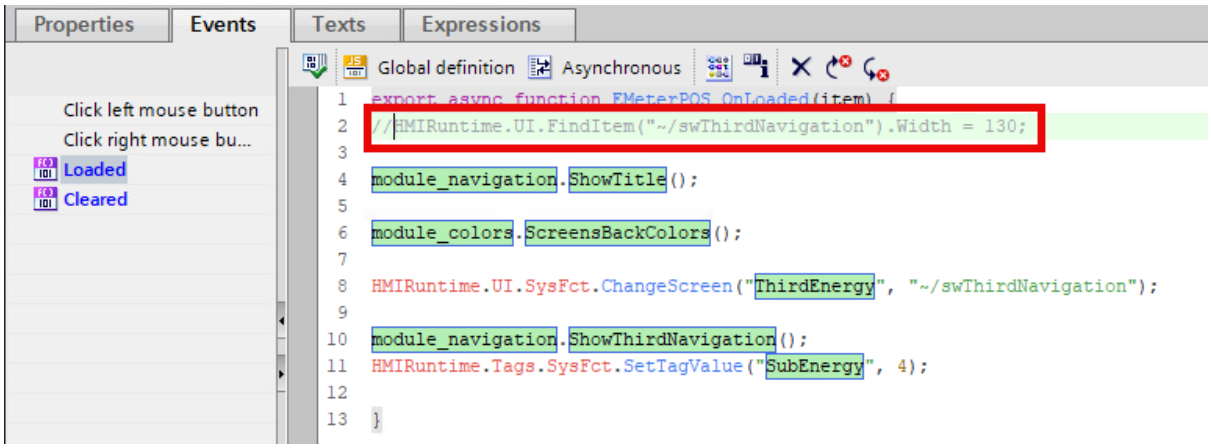


Figure 9-37: Static values removed in the scripts.

5. **Final Adjustments and Testing**

   After completing the resize operations, it is recommended to review and test the screens on the target device. Ensure that all objects, text, and navigational elements are appropriately scaled, and that the user interface remains functional and visually appealing.

10. Equipment modules

10.1. General structure

An EM always consists of exactly one Software Unit and is part of the structure of the command and data flow as well as the HMI connection concept presented in chapter PLC Software Architecture.

The EMs are including specific CMs of the machine which execute the functionality the equipment module is specifying.

Main program structure

- On the EM level, the current linking status of the module itself to the Unit is first evaluated. What linking and unlinking one EM from the Unit means will be discussed in the chapter Interlink .
- Next, the control module blocks representing the sensors are processed to obtain the status of the process inputs of the equipment module.
- The EM next maps the different statuses from the Unit to different behaviours of the user programm or sequence, which should provide the main functionality of the module. Therefore the functionalities can be devided into the job control concept, which will be further presented in the chapter Job control.
- The general evaluation of whether and when the CMs, that represent actuators, are reset and activated takes place before the CMs are called.
- The feedback of the completion of the current state and the commands to the Unit will then be built.
- Finally, the module's alarms are processed

Signal Flow

The Figure 10-1: Signal flow in the equipment module. shows the interconnection of the signals (commands and status) within an EM.

General mode and state are coming from the unit level. This determines if the equipment level runs in automatic mode, maintenance, manual or any other user defined mode. If manual mode is enabled, manual operation of the control modules, the job selection and the sequence is allowed via the HMI. DB "HmiInterface" is connected to the according faceplates in the HMI screens. The provided example sequence allows a manual continuous operation as well as step-by-step mode.

The Mode and State of the unit are evaluated via the FC "LUC_InterlinkToParent" and "LAF_ControlEm". Depending on the linking state of the equipment module, either the unit signals are used within the EM or if unlinked the HmiInterface or ControlNodes signals are used. The code of the FB "CallEquipment" is highly customer specific. The shown implementations are just examples. In the AF, multiple EMs with different ways of implementations are shown. I.e. different sequences can be called depending on the recipe or job. Sequences can be programmed in LAD/FBD, SCL and GRAPH.

The sequence communicates with the control modules using the central data block: DB "ControlNodes". This has the benefit of high transparency and flexibility. It is advisable to limit the number of direct connections between control module blocks and the sequence and instead utilize the DB "ControlNodes" as a centralized data hub for process commands and status. Any customer specific control module blocks can be programmed in a way, so that the commands and status are placed in the DB "ControlNodes".

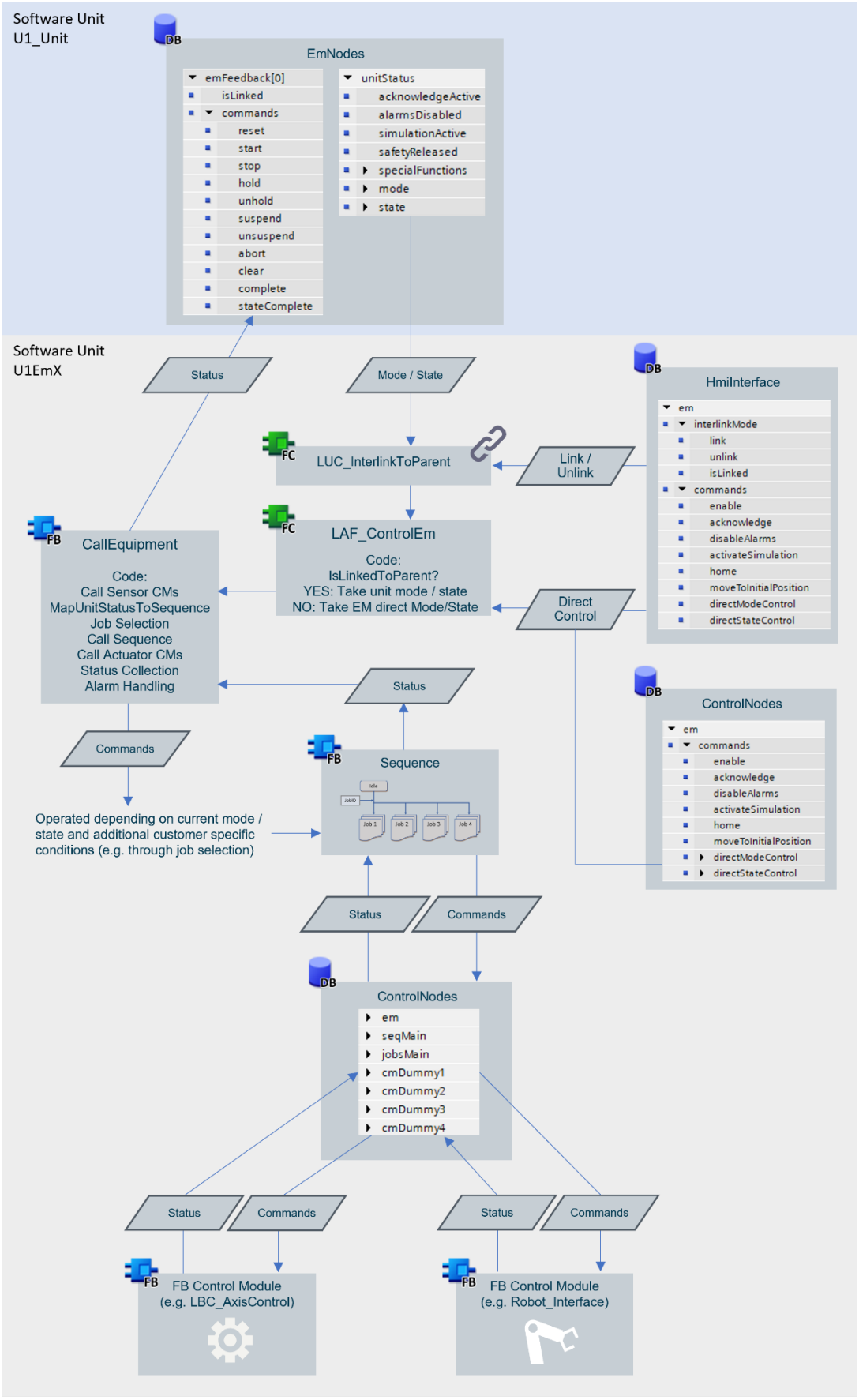


Figure 10-1: Signal flow in the equipment module.

10.2. Interfaces and blocks

The main responsibilities of the equipment modules include:

- embedding the process logic into the system structure
- interface to the hardware level

To achieve these functionalities, the following settings and blocks are available in the Software Unit that represents an equipment modules:

Relations

Every EM needs a relation to the ObjectsOfProjectLibraryUsed and the associated unit. Furthermore, the EM relations could include technology objects which are handled within the module. The figure illustrated below shows the Demo EM High Performance Motion which includes the discussed relations.

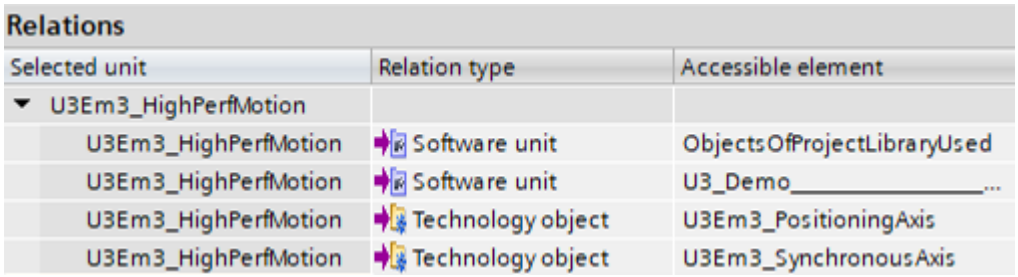


Figure 10-2: Relations of an equipment module.

Program blocks

Also at the equipment module level, the general program architecture is based on the description in the chapter Program structure. In order to clarify how the individual blocks work at the equipment module level, the following section will take a closer look at the individual program blocks of an EM.

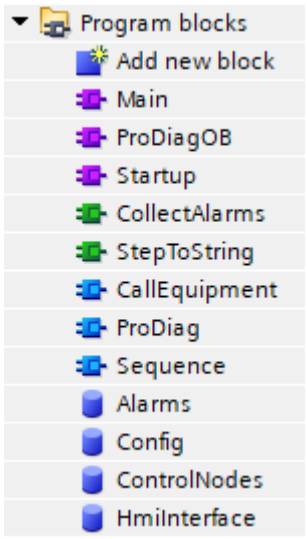


Figure 10-3: Program blocks of the equipment module.

OB Startup (Em)

Within the OB "Startup" of the EMs the configuration of the reference designator and Axis data are set.

FC StepToString

The function "StepToString" converts the integer value of the sequence steps into string variables. That increases the readability. The function must be adjusted for each sequence according to the defined steps and step names.

FB Sequence

The main purpose of the EM level is to integrate the process logic. The FB "Sequence" hosts the sequential logic of the module.

DB Config (EM)

The DB "Config" contains all configuration data of the EM and underlaying modules. That includes the configuration data of the em reference designator, axis and control modules. Configuration is usually set once during commissioning and not changed during operation of the machine. By saving this DB (e.g. export), the configuration of the EM has been stored and can be easily imported in a new PLC.

DB ControlNodes (EM)

The DB "ControlNodes" hosts the runtime data of the EM itself, the sequence as well as the job data and variables to control and monitor the CMs.

DB HmiInterface

At the EM level, all commands to and from the HMI are passed through the DB "HmiInterface". It contains the same objects as stored in the DB "ControlNodes" but provides the HMI interface values for these objects.

FB CallEquipment performs the followings actions

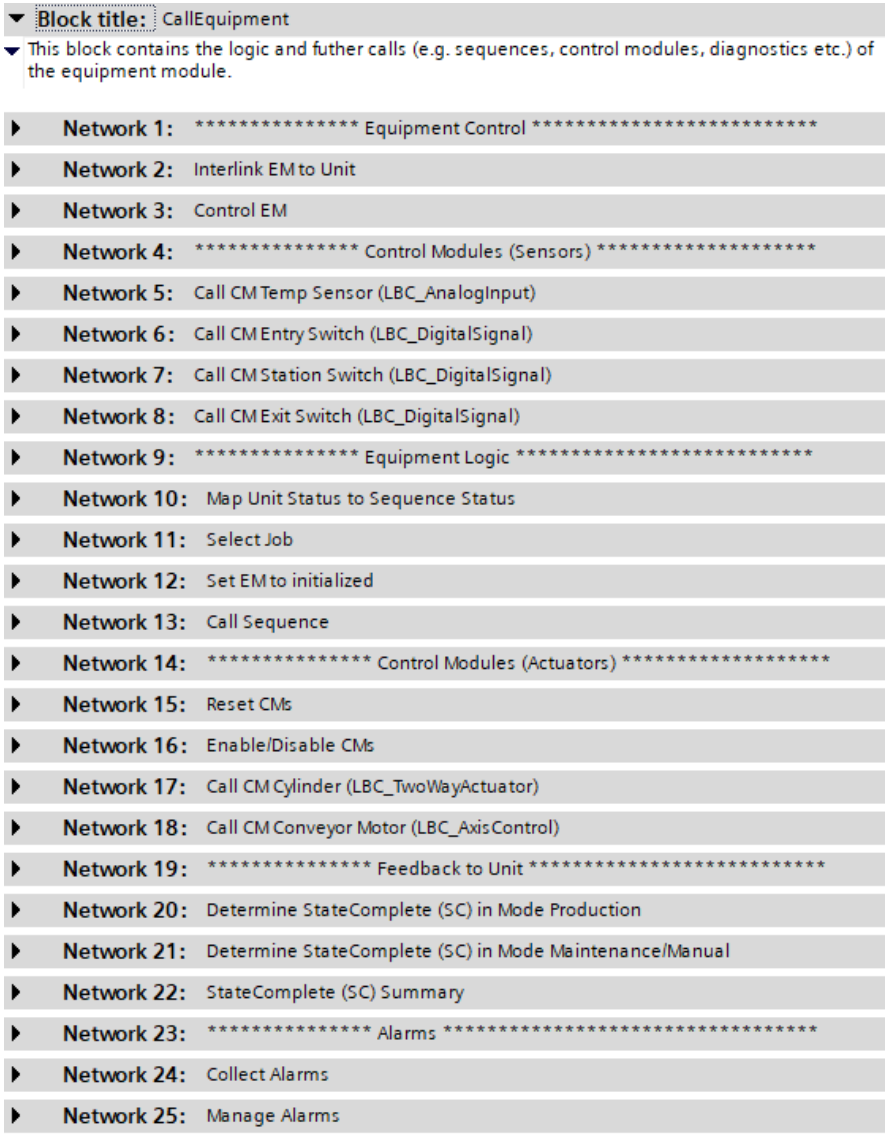


Figure 10-4: Content of FB "CallEquipment" based on the example of Em2 Conveyor Station.

Interlink EM to Unit / Control EM

Before using the current unit mode and state, the EM evaluates if it is currently linked to the Unit or if an unlink command has been set on the HMI. This is handled by the "LUC_InterlinkToParent" block. If the EM is unlinked, the Unit Mode and State can be simulated with the "LAF_ControlEm" block.

Control Modules (Sensors)

These networks are calling the CMs that evaluate the sensor values of the equipment module. The sensors and actuators are separated to always have the newest values within the PLC cycle. The actual input values are evaluated, then processed in the program and used to update the CMs of the actuators

Map Unit Status to Sequence Status

This network calls a function to set preconditions for the jobs and the sequencer of the equipment module based on the current unit mode and state. For instance, when the manual mode of the unit is enabled, the job and sequence management are set to manual mode as well. Additionally, the block has input bits to define the behavior of the job management within different unit states. Each input therefore gives the possibility to configure if the current job should be interrupted while the unit changes to the specific state or not. If the input signal is true, the job will be interrupted if the specific unit state is activated.

Select Job

To separate the functionality of the EM a Job Management is added. Via several "LAF_SelectJob" blocks a function is then triggered during runtime. A detailed explanation is following in the chapter Job control.

Set EM to initialized

This network is used to show an example of how the status variables of the equipment module could be processed. There are also additional status bits which could be processed within this network or in additional networks.
In this example here, the equipment module is set to initialized, if the initialization job has been completed. Other jobs could take that status as precondition for their own execution.

Call Sequence

After the evaluation of the job which should be executed the Sequence is then called. In this case a S7-Graph sequence. The process of the step sequence is structured according to the job management.

Reset CMs / Enable/Disable CMs

The general evaluation of whether and when the CMs are reset, enabled and disabled takes place before the CMs are called.

Control Modules (Actuators)

As mentioned after processing the step sequence the output of the actuators are evaluated with the specific CMs.

Determine StateComplete (SC) in Mode Production / Manual / StateComplete (SC) Summary

After the process logic and hardware have been processed, the feedback from the EMs is output. This is done by analyzing whether the conditions are met in production or manual mode to send a State Complete to the unit.

Collect Alarms

Any event can be assigned to an alarm within the collect alarms function. This specific alarm can be monitored by ProDiag and mapped to a reaction for the unit (e.g. abort, stop, hold). More information on the process diagnostic within the AF is given in chapter Diagnostic Concept.

Manage Alarms

This block derives commands from the ProDiag supervision. The ProDiag supervisions contain categories which define the reaction of these supervisions (e.g. abort, stop, hold). These are handed over as commands to the status collector.

10.3. Hmi Screen

The equipment screens are freely programmable by the user and depend on the machine. In this demonstration, simple examples are realized for EM Hopper, EM ConveyorStation and EM HighPerformanceMotion.

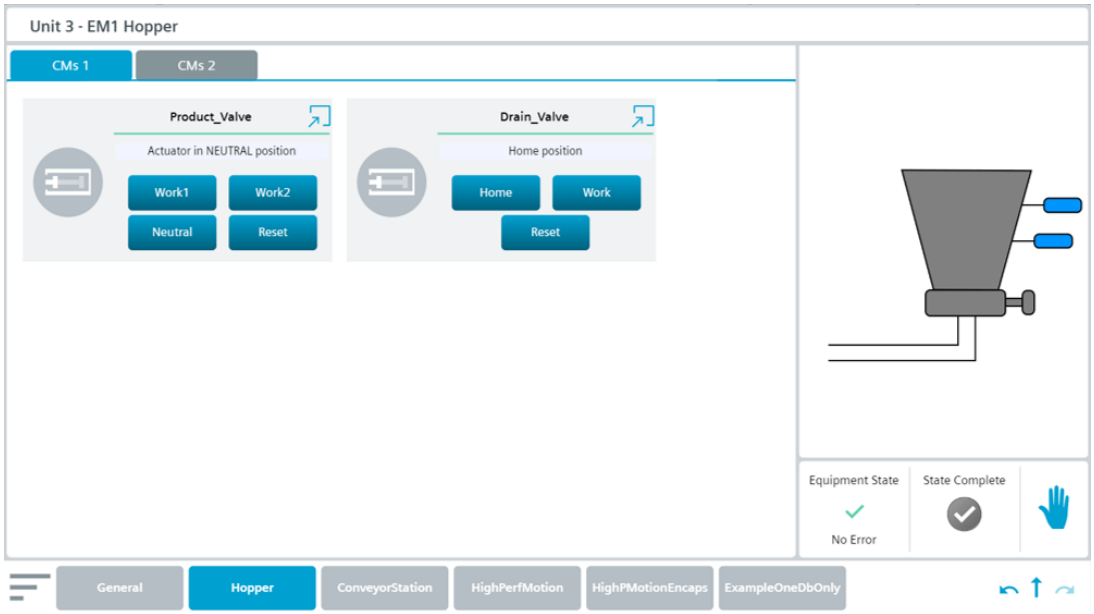


Figure 10-5: Equipment screen of EmHopper.

The equipment screen contains the following faceplates and therefore is able to manually control and monitor:

- jobs
- sequence
- the linking state and if the EM is unlinked, the simulated unit mode and state
- conrol modules

In this example, faceplates of the LBC library are used. All faceplates are enabled when the unit state machine is in manual mode. The faceplates for controlling the sequence, jobs and linking state can be accessed by pressing the button with the image of a hand in the bottom right corner. A popup will be opened, which is shown in the following figure:

Figure 10-6: Manual Control Faceplates of the EM.

10.4. Interlink to unit

The AF has the possibility to link and unlink equipment modules from the assigned Unit. This can be useful during commission, when not all EMs should be integrated yet. Another use case is, that some machine could have redundant EMs and the machine offers the possibility to take some EMs off, for maintenance or due to lower needed throughput, e.g. a redundant infeed system.

If an EM is linked to the unit, the EM applies the unit mode and state and follows the unit.
If an EM is not linked to the unit, the EM does not follow the unit and doesn't apply the unit mode and state. It is disconnected from the unit and can be operated directly. If the EM if unlinked, the unit will not take the EM feedback into consideration. The unit ignores the EM.

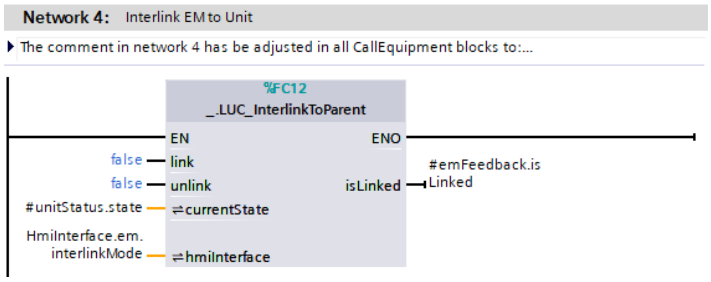


Figure 10-7: Interlink functions of an EM.

The function "LUC_InterlinkToParent" allows the EM to link or unlink from the unit internally using the inputs "link" and "unlink". Additionally, the operator can also establish or remove the connection through the hmiInterface. The process of linking or unlinking is only possible when the parent, the Unit that contains the respective equipment module, is in the active state of Aborted, Clearing or Stopped.

Link an Equipment Module to a Unit:

Connect a logic to the input link, or send a link command from the HMI, when manual mode is enabled.

Unlink an Equipment Module from a Unit:

Connect a logic to the input unlink, or send a unlink command from the HMI, when manual mode is enabled.

10.5. Direct control

The function "LAF_ControlEM" evaluates if the EM is linked to the parent. If it is linked, it takes the "unitStatus" input and copies the values to the output "unitStatusUsed". That variable is used in the further logic of the EM. So in case the EM is linked, it fully applies the unit status to the EM.

In case the EM is not linked to the unit, only acknowledge and safetyReleased from the unitStatus are applied to the EM. The mode and the state can be controlled directly by the "localInterface" or by the operator through the "hmiInterface" InOut. That means that any mode and state can be given to the EM. There is no state machine in the EM, so no automatic mode or state transitions are possible, the order of going through the states is completely flexible.

With the direct EM control, it can be easily tested if the EMs perform the right jobs and tasks in the respective states. E.g., you can apply the state resetting and test the logic executed in that state.

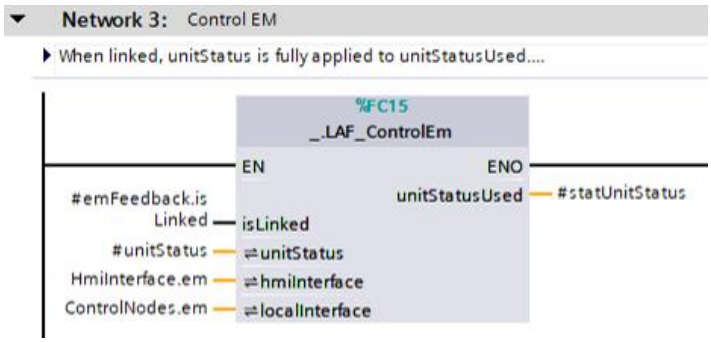


Figure 10-8: Direct control function of an EM.

10.6. EM – EM link

As the EMs of a unit are interdependent on the process, it is possible to implement an internal signal exchange of the EMs. The DB "EmNodes" of the unit serves as the central data point here. The interfaces between the individual EMs can be integrated into the emRelations data structure. The Figure below shows a demo implementation of the emRelations.

emRelations	Struct
EM1	Struct
readyToReceive	Bool
readyToDeliver	Bool
EM2	Struct
EM3	Struct

Figure 10-9: Demo Implementation of the EM Relation signals.

This example Struct is used to transfer signals from an EM which is placed in the middle of two other EMs. It receives parts from the first Em and delivers them to the second one. Depending on the state it sets the "readyToDeliver" or "readyToReceive" bits. The neighbor EMs can therefore react on these bits and deliver or receive parts.

The Em signal exchange can be used to sequentially execute different functions of the EMs. This interconnection can be used to achieve a process of the unit.

10.7. Job control

To achieve a modular behavior of the EM, the different functionalities that could be provided by the equipment are separated into different Jobs. A job could be e.g. the initialization of the EM or a part of the production process, such as the transport of a produced object towards the next EM. The Handling of this separation is made by the "LAF_SelectJob" function block. It provides the control of one job that the EM could execute. Therefore, every job that was defined for the EM must have its "LAF_SelectJob" instance and a specific number in the user constants of the EM.

By activating the Job with the "startCondition" Input of the "LAF_SelectJob" block the assigned number will be transferred to the Sequence of the EM and the branch of the Job will start. Every Job then includes all the steps to fulfill the specific function that should be executed. Therefore, the CM interaction will be happening in these steps of the different jobs.

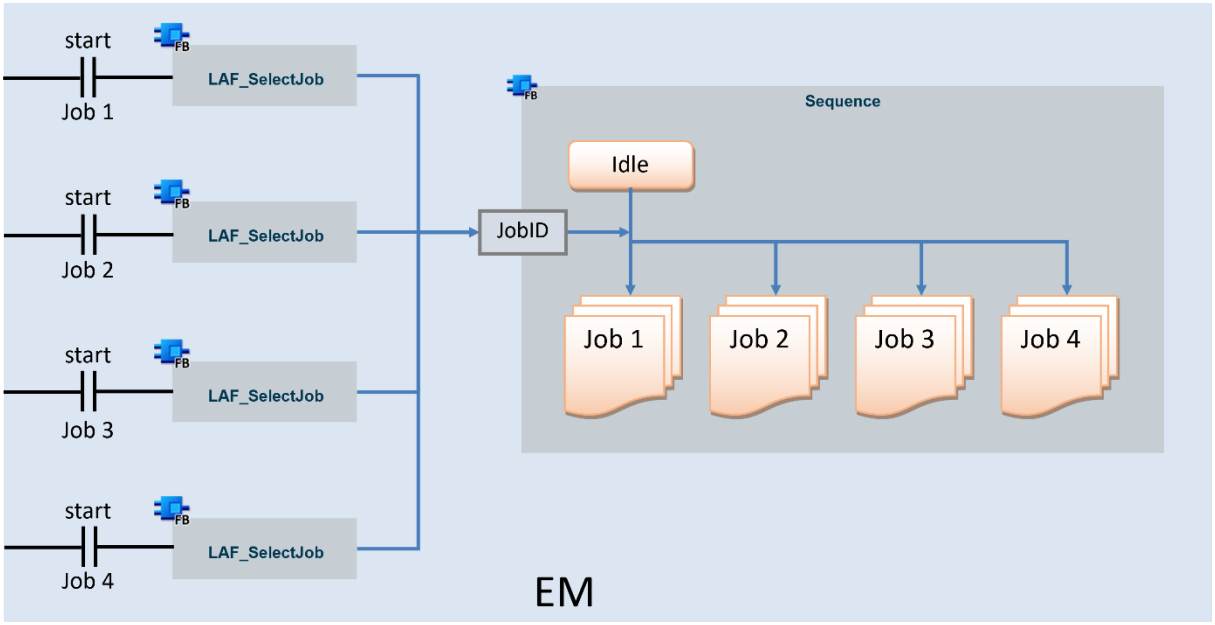


Figure 10-10: Job Management of an EM.

When using the block, the defined job number must be linked to the instance and it must be defined whether the job is a "singleCycleJob" or not (True = job is executed once when the "startCondition" is fulfilled, False = job is executed cyclically when the conditions at the "startCondition" input are fulfilled). Additionally, the "parentReferenceDesignator" of the EM could be connected to include the blocks into the diagnostic concept of the AF.

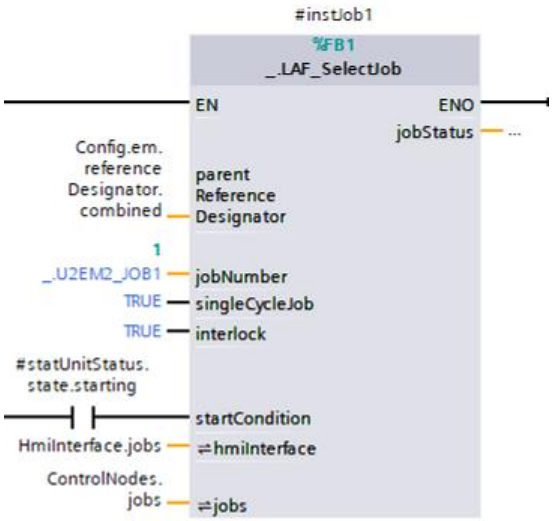


Figure 10-11: Call of the FB "LAF_SelectJob".

The "interlock" input prevents the execution of the specific job if the conditions are not fulfilled. Therefore, the Input must always remain "True" to be able to execute the job. As already mentioned, the "startCondition" input defines the conditions that must be fulfilled to start the job.

The "hmiInterface" InOut should contain the necessary data to establish a connection between the block and the HMI for control purposes. The "jobs" InOut will then transfer the data to the specific points within the PLC to share the jobs information. In addition, the current status of the job is available in the output signal "jobStatus". It contains the "LAF_typeJobStatus" UDT that provides information on whether the job is locked, active, completed or if it was aborted while execution. These signals therefore can be used to be linked to further processing, e.g. to feedback the "stateComplete" condition for specific unit states.

If the current unit mode is maintenance or manual mode, the "LAF_SelectJob" blocks will also change to a manual behavior. The block now accepts the job number via the "hmiInterface" and forwards the information to the "jobs" interface. The status of the jobs will still be processed and can be monitored.

10.8. Sequencers

There are three different types of sequencers available in the AF. Depending on the requirements, a S7-GRAPH, LAD or SCL sequence can be chosen. All sequencers are implemented with at least one template and one example implementation. The exact difference between the implementations is explained in the next chapter. The Implementations can be found within the Software Units that are shown below.

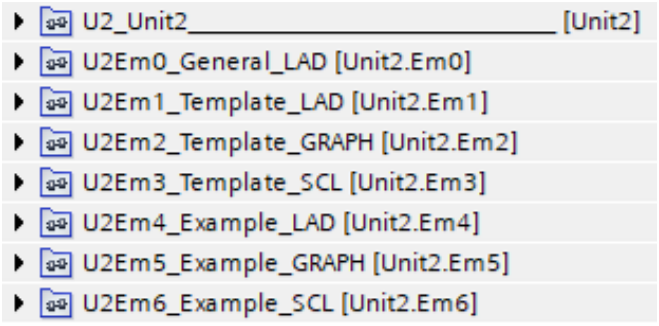


Figure 10-12: Software Units that contain the Template and Example EMs.

The structure of the different equipment modules is all based on the explanation given earlier in this chapter. The only difference is the Sequencer and the Jobs. Each sequencer follows the concept introduced in the Job control chapter, where the individual functions of the EM are divided into jobs. In order to provide the general function of the sequences and to implement the architecture without further programming effort at the EM level, the Automation Framework offers the possibility to use a control and a monitor FB for each of the three programming languages mentioned before.

S7-GRAPH

Within the function block of the S7-Graph Sequencer, the two FBs that are provided are linking the sequence control variables to the internal commands of the Graph FB, called "RT_DATA.MOP". The first function "LAF_ControlGraphSequence" is added to the permanent pre-instruction network and sets the sequence mode, initializes, holds and changes the sequencer to single step mode. Additionally, it is evaluated if the current Job was cancelled, which will be shown at the "resetJobs" output and can be used to reset the status bits of the "LAF_SelectJob" blocks. The following figure shows the call of the block with the connections that must be made to map the signals of the DB "HmiInterface" an "ControlNodes" to the function block and therefore to the "rtDataMop" InOut.

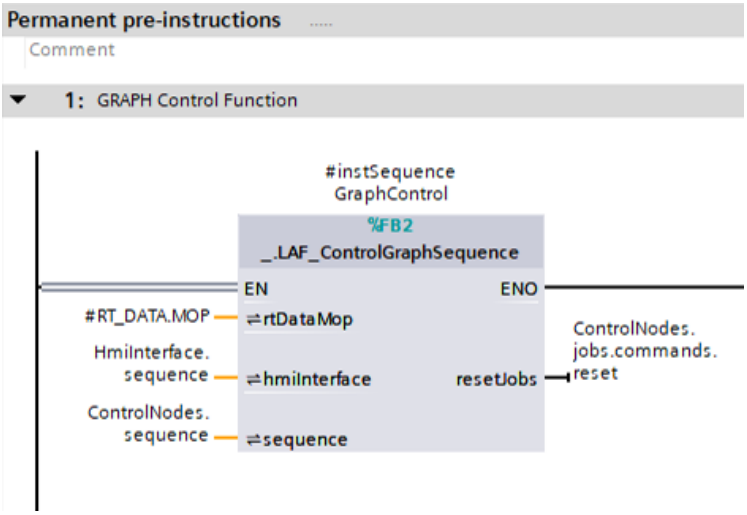


Figure 10-13: Call of the FB "LAF_ControlGraphSequence".

The second function block "LAF_StatusGraphSequence" is called in the permanent post-instruction section and sets the current state of the sequence. Therefore, the internal Graph Signals are used to be mapped to the DB "HmiInterface" and "ControlNodes" UDT of the sequence. Also, the "setJobNumber" input value is used to set the active Job within the Interface signals. In order to evaluate whether the sequencer is in the required steps of Idle or Stopped, the internal S7-Graph variables of the steps have to be assigned. To include the block in the diagnostic concept of the AF, the combined reference designator of the EM could be connected to the "parentReferenceDesignator" input of the block. The call and the corresponding interface assignment are shown in the illustration.

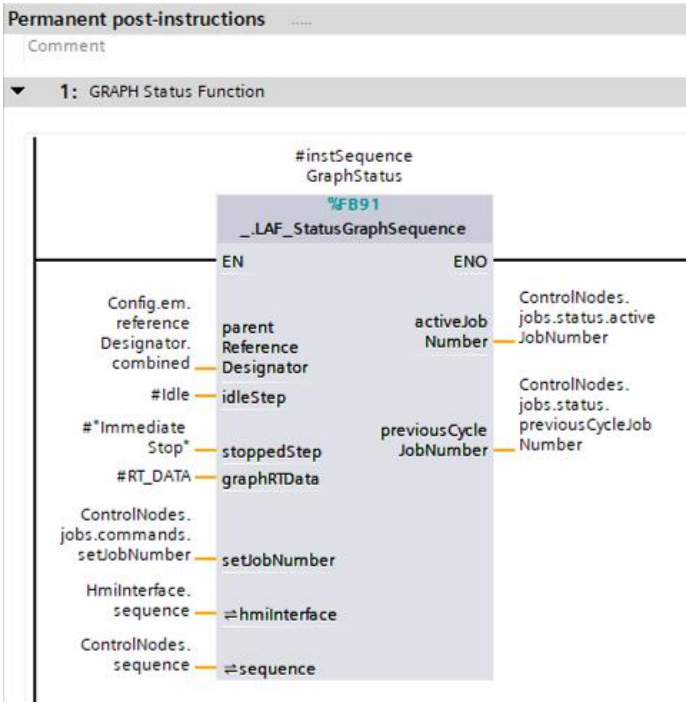


Figure 10-14: Call of the FB "LAF_StatusGraphSequence".

LAD

The difference of the function blocks "LAF_ControlLadSequence" and "LAF_StatusLadSequence" to the ones used in the Graph Sequence is, that there is no necessity to map the DB "HmiInterface" and "ControlNodes" signals to internal data structures of the sequencer. This in turn has the characteristic that the Idle and Stopped steps must also be specified within the Interface of the " LAF_ControlLadSequence". The rest of the functionality and behavior is similar to the handling of the graph sequence. The correct connection of the two blocks is shown in the diagram below.

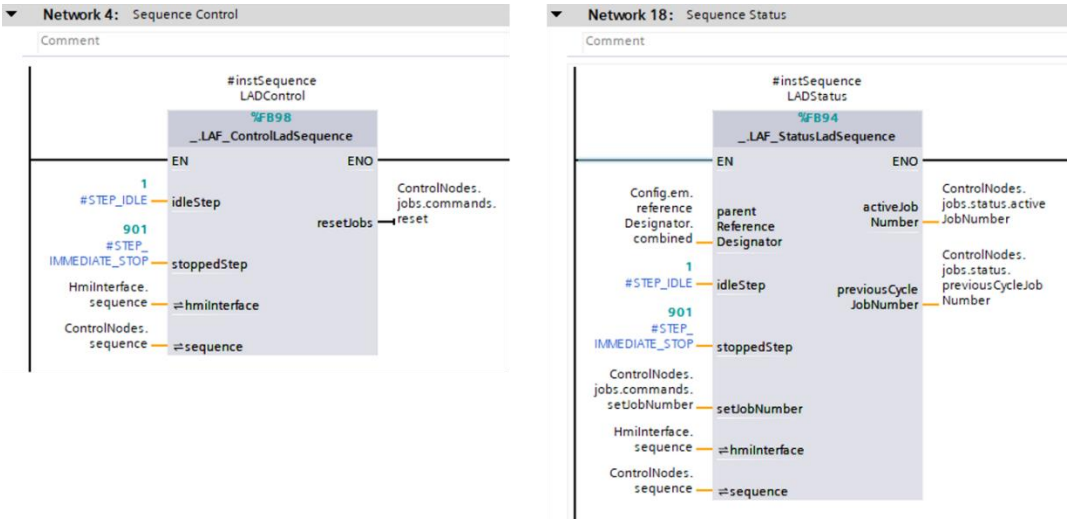


Figure 10-15: Call of the FBs "LAF_ControlLadSequence" and "LAF_StatusLadSequence".

SCL

The concept of sequence and job handling also covers the SCL programming language. The implementations of LAD and SCL only differ in a few internal details in the control and monitor blocks. For SCL, the Idle and Stopped steps must also be assigned to the FBs "LAF_ControlSclSequence " and "LAF_StatusSclSequence ". The interface connection for SCL is therefore the same as the one shown for the LAD blocks.

Post Processing

The data structure of the Job control and the provided sequencers allow to include additional information, which is not already handled by the sequence blocks. Therefore, the equipment modules of the Automation Framework include two additional networks for the purpose of post-processing status information related to the job and sequence execution. These networks are called below the status blocks of the sequence. The following picture shows the call of the helper functions that provide the additional information.

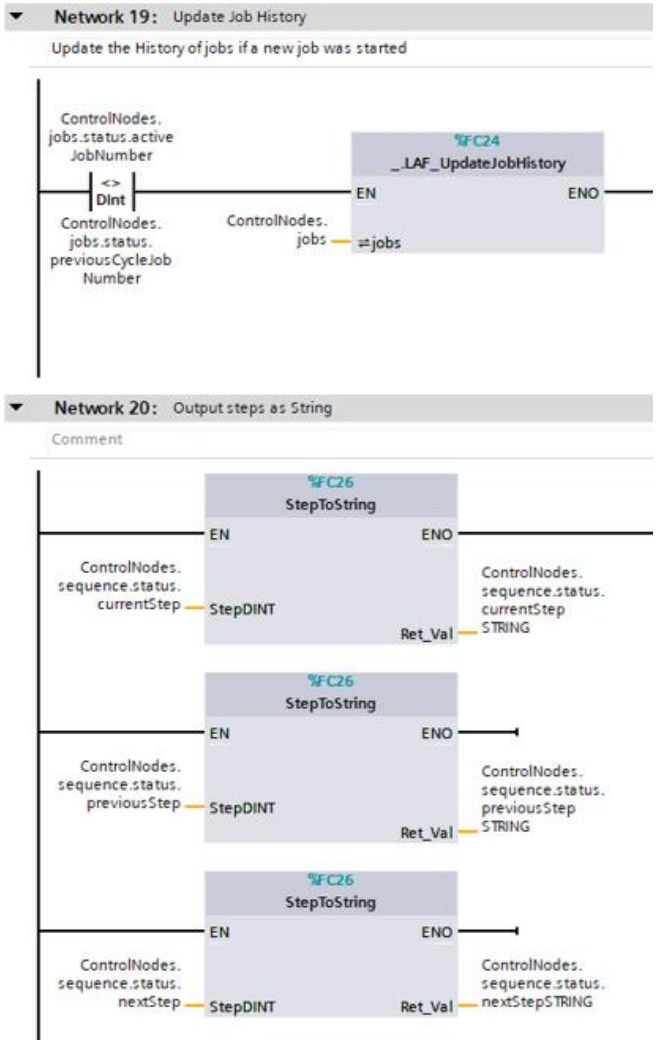


Figure 10-16: Call of the functions for additional information of the jobs and sequence.

The function "LAF_UpdateJobHistory" is used to insert the executed jobs into a history. As shown in the figure the function will be executed, if the active job number is not equal to the job number that was active in the previous cycle, which means the last job was finished. The function "StepToString" just converts the step number into a string. Therefore the block is called for the current, previous and next step.

10.9. EM Template / Example

Technical description

The template or example equipment modules are blueprint EMs that show the main structure and interface and can be used as a base to create own equipment modules. Within the Automation Framework there are different Template equipment modules provided depending on the programming language of the whole unit, the EM and the sequence of the module.

The use case of the two EMs differs as follows:

- **Em_Template**: Contain only the mandatory blocks and calls. They are the base to start own equipment modules. One template for each programming language.
- **Em_Example**: Contain a basic example of an equipment module. Each Em_Example contain four CMs and implements one sequence example. All Em_Examples are designed to execute the same use case such that the user can easy compare the different implemtation in the different programming languages.

Architecture

The basic structure of the EMs (like described in chapter Interfaces and blocks) is also adopted in the structure of the example and template EMs. To get an overview about the differences between the individual EMs the following table is showing their content:

Software Unit	Description
U1_Unit1_SCL	Programmed completely in SCL
U1Em0_General_SCL	Programmed completely in SCL
U1Em0_Template_SCL	Programmed completely in SCL
U1Em1_Example_SCL	Programmed completely in SCL
U2_Unit2	Programmed in LAD
U2Em0_General	Programmed in LAD
U2Em1_Template_LAD	General logic programmed in LAD, sequencer programmed in LAD
U2Em2_Template_GRAPH	General logic programmed in LAD, sequencer programmed in GRAPH
U2Em3_Template_SCL	General logic programmed in LAD, sequencer programmed in SCL
U2Em4_Example_LAD	General logic programmed in LAD, sequencer programmed in LAD
U2Em4_Example_GRAPH	General logic programmed in LAD, sequencer programmed in GRAPH
U2Em4_Example_SCL	General logic programmed in LAD, sequencer programmed in SCL

Table 10-1: Overview about the template and example EMs

The EM Template could be used to start an own individual equipment module while maintaining the structure of the project. In addition to the template EMs, the example EMs provide a more use case specific program with representative CMs and a sequence. The enhanced program of the function block "CallEquipment" is shown in the following figure.

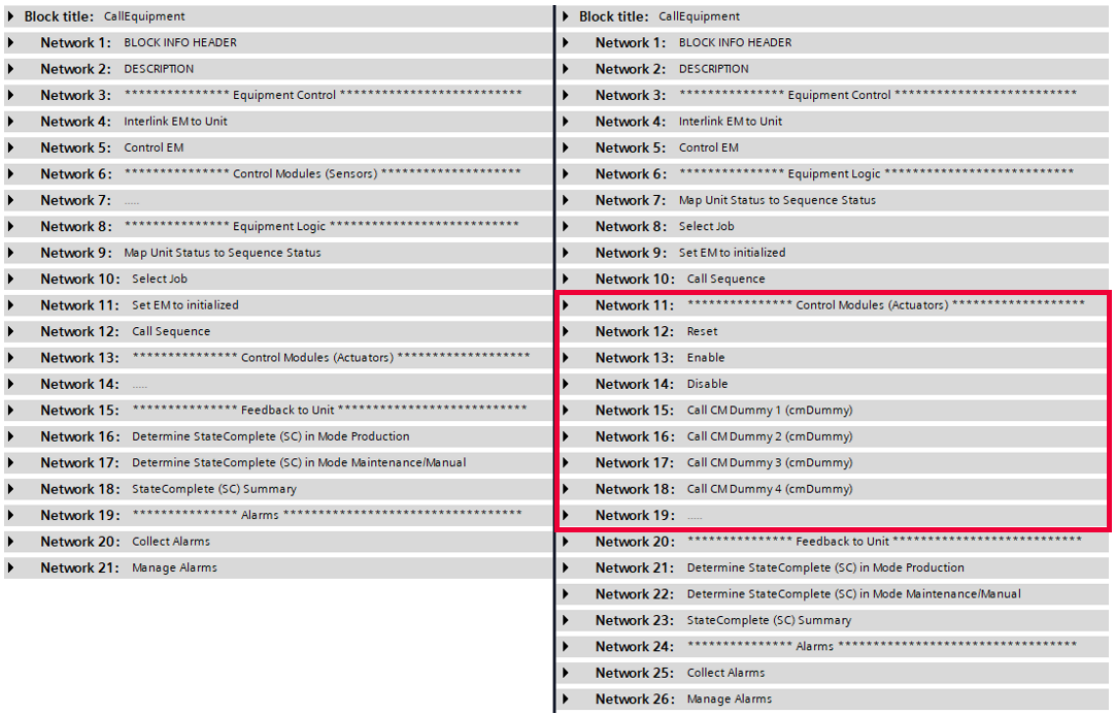


Figure 10-17: Comparison of a Template and Example EM.

The dummy CMs are used in the FB "Sequence" to show how to build a sequence. In the sequence block, the steps of the respective jobs are programmed, as well as the conditions for the individual steps. Each job is divided into different steps. A step represents an action in the process e.g. filling water or move motor. In the example the jobs and the steps are more generic, because of their purpose. A fully programmed sequence can be found in the EMs of unit 3 (chapter EM Demo Implementation – Hopper till EM Demo Implementation – High Performance Motion). The figure below shows the generic structure of an individual step.

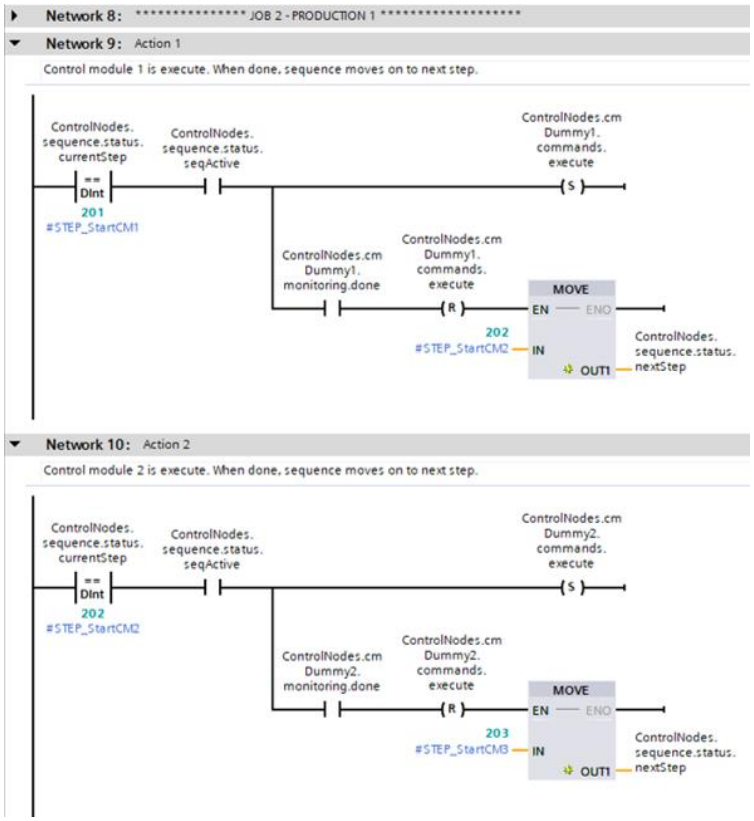


Figure 10-18: Example for generic programming with step structure (Unit2Em4).

Interface function description

The DB "Config" provides a centralized point where the reference designator, axis and CM configuration can be made. Furthermore, the interface of this EM is distributed to the DB "ControlNodes" as well as the DB "HmiInterface". The DB "ControlNodes" hosts the runtime data of the EM itself, the sequence as well as the job data and variables to control and monitor the CMs. At the EM level, all commands to and from the HMI are passed through the DB "HmiInterface". It contains the same objects as stored in the DB "ControlNodes" but provides the HMI interface values for these objects.

10.10. EM General

Technical description

The equipment module contains General Functions, checks and logics that is needed in the entire unit. It can include general releases and system wide functionalities, like air pressure and power handling as well as reactions to events.

Architecture

The architecture of the EM General is derived from the general structure, which was already explained at the beginning of this chapter General structure. Because there is no sequential process within this EM, there is no FB "Sequence". The architecture of FB "CallEquipment" also follows the generally presented one. Within this function block the implementation of functionality can be seen in the following points:

- **Interlink**; the EM General always remains linked to the Unit.
- **General Releases**; Includes information of general fuctions like the feedback of circuit breakers or power supplies are evaluated.
- **Control modules**; Different CMs like the stack lights are part of the general functions.

Interface function description

The interface of this EM is distributed to the DB "ControlNodes" as well as the DB "HmiInterface". The DB "ControlNodes" provides signals regarding the status of the EM, including interlinkMode, commands to the EM and status information. There are also variables that represent the status of the CMs and linked HW that is processed within the general releases. This could be for example fuses or the state of the pressurized air.

The DB "HmiInterface" only contains the EM information, but extended by the alarm status of the EM as well as the UDT to achieve a direct EM control by means of a simulated unit status.

10.11. EM Demo Implementation – Hopper

The equipment module "Hopper" is an example to demonstrate how to implement a hopper application. It is not fully programmed in all detail. It demonstrates how to implement and manage a more complex control module by creating a technical module (TM).

The example implementation contains the following types of control modules:

- "LBC_AnalogScaleCn" is inserted to implement a load cell example.

Additionally, the technical module "TM Aspiration" contains:

- "LBC_AnalogInput" to measure the liquid level of the tank.
- "LBC_PT1-Filter" to filter the measured filling level.
- "LBC_AxisControl_TectPlcCn" to control the speed of the mixer.
- "LBC_ThreeWayActuator" is implemented for the outlet of the hopper.
- "LBC_TwoWayActuator" is added to control a drain valve that opens and closes to let remaining amount out of the hopper.

Technical description

By implementing the equipment module, the OMAC state model is separated from the EM level. To create a modular behavior for the EM, various functionalities that the equipment could offer are divided into separate Jobs. A job could be the starting of the EM or a subprocess of the production, such as the discharging of the remaining quantity before a new loading and production cycle can start. In these example five jobs where defined, the details about the individual jobs can be found below.

In the starting job the hopper is emptied, since there may be some residue from the previous production batch. The remains are discharged through the drain valve (two-way actuator). The next job describes how a tank is filled with water via a three-way actuator, after the starting process. The axis is then set in motion and the valve is returned to the neutral position. The substance is then added manually from above into the hopper. This is followed by the production job, in which the production timer runs down. Followed by the discharge job. In this job, the outlet of the production valve (three-way actuator) is opened. When the hopper is empty, the axis is stopped. The process can now be restarted.

To establish a connection between the OMAC state model and the EM it is necessary to link the states of the unit with the different jobs of the EM. For example, the OMAC starting state would initiate the EM's initialization job. Furthermore, the ending of the initialization job would update the EM confirmation bit to report the state back to the unit level.

Architecture

The EM Sequencer architecture allows customers to link different unit variations to the equipment module. It is structured, based on the template equipment module's architecture, encompassing all functionalities within the FB "CallEquipment". The function block establishes the connection between unit states and the sequence. Additional networks are included to manage the selection of the sequence mode, the job, and the conditions for stopping the sequence and returning it to its initial state. If no job is chosen, the step sequencer will stay in the idle step.

The EM Hopper contains the following Jobs:

- Starting
  - Triggered by the OMAC Starting State
  - Empty the hopper before next job starts (in case that there are any remains)
  - Actuators are set to neutral
- Loading
  - Triggered by the OMAC Executing State
  - Fillling with liquids starts and mixer starts moving
  - Manual filling is starting
- Production
  - Triggered when the previous job is done
  - Mixer is moving
  - Mixing timer is running
- Discharge
  - Will be then executed after the Production job is finished
  - Mixer is moving
  - Production valve is in work position to enable the emtying process
- Ending Production
  - Triggered by the OMAC Stopping State
  - The axis will stop and the actuators are set in a neutral position

The main processes of the equipment module and the individual steps are implemented in the FB "Sequence".

Technical Modules (TM) and Control Modules (CM)

In this EM the structure of a technical module (TM) is used. A TM is a module that contains multiple control modules (CM) and therefore it makes an own module. For the equipment logic however, it behaves similar to a control module as it can be controlled and monitored via the commands and monitoring bits in the DB "ControlNodes". This structure enables central access.

In this example, a control module "cmAnalogScale" and a technical module "tmAspiration" are implemented.

The control module "cmAnalogScale", which represents a load cell, is called directly in FB "CallEquipment. In this example the module is not used in the simulation. It can be added in a specific use case.

For the more complex module, a technical module called "TmAspiration" is defined. It contains the functionality which was listed in the upper part, the LBC blocks are used for that purpose. The "TmAspiration" is an example how to create user defined complex modules. Following the ISA-88 standard, the possibility to have a control module in a control module is used.

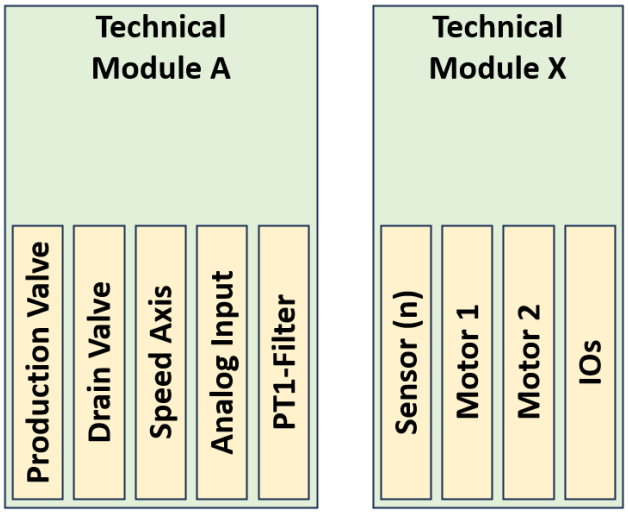


Figure 10-19: Concept of a technical module.

Interface function description

The Interface DBs contain an EM control UDT that includes "interlinkMode" commands to the EM, status information, simulated Unit Status for the HMI part, and current alarm status. Additionally, there is a job control UDT responsible for managing different jobs. This UDT includes the selected job number, the currently active job number, and indicates whether the job management is in manual mode or if a job gets canceled. Moreover, the DB "ControlNodes" stores a history of executed jobs. The main interface of the step sequencer is found in the DB "ControlNodes" and the DB "HmiInterface". The HMI section includes mode handling, commands, and monitoring signals. Within the DB "ControlNodes", the interface consists out of the mode handling, the commands and the status, along with a history of previously executed steps.

The CMs store their interface data in both: the DB "ControlNodes" and DB "HmiInterface".

10.12. EM Demo Implementation – Conveyor

The equipment module "Conveyor Station" is an implementation to demonstrate how to integrate a production sequence with a S7-Graph block. The main intention is to define a modular way of integrating Graph Sequences into an existing unit control model. Therefore, the implemented EM interface can be connected to different types of higher-level control variants.

The example implementation contains four different types of control modules.

- "LBC_AxisControl" function block to control a conveyor belt
- "LBC_TwoWayActuator" is added to simulate a cylinder that opens and closes a heating station
- "LBC_DigitalSignal" (3 times) for proximity switches
- "LBC_AnalogInput" to simulate the position and temperature tracking of the heating station.

Technical description

The implementation of the equipment module separates the OMAC state model from the EM level. To achieve a modular behavior of the EM, the different functionalities that could be provided by the equipment are separated into different jobs. A job could be, for example, the initialization of the EM or a part of the production process, such as the transport of a produced object towards the next EM. The link between the OMAC state model and the EM is established by associating the states of the unit with the different jobs of the EM. The OMAC starting state would trigger the initialization job of the EM or the aborting state would set the S7-Graph sequencer to the idle step. The other way round, the finish of the initialization job would set the EM confirmation bit to report to the state to the unit level.

Architecture

The architecture of the EM Sequencer in Graph allows the customer to connect different unit variants to the equipment module provided. It is based on the architecture of the template equipment module with all the different functionalities within the FB "CallEquipment". In addition, the connection between the unit states and the Graph sequencer is made in this function block by providing special Controlling a monitoring blocks for the step sequencer. The EM Conveyor Station contains the following Jobs:

- Starting
  - Triggered by the OMAC Starting State
  - Cylinder to home position and conveyor runs a specific time to empty remaining objects
- Execute Production
  - Triggered if an object arrives at the entry sensor of the station
  - Object will be transported to the station and heated up
- Execute Removal
  - Triggered if an object is at the station sensor
  - Will be then executed after the Execute Production job is finished
  - Object is transported to the Exit Sensor of the station
- Ending Production
  - Triggered by the OMAC Stopping State
  - The axis will be disabled

Interface function description

The main interface of the em itself, the jobs and the step sequencer is contained in the DB "ControlNodes" and the DB "HmiInterface". The Hmi part contains commands and monitoring signals which will be connected to any kind of operating point. The DB "HmiInterface" also contains for the control of the EM a specific UDT, where the interlinkMode could be changed and the current alarm status could be monitored.Within the DB "ControlNodes" the interface consists mostly of the same elements with additional histories of previous jobs or steps that were executed or different enable signals for the jobs and the sequence. In addition, the CMs are also hosting their interface data within the DB "ControlNodes" and DB "HmiInterface".

ControlNodes [Unit3.Em2]

Name	Data type
Static	
em	_.LAF_typeEmControlNode
commands	_.LAF_typeEmCommands
status	_.LAF_typeEmStatus
jobs	_.LAF_typeJobs
enableManualCtrl	Bool
commands	_.LAF_typeJobsCommands
status	_.LAF_typeJobsStatus
previousJobs	Array[0..63] of _.LAF_typeJobPrevious
sequence	_.LAF_typeSequence
enableManualCtrl	Bool
enableManualMod...	Bool
commands	_.LAF_typeSequenceCommands
status	_.LAF_typeSequenceStatus
previousSteps	Array[0..63] of _.LAF_typeSequenceStep
cmCylinder	_.LBC_typeTwoWayActuatorControlNode
cmTempSensor	_.LBC_typeAnalogInputControlNode
cmConveyor	_.LBC_typeAxisControlNode
cmEntrySwitch	_.LBC_typeDigitalSignalControlNode
cmStationSwitch	_.LBC_typeDigitalSignalControlNode
cmExitSwitch	_.LBC_typeDigitalSignalControlNode

HmiInterface [Unit3.Em2]

Name	Data type
Static	
em	_.LAF_typeEmInterface
interlinkMode	_.LUC_typeInterlinkMode
commands	_.LAF_typeEmCommandsInterface
status	_.LAF_typeEmStatus
alarms	_.LAF_typeAlarms
jobs	_.LAF_typeJobsInterface
commands	_.LAF_typeJobsCommands
status	_.LAF_typeJobsStatus
sequence	_.LAF_typeSequenceInterface
commands	_.LAF_typeSequenceCommands
monitoring	_.LAF_typeSequenceStatus
cmCylinder	_.LBC_typeTwoWayActuatorInterface
cmTempSensor	_.LBC_typeAnalogInputInterface
cmConveyor	_.LBC_typeAxisControlInterface
cmEntrySwitch	_.LBC_typeDigitalSignalInterface
cmStationSwitch	_.LBC_typeDigitalSignalInterface
cmExitSwitch	_.LBC_typeDigitalSignalInterface

Figure 10-20: Interfaces of the Conveyor equipment module.

10.13. EM Demo Implementation – High Performance Motion

The equipment module "High Performance Motion" (HPM) is an example how to implement high-prior motion tasks.

For many machines, the motion control program must be executed very fast, meaning in the same cycle as the calculation of the axes themselves. With that, new motion commands can be given instantly, as well as reactions to any motion events. To maximize the use of the CPU, it makes sense to divide the logic into time-critical and non-time-critical tasks. With that, the CPU load is reduced, and a higher number of axes can be operated. Non-motion-time-critical logic can be executed in a slower cycle, where the motion critical logic is calculated in every motion cycle. Non-motion-time-critical logic also includes basic motion logic like enable, reset or homing. Motion-time-critical logic usually contains fast positioning actions or reactions to sensors.

The example implementation contains two different types of control modules with the following three functionalities:

- "LBC_SinaSpeed" to control a normal Speed Axis.
- "LBC_AxisControl" Positioning Axis, to control a Axis in the High Performance Cycle.
- "LBC_AxisControl" Synchronous Axis, to control a synchronized Axis for the Positioning Axis.

Technical description

The Implementation of the High-Performance Motion equipment module follows the same technical implementation as the other demo EMs by separating the OMAC state model from the EM level. The job- and sequence management blocks are implementing this separation.

Architecture

The HPM equipment module is providing two basic jobs to fulfill the functionality of the module.

- Homing
  - Triggered by the OMAC Starting State
  - Positioning and Synchronous Axis are homed
- Production
  - Triggered by the OMAC Execute State
  - Start the Speed Axis handled by the "LBC_SinaSpeed" block
  - Start the High Performance Program

The HPM equipment module is as already mentioned including two instances of the FB "LBC_AxisControl". These function blocks are called as usual in the FB "CallEquipment" within the OB "Main" cycle of the equipment module. Thus, the existing diagnostic and HMI functionalities, which are provided for example by the LBC blocks, continue to be processed within the application in the standard priority of the program.

Within the high-priority motion task, MC_Move commands from the system instructions are used directly. This is the least calculation time consuming way of implementation. A DB called "SyncNode" is used for data exchange between the standard program and the high-performance motion program. The following figure shows the architecture of the data flow between the standard program and the high-performance motion program.

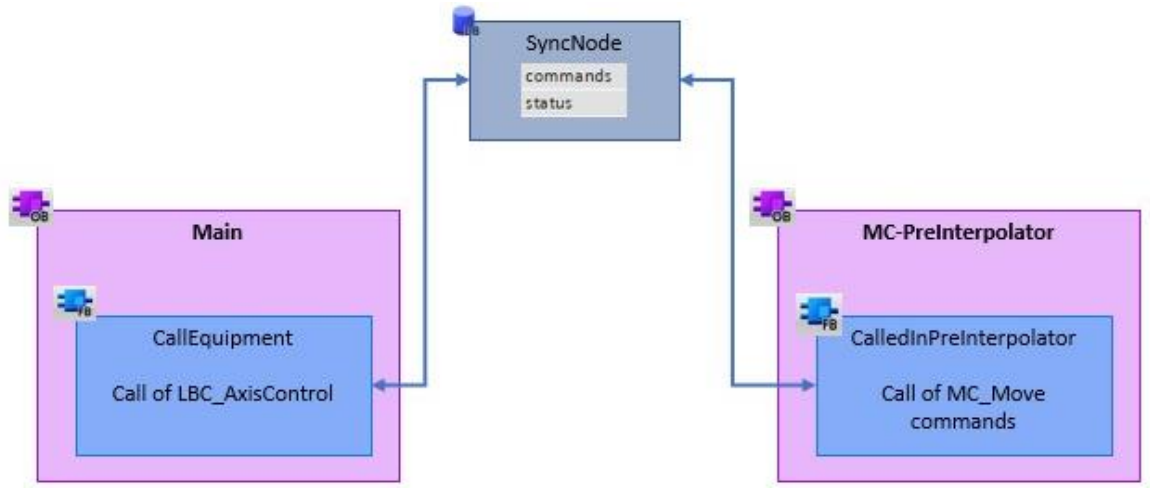


Figure 10-21: Interconnection of standard cycle and motion sync cycle.

Actions that are not critical to performance can be executed as usual in the Sequence of the equipment module, such as enabling or resetting the axes and the reactions to the global operating mode and states. The activation of the part of fast execution in the motion cycle is then triggered in specific steps of the sequencer. For this purpose, a slim interface is used, in this example. The command "enableHpMotion" is send to the sequence in the MotionCycle.

It is important to ensure data consistency from IOs is ensured as well. Therefore, assign the signals that are used in the motion cycle are assigned to the motion process image and evaluated with OB "MC_Servo". In general, it is recommended to perform only the necessary actions in the motion cycle to avoid unnecessary load.

To achieve this architecture and optimize the functionality of the EM, the blocks called in the motion cycle are stored in a special folder "CalledInMotionCycle" in the Software Unit high-performance motion. These blocks are called in the OB "PreInterpolator" in the Software Unit Motion.

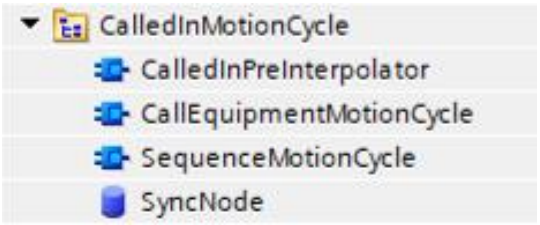


Figure 10-22: Additional blocks of the HPM equipment module, called in the motion cycle.

Interface function description

The Interface of the HPM equipment module also follows the standard of the EMs, with the implementation of PLC internal signals within the DB "ControlNodes" and HMI related signals within the DB "HmiInterface". Additionally, there is the special Interface DB "SyncNode", already mentioned within the architecture explanation. This interface is including the activation of the high-performance motion commands and also gives a status update on whether the motion tasks are busy or any error occurred.

10.14. How to add a new Equipment Module

The following steps have to be performed when adding an additional equipment module. As an example, another equipment module is added to the U4_Unit4_Dummy.

- In the Software Unit U4_Unit4_Dummy, go to PLC tags and select the Unit4 PLC tag table. Within the User constants Tab the following steps have to be taken.

Unit4

	Name	Data type	Value	Comment
1	U4_NO_OF_EMs	Int	1	Highest Value of equipment modules in the unit (Counting starts from 0)
2	U4_EM0	Int	0	Equipment Module 0
3	U4_EM_NEW	Int	1	This is the new EM
4	<Add new>			

Figure 10-23: Unit4 PLC Tag list.

- Increase U4_NO_OF_EMs by 1 so that is shows the highest value of the last equipment module within the unit
- Add a new constant with the name of the additional EM and assign the next possible higher number.
- Copy an existing EM of the AF that fits the best to the needs. In this case the Software Unit "U2Em1_Template_LAD" is copied because a new equipment module with a LAD sequencer is needed. Paste it to the project by right clicking on the Software Units folder in the PLC project tree and select paste.
- In the new Software Unit, adjust the name and namespace as needed. Therefore right click on the new Software Unit and open the properties.
- After putting in the new name and namespace, click "Apply to all underlying elements" to apply the namespace to all objects within the Software Unit.

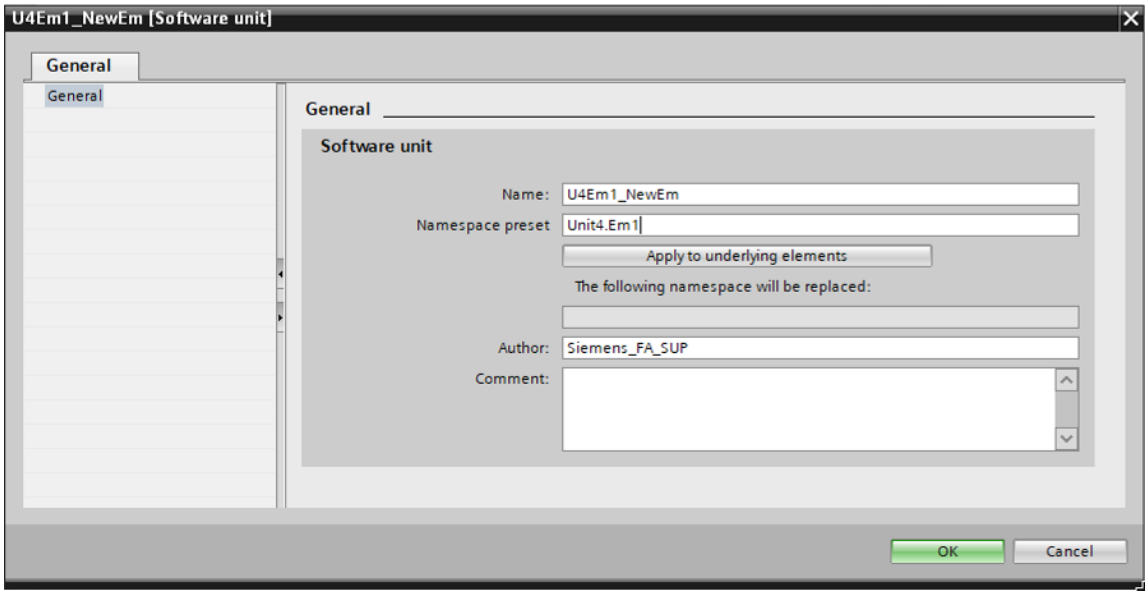


Figure 10-24: Properties of the new Software Unit.

- Adjust the relation to the unit to which it is assigned by selecting the relations menu and adding the new connection.

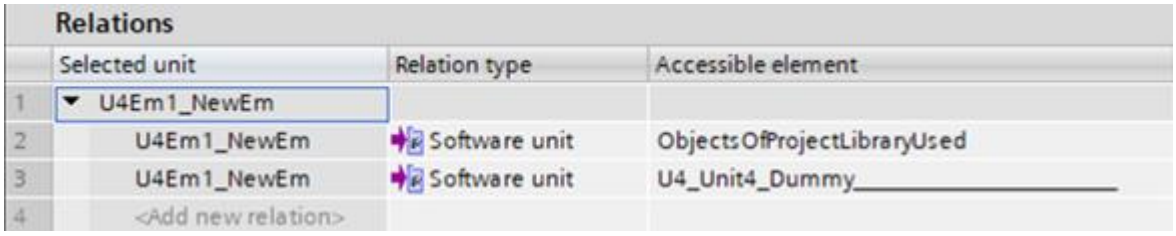


Figure 10-25: Adaption of the newly added EM Relations.

- Open OB "Main" of the new EM and assign the tag that was generated within the PLC Tags of the unit. The tag is assigning the EM to the array structure of the unit data.
- Change the interface of the CallEquipment block to the corresponding unit signals.

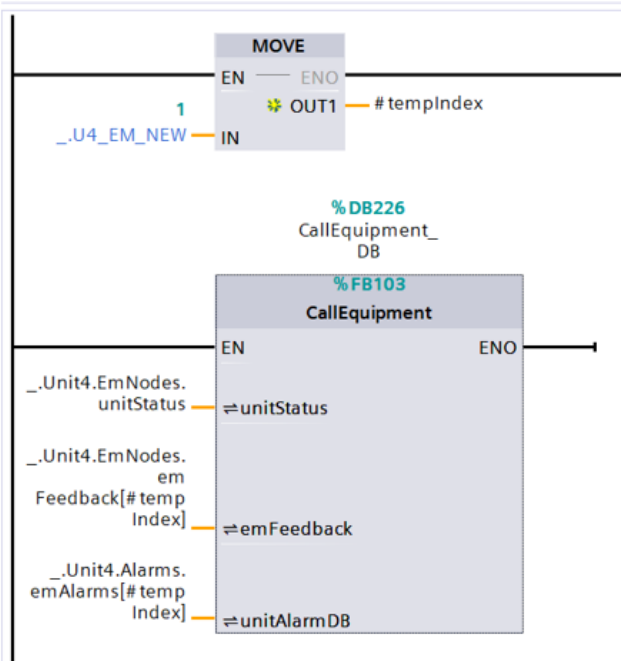


Figure 10-26: OB Main adjustments.

- The template EMs contain two jobs, which can be adapted or new jobs added as required. Job numbers are defined within the PLC tags of the new EM in the User constants. Also adapt the name of the PLC tag table to match the name of the Software Unit.

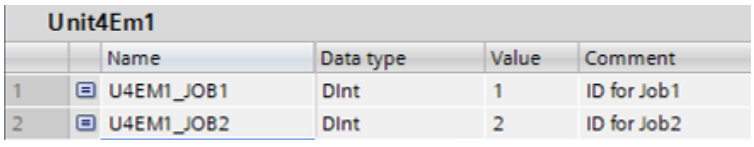


Figure 10-27: EM PLC Tags definition of the Jobs within the User constants tab.

- Every Job then needs a "LAF_SelectJob" instance which are called in the appropriate area of the FB "CallEquipment".
- Change the OB numbers of the OB "Main", "Startup" and "ProDiag" according to the OB numbering convention presented in chapter PLC start and cyclic call sequence.

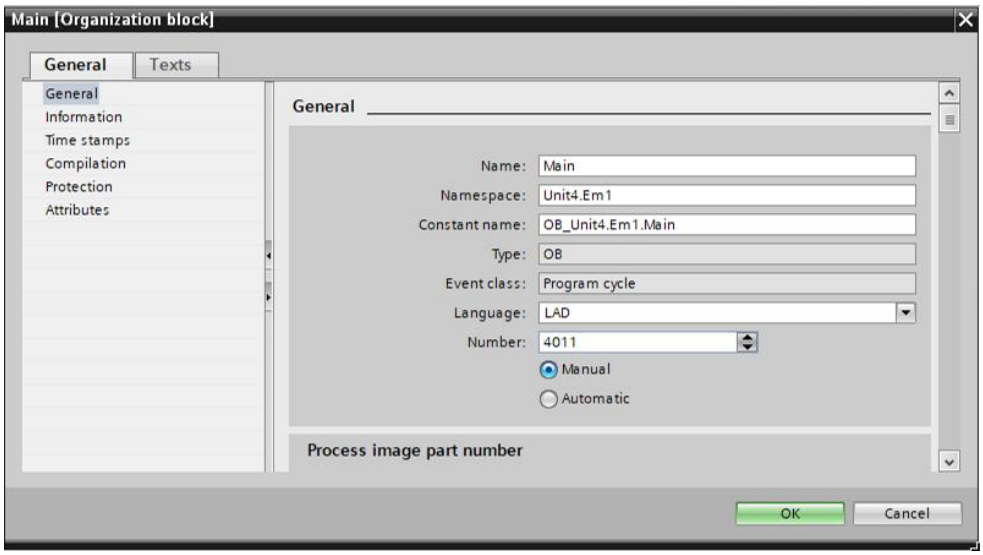


Figure 10-28: Properties of OB Main.

- Renumber the rest of the blocks by compiling the PLC and select to solve the conflict automaticaly, as shown in figure Figure 10-29: Prompt to solve block numbering.
- The Software Unit is then ready and can be adjusted to the process that should be fullflilled by the equipment module. Therefore adjust the sequence and add all control modules that are needed.
- In order to obtain a standardised diagnosis within the EM, the Diagnostic Concept can be followed.

Figure 10-29: Prompt to solve block numbering.

11. Control modules

This chapter focuses on the control modules (CMs). It describes the interfaces structures, how the configuration of each module is handled and the behavior during the manual and automatic control.

The control modules are used to either control the real machine parts or to translate the signals from the machine to the equipment module.

11.1. Interfaces

The interface of the CMs does follow the PLCOpen, all have at least the input "enable" and the outputs "valid", "busy", "error" and "status". In the FBs input there are also the commands to execute functionalities and the hardware feedback. For the project, the control modules of the LBC were used.

More information about the LBC library and the interfaces therein:
https://support.industry.siemens.com/cs/ww/en/view/109792175.

For the ease of use, a special control node version of the LBC is used in the AF. The control node FBs do simplify the FBs interface by moving the commands inputs to a commands UDT and the outputs to a monitoring UDT. Both, commands and monitoring UDTs, are within the control node UDT. Additionally, there is also the command configuration UDT that is used to set the values that will be used for the commands. This simplifies the use of the blocks and a central data point of the CMs can be used to handle the signals within the equipment module

11.2. Configuration

The control modules in the LBC do have a configuration UDT that is used to configure at least the control modules name for the HMI and supervision, and whether the supervision alarms should be disabled.

As shown in the chapter Interfaces and blocks, the control modules configuration in an equipment module are stored in the Config DB. There are two recommended possibilities to configure the DB.

- The first is to set the start value within the DB and shown in the following image:

Config [Unit3.Em2]

	Name	Data type	Start value
1	Static		
2	em	_.LAF_typeEmConfi...	
3	axis	Array[1..._.AMOUNT...	
4	cmCylinder	_.LBC_typeTwoWay...	
5	cmTempSensor	_.LBC_typeAnalogI...	
6	referenceDesignator	WString[25]	WSTRING#'Temperature sensor'
7	physicalUnit	String[10]	'Celsius'
8	isUnipolarSignal	Bool	TRUE
9	default	LReal	0.0
10	limitHigh2	LReal	100.0
11	limitHigh1	LReal	90.0
12	limitLow1	LReal	-10.0
13	limitLow2	LReal	-20.0
14	processValueMax	LReal	100.0
15	processValueMin	LReal	0.0
16	scaleAnalogUppPoint	LReal	27648.0
17	scaleAnalogLowPoint	LReal	0.0
18	scaleProcessUppPoint	LReal	100.0
19	scaleProcessLowPoint	LReal	0.0
20	disableAlarms	Bool	false

Figure 11-1 Configuration example: Directly in the DB.

- The second option is to set the values during the Startup OB, as shown in the following image:

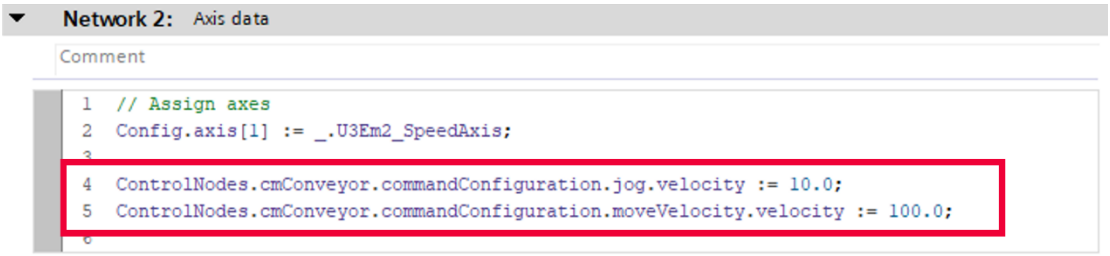


Figure 11-2 Configuration example: Startup OB.

The advantage of using the Startup OB is, that the values can be set dynamically depending on other values.

11.3. Operating modes

The control modules do have a mode for being controlled manually and automatically. Here a manual control is enabled by the PLC and controlled by the user in the HMI. The manual control is enabled only during the manual mode of the unit. Using the LBC as an example, the input "enableManualCmds" is only enabled with the "#statUnitStatus.mode.manual", as seen in the next figure:

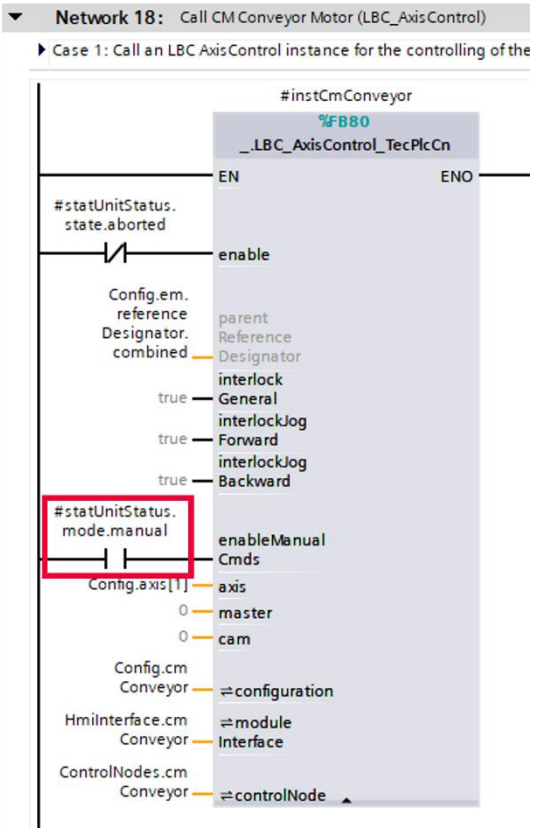


Figure 11-3: Enable Manual Commands of the LBC block.

When activating the Manual Mode of the LBC blocks both interfaces can control the internal functionality of the block. The "moduleInterface" is enabled to be controlled via the HMI and PLC internal commands can be used to control the block via the "controlNode" InOut.

If the "enableManualCommands" Input is not set, the CM is acting in Automatic Control. Therefore, it can only be controlled via the "controlNode" Input. The "moduleInterface" will not be processed while the automatic mode is activated.

12. Safety

The AF introduces a modular concept for a safety program. It demonstrates how to integrate a safety program, how to split the machine into different safety zones and how to structure the safety program.

NOTE For more information about best practice Siemens Safety programming please refer to the Programming Guideline Safety:
https://support.industry.siemens.com/cs/ww/en/view/109750255

12.1. Basics

Functional Safety

Implementing Functional Safety is a mandatory process, which starts with a probably hazardous machine. To get to a safe machine, the risks must be known and sufficiently reduced.

A safety system performs safety functions and is composed of subsystems:

- Subsystem T1: Detection (Position switch, light curtain, ...)
- Subsystem T2: Evaluation (Fail-safe CPU, safety relay, ...)
- Subsystem T3: Reaction (Contactor, frequency converter, ...)

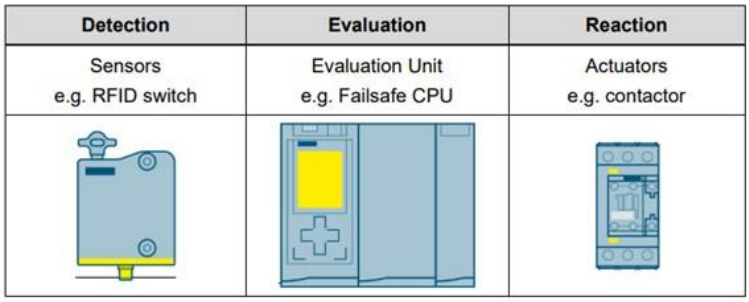


Figure 12-1: The safety system is composed of subsystems.

TIA Safety consists, as well as standard TIA, of two main parts: parameterization and programming. For safety functions, both parts must be configured correctly to reach the necessary and deliberate safety levels.

Safety Administration Editor:

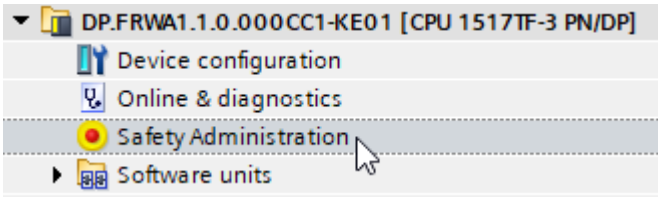


Figure 12-2: Safety Administration.

All main safety settings for the safety program can be set in the Safety Administration. Please refer to the TIA Portal Safety documentation for details.

For the AF, the default settings are used except for those adjustments:

- Activation of the "F-change history". This setting allows to track which user has changed or downloaded the last changes. This cannot be used to detect changes in the safety program for safety reasons, only for programming purpose.
- It is recommended to enable "Consistent upload from the F-CPU" at the end of the commissioning phase. This enables a programmer to load the complete PLC program from a CPU into an empty TIA Portal project. It is not recommended to activate that setting during commission phase as it increases every download time due to the increase data that has to be loaded to the PLC.

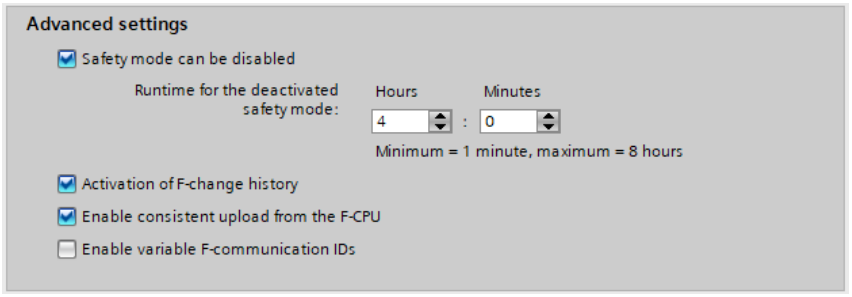


Figure 12-3: Safety program settings in the TIA Portal.

Furthermore, to prevent unwanted changes in the safety program, make sure to create a password.

PROFIsafe address

The PROFIsafe addresses must be unique. The PROFIsafe addresses must be assigned to the real hardware during commissioning.

NOTE Further information about address types etc. can be found in Siemens Safety Documentation and this application example: Configuration examples for unique network-wide and CPU-wide PROFIsafe addresses.

https://support.industry.siemens.com/cs/ww/en/view/109740240

Hardware - Channel Settings

Like the standard hardware, channel settings can be used to set up input and output signals directly. Parameters like sensor evaluation, discrepancy behavior, discrepancy time or sensor supply can be set up for inputs. These settings depend highly on the application. It is mandatory to evaluate the necessary and correct safety settings for each application and each safety function independently. Channels which are not used should be deactivated in all projects and modules.

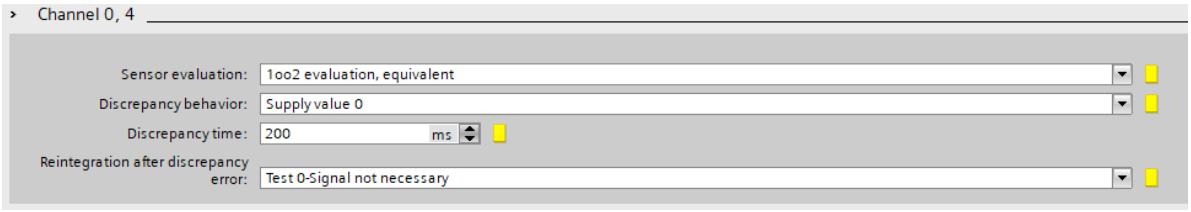


Figure 12-4: Example of safety channel setting in the TIA Portal.

Safety Programming – FB Main_Safety_RTG1

The function block Main_Safety_RTG1 is the first F-block of the safety program that you program yourself. This block calls all logic and underlaying additional functions and function blocks. FB "Main_Safety_RTG1" is called by the F-OB assigned to the F-runtime group. This setting can be done in the safety administration – F-runtime group.

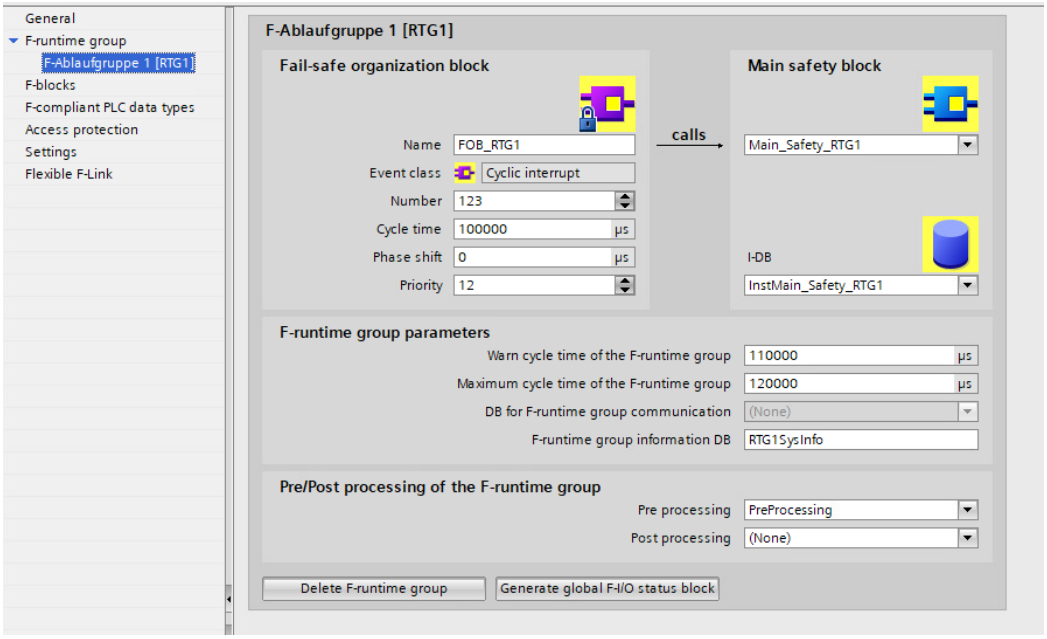


Figure 12-5: Example of the settings for the F-runtime group.

Within the FB "Main_Safety_RTG1", the following sequence of block and logic calls should be followed:

- Receive blocks from other CPUs (F-CPU-to-F-CPU communication)
- Error acknowledgment/reintegration of F-modules/F-channels
- Evaluation block of the sensors
- Operating mode evaluation
- Logic operations, calculations, evaluations, etc.
- Control blocks for safe actuators
- Send blocks to other CPUs (F-CPU-to-F-CPU communication)

Safety Functions

For simple solutions with standardized safety functions, it is recommended to use libraries. In this example especially the "LSafe" and the "LDrvSafe" library is used. The Safety libraries can be found in the Safety unit in the ObjectsOfProjectLibraryUsed folder.

LSafe uses basic safety applications such as E-stop or f-door and combinations for drives or other actuators.

NOTE A further example can be found in the application example: Safety door monitoring with RFID safety switch 3SE64 with guard locking and ET 200SP.

https://support.industry.siemens.com/cs/ww/en/view/109811081

The "LDrvSafe" library can be used to control SINAMIC safety functions via PROFIsafe with the SIMATIC PLCs.

NOTE Further information about it can be found under SIMATIC Failsafe library LDrvSafe to control the Safety Integrated functions of the SINAMICS drive family.

https://support.industry.siemens.com/cs/ww/en/view/109485794

12.2. Safety and ISA-88

A safety zone is a part of the machine, that is affected by a safety event.
Two scenarios are common:

- One safety zone for the entire machine. In this case, a safety event (e.g. E-stop) would have an effect of the entire machine and all safety actuators.
- One safety zone for each ISA-88 unit. In this case, one unit can go to safety stop, while others continue executing. Safety sensors could have an impact on one or multiple safety zones. E.g. a machine might have a global E-stop button that stops the entire machine, while a light curtain at one part of the machine only triggers the safety event for the respective unit.

The AF demonstrates a way how to implement a separate safety zone for each unit.

12.3. Data exchange between standard program and safety program

Direct data exchange between standard and safety program is not allowed.

During engineering, changes to standard blocks that are read or written by the safety program would cause the F program to lose its consistency. A recompile of the safety program is necessary and a download to the CPU is only possible via CPU STOP.

During operation, changes to safety-related data during the execution of the safety runtime group will cause the CPU to stop.

Therefore, a strict separation between standard and safety program is necessary. For data exchange between standard user program and safety program, special data blocks are defined, in which the data to be exchanged is stored. This action allows you to decouple the blocks of the standard user program and the safety program. The changes in the standard user program do not affect the safety program (and vice versa), given that these data blocks are not modified.

An overview of the data flow can be seen here:

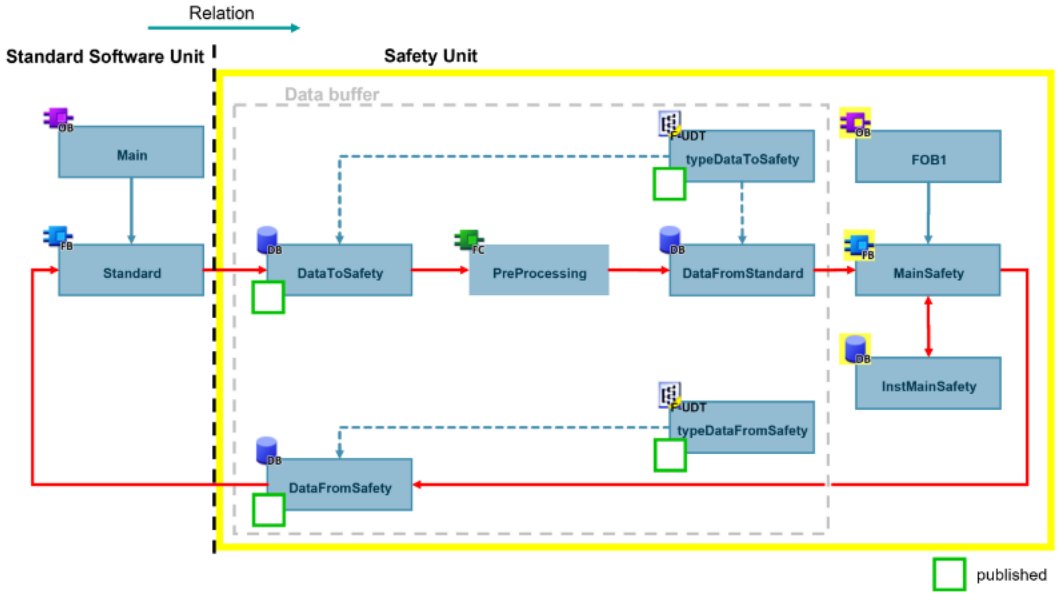


Figure 12-6: Data exchange between the standard user program and the safety program.

The following data blocks are defined:

DataToSafety

This block includes information that is send from the standard user program to the safety program, e.g. acknowledge commands.

In the AF, data sent from the standard user program to the safety program contains the following information:

DataToSafety [Safety]

	Name	Data type	Start value	Retain	Comment
1	Static				
2	ackSafetyFromHmi	Int	0		Acknowledge variable from operator control ...
3	ackSafetyGlobal	Bool	false		Acknowledge command to ack all safety
4	ackFromUnit1	Bool	false		Acknowledge command comming from Unit 1
5	ackFromUnit2	Bool	false		Acknowledge command comming from Unit 2
6	ackFromUnit3	Bool	false		Acknowledge command comming from Unit 3
7	ackFromUnit4	Bool	false		Acknowledge command comming from Unit 4

Figure 12-7: Content of DB "DataToSafety".

AckSafetyFromHmi enables failsafe acknowledgement from an operator control and monitoring system. An integer variable is used to realize a multi-step acknowledgement. See details further below. In the AF, it is linked to the ACK button the HMI.

AckSafetyGlobal provides the possibility to give a global acknowledge command. E.g. a mechanical hard-wired reset button could trigger that bit.

Additionally, each unit could send an acknowledge command to the safety program, if it contained a hard-wired reset button. In network 2 of FB "Main_Safety_RTG1" it can be configured, which safety zones should be acknowledged by the individual commands.

DataFromStandard

This block is an additionally buffer block, so that the data used in the safety program does not change while safety execution. If that block wasn't used, any change in "DataToSafety" during safety execution would cause the CPU to go to stop. FC "PreProcessing" copies the data from DB "DataToSafety" into DB "DataFromStandard". Both DBs have the same structure "typeDataToSafety". FC "PreProcessing" can be configured in the safety administration.

DataFromSafety

This block contains all information from the safety program that is relevant for the standard user program. That includes status information, of for example active optoelectronic protective devices (short AOPD), emergency stop buttons and drives. Further processing of the data, for example the diagnostic and reporting concept is carried out within the standard program to keep the safety program compact. In many cases the standard program does a controlled stop in a safety event or decreases the speed of an axis, while safety is just monitoring safely and interacting e.g. if set speed boundaries are exceeded.

The data sent from the safety program to the unit contains the following information:

DataFromSafety [Safety]

	Name	Data type	Start value	Accessible f...	Writa...	Visible in ...	Supervision	Comment
1	Static							
2	safetyReleasedAllSafetyZones	Bool	false	☑	☐	☑		All safety zones are released
3	ackRequiredAnySafetyZone	Bool	false	☑	☐	☑		At least one safety zone requires an acknowledge
4	ackOP_changeRequired	Bool	false	☑	☐	☑		ACK_OP is waiting for value change
5	devicesSummary	typeDevicesStatusSummary		☑	☐	☑		General safety status of devices
6	eStopButton0pressed	Bool	false	☑	☐	☑		Global - E-stop button E0 is pressed
7	eStopButton1pressed	Bool	false	☑	☐	☑		Zone 1 - E-stop button E1 is pressed
8	eStopButton2pressed	Bool	false	☑	☐	☑		Zone 2 - E-stop button E2 is pressed
9	eStopButton3pressed	Bool	false	☑	☐	☑		Zone 3 - E-stop button E3 is pressed
10	eStopButton4pressed	Bool	false	☑	☐	☑		Zone 4 - E-stop button E4 is pressed
11	fDoor1Open	Bool	false	☑	☐	☑		Zone 1 - F-door is open
12	fDoor2Open	Bool	false	☑	☐	☑		Zone 2 - F-door is open
13	fDoor3Open	Bool	false	☑	☐	☑		Zone 3 - F-door is open
14	fDoor4Open	Bool	false	☑	☐	☑		Zone 4 - F-door is open
15	AOPD1interrupted	Bool	false	☑	☐	☑		Zone 1 - Light curtain is interrupted
16	AOPD2interrupted	Bool	false	☑	☐	☑		Zone 2 - Light curtain is interrupted
17	AOPD3interrupted	Bool	false	☑	☐	☑		Zone 3 - Light curtain is interrupted
18	AOPD4interrupted	Bool	false	☑	☐	☑		Zone 4 - Light curtain is interrupted
19	eStopZone1	typeEstopFunction		☑	☐	☑		Safety status of estop button
20	eStopZone2	typeEstopFunction		☑	☐	☑		Safety status of estop button
21	eStopZone4	typeEstopFunction		☑	☐	☑		Safety status of estop button
22	eStopZone3	typeEstopFunction		☑	☐	☑		Safety status of estop button
23	fDoorZone1	typeFDoorFunction		☑	☐	☑		Safety status of door
24	fDoorZone2	typeFDoorFunction		☑	☐	☑		Safety status of door
25	fDoorZone3	typeFDoorFunction		☑	☐	☑		Safety status of door
26	fDoorZone4	typeFDoorFunction		☑	☐	☑		Safety status of door
27	lightCurtainZone1	typeAOPD_Function		☑	☐	☑		Safety status of light curtain
28	lightCurtainZone2	typeAOPD_Function		☑	☐	☑		Safety status of light curtain
29	lightCurtainZone3	typeAOPD_Function		☑	☐	☑		Safety status of light curtain
30	lightCurtainZone4	typeAOPD_Function		☑	☐	☑		Safety status of light curtain
31	statusZone1	typeSafetyZoneStatus		☑	☐	☑		Status of safety zone
32	statusZone2	typeSafetyZoneStatus		☑	☐	☑		Status of safety zone
33	statusZone3	typeSafetyZoneStatus		☑	☐	☑		Status of safety zone
34	statusZone4	typeSafetyZoneStatus		☑	☐	☑		Status of safety zone
35	activeSTO_Zone3	Bool	false	☑	☐	☑		Zone 3 - STO active
36	drive1	typeSafetyDrive		☑	☐	☑		Safety status of drive
37	drive2	typeSafetyDrive		☑	☐	☑		Safety status of drive
38	drive3	typeSafetyDrive		☑	☐	☑		Safety status of drive
39	drive4	typeSafetyDrive		☑	☐	☑		Safety status of drive
40	drive5	typeSafetyDrive		☑	☐	☑		Safety status of drive
41	drive6	typeSafetyDrive		☑	☐	☑		Safety status of drive

Figure 12-8: Content of DB "DataFromSafety"

For accessing the data blocks "DataToSafety" and "DataFromSafety" from outside the safety unit, the two data blocks and the corresponding F-UDTs including all subordinate F-UDTs must be published, as shown in Figure 12-6. Within the accessing standard software units a relation to the safety software unit must be created. This is shown as an example for Unit 3 in the following Figure 12-9.

Relations

	Selected unit	Relation type	Accessible element
1	U3_Demo_____...		
2	U3_Demo_____...	Software unit	ObjectsOfProjectLibraryUsed
3	U3_Demo_____...	Software unit	Global
4	U3_Demo_____...	Software unit	Safety

Figure 12-9: Relation to Safety Unit

In the AF, the following safety information is provided:

- Summary bit that shows if all safety zones are released and a summary bit that shows if any safety zone requires an acknowledgement.
- Detailed information for each safety zone
- Summary of all safety devices
- Detailed information for each safety device (sensors and actuators)

Summary bits can be used to monitor the overall status. The detailed bits can be used track down the root cause of any safety event. The individual status bits of the devices could be monitored with ProDiag supervisions, so that the HMI displays which safety devices have been trigger, for example.

12.4. Safety Program

In the AF, the safety program is split into different safety zones which are structural related to one unit, and each safety zone has multiple subordinated safety functions, e.g. Emergency Stop or Protective door monitoring. Each safety function internally contains the structure recommended by the safety programming guidelines.

- Evaluation (of the corresponding sensor, e.g. emergency stops, protective doors,...)
- Logic (Defining how the corresponding actuator should react on different input states)
- Reaction (Controlling the corresponding actuator(s) of the safety function)

The structure of the safety program in the Automation Framework follows the above-mentioned general program structure:

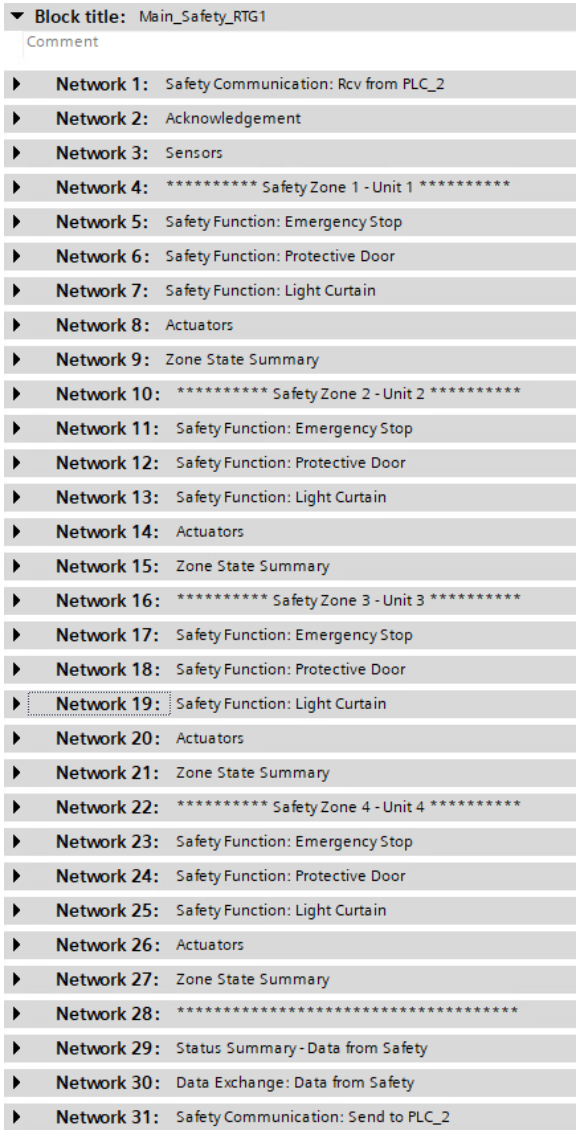


Figure 12-10: Content of FB "Main_Safety_RTG1" in the Automation Framework.

The structure implemented in the AF is an example. Depending on size and complexity of the safety program, a different structure can be chosen.

Safety Communication: Receive from and send to PLC 2

These networks implement fail-safe sending and receiving of data from another fail-safe PLC over PRIOFINET. The RCVDP instruction receives the data from the other F-CPU. The instruction must be called in the first network. The SENDDP instruction sends the data to the other F-CPU. The instruction must be called in the last network. Detailed description of both instruction, and configuration of the fail-safe communication can be found in the SIMATIC Safety- Configuring and Programming manual. https://support.industry.siemens.com/cs/ww/en/view/54110126/171195072139

Acknowledgement

This network implements the acknowledgement for the safety program and the reintegration of passivated fail-safe modules.

All safety zones are acknowledged by one global acknowledgement signal. The acknowledgement of each safety zone/ function must be adapted to the requirements of the real machine.

- Acknowledge from HMI :
  Communication between the HMI and the CPU is not safe. Transferring safety-related data requires measures that ensure the safe transfer. For resetting safety functions or acknowledging errors using an HMI, TIA Portal provides the "ACK_OP" system block. This block is called in here.
- Global acknowledge and acknowledge from units :
  These bits could be set in FB "CallGlobal" or in FB "CallGlobal" respectively. Dummy networks exist in the AF, but nothing is connected. If mechanical and hard-wired reset buttons exist on a global level or in a unit, these could be connected and trigger these bits.
- Global acknowledge and reintegration :
  The instruction ACK_GL is called to create an acknowledgment command for the simultaneous reintegration of all F-I/O or channels of the F-I/O of an F-runtime group after communication errors, F-I/O errors, or channel faults.

Sensors

This sensors network interconnects the global addresses from the input module to local variables for further use in the safety program.

Safety Zone

A safety zone is corresponding to one Unit and consists of multiple safety functions and one actuator network. If necessary, the safety zones can be divided further into safety zones corresponding to the different equipment modules.

Safety Function

A safety function implements one safety function resulting from the risk assessment. This includes the detection, evaluation and reaction subsystem in one block. Each safety function should be separated from other safety functions, making it scalable and reducing testing and validation efforts.

Within Safety Zone 3 preconfigured safety functions can be found based on the LSafe Library, e.g. Emergency Stop. This safety function implements a safe shutdown (STO) of drives in emergency. When the emergency button Is pressed, the actuator(s) are switched off with STO.

More information about the LSafe and other available applications examples commonly used safety functions can be found in SIOS and selected via Safety Select: https://support.industry.siemens.com/cs/ww/en/view/109773295

Actuators

As one actuator can correspond to different safety functions, the actuator(s) are not directly controlled within the safety function network. The network contains the controlling of all actuators of the related safety zone. That could mean sending a STO command to a drive or switching of contactors. Each safety zone has its own FB Actuators.

Zone State Summary

Summarize the diagnostic information from each safety function, this could include release of the safety zone, and if an acknowledge is required by any safety function or actuator of the safety zone.

Status Summary

Further summarizes all diagnostic information from each safety zone.

Data Exchange

Moves the summarized diagnostic information to the data exchange DB for further use in the standard user program.

12.5. Demo

In the AF demo, four emergency-stop- and one protective-door-monitoring-functions are preconfigured and evaluated. In safety zone 3, one emergency-stop and the protective-door-monitoring function will lead to a safe shutdown (STO) of the drives.

The AF default safety password is *Siemens!123*.

With the help of simulation tools, the safety program can be validated in a virtual, risk-free environment without the need of extensive hardware, leading to deployment ready PLC code and greatly reducing engineering cost. The safety unit can be simulated with different methods:

PLCSIM:

With PLCSIM it is possible to create SimTables to monitor and modify the fail-safe inputs to test and verify basic logic of the implemented safety functions.

PLCSIM Advanced and periphery simulation with SIMIT

With PLCSIM Advanced and the simulation software SIMIT, both the PLC behavior and the behavior of sensors and actuators can be simulated with simulation models. This allows validation and verification of the program and to test the physical behavior of the devices.

A detailed description of the different simulation workflows can be found in chapter Simulation.

13. Motion Control

13.1. General

The S7-1500 motion control (MC) is executed centrally by the Motion-OBs which are in the Software Unit "Motion". The motion control commands of the technology object (TO), are programmed in the equipment modules that belongs to the axis from a technology perspective. However, the call of the motion commands can be programmed in the standard cycle OB "Main" of the equipment module, or in the OB "MC_PreInterpolator", which is running in the same fast cycle as OB "MC_Servo" (= motion control application cycle).

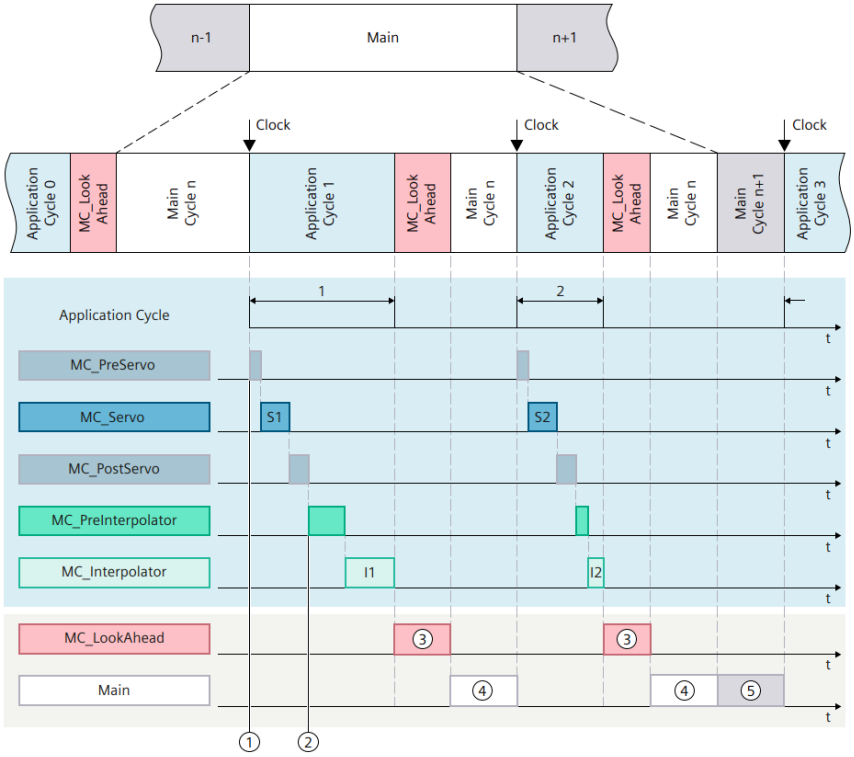
MC_Servo/ MC_Interpolator

When the first TO is added to the S7-1500 PLC, the OBs "MC_Interpolator" and "MC_Servo" for processing the TOs are created automatically. The functionality of the TOs creates its own execution level according to the application cycle. These OBs are know-how protected and the program cannot be edited.

MC_PreInterpolator

The OB "MC_PreInterpolator" and other not know-how protected Motion-OBs can be used to call motion commands in the motion cycle and must be inserted manually. These OBs consider special requirements regarding to time-critical events. The programming these OBs, should only be done if needed, for example, to start motion commands immediately when a fast reaction or an instant execution for positioning is required. Standard motion commands like "MC_Power" and "MC_Reset" should be called in the standard program cycle of the equipment module. Otherwise, the PLC is loaded with too much unnecessary program which decreases the overall PLC performance.

The figure below illustrates the motion control application cycle and the hierarchy of the different Motion-OBs in detail:



① "TPA OB Servo" input process image partition
② "TPA OB Servo" output process image partition
③ MC_LookAhead cycle
④ Main cycle n
⑤ Main cycle n+1

Figure 13-1: Motion control application cycle and call hierarchy of Motion-OBs.

13.2. Motion Software Unit

The Software Unit Motion is very lean. It only contains the Motion-OBs and calls of time-critical program code. The standard motion commands are programmed and called in the respective equipment module itself. Time-critical motion commands are programmed in the respective equipment module too, but instanced in the OB "MC_PreInterpolator".

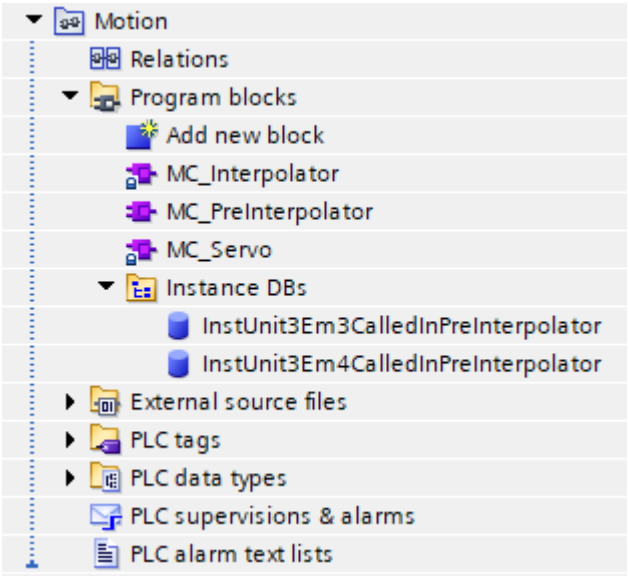


Figure 13-2: Blocks of Software Unit "Motion".

The technology objects are configured outside of the Software Units, globally in the PLC. To access the data of the TO, a relation from the respective equipment module Software Unit to the TO is needed. If a time-critical motion application is required, an additional relation from the Software Unit Motion to the respective equipment module Software Unit is necessary to call the time-critical motion commands in the motion cycle.

In the AF, the time-critical motion commands are called in the OB "MC_PreInterpolator", for example for equipment module 3 of unit 3:

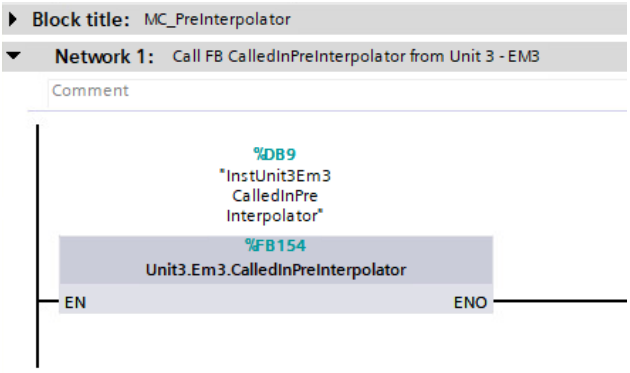


Figure 13-3: Call of time-critical motion commands from EM in motion control application cycle.

In the preconfigured TIA project of the AF, a demo equipment module "U1Em3_HighPerformanceMotion" is included, this demo EM, shows cases about how to integrate motion control into the user program.

NOTE For a detailed guide how to implement a high-performance equipment module see chapter EM Demo Implementation – High Performance Motion.

13.3. Technology objects vs. Sina-blocks

The drives can be controlled via technology objects within the PLC or standalone with Sina-blocks. The general difference in using TOs and Sina-blocks is the location of the control.

When using TOs the position controller and interpolator is in the PLC and sends control commands as well as speed setpoint values to the drive. The speed and current control are done by the drive. So, the intelligence is inside the PLC which offers a greater functionality in terms of gearing, camming and kinematics.

When using Sina-blocks like "SinaPos", the position controller and interpolator is in the drive. The Sina-blocks in the PLC are an interface to send control commands to the drive. The intelligence is inside the drive and limited to independent axes with basic functionalities like speed control and positioning.

Since both concepts are different, both have different advantages. The following table compares both concepts. The Advantages are highlightend in Green.

	Technology object	**Sinablock**
Functionality	• Speed control • Positioning • Gearing • Camming • Kinematics	• Speed control • Positioning (with EPOS)
Axis quantity	Depends on the CPU. • Up to 192 axes are supported with CPU1518TF • Up to 310 axes are supported with CPU1508S TF	Number of axes only limited to network limitations.
Performance	• Depends on the CPU and the quantity of axes. • At least medium impact on CPU performance.	• Independent on the quantity of axes. • Lower impact on CPU performance.
SINAMICS drives	All	All
PROFIdrive telegrams	A wide range of telegrams are supported.	Limited to telegram 1 (speed control) and telegram 111 (positioning via EPOS).
PROFINET IRT	Required for servo drives	Not required
Axis configuration	• In TIA Portal • Intuitive • Wide range of options	• In Startdrive • Additional know-how required • Limited options
PLC programming	• Variety of PLCOpen blocks for MC • High functionality • High flexibility	• Limited to "Sinablocks" • Low functionality • Low flexibility
Usability	High	Medium
Axis data availability	• Directly available via the DB of the TO • Low complexity • Low engineering effort	• Via additional blocks which establish an acyclic communication to write or read parameters from the drive • High complexity • High engineering effort
Diagnostics	• CPU and Startdrive trace supported • Online diagnostics in TIA Portal • Technology alarms are available in diagnostic buffer of the CPU, via the CPU display and webserver and in the alarm view in the HMI	• Limited to Startdrive trace • Detailed diagnostics only online via Startdrive • Faults and alarms normally only available online via Startdrive (with no additional engineering effort)

	Technology object	Sinablock
Simulation	• TO integrated simulation • No additional software and licenses required	• Additional software and licenses (SIMIT or DriveSim) required
Openness / Add-Ins	Supported	Supported

Table 13-1: Comparison of TOs and Sinablocks

The equipment module "High Performance Motion" shows a "TO_PositioningAxis" with a geared "TO_SychnronousAxis" as well as a speed-controlled drive with the Sinablock "SinaSpeed" as an example.

Because the axes must move synchronously, the intelligence of the control needs to be in the PLC and is implemented with "TO_PositioningAxis" and "TO_SychnronousAxis". The non time-critical motion commands (e.g. "MC_Power") are called in the standard program cycle of the equipment module "High Performance Motion". The time-critical positioning commands for the geared movements are called in the faster motion control application cycle.

Because the movement for the speed-controlled drive is independent, it is possible to use a "TO_SpeedAxis" or the SinaSpeed block. To reduce the CPU load as much as possible for the high-performance motion application, the SinaSpeed block within "LBC_AxisControl" is used to control the drive.

14. Communication

NOTE Please be aware that without TIA Portal V19 Update 3, TIA Portal is crashing when compiling the PLC with a configured OPC UA Server interface. Further information, as well as solution and workaround are available under:

https://support.industry.siemens.com/cs/ww/en/view/109971630

Latest state: 18/09/2024

14.1. Machine-MES communication

In the smart manufacturing industry, various IT applications must exchange data with the shopfloor automation (OT – operation technology). This introduces challenges on the IT/OT integration like different fieldbus protocols, scalability with the number of devices to connect, consistency of the exchanged data as well as IT security and availability aspects.

To address these challenges, OPC UA offers a rich set of features, enables vendor and supplier independent integration, flexible interface data modeling independent of the PLC code implementation and various data exchange methods to reach consistency and indispensable IT security.

Siemens offers an information model kit as a tool kit for creating a production wide standardized machine interface connecting the shopfloor to IT systems like MES/MOM.

The Siemens Information Model Kit provides standard OPC UA NodeSets containing types and objects commonly used in industry-independent IT/OT integrations. Besides general machine models like machine information and status you can find communication approaches to exchange common messages for a seamless production workflow integration. The guideline helps to understand how to create an own standardized interface, which consideration need to be taken in mind when modeling and how to implement integration use cases like KPI/OEE monitoring or Tracking & Tracing. The Siemens Information Model Kit also contains the interface for an out-of-the-box integration of a controller with Siemens MOM products.

The Siemens Information Model Kit provides standard OPC UA NodeSets containing types and objects commonly used in industry-independent IT/OT integration \6\.

https://support.industry.siemens.com/cs/ww/en/view/109814835

In addition, the AF provides an explanatory OPC UA server interface based on the Siemens Information Model Kit and an OMAC PackML conform data structure. The idea of the example interface is to share and introduce best practices of interface modeling to achieve an easier, faster and cost-efficient IT/OT integration.

The provided OPC UA server interface covers four integration use cases:

- Machine information, based on a static data set with basic information and description of the machine.
- Machine status, which indicates the current unit state and mode as well as basic operation KPIs.
- Machine monitoring, which supervises the specific status or process value of modules or components in the machine.
- Machine control, which enables simple command based remote control of the unit state and mode.

NOTE With free "Siemens OPC UA Modeling Editor (SiOME) \3\, the PLC engineer can easily modify and/or extend the interface and deploy it to the OPC UA Server of the PLC.

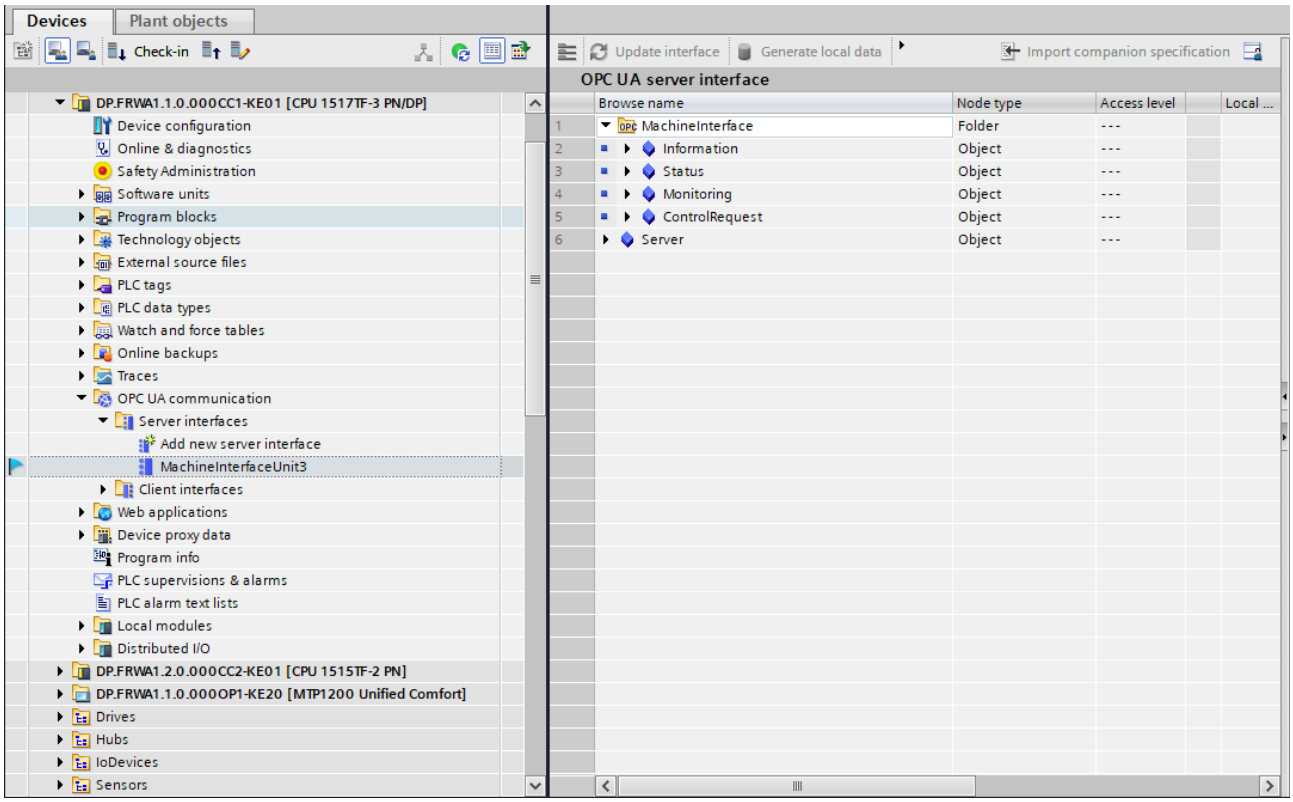


Figure 14-1: OPC UA server interface on unit level.

NOTE The OPC UA server interface is implemented and mapped as an example for unit 3. **It is not tested!**

In addition, the remote Control Interface is disabled within the Software Unit Global for unit 3, which means the OPC UA methods won't influence the current mode and state of the unit.

Furthermore, the AF provides a predefined icon to visualize the connection state between the machine and the IT system in the HMI header. The IT connection icon shows two states:

State	Figure
Connected (light grey)	
Not connected (red)	

Table 14-1: IT connection state icon

NOTE The PLC application must identify its IT connection state and map it into the following tag: "CentralFunctions.Settings.ITConnection"

14.2. Machine-Machine communication

PN/PN Coupler

The PN/PN Coupler allows data transmission between two PROFINET controllers located in separated networks. The PN/PN Coupler physically separates the networks and offers PROFINET real-time (RT) communication as well as full support of the PROFIsafe profile for safe communication over network boundaries.

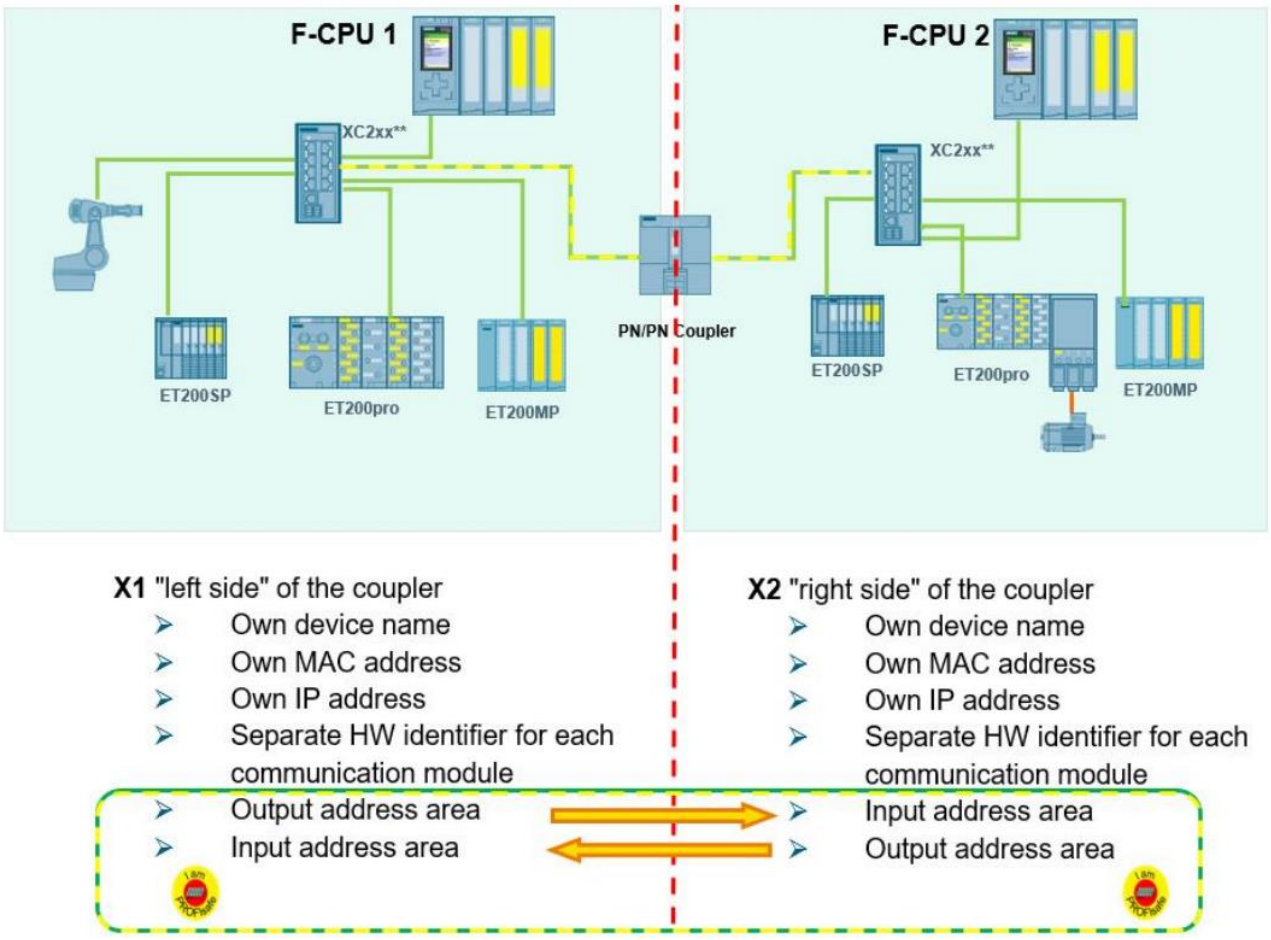


Figure 14-2: Controller-Controller communication via PN/PN Coupler.

Detailed information as well as advantages and disadvantages of using PN/PN Coupler are available in the Information system for the TIA Portal.

NOTE The AF provides a preconfigured exemplary real-time communication with PROFIsafe data between two fail-safe PLCs by using a PN/PN Coupler.

I-Device

The I-Device functionality in the S7-1500 allows the IO-controller to be an IO-device and IO-controller simultaneously. This communication could be used in approbation with the end customer when a network separation between the machines is not required or when more PLCs are in the same subnet and data needs to be exchanged in real-time. In this case the communication between PROFINET controllers could be implemented using the I-Device function.

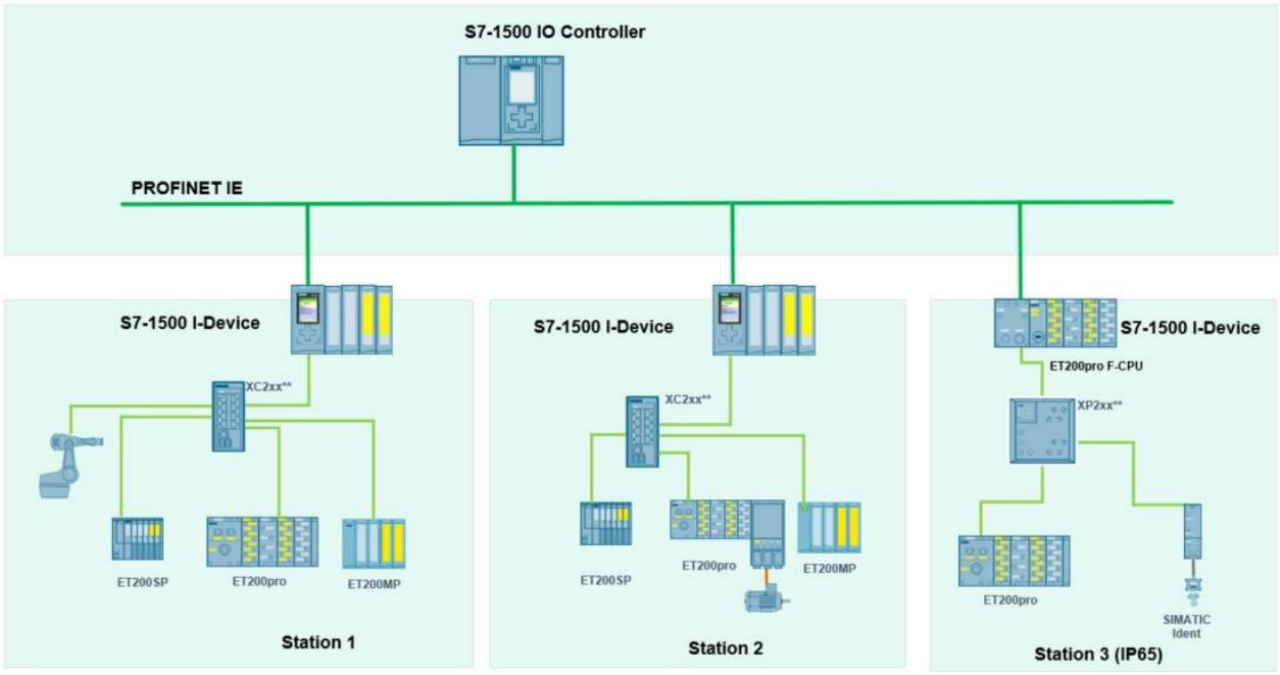


Figure 14-3: S7-1500 as I-Device.

Detailed information as well as advantages and disadvantages of using the S7-1500 as an I-Device are available in the Information system for the TIA Portal.

Flexible F-Link

The Flexible F-Link supports safety-related CPU to CPU communication across machine/subnet boundaries and relies on standard communication protocols (TCP/IP), providing the option of routing from one network into a different network (via layer 3 and aggregation level).

This does not compromise the machine security concept and does not require any special requirements on the configuration of the router or ports. Just ensure that the Open User Communication (e.g. TCP or UDP) between the networks is routed.

Because standard communication protocols are used as the underlying communication method, the Flexible F-Link is not a deterministic communication method (no real-time communication). Especially for very rapid safety reaction times, it is not suitable for every application. If a safety-related communication via Flexible F-Link can be implemented, must be verified on a case-by-case basis.

In addition, Flexible F-Link does not require any additional hardware and can be implemented on existing network infrastructure.

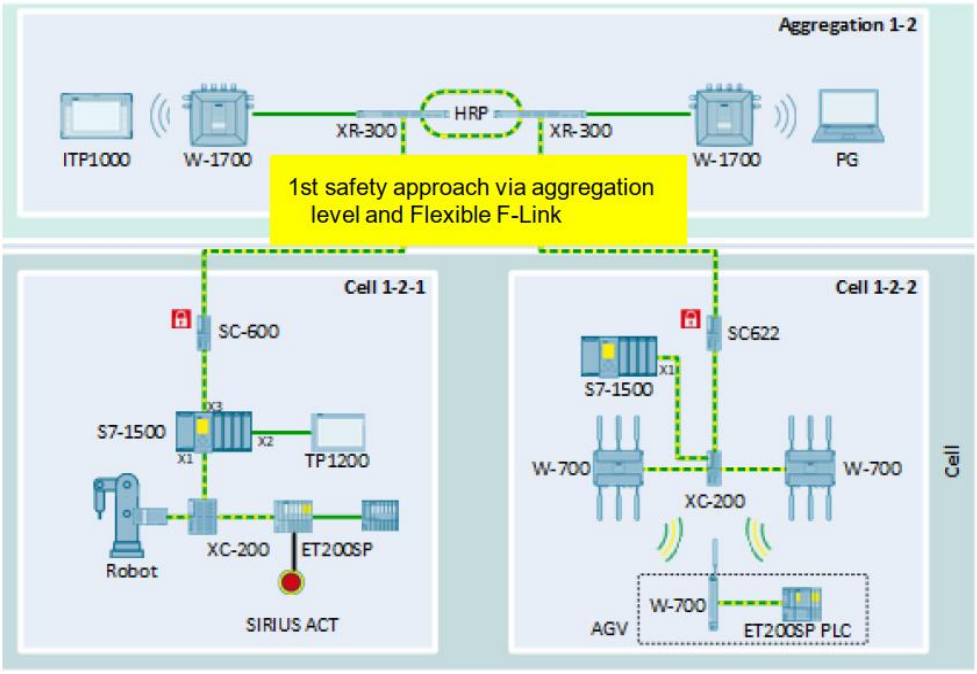


Figure 14-4: CPU to CPU communication via OUC and "Flexible F-Link".

Detailed information of using "Flexible F-Link" as well as an application example of the configuration of "Flexible F-Link" are available in the Information system for the TIA Portal.

NOTE

It is recommended to specify the safety related data early in the development phase to reduce ad hoc changes during or after engineering and commissioning. The selected safety network architecture must ensure that PROFIsafe addresses are unique in the whole network.

Additional information is available in this FAQ:
https://support.industry.siemens.com/cs/ww/en/view/109740240

14.3. Communication to third-party systems

The standardized communication concept OPC UA supplies the base for successfully integrating the equipment control level (e.g. PLC) with the higher levels of factory automation (e.g. IT systems), such as the manufacturing execution system.

The "Consistent message-based communication via OPC UA" application example provides a solution to transfer data in a consistent and reliable way either through tag-based or method-based communication. A handshake mechanism and buffering (ring buffer) are also included. Every partner that wants to communicate with the machine, just needs to follow the standard procedure to exchange data with the machine.

Requirements that are solved by this application are:

- Message contents of arbitrary types can be processed.
- Communication with several partners can be handled.
- Ingoing messages can be handled specifically as well as outgoing messages.
- Messages are sent in the same order in which they are received.
- Buffer to prevent loss of messages in case of communication interrupt.
- Communication mechanisms on OPC UA (registered) read / write and subscribe or OPC UA methods can be used.

The concept can be extended by custom message types. It provides a reliable, consistent, and serialized communication: writing to and reading from a message buffer. Separate alarming mechanism including a handshake between the PLC and the information system must be set up. Therefore, the diagnostics information needs to be uniform. Each alarm message of the information system has an ID, category, and value to point out the detailed explanation in additional diagnostics table. The handshake for alarm messages will be organized by an alarm manager, that triggers the active alarm and confirms if the alarm is acknowledged by the information system.

Detailed documentation as well as an application example for TIA Portal of "Consistent message-based communication via OPC UA" are available here:

https://support.industry.siemens.com/cs/ww/en/view/109795979

15. Process diagnostics

15.1. General

The Automation Framework offers a holistic diagnostic concept that allows an efficient localization of faults and errors in the machine application. Within the TIA Portal we distinguish between "System diagnostics" and "Process diagnostics".

- The System diagnostics is a hardware-related and described in the chapter Diagnostics.
- The Process diagnostic is related to the machine actions. Defined tags in the PLC-Program are supervised and create a full text message for HMI or 3rdParty systems.

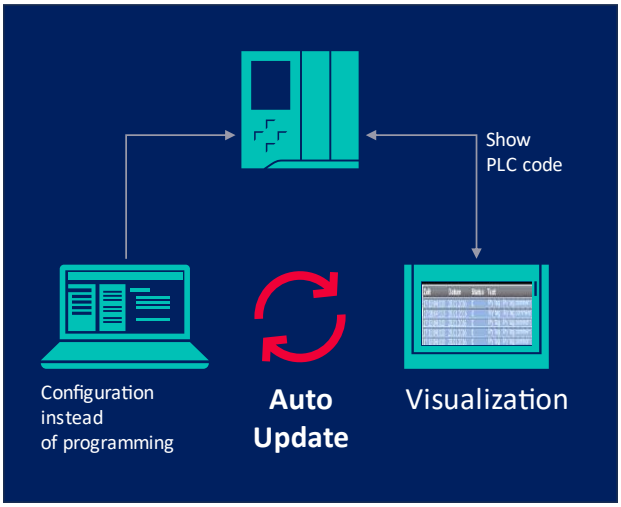


Figure 15-1: TIA Portal option for Process Diagnostics.

In the Automation Framework (AF), the option ProDiag is used for Process Diagnostics.

The use of ProDiag has the following advantages:

- Consistent alarm texts, generated automatically in the PLC with up to three different languages in the runtime.
- No engineering on the HMI.
- Automatic generated collection bits for managing the machine behaviour.
- Almost no increase in cycle time within alarm surge.
- Criteria analysis in the HMI – view the root cause.
- PLC code viewer on HMI for interlocks.
- Create new supervisions with just a few clicks.

The following chapters will introduce the necessary TIA Portal licenses and system settings for ProDiag and the AF implementation.

NOTE Due to its modular structure, ProDiag can be deleted and replaced by your own implementations.

15.2. Prerequisites

This chapter explains the necessary settings for ProDiag. The final settings to meet end-user requirement may differ.

Detailed documentation about ProDiag can be found within the Information system of TIA Portal as well as an application example for understanding the usage:
https://support.industry.siemens.com/cs/ww/en/view/109740151

15.2.1. PLC device settings

Central Alarming Activation

"Central alarm management in the PLC" must be activated in the properties of the PLC device. This enables the Alarm Server on the PLC. Full text messages are generated in the PLC program and then sent to PLC Alarm Server which pushes the messages to all connected and available alarm clients for example the TIA Portal engineering, the display on the PLC, all connected HMIs, or the PLC's SIMATIC web server.

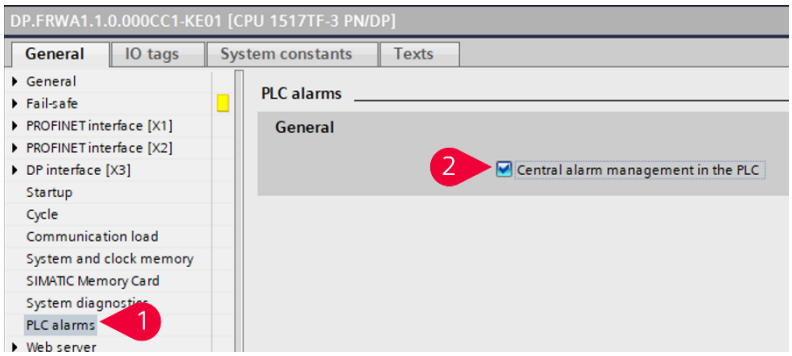


Figure 15-2: Activation of the central alarm management in the PLC.

15.2.2. Multilingual support

In the AF project the default ProDiag language setting is English, but multilingual is supported. To display the messages in the correct language:

- Open the project languages Configuration in the Languages & resources folder.
- Select all the project languages you want to add to the project.

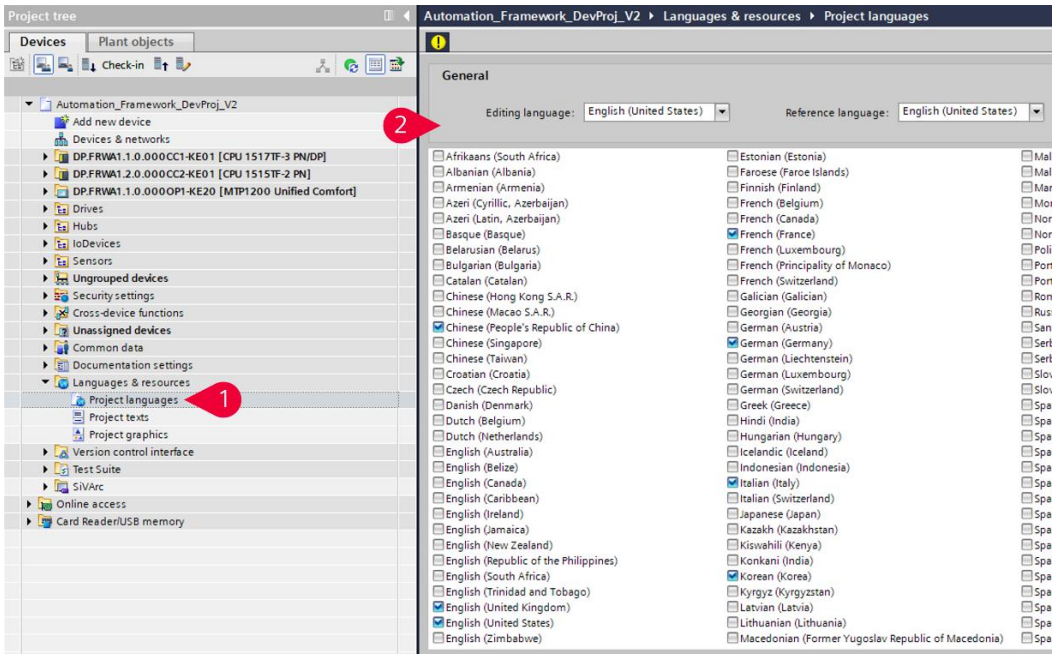


Figure 15-3: Overall project language settings.

All languages that are activated in this configuration can now be adapted within the project texts. A further step that must be taken is to assign the project languages to the respective system languages of the PLC and its web server. To do so, the following steps must be taken:

- Select the Multilingual support setting within the properties of the PLC.
- Assign the project languages that should show up to the system languages.

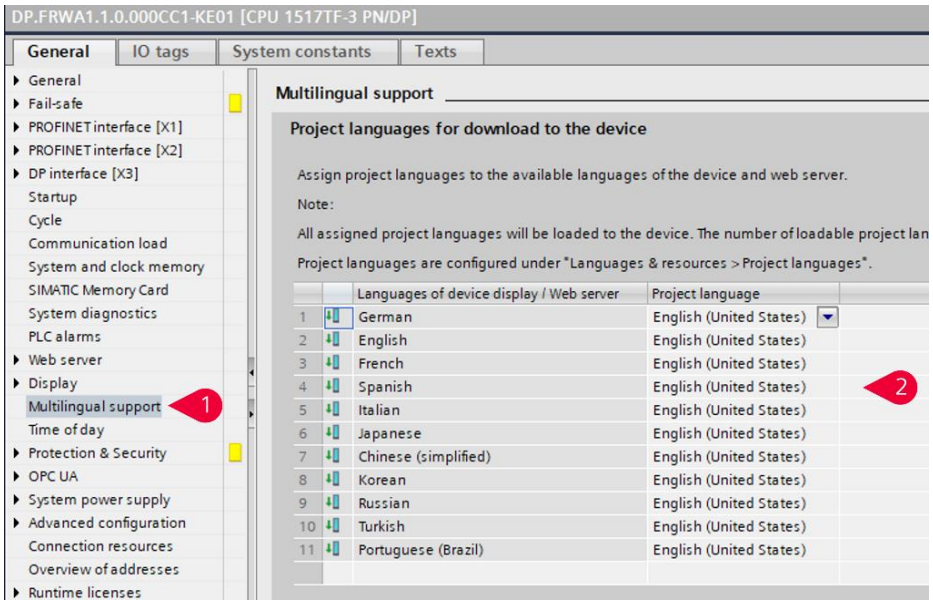


Figure 15-4: Project language assignment.

All assigned project languages will be loaded to the PLC. A maximum of three different project languages could be loaded to the PLC at the same time. If the system language is then switched, the corresponding project texts are changed to the assigned language.

NOTE Additionally, pay attention to the HMI language settings.

15.2.3. ProDiag Function blocks

ProDiag FBs are required to process the ProDiag supervisions. The first ProDiag FB is automatically created if you enable supervisions.

NOTE For detailed information on ProDiag, please refer to the TIA Portal documentation (chapter 12.26.) https://support.industry.siemens.com/cs/ww/en/view/109826862.

One ProDiag FB Version 2.0 can process up to 1.000 supervisions. For more ProDiag supervisions an additional ProDiag FB needs to be generated. In that case, ProDiag supervisions must be assigned to the specific ProDiag FB for processing.

You can structure the user program and the composition of the supervision messages according to the structure of the plant: concerning the number of supervisions, the symbolic names and assigning the "FB supervision instances".

To realize this structure all ProDiag FBs are stored in the folder of the corresponding unit and EM. After the blocks have been generated, the instance data blocks and the organization block "ProDiagOB" will be generated when compiling the software. All ProDiag FBs are called in this ProDiag OB.

Function blocks with supervisions (e.g. supervisions of operands) in the same module are now assigned to this ProDiag FB. This leads to a hierarchically structured supervision of the modules.

A detailed overview of the structure of the diagnosis with ProDiag is given in chapter Diagnostic Concept.

15.2.4. ProDiag License – PLC

The license "SIMATIC ProDiag S7-1500" is required for the usage of ProDiag. The number of used ProDiag licenses is set in the properties of the PLC. There it is also shown the number of ProDiag supervisions used for the configured PLC.

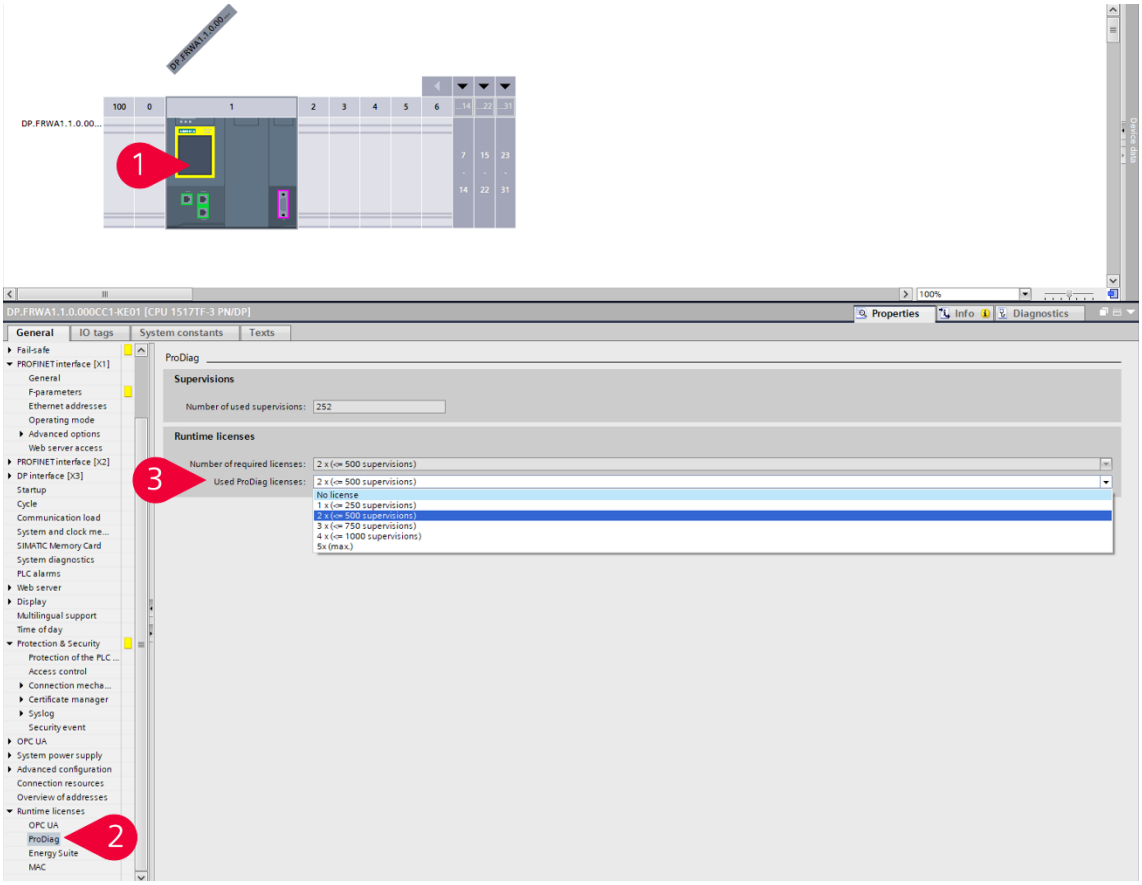


Figure 15-5: Configure the license in the PLC for the ProDiag functionality.

15.2.5. Enable acknowledge

For diagnostics purposes, some messages need to persist even if the trigger condition is no longer active. For such messages a dedicated acknowledge must be configured.

In the properties of the ProDiag block, open tab "Supervision settings" then "categories". Enter the reset / acknowledge signal which is assigned to the HMI acknowledge button as the acknowledge tag for category "AV/AM, FV/FM" (Alarms, Failsafe alarms).

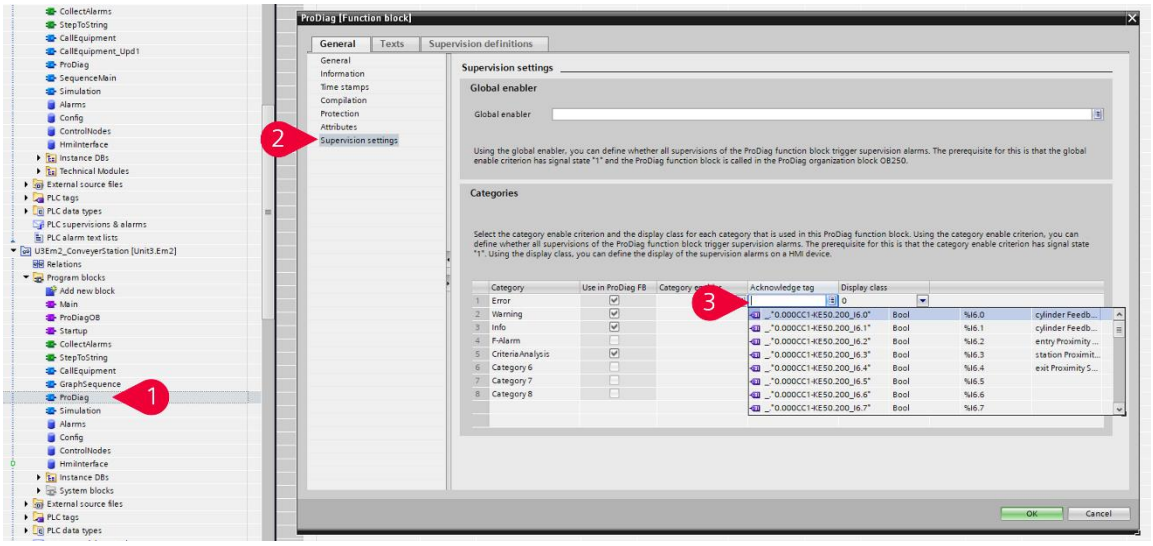


Figure 15-6: Activation of the acknowledge tag in the ProDiag properties of the function block.

15.2.6. PLC supervisions & alarms

In "PLC supervisions & alarms" (1) the view "Global supervisions" (2) show all details of the existing global supervisions. Here you can find all the supervisions that has been added manually by the system integrator.

The "Supervision instances" (3) view shows all the supervisions that are instantiated by adding FBs that already contain predefined supervisions.

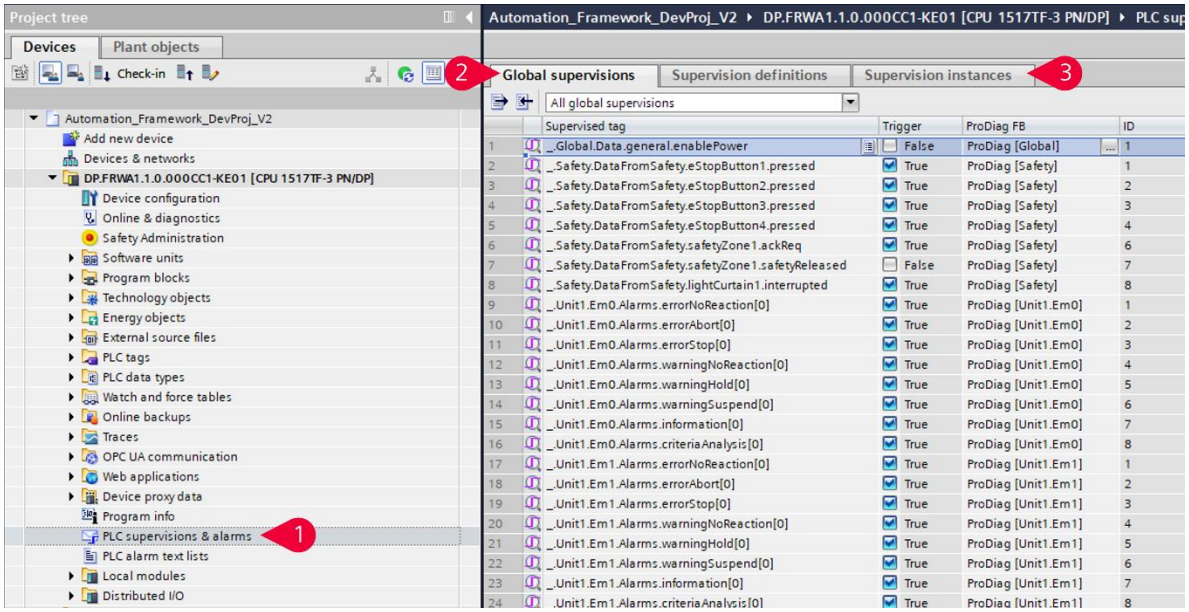


Figure 15-7: PLC supervisions & alarms configuration.

15.2.7. Common alarm class settings

For each project in TIA Portal, it is possible to create specially defined alarm classes in addition to the standard alarm classes "Acknowledgment" and "No Acknowledgment". Names, display names, priorities and whether the alarms must be acknowledged, can be set for new defined alarm classes.

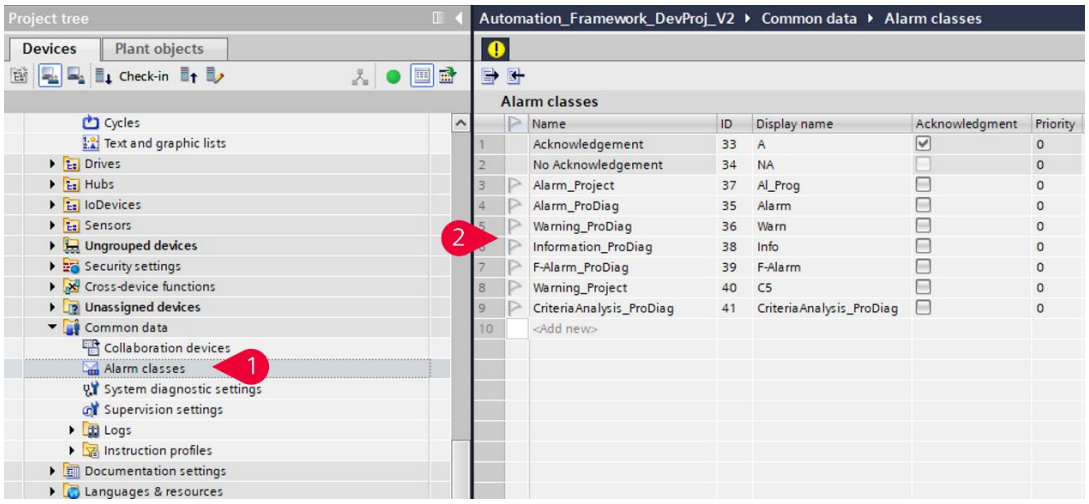


Figure 15-8: AF Alarm Classes configuration.

The AF provides the following user defined Alarm classes:

- Alarm_Project
- Alarm_ProDiag
- Warning_ProDiag
- Information_ProDiag
- F-Alarm_ProDiag
- Warning_Project
- CriteriaAnalysis_ProDiag

The Alarm classes can then be assigned to the diagnostic settings described in the following chapters.

15.2.8. System diagnostic settings

The PLC already implements certain diagnostic functions. The previously defined alarm classes can be assigned to the different categories of system diagnostics. For this purpose, first select the system diagnostic settings area in the Common data folder (1) and then adjust the assignment (2). Furthermore, is also possible to deactivate the respective category for the project.

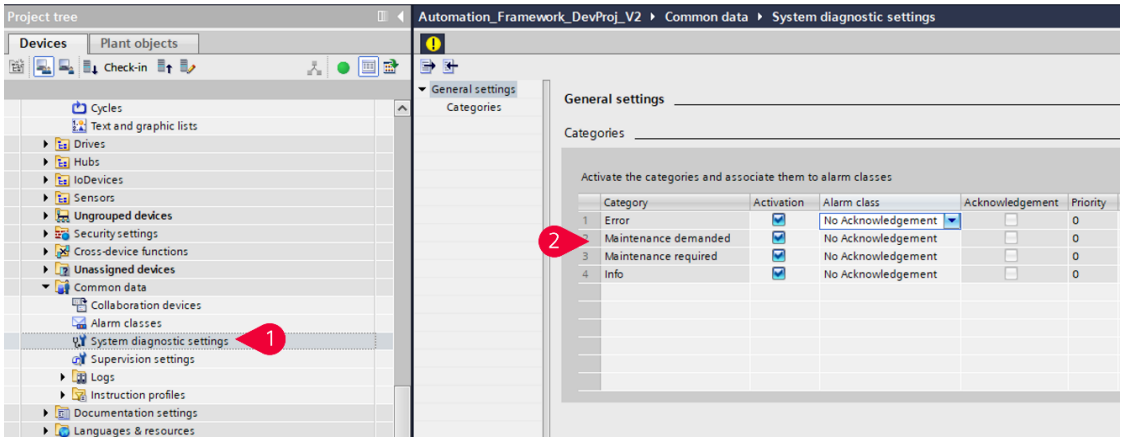


Figure 15-9: System diagnostic settings in the AF.

15.2.9. Supervision settings

The final project-wide configuration of the diagnostics, is the setting of the used ProDiag supervisions. The supervision settings are selected in the common data folder (1). The common alarm classes are assigned to the supervisions categories (2). There are eight categories available, which can be activated and named accordingly. In addition, it is possible to define up to 16 further subcategories in two subcategory groups (3). How this categories are used within the AF, will be explained in chapter Diagnostic Concept.

The structure of the alarm texts is also defined here (4). The texts can be composed of different parameters and customized individually for GRAPH, basic Supervisions, Supervisions with an error message and Supervisions with a text message. It is also possible to display globally created and instantiated supervisions differently (5).

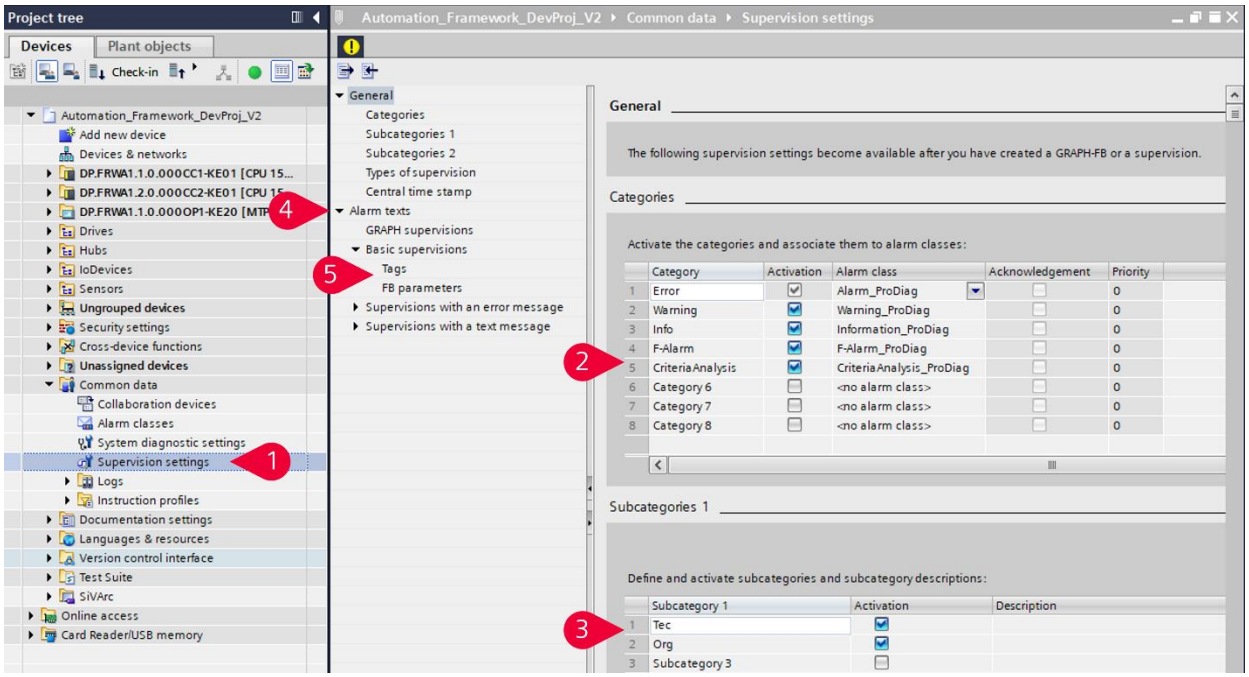


Figure 15-10: Supervision settings within the AF.

The configuration of the alarm texts for the Automation Framework is shown in the following figure (Supervision of a global Tag (1), Supervision of a FB parameter (2):

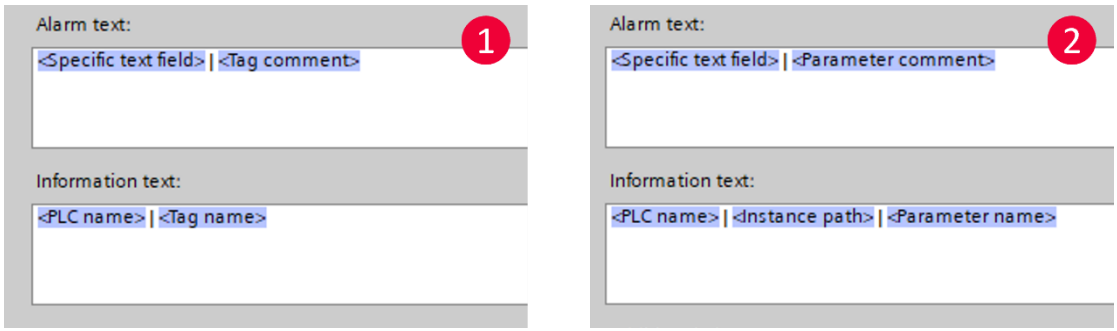


Figure 15-11: Supervision alarm text settings.

15.2.10. HMI alarm class settings

The HMI alarms can also be freely configured, and the alarm classes created, can be assigned. Different colors can be set depending on the status of the alarm. Via HMI alarms within the configured visualization device (1), you can access on the alarm classes tab (2) the configuration overview (3).

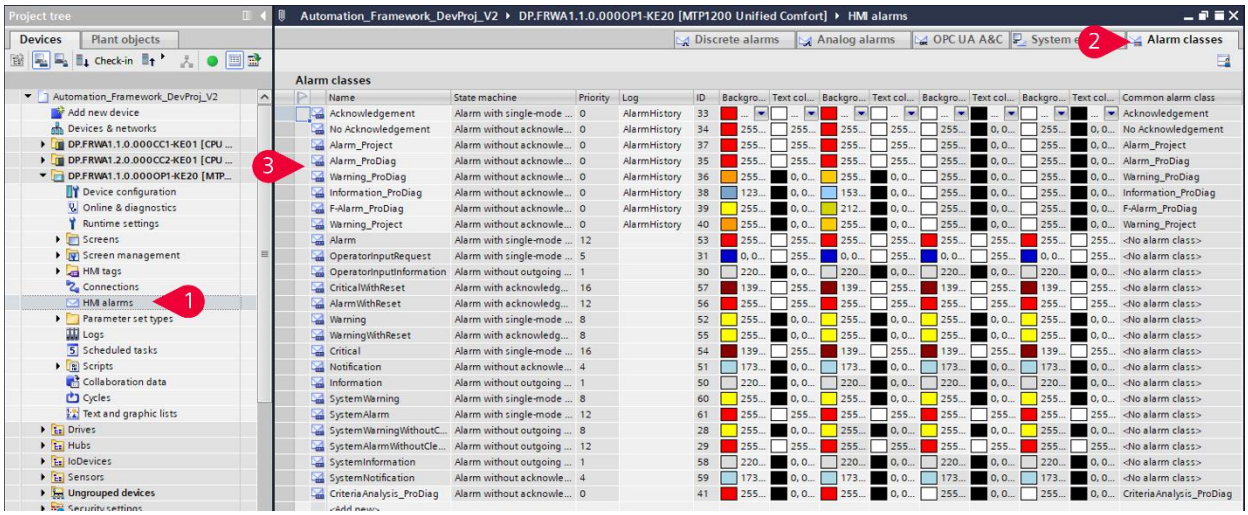


Figure 15-12: HMI alarm setting for the AF.

The following default background color settings for the AF are:

- Alarm: red
- Warnings: orange
- Information: blue
- F-Alarms: yellow

15.2.11. Customized supervision configuration

Supervisions are created directly within the interface description of an FB or in global DBs (1). In the Properties area (2) the Supervision configuration can be made (3). The supervision settings (Category, subcategory 1, subcategory 2) must be selected when a ProDiag supervision is created (4). Some settings are derived directly from the supervision type and the global supervision settings. The following chapter describes in detail how the configuration of the supervisions within the AF is realized.

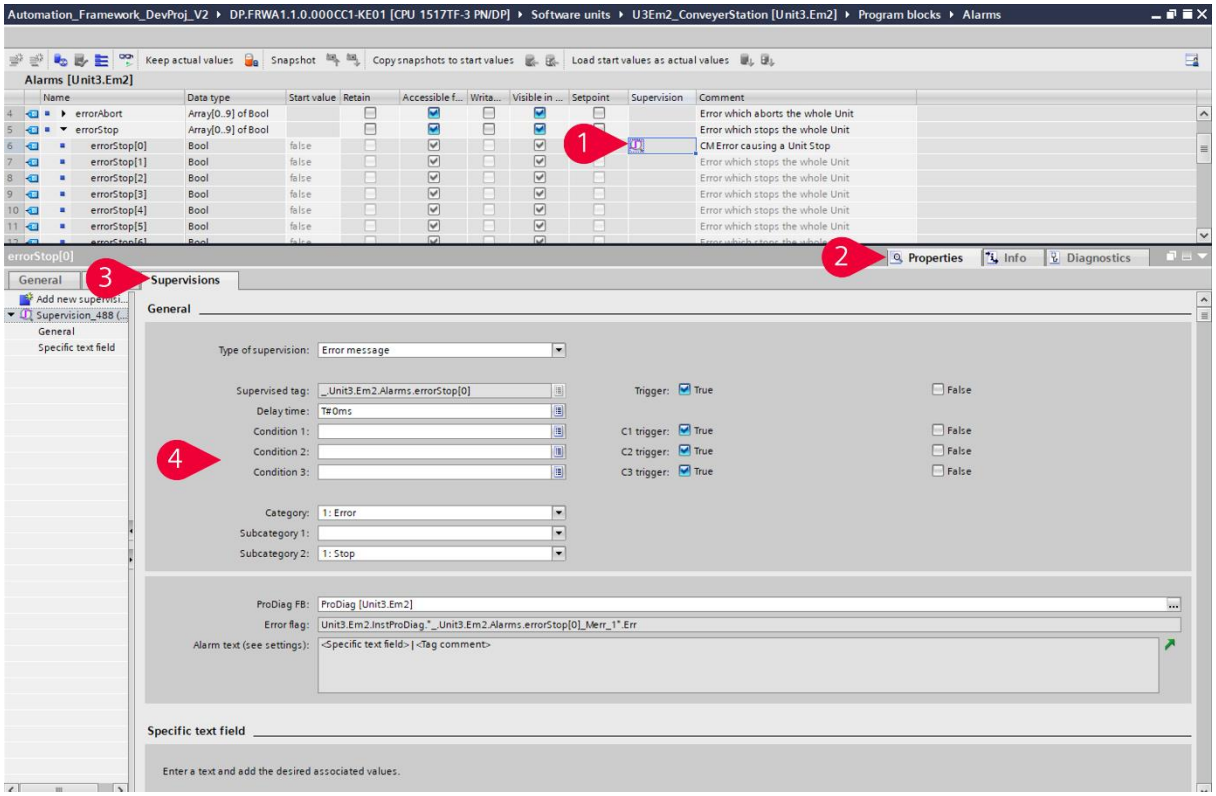


Figure 15-13: Customized ProDiag settings for the AF.

15.3. Diagnostic Concept

The holistic alarm concept within the Automation Framework, provides a standardized and modular approach for creating alarms in a machine. The reason behind this is to enable the operator to identify the cause and location of the failures in a technical system. This can result in increased production availability and decreased downtime.

In the context of the Automation Framework, it implies the need for a well-defined structure that outlines how the alarm concept's relationships are integrated into the functional architecture of the Framework. This includes the alarming of the different layers of the ISA-88 model from CM to Unit-Level.

The implemented alarm concept consists of two components for machine alarming. The first component involves predefined alarms within blocks that already have the concept by default, such as the LBC on the CM Level. The second component are blocks such as the "CollectAlarms" block, which allows for the easy definition of unique alarms. It is also possible to supervise specific data related to the step sequence and include these supervisions to the diagnostic concept.

These different supervisions are used to map the different alarms to the machine's process, allowing them to directly influence the behavior of the machine. This process is handled by the "LAF_ManageAlarms" block. This block sets the Interface signals of the different layers to influence the LPML Unit Mode State Manager. The figure below presents the influences of each level on the higher instance by the implementation of the diagnostic concept. All blocks shown in yellow, or purple, are library objects and do not require any further programming.

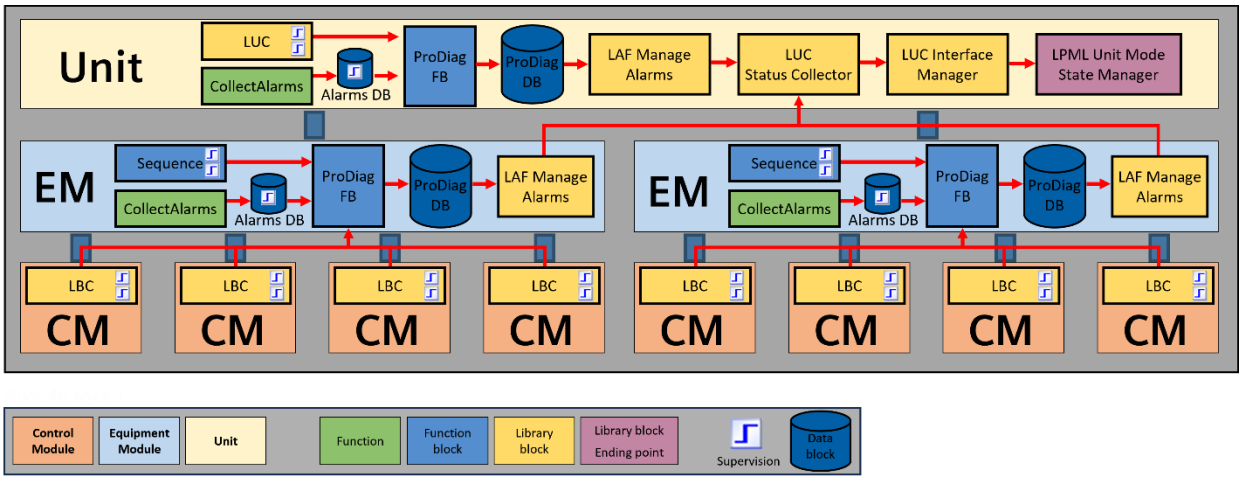


Figure 15-14: Architecture of the Diagnostic Concept.

15.3.1. Alarm Configuration

The Automation Framework provides different categories and subcategories for the supervisions to define how the alarms influence the process. For this purpose, five user-specific alarm classes were created in the project and assigned to the main categories C1-C5 of the process diagnostics. The process of editing these settings is described in chapters Supervision settings and Common alarm class settings. In addition, there are five sub-categories that are used for the effect of the alarm on the process.

Alarm Classes	Categories	Subcategories
Error_ProDiag	Error (C1)	Stop (SC2.1)
Warning_ProDiag	Warning (C2)	Continue (SC2.2)
Information_ProDiag	Information (C3)	Abort (SC2.3)
F-Alarm_ProDiag	F-Alarm (C4)	Hold (SC2.4)
CriteriaAnalysis_ProDiag	Criteria Analysis (C5)	Suspend (SC2.5)

Table 15-1 Supervision settings of the Diagnostic Concept

The assignment of the category and subcategory to a supervision influences how the alarm is displayed and to what extent the occurrence affects the process of the unit.

15.3.2. Alarm Implementation

To implement specific alarms of the different levels the function "CollectAlarms" can be used. To add an alarm, the following steps must be taken:

- Define a condition that should trigger the alarm and place it in the section of the "CollectAlarm" block according to the category and subcategory of the alarm.
- Insert the assignment and connect a free element of the array that is related to the combination of category and subcategory from the DB "Alarms".

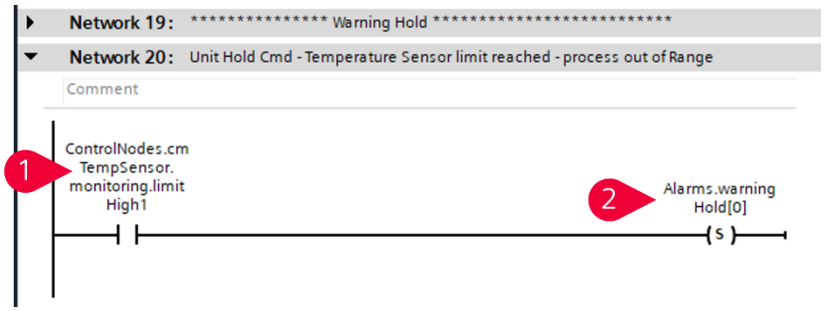


Figure 15-15: Assignment of the signal to be supervised (FC "CollectAlarms").

- Open the DB "Alarms", select the assigned element and add a comment that is describing the alarm. The comment can be added for all the languages that has been defined for the project.

Figure 15-16: Comment definition.

- Add a new supervision to the variable. If a supervision already exists within the same array, it can be transferred to the desired element using copy and paste and the next steps can be skipped.

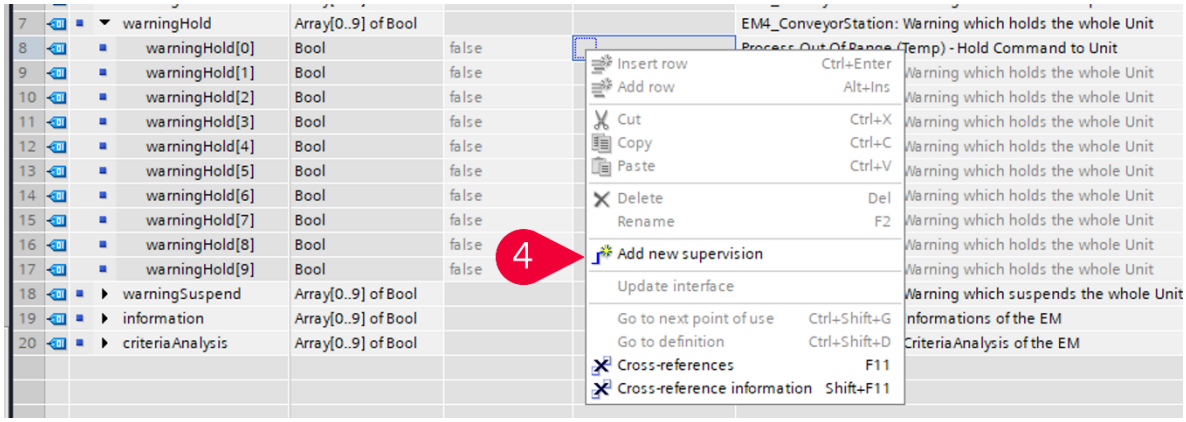


Figure 15-17: Creating a supervision in the Alarm DB.

- Configure the Supervision by selecting the matching category and subcategory.

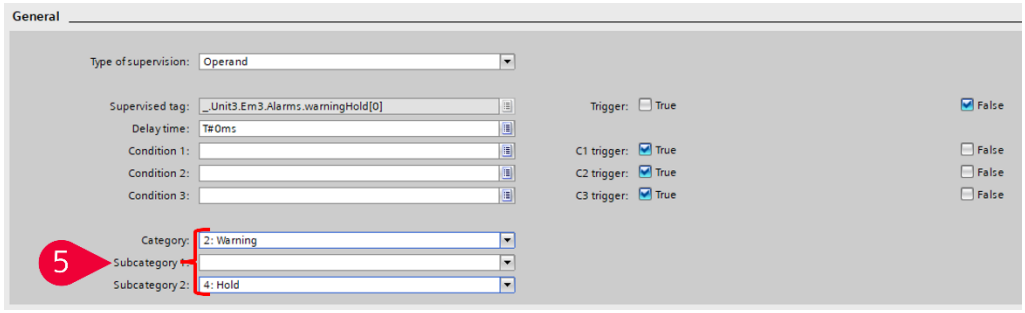


Figure 15-18: Supervision category and subcategory definition.

- Connect the Reference Designator of the level at the Tag 1 (SD_4) Input in the specific text field area.
- Define the usage of Tag 1 (SD_4) within the specific text field.

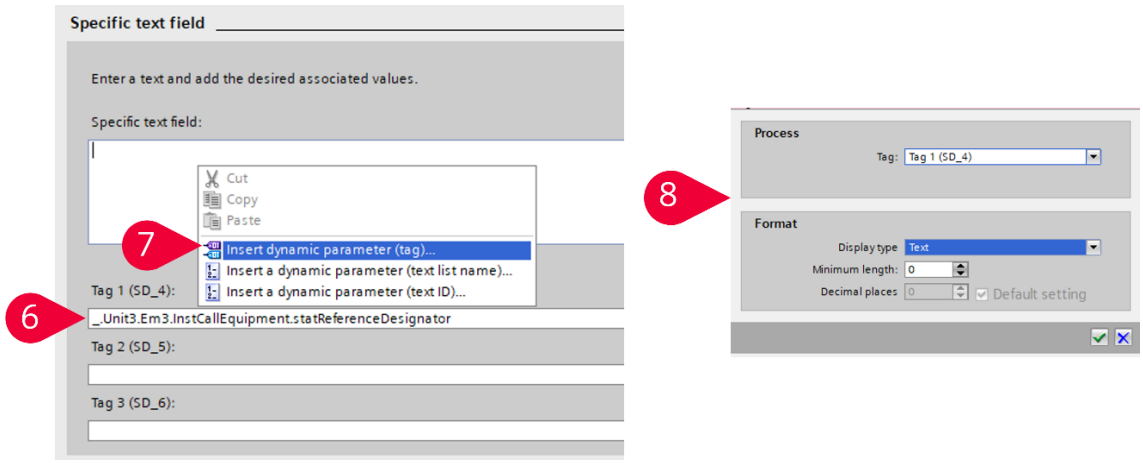


Figure 15-19 Specific Text field configuration.

It is also possible to create the supervisions directly on the signal to be monitored and to define the corresponding category and subcategory. The correct alarms definition must be considered when analyzing the process. Figure 15-20: Creating a supervision at the monitored signal itself. shows how the alarms can be implemented directly in the signal which is monitored without mapping the alarm to the DB "Alarms".

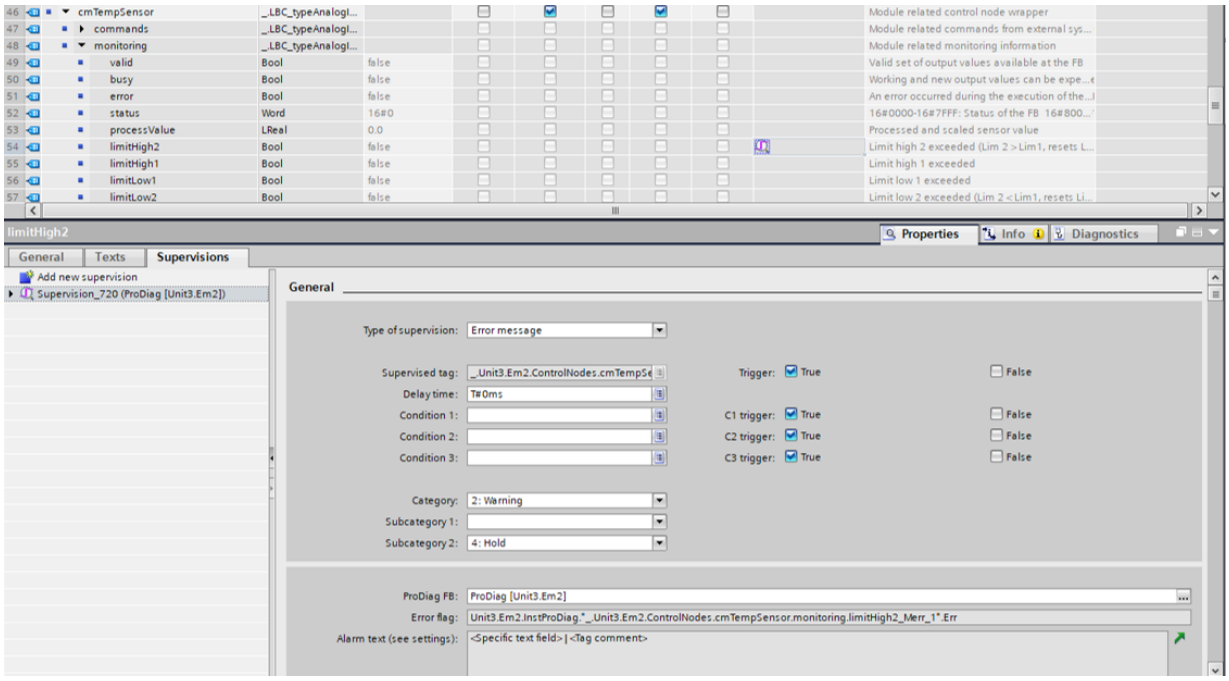


Figure 15-20: Creating a supervision at the monitored signal itself.

If there is already a supervision for the signal existing, as it is the case with the LBC blocks, only a reaction to the process can be added within the "CollectAlarm" block. The command to the unit will be set if the condition is true but no additional alarm will be triggered from the DB "Alarms".

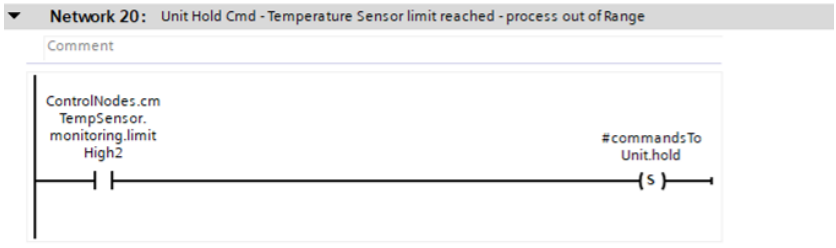


Figure 15-21: Adding a reaction to a tag that is already supervised.

15.3.3. Error handling within the LBC Library Blocks

Error Detection

The LBC block have internally error detection and analysis so that a status with the current cause is given to the user. By handling the error analysis, it becomes easier for the user to find the error source and solution.

When an error occurs, an alarm should be generated, and the correct alarm message should be displayed. If an error occurs (e.g. error in block operation), the error must be given as output, and the status set (e.g. 16#8001). Constants are assigned to the status code, which contain the specific information about the error (e.g. 16#8001: Error: Wrong operation of the function block).

These constants are included in a PLC alarm text list LBC_AxisControl_TecPlc.

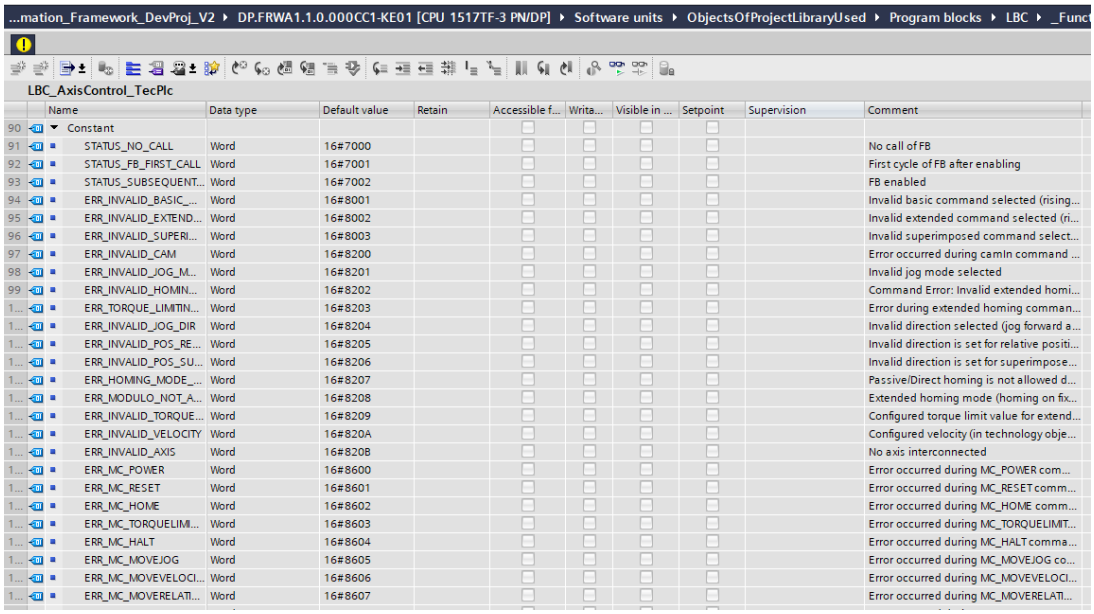


Figure 15-22: Status code and constant for error handling in the LBC_AxisControl_TecPlcCn block.

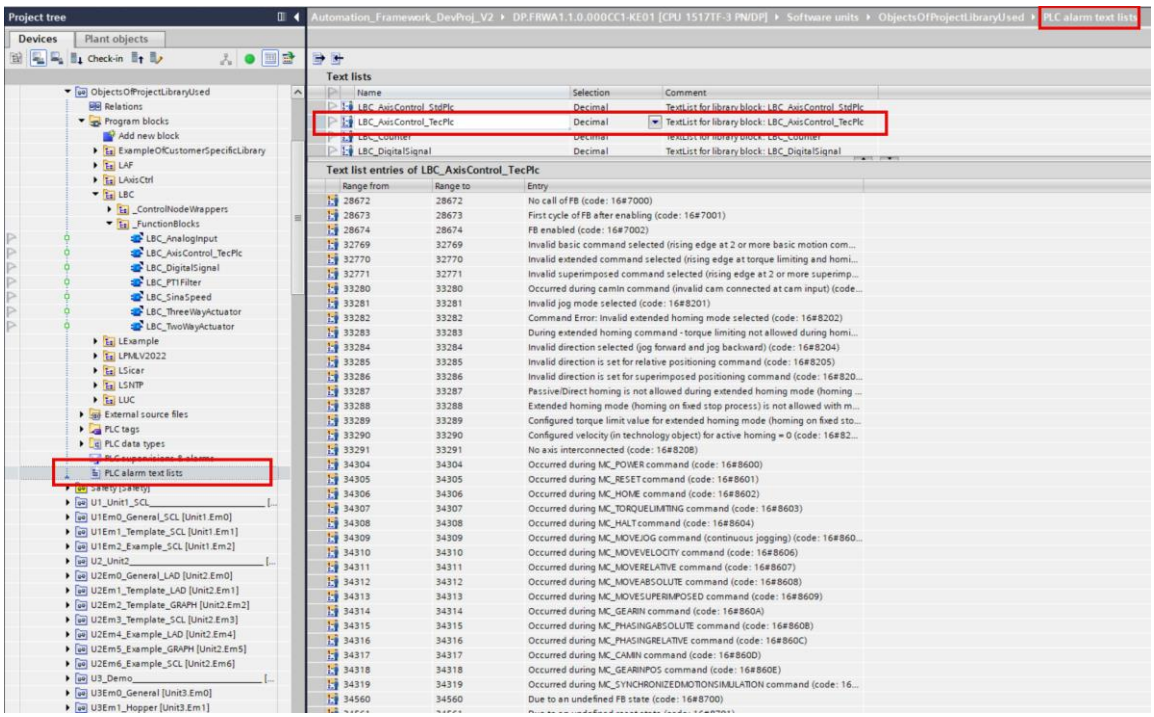


Figure 15-23: Diagnostic information of LBC_AxisControl_TecPlcCn represented in a PLC alarm text list.

Alarm Generating

The error bit in the "statProcessValues" area of the FB "LBC_AxisControl_TecPlcCn" is configured with a ProDiag supervision and the status refers the corresponding alarm message.

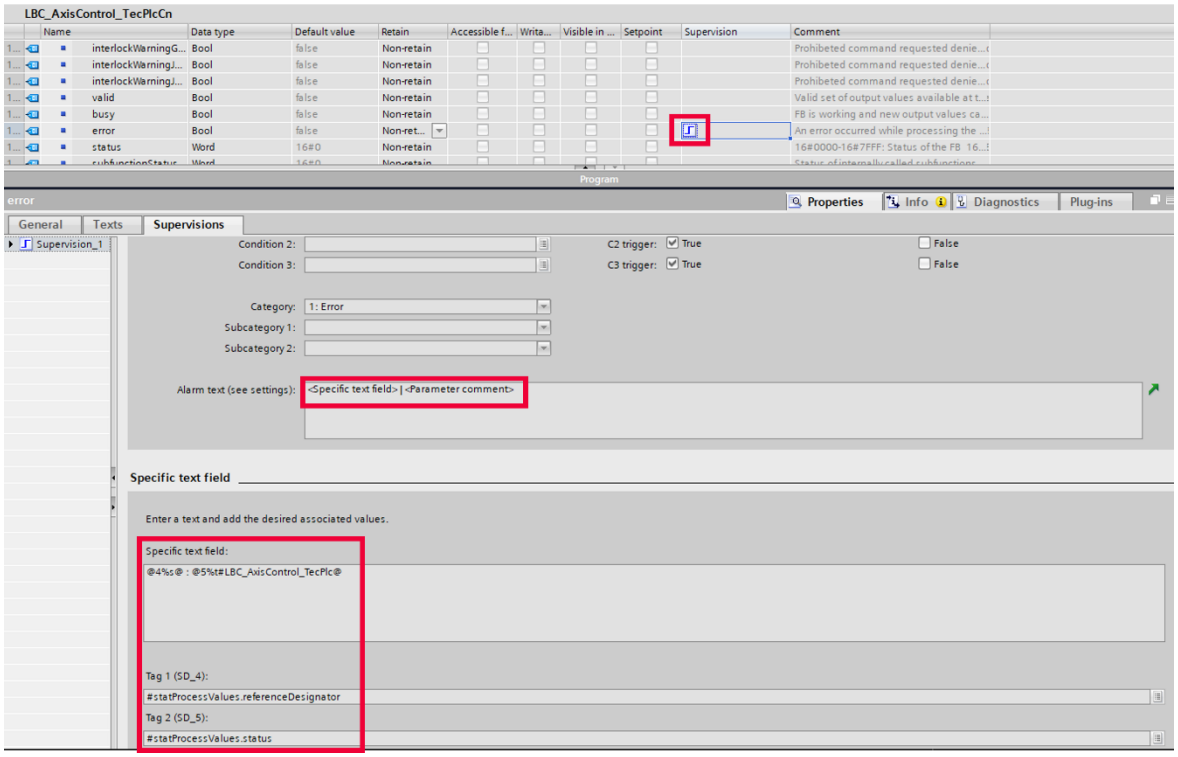


Figure 15-24: ProDiag supervision configuration.

In the properties of the selected ProDiag supervision, specific information about the alarm text is shown and can be configured. In this example, the displayed alarm message includes the specific text field and the tag comment.

In the specific text field, the associated values are used. In the AF project, a dynamic parameter (text list) is configured.

<"Tag 1 (SD_4)": Text list: LBC_AxisControl_TecPlc >

The alarm message is generated using a provided text list in combination with the status word of the FB.

These alarm messages generated in the EMs are forwarded to the unit via the EM Status UDT and visualized via a status display in the HMI. The unit, summarizes the signals generated with ProDiag and creates a status that is also including whether an EM is coupled to the unit or not. The safety status of the unit is also included in this collection.

NOTE More information about ProDiag can be found in the following entry https://support.industry.siemens.com/cs/ww/en/view/109740151.

16. Simulation

16.1. Overview

S7-PLCSIM products

A comparision of the S7-PLCSIM products and their technical features and differences are available here: https://support.industry.siemens.com/cs/ww/en/view/109955578/173421632907

NOTE

Additional software packages and licenses are required for simulation with PLCSIM Advanced, SIMIT and HMI simulation. More details see chapter Hardware and Software Requirements.

PLC simulation with S7-PLCSIM

S7-PLCSIM offers tools for troubleshooting, validation and verification of the implemented logic in the PLC program. Simulation tables, allow for the monitoring and modification of PLC inputs and outputs as well as DB tags. Simulation sequences can be used to simulate an external process interacting with the PLC program. In addition, function blocks can be programmed to provide the expected behavior of the process within the PLC.

This is a manual approach for testing the logic in the PLC program. The quality of the simulation depends on how accurate the process is modeled in the simulation sequences of S7-PLCSIM or programmed as function block within the PLC.

Simulation of system behavior with S7-PLCSIM Advanced and SIMIT

Comprehensive validation and verification of the PLC program with simulation of the periphery behavior and reaction of sensors and actors as well as simulation of the technological behavior of the process. In this scenario, the PLC program is simulated with S7-PLCSIM Advanced, and the periphery and the process are simulated with SIMIT.

This approach makes possible to detect potential issues early in the development phase, as well as to test the robustness of the PLC program in different fault and error scenarios without the risk of personal or material damage.

Advantages comparison of simulation options

Advantages	S7-PLCSIM	S7-PLCSIM Advanced	S7-PLCSIM Advanced with SIMIT
Debugging and monitoring of PLC program and tags	**Yes**	**Yes**	**Yes**
Validation and verification of PCL program without real hardware	**Basic** PLC logic, basic I/O simulation	**Advanced** PLC program, basic I/O simulation	**Comprehensive** PLC program, periphery behavior of drives, sensors, actors, safety, etc.
Test level options	**Unit tests** only manual testing	**Integration tests** optional automatic unit tests with Test Suite Advanced	**System tests** optional automatic simulation tests with SIMIT Rapid Tester
Validate communication to external systems, e.g. MES, M2M communication, etc.	**Limited** only softbus between max. two PLCs and HMI	**Extensive** various protocols, secure communication	**Extensive** various protocols, secure communication
Scope of simulation	**Basic** PLC logic, digital and analog I/O	**Advanced** PLC program, digital and analog I/O communication	**Comprehensive** Complex processes, mechanical interactions, system behavior
Simulation of faults and error	**Limited** Simulation of fault and error routines in PLC program	**Limited** Simulation of fault and error routines in PLC program	**Extensive** Simulation of fault and error scenarios to test robustness of the system
Process simulation	**No** applicative solution	**No** applicative solution	**Yes** Simulation of process behavior, including fluid dynamics, temperature control, etc.
Operator / maintenance training	**Limited** HMI interactions	**Limited** HMI interactions Webserver	**Extensive** HMI interactions, Webserver, physical buttons, etc.
Integration with other simulation tools	**No** not applicable	**Yes** e.g. SIMIT, Simcenter Amesim, Tecnomatix Process Simulate	**Yes** e.g. NX MCD, SINAMICS DriveSim Basic, Tecnomatix Process Simulate, Tecnomatix Plant Simulation
Quality assurance	**Low** detection of bugs in PLC logic and blocks	**Medium** detection of bugs in PLC program and blocks, detection of incorrect communication settings	**High** detection of potential issues early in the development phase, reduces risks during actual commissioning
Setup complexity	**Low** easy to set up, no additional training required	**Low** easy to set up, no additional training required	**High** More complex to set up behavioral model/digital twin, additional training and expertise required to fully utilize its capabilities
Ease of use	**High**	**High**	**Moderate**

Advantages	S7-PLCSIM	S7-PLCSIM Advanced	S7-PLCSIM Advanced with SIMIT
Costs	**Low** no additional license required	**Medium** additional license required	**High** additional licenses required

Table 16-1: Advantages comparison of simulation options

16.2. PLC and HMI integration of simulation signals

PLC integration

All signals regarding simulation are centrally managed by FC "CallSimulation" in the software unit of central functions and distributed to the units via the global data. Each software unit uses simulation tags are organized in separated tag tables and programmed in separated function blocks, making it easier to delete the simulation from the PLC program. Figure 16-1 shows this program architecture for central functions, safety unit and equipment module "Hopper" exemplary.

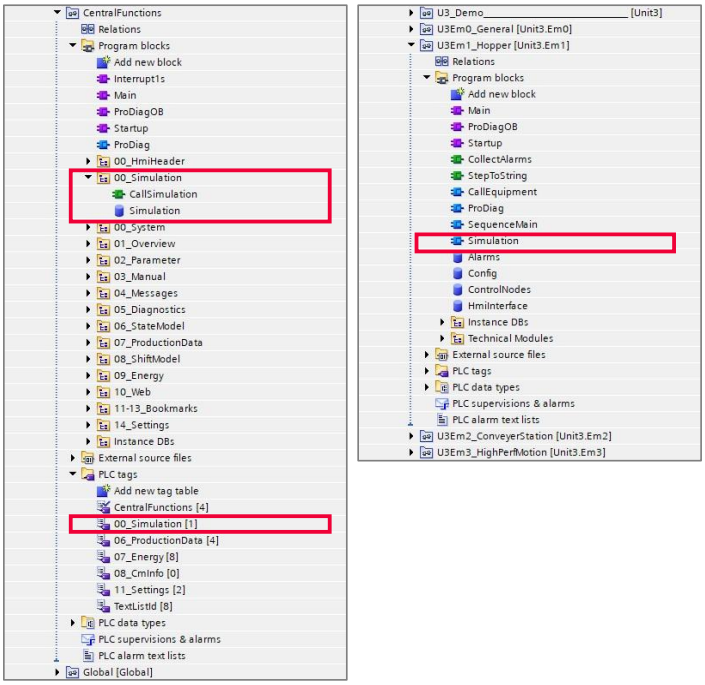


Figure 16-1: PLC blocks and tag tables regarding to simulation

HMI integration

All simulation tags of the global data are also centrally organized in one HMI tag table, which makes it easier to delete the simulation in the HMI. The HMI tags are linked to HMI objects and implemented in different HMI screens in the AF by default. The PLC simulation with function blocks can be activated or deactivated by the simulation buttons (1).

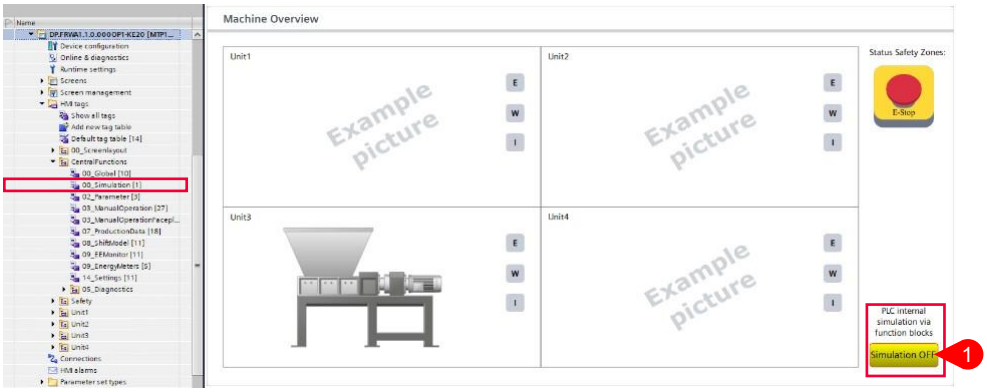


Figure 16-2: HMI objects and tag tables regarding to simulation

16.3. PLC simulation with PLCSIM

The simulation function blocks in the PLC program are activated by default in the AF example project. While the PLC simulation is active, the technology objects from type axis are set to simulation mode. Following the steps below allow us to use the PLC simulation with both the simulation tables and internal simulation function blocks.

1. Start S7-PLCSIM.
2. Open the provided preconfigured Workspace (1) & (2). Select the superordinate folder "AF_PLCSim_Workspace_V1_2" (3) and open it (4).

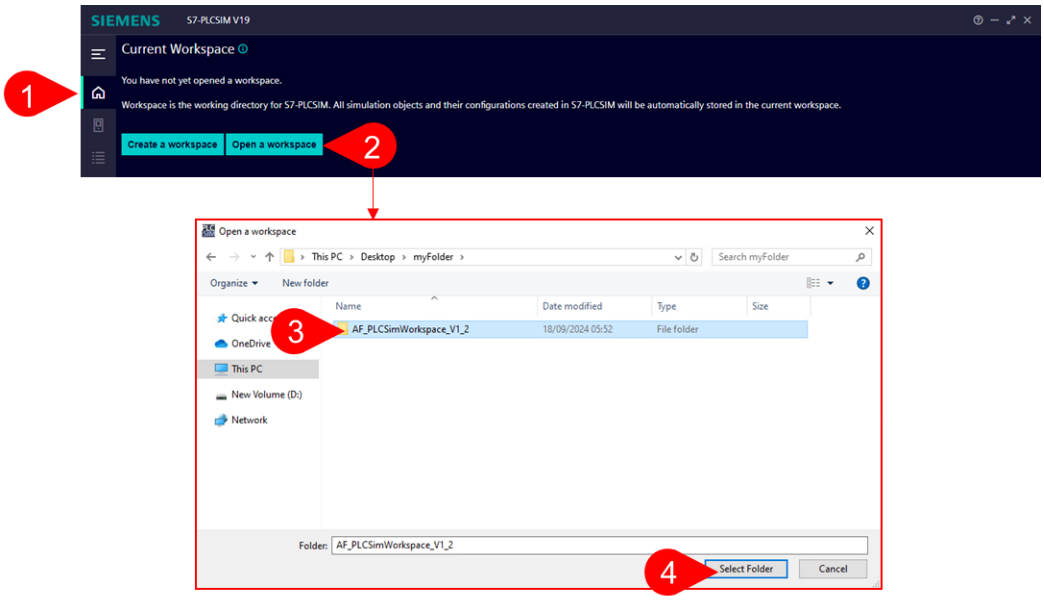


Figure 16-3: How to open a saved PLCSIM workspace

3. Start the simulated PLC instances by following the steps in Figure 16-4.

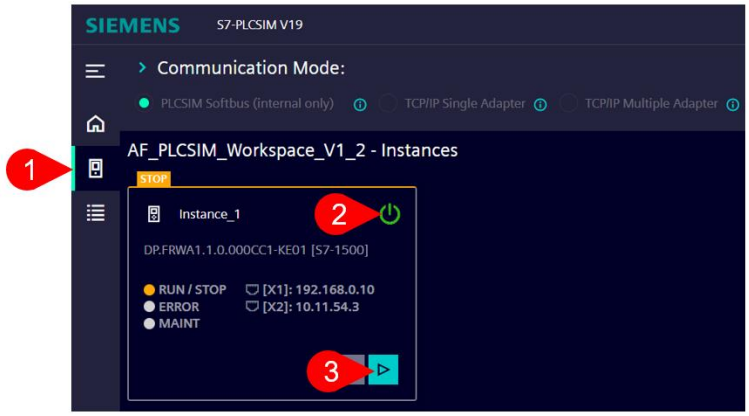


Figure 16-4: How to start simulated PLC instances in S7-PLCSIM

4. Go to TIA Portal and download the PLC (1). PLCSIM is preselected as PG/PC interface when a simulated PLC instance is currently running (2). This step is necessary when changes in the project were made, otherwise the preconfigured instances already contain the project and this step can be skipped.

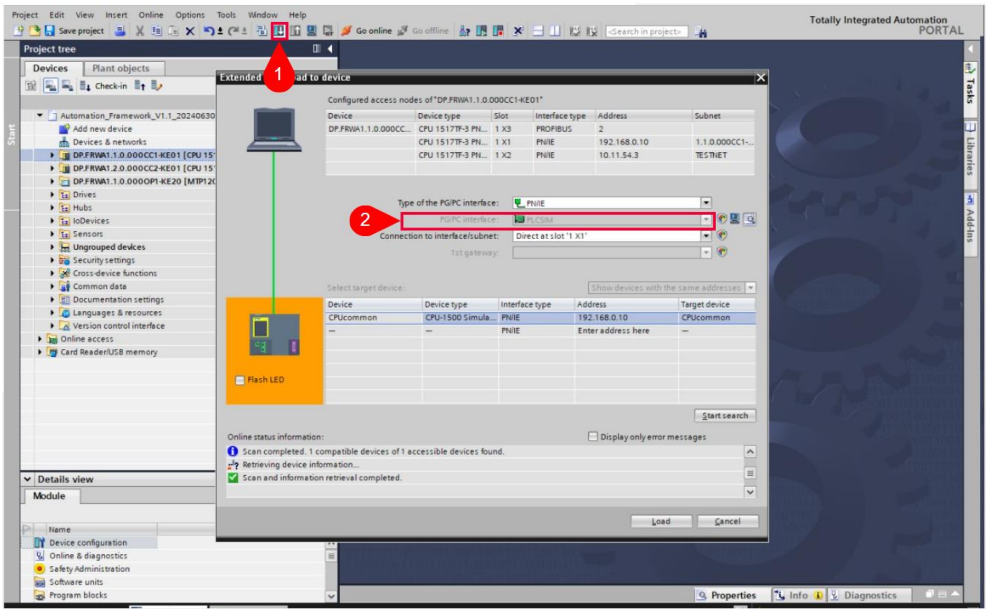


Figure 16-5: How to download the PLC into a simulated PLC instance

5. Optional: Start the simulation of the HMI (1). The WinCC Unified runtime will open automatically in the browser. HMI simulation may require additional software and license depending on the engineered power tags.

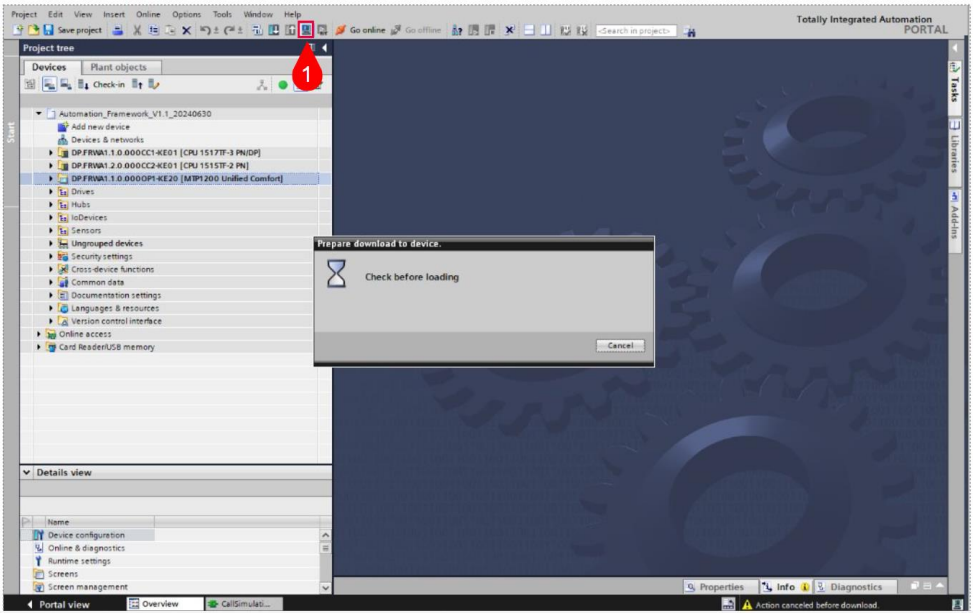


Figure 16-6: How to start the simulation of the HMI application

6. Go back to S7-PLCSIM and open the simulation tab (1). The required tags for simulating the fail-safe inputs are preconfigured. Start the simulation table to enable the online view (2).

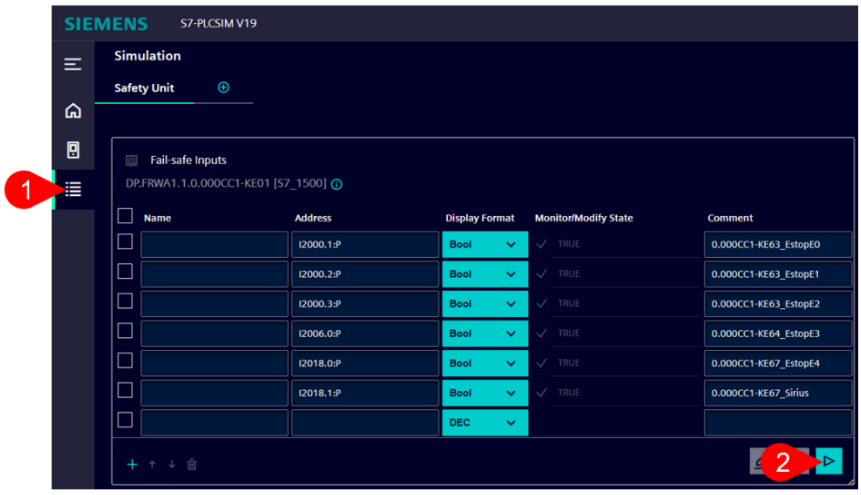


Figure 16-7: How to start the simulation table in S7-PLCSIM

7. For example, test and validate the logic of the safety program by modifying the fail-safe inputs in the simulation table (1). The state of the fail-safe input for an emergency button is set to "FALSE" (2 & 3) which triggers the STO functionality of all drives, by changing the value of STO to "FALSE" (4).

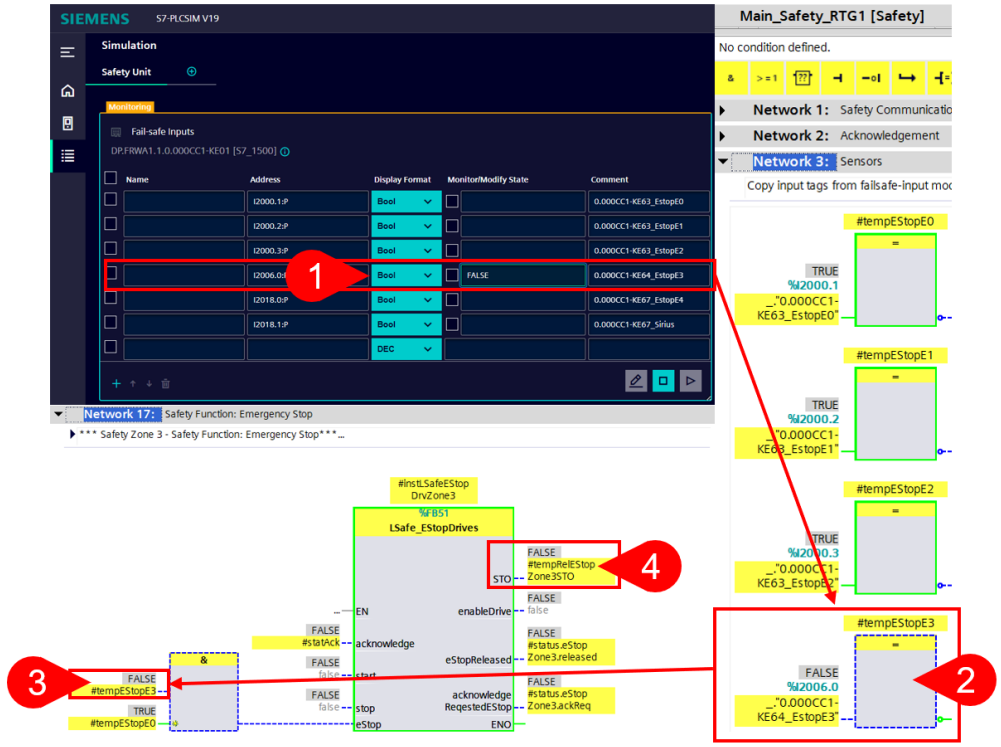


Figure 16-8: Reaction in the safety program after simulated state change of fail-safe input

Result

The correct logic of the safety program is validated. However, the correct response of the drive is not. It must also be simulated manually, to validate the response in the PLC program.

16.4. Simulation of system behavior with SIMIT

NOTE The provided example SIMIT project is based on the Demo Unit and the subordinated demo equipment modules EM Demo Implementation – Hopper, EM Demo Implementation – Conveyor, EM Demo Implementation – High Performance Motion.

NOTE The provided SIMIT project starts the simulated PLC instances in S7-PLSIM Advanced in modus PLCSIM softbus. In this mode the OPC UA server and the webserver are not accessible. In order to test the communication, start a simulated PLC instance in modus TCP/IP single adapter directly in S7-PLCSIM Advanced.

When using SIMIT, it is recommended to implement a life bit in the SIMIT project that is connected to a digital input in the TIA Project. The life bit is always set to "TRUE" while SIMIT is running. This connection allows to trigger reaction in the PLC program (e.g. show info message in HMI, disable simulation FBs in the PLC program, etc.).

In case of the Automation Framework, the PLC simulation with PLCSIM is deactivated and the HMI objects (see Figure 16-2) are hidden. This feature is implemented in the FC "CallSimulation" in the software unit "CentralFunctions". Follow the next steps to use the PLC simulation with SIMIT:

1. Start SIMIT SP and open the provided preconfigured project.
2. Start the simulation via the play button (1). S7-PLCSIM Advanced and the simulated PLC instances are started automatically in the background.

Figure 16-9: How to start the simulation in SIMIT

3. Go to TIA Portal and download the PLC. PLCSIM is preselected as PG/PC interface when a simulated PLC instance is currently running.
4. Start the simulation of the HMI. The WinCC Unified runtime will open automatically in the browser. HMI simulation may require additional software and license depending on the engineered power tags.

5. Figure 16-10 shows a running simulation of EM Demo Implementation – Conveyor in SIMIT (2). A running simulation is highlighted in orange. The simulation allows to validate I/Os (3) and functions blocks for control modules (e.g. LBC_TwoWayActuator, LBC_AnalogInput (4)), verify the feedback signals to the OMAC PackML state machine or test the sequence.

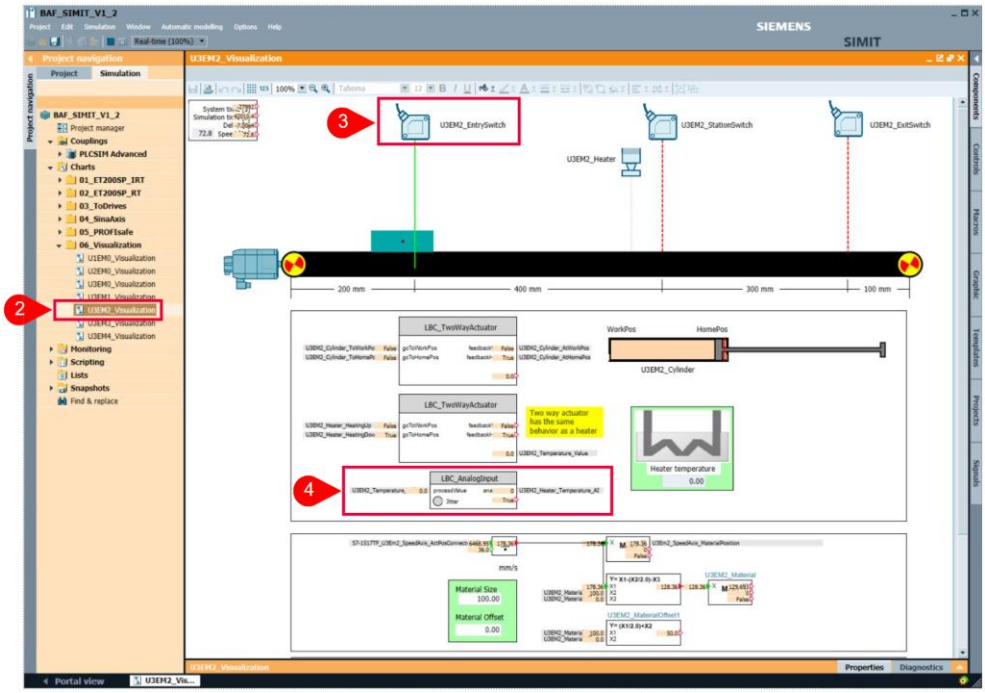


Figure 16-10: Running simulation in SIMIT

6. Figure 16-11 shows the validation of the safety program as typical simulation use case. The emergency stop button is pressed (5). The correct response is validated with the behavior model of the PROFIsafe telegram 30. In this example, the value of STO goes to "FALSE" after the emergency stop button is pressed and the power removed status from the drive responses with the correct value "TRUE" (6).

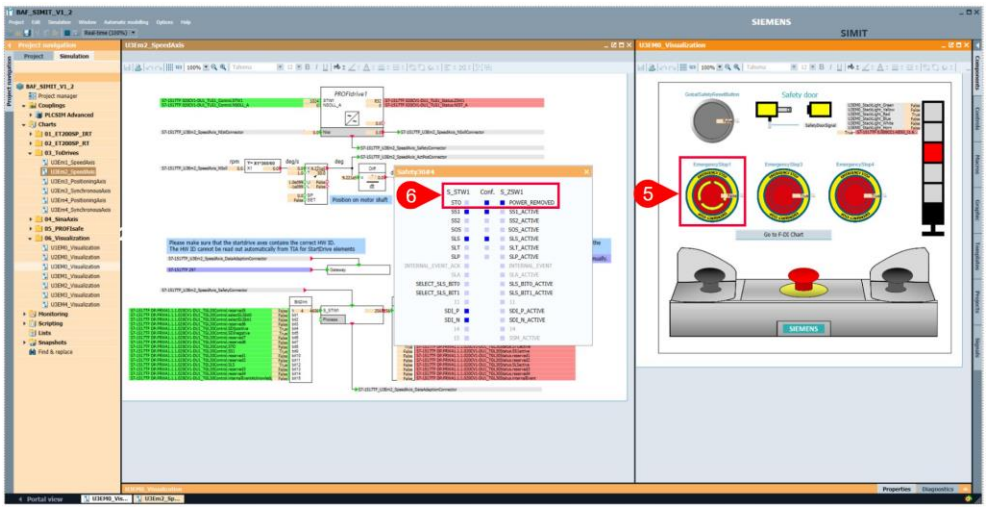


Figure 16-11: Simulation use case – safety program validation of STO response from drive triggered by emergency stop

Result

The correct logic of the safety program and the correct reaction in the PLC program based on the drive response is validated.

17. User Management

17.1. General

To centrally administrate users and user groups across a factory network and use domain users as well, TIA Portal offers a ready to run user management component (UMC). TIA Portal already provides a range of preconfigured roles. The customer can add additional roles und configure them according to his requirements.

The User/ Groups-configuration within the UMC server can be imported into your TIA Portal projects, so this configuration is a single action independent from the number of devices.

To be able to manage users in your project, you can perform the following actions:

- Open the editor "Security settings > Users and roles" (1).
- Open the Roles Window (2)
- By selecting one of the users (3), the settings of the user are shown in the bottom screen
- Open the Runtime Rights Window (4), there you can see the Runtime rights of the role (5).
- To assign roles to a user, follow these steps:
- Open the "Users" tab (6).
- Select the user to whom you want to assign roles (7). Note that you cannot use multiple selection.
- Enable the desired roles (9) in the "Assigned roles" section (8).

The procedure described is illustrated by the pictures below:

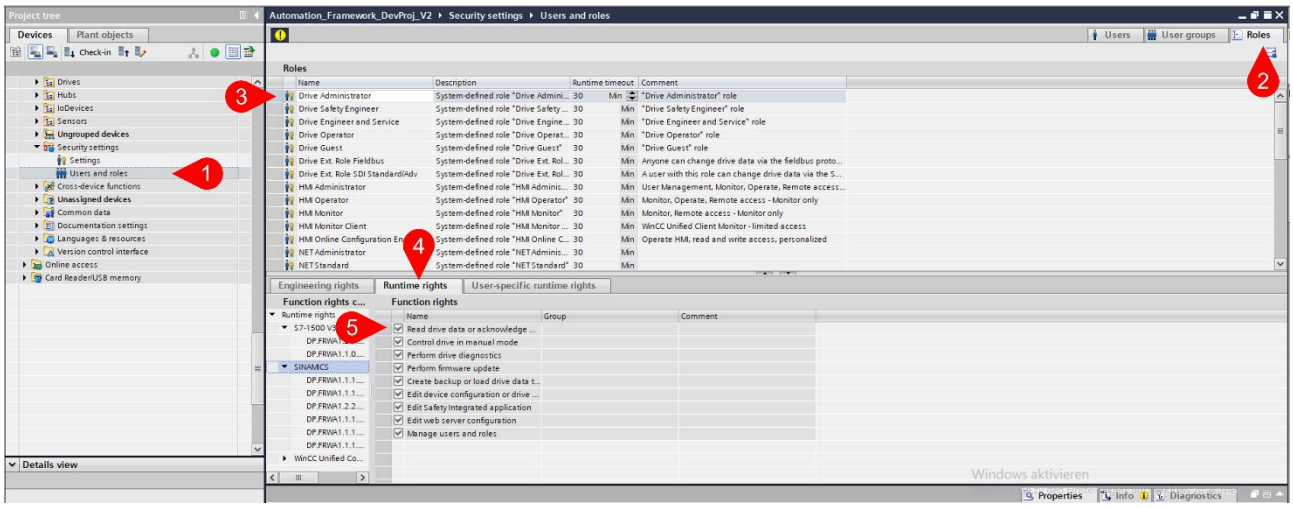


Figure 17-1: Configurate roles and rights.

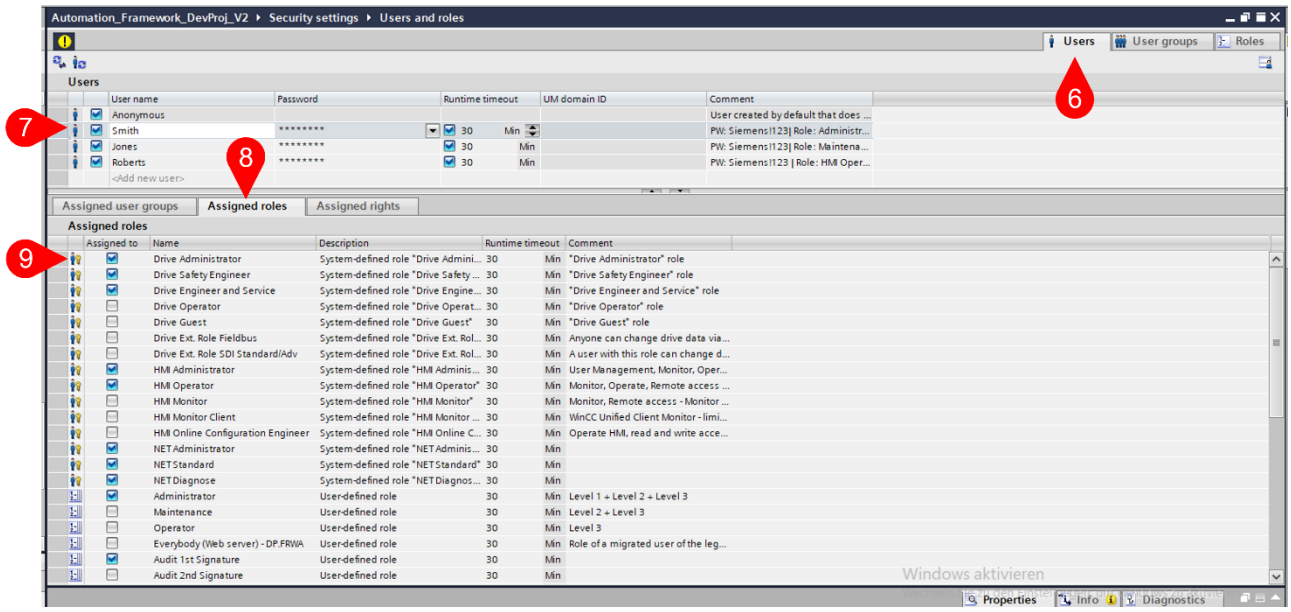


Figure 17-2: Configurate users and assigned roles.

Central User Management with "User Management Component (UMC)" for WinCC Unified runtime

All users can login with their personal credentials and can operate the HMI along their dedicated role.

User Management and Access control (UMAC) to protect the Project, PLC, Drives and Scalance

TIA Portal offers comprehensive UMAC for the TIA projects and devices. You can administrate the access to the engineering. Users can run only functions along their roles.

To active UMAC it is necessary to enable project protection.

WARNING

Project protection cannot be revoked.

With loss of login information from protected projects – the project will be lost.

Use of project protection is advised only with proper login and password information safekeeping.

NOTE

It is recommended to activate UMAC or Project protection earliest in commissioning phase if you need to make sure that functions or configurations are not changed from anybody anymore, latest after the handover to the end user.

Please find more information in the TIA Portal online help under "UMAC" or in the manual for SIMATIC Step7 V19 (chapter 9):

https://support.industry.siemens.com/cs/ww/en/view/109826862.

17.2. Demo Implementation

The AF project includes an example of how various user management/access control can be configured based on "Local Users". There exists three different "function rights" in the "User-specific runtime rights" (4), which were created to set up different access levels for HMI users. In the picture below for the Administrator user(3), the "User-specific runtime rights" can be found in the tab "Roles" (2), in the "Users and roles" menu (1).

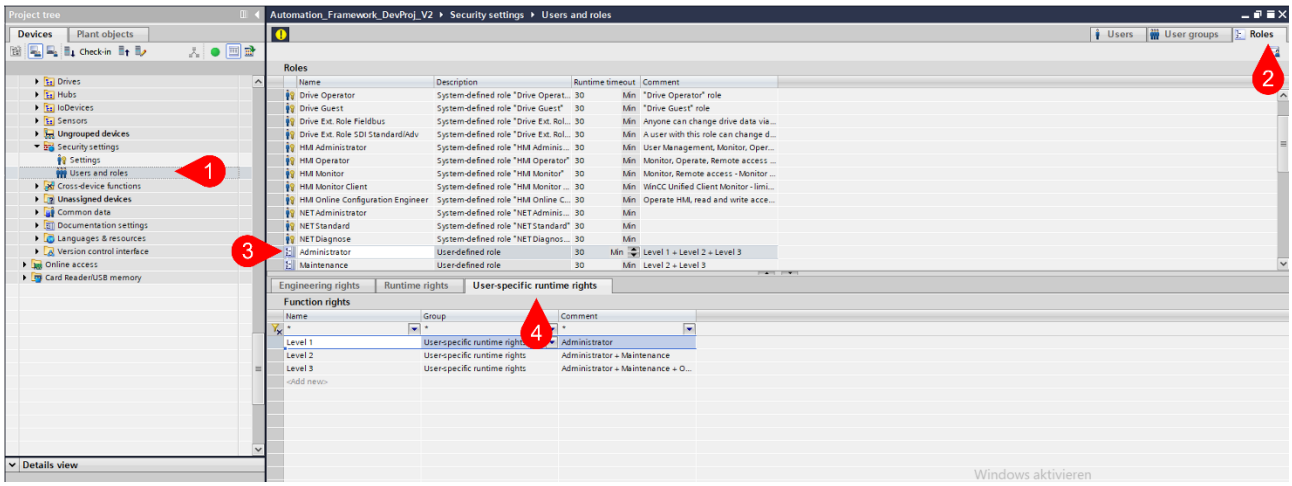


Figure 17-3: User and roles page.

To assign function rights to a role or remove assigned function rights, follow these steps:

- Open the "Roles" tab.
- Select a role (1). Please note that you cannot use multi-selection for assignment.
- In the lower area, open the tab with the category from which you want to assign or delete function rights.
- Activate the function rights that you wish to assign to the role (2).
- Deactivate the function rights that are no longer assigned to the role (2).

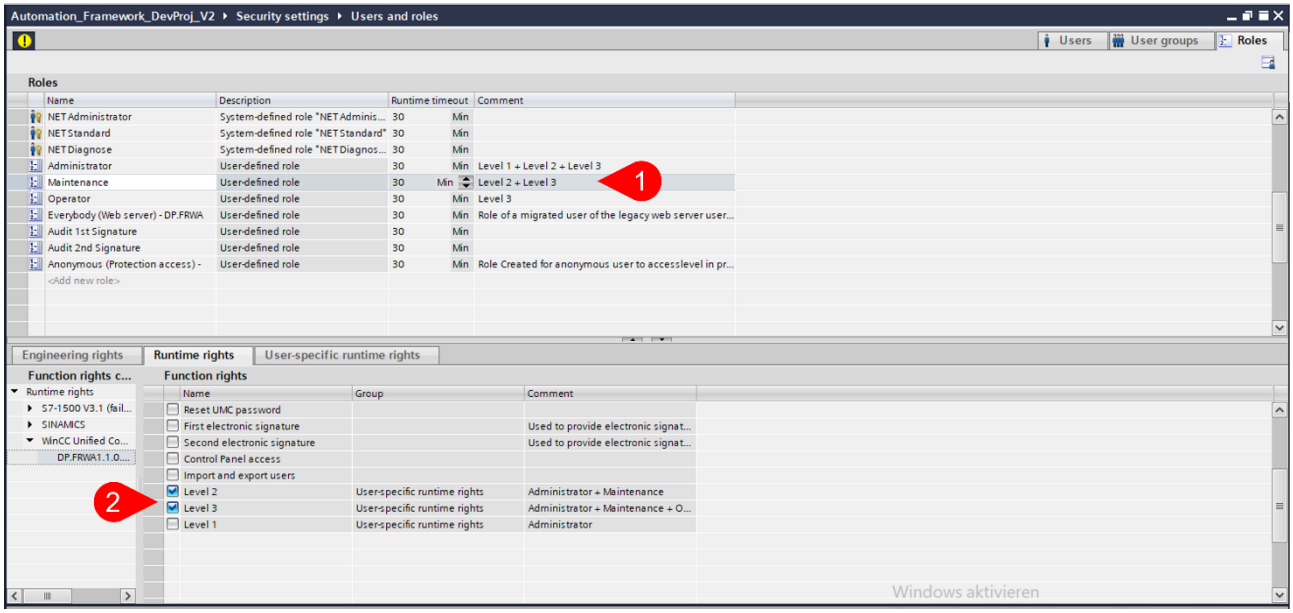


Figure 17-4: Runtime rights.

These roles can be renamed, or the list can be extended by another role. To assign roles to a user, follow these steps:

- Open the "Users" tab (1).
- Select the user to whom you want to assign roles (2). Note that you cannot use multiple selection.
- Enable the desired roles in the "Assigned roles" section (3).

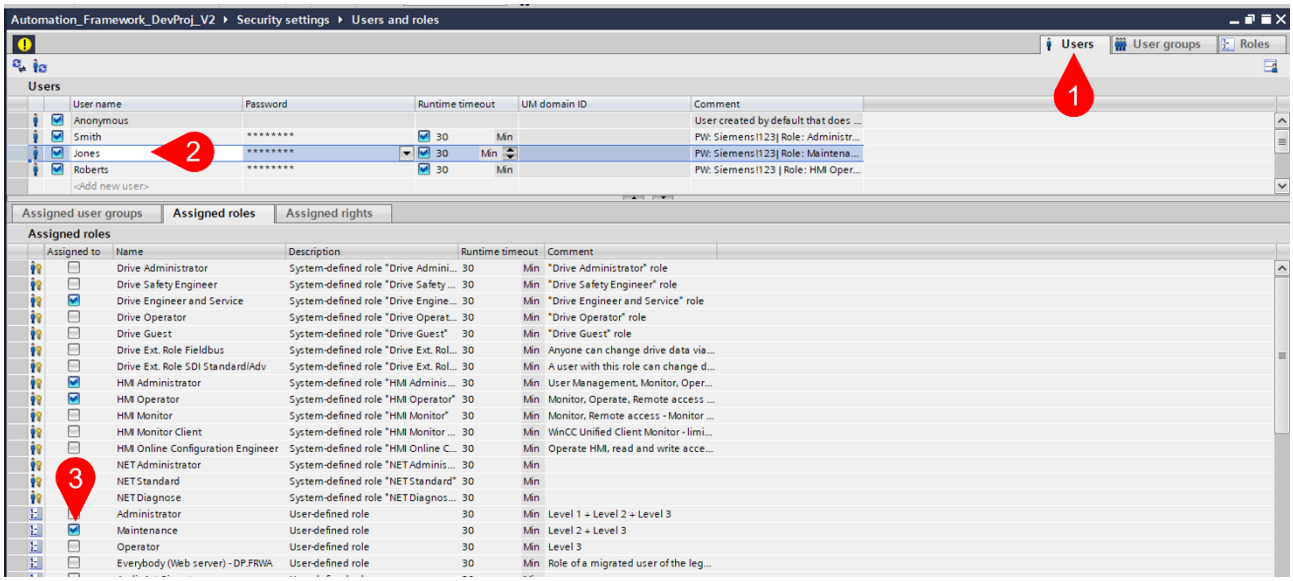


Figure 17-5: Assign roles.

After the user and the roles are configured, it is necessary to set up the project and screens with the related access levels. In the AF project, an example has been implemented. The "Setting" screen (2) should be only accessible for the users with the "Role" "Level1". Therefore, the "Security" settings (3) need to be configured and "Authorization" must be set to "Level1". This configuration led to the behavior, that only users with role "Level1" can access the settings page of the AF HMI.

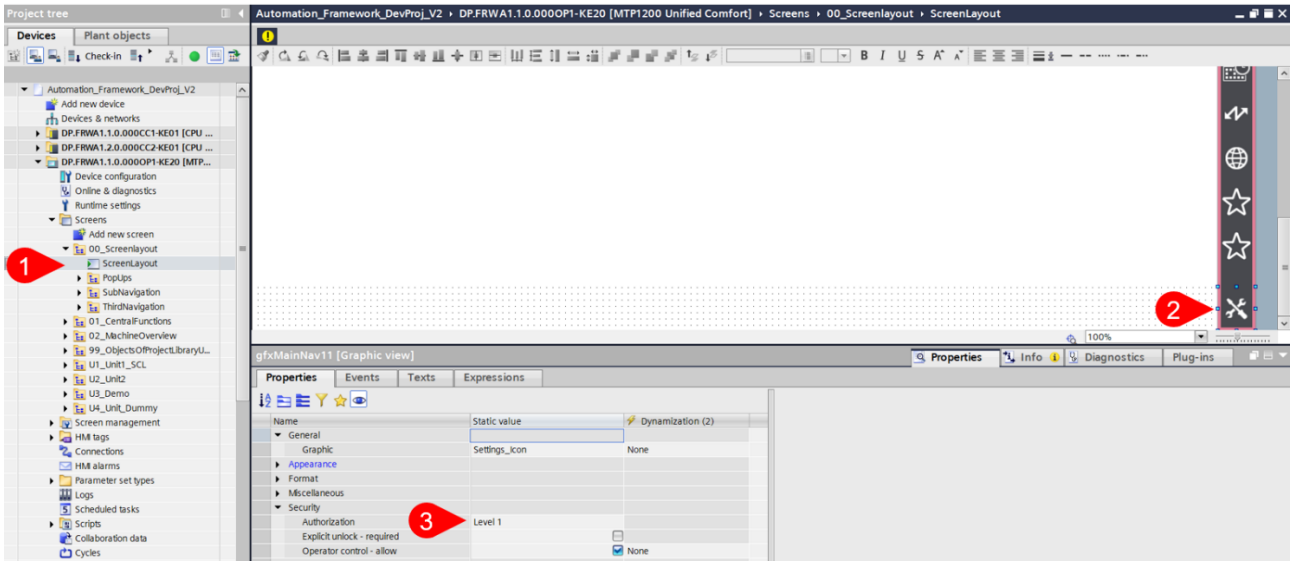


Figure 17-6: Assign an Authorization level to a screen object.

System defined roles:

- Drive Administrator
- Drive Safety Engineer
- Drive Engineer and Service
- Drive Operator
- Drive Guest
- Drive Ext. Role Fieldbus – Anyone can change drive data via the fieldbus protocol if the "Anonymous" user has this role (with the exception of the settings for Safety Integrated and user and role management). This also applies to the "Drive Engineer and Service" role.
- Drive Ext. Role SDI Standard/Adv - A user with this role can change drive data via the SDI Standard Panel (with the exception of the settings for Safety Integrated and user and role management). This also applies to the "Drive Engineer and Service" role.
- HMI Administrator - User Management, Monitor, Operate, Remote access, Remote access - Monitor only, Openness Runtime - read and write access, OPC UA - read and write access, Import & export users, Reset UMC Password, GraphQL access.
- HMI Operator - Monitor, Operate, Remote access - Monitor only.
- HMI Monitor - Monitor, Remote access - Monitor only.
- HMI Monitor Client - WinCC Unified Client Monitor - limited access.
- NET Administrator
- NET Standard
- NET Diagnose

User defined roles:

- Administrator – Level 1 + Level 2 + Level 3
- Maintenance – Level 2 + Level 3
- Operator – Level 3

User group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Administrator	X	X	X					X	X			X	X	X	X		
Maintenance			X					X	X							X	
Operator									X								X
Anonymous																	

Table 17-1: Runtime and user specific rights

18. Run the demo

The AF is tested using real hardware and is a runnable program. However, some use cases involve a higher number of devices than are available in the test environment. As a result, some devices and signals are simulated or simply represented as tags in a data block. The AF project can be executed either with real hardware or through simulation. For running in simulation mode, PLCSIM and PC Runtime are required.

This approach allows for the evaluation and verification of the program's functionality in a controlled environment before final deployment with all physical devices. Simulation ensures that multiple scenarios and configurations can be tested without needing all the necessary hardware from the beginning. In summary, the AF provides flexibility and robustness in the testing process, ensuring that both real hardware execution and simulation deliver consistent and reliable results.

When the user tries to download the compiled program in the PLC (1), a window will pop up asking for login with the credentials of the user that is authorized to write in the PLC.

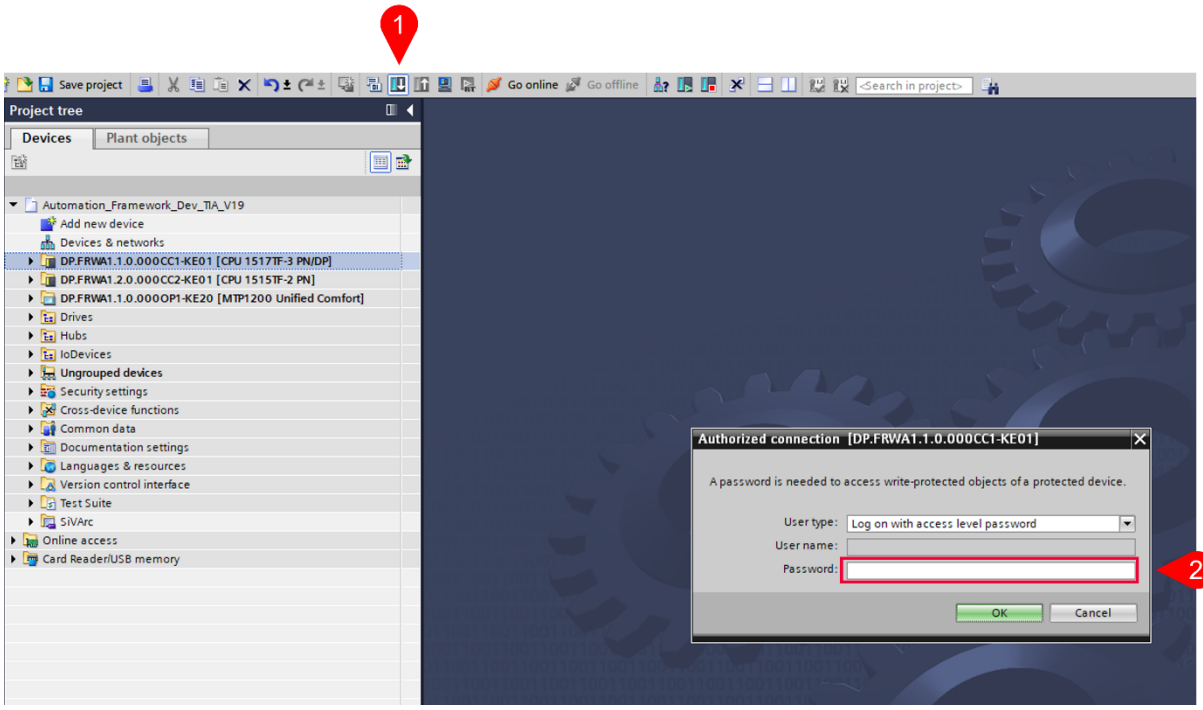


Figure 18-1: Download of the compiled program in the PLC.

The user type to choose is "Project user" (2). Then the "User name" and "Password" field must be fulfilled. The information of this credentials can be found within the Security settings in the "User and roles" menu (See Chapter 17. User Management). For demo purposes, the user must have an administrator role. In this case the login data will be the following:

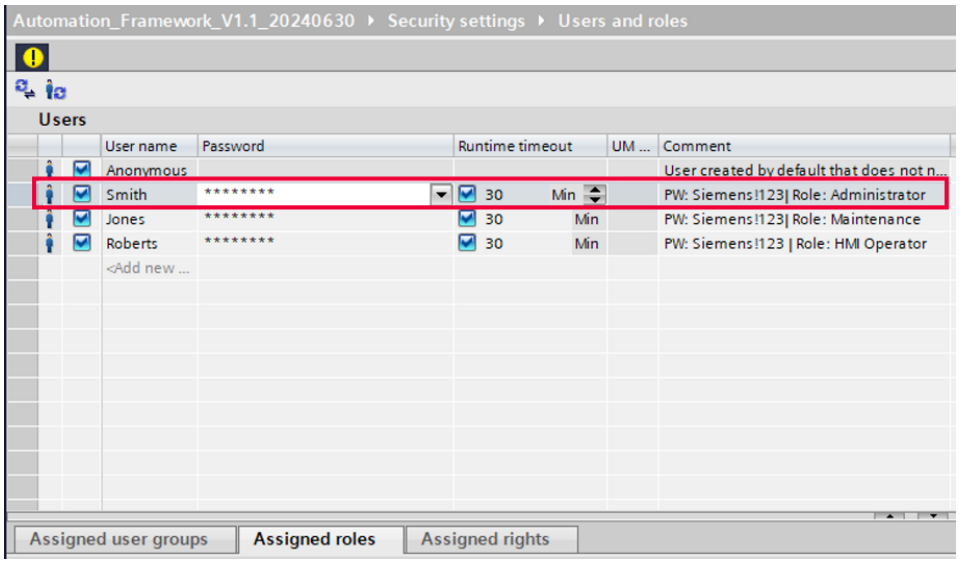


Figure 18-2: Login data of user with administrator role.

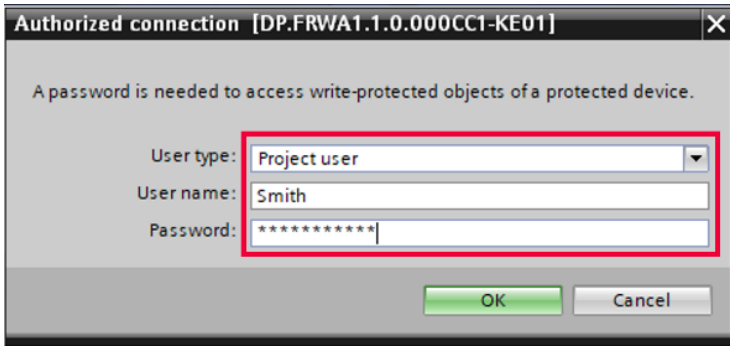


Figure 18-3: The user must fulfill the login data to authorize the connection with the PLC.

From the HMI side, to run the demo, the user must simulate the panel. Once the simulation is running, the HMI is not operable. The user has to login with the same login data as in the PLC with administrator credentials. For that, by pressing/clicking in the "User" field, a pop up for sign in will appear (1). Now the user can fulfill the login data (2), then the HMI will be operable.

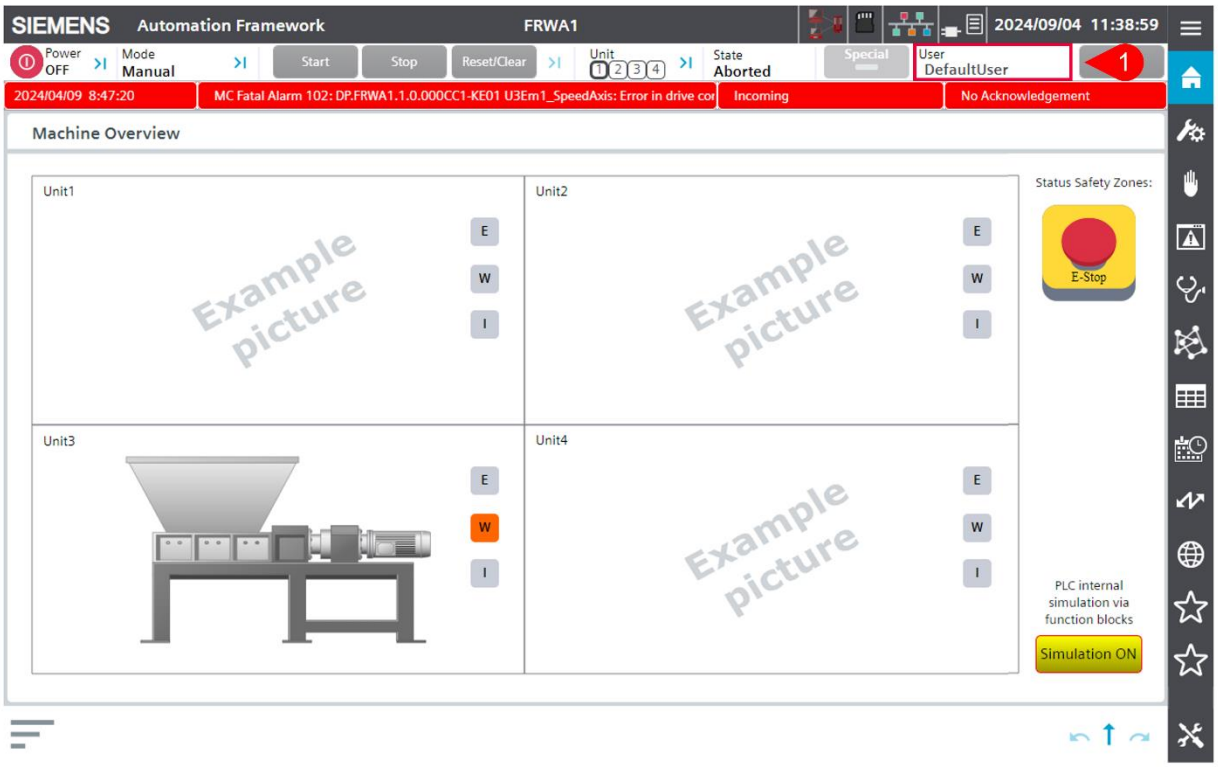


Figure 18-4: Non operable HMI. The user has to login with administrator credentials.

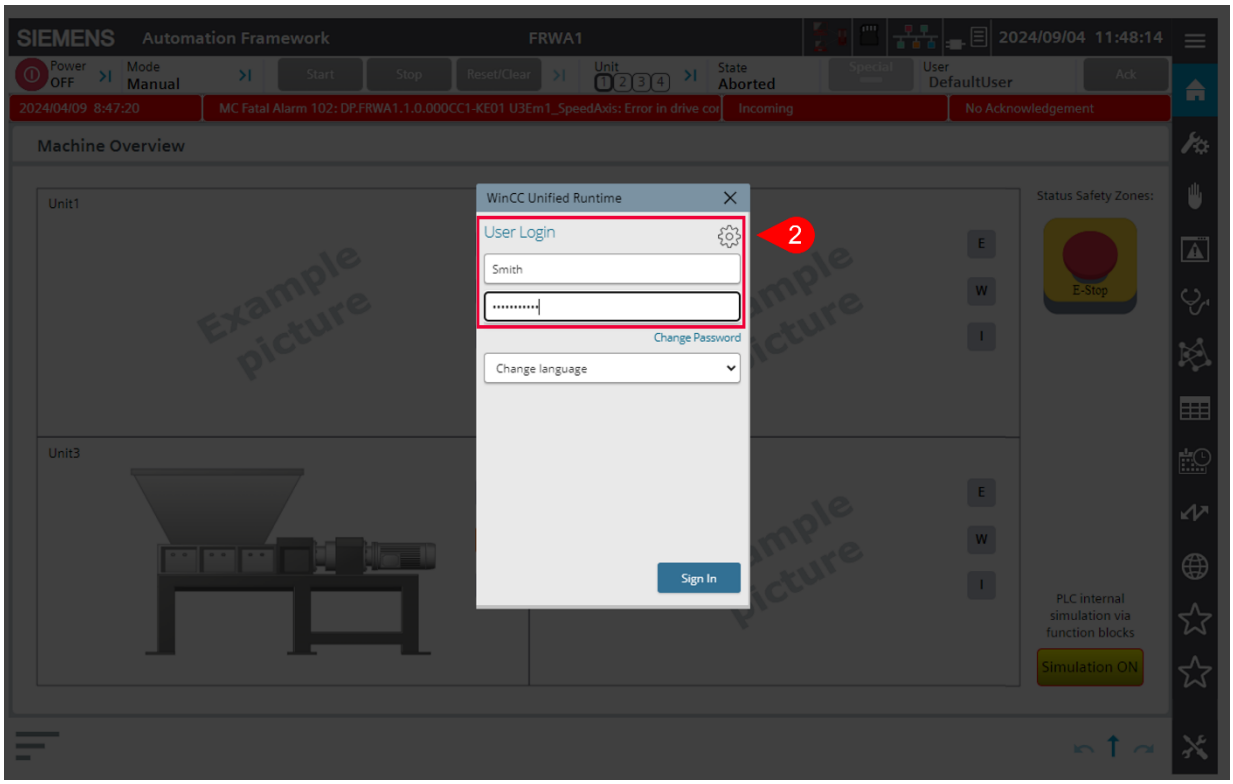


Figure 18-5: Login in the HMI with administrator credentials.

19. Additional information

19.1. Siemens Guidelines

Standardization

The standardization guide shows you how you can modularize your machines and systems. It gives you recommendations and hints for structured and standardized programming of your automation solutions.

https://support.industry.siemens.com/cs/ww/en/view/109756737

Programming Guideline

The controller generation SIMATIC S7-1500 with its up-to-date system architecture should always be used together with the TIA Portal. Both are perfectly coordinated and offer efficient options of programming and configuration. The Programming Guideline for SIMATIC S7 1200 and S7-1500 provides information about innovations, recommendations, and tips to create a powerful user program with TIA Portal in combination with SIMATIC S7-1200 and S7 1500 controller.

https://support.industry.siemens.com/cs/ww/en/view/81318674

Programming Guideline Safety

The Safety Programming Guideline for SIMATIC S7-1200 and S7 1500 describes how a Fail-Safe Controller should be configured, demonstrates different methods for safety programming and tips on how to optimize it.

https://support.industry.siemens.com/cs/ww/en/view/109750255

Programming Styleguide

A PLC program should be as readable and structured as possible so that it is feasible for each software developer, commissioner and maintenance engineer to read and understand the code.

If each developer realized his tasks in its own philosophy, it would be unavoidable that the resulting code is only interpretable by the respective creator.

With the Programming Styleguide for SIMATIC S7-1200 and S7-1500 Siemens offers some help to create a uniform program code which is maintainable and reusable even in case of multiple developers working on the same application program.

https://support.industry.siemens.com/cs/ww/en/view/81318674

Engineering guideline for WinCC Unified

Unified Panel and PC-RT devices offer the latest technologies such as HTML5, JavaScript and scalable vector graphics (SVGs). In order to use them efficiently, you should follow some engineering recommendations.

The presented implementations can be helpful if you start a new project but also if you have an existing project already there.

https://support.industry.siemens.com/cs/ww/en/view/109827603

Test Suite Advanced: Checking TIA Portal projects for compliance with a programming style guide

The application example provides a rule set for checking the Styleguide with TIA Portal Test Suite Advanced. A demo project shows you how the Styleguide check works.

https://support.industry.siemens.com/cs/ww/en/view/109779806

Guideline on Library Handling in TIA Portal

The Guideline on Library Handling in TIA Portal describes the creation of a corporate library using typified library elements in TIA Portal. Using such types offers some advantages, for example a central update function of all instances, system-supported version management and change tracking of library elements.

https://support.industry.siemens.com/cs/ww/en/view/109747503

Multiuser Engineering in TIA Portal

With Multiuser Engineering in TIA Portal, it is possible to work with multiple users together and simultaneously on a project. By processing different objects in parallel within a multiuser project, you can significantly shorten the project planning and commissioning times.

https://support.industry.siemens.com/cs/ww/en/view/109740141

19.2. TIA Portal Options

TIA Portal Test Suite

The TIA Portal Test Suite enables you to define a rule set which contains several rules against which Function Blocks, Functions, Organization Blocks, varieties of global DBs/Instance DBs, PLC tags and User Defined Datatypes are validated.

https://support.industry.siemens.com/cs/ww/en/view/109825229

SIMATIC Visualization Architect (SiVArc)

With the SIMATIC Visualization Architect (SiVArc), TIA Portal provides an option to generate the visualization with the aid of rules which define the assignment between the control program and the visualization. Using this option saves configuration time and prevents errors, particularly in recurring tasks.

https://support.industry.siemens.com/cs/ww/en/view/109826234

19.3. Software for shared tasks at a glance

The powerful tools offered by SIMATIC software will save you valuable time, allowing you to focus on your core automation tasks. An overview of the software is shown in the following link:

https://new.siemens.com/global/en/products/automation/industry-software/automation-software/software-for-shared-tasks.html

TIA Selection Tool

For your application we offer the TIA Selection Tool to support all project planners, beginners and experts alike. No detailed portfolio knowledge is necessary. TIA Selection Tool is available for download as a free desktop version or a cloud variant.

http://www.siemens.com/TIA-Selection-Tool

Start directly https://mall.industry.siemens.com/tstcloud/#/Start

Sinetplan

The Siemens Network Planner supports planners of automation systems based on PROFINET and supports the professional and proactive simulation of the network of a system.

https://support.industry.siemens.com/cs/ww/en/view/109763136

PRONETA

PRONETA Professional supports the operator of automation systems in the automated data acquisition of the components used as well as the comprehensive diagnostics functions used in the PROFINET network.

Professional https://support.industry.siemens.com/cs/ww/en/view/109781283

Basic https://support.industry.siemens.com/cs/ww/en/view/67460624

SIMATIC Automation Tool

SIMATIC Automation Tool V5.0 supports the latest firmware versions related to the TIA Portal V18 and provides additional features as well an extra SDK variant.

The following features are new in this release:

- Support for additional new devices and latest firmware versions related to TIA Portal V18 release as indicated in the device catalog
- Comparison of hardware configuration to actual hardware
- Device table display of the F-Signature for F-CPUs
- Improvements to the Insert Device dialog
- Status bar with device and operation information
- A quick ping operation to determine whether devices are connected.

https://support.industry.siemens.com/cs/ww/en/view/109816217

Siemens OPC UA Modeling Editor (SiOME)

With the free "Siemens OPC UA Modelling Editor" (SiOME) tool, we have created an editor for defining your own OPC UA Information Models or mapping existing companion specifications on your SIMATIC PLC/SINUMERIK. Using this tool, you can import and edit Information Models as XML files or generate and export individualized models.

https://support.industry.siemens.com/cs/ww/en/view/109755133

19.4. Applications for Drives

SINAMICS SDC: Serial Drive Commissioner

The openness application SINAMICS SDC (Serial Drive Commissioner) provides functionality to copy, update and download multiple drives. The devices of the SINAMICS G product line and SINAMICS S210 drives are supported.

https://support.industry.siemens.com/cs/ww/en/view/109774753

Edit parameters in several drives Add-In

TIA Portal V16 facilitates the commissioning of similar drives by copying an existing configuration from one drive object to other drive objects. Parameter values can be written offline or online.

https://support.industry.siemens.com/cs/ww/en/view/109777633

SIMATIC S7-1500 / S7-1500T: Standard application axis control

Function blocks for simple and central control of the basic motion control functions of axes (technology objects) of a SIMATIC S7-1500 or S7-1500T. The central view of each axis via this standard application enables easy programming, fast commissioning and direct testing of your application.

Recommendation is to use the "LAxisCtrl" library.

https://support.industry.siemens.com/cs/ww/en/view/109749348

SIMATIC - Failsafe library LDrvSafe to control the Safety Integrated functions of the SINAMICS drive family

The library includes fail-safe SIMATIC S7 blocks to implement various Safety applications in conjunction with a S7-1200F, S7-1500F, failsafe Open/Software Controller and SINAMICS drives coupled through PROFIsafe.

https://support.industry.siemens.com/cs/ww/en/view/109485794

Energy efficiency evaluation and media analysis for machines with SIMATIC Energy Management (S7-EE Monitor)

The concept and the application examples for the standardized efficiency evaluation of machines according to VDMA34179 or ISO14955 allow an evaluation of the energy efficiency of machines (all media).

https://support.industry.siemens.com/cs/ww/en/view/109753230

20. Appendix

20.1. Service and support

SiePortal

The integrated platform for product selection, purchasing and support - and connection of Industry Mall and Online support. The SiePortal home page replaces the previous home pages of the Industry Mall and the Online Support Portal (SIOS) and combines them.

- Products & Services
  In Products & Services, you can find all our offerings as previously available in Mall Catalog.
- Support
  In Support, you can find all information helpful for resolving technical issues with our products.
- mySieportal
  mySiePortal collects all your personal data and processes, from your account to current orders, service requests and more. You can only see the full range of functions here after you have logged in.

You can access SiePortal via this address: sieportal.siemens.com

Technical Support

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

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.
For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page: siemens.com/sitrain

Industry Online Support app

You will receive optimum support wherever you are with the "Industry Online Support" app. The app is available for iOS and Android:

20.2. Links and literature

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109817223
\3\	Siemens OPC UA Modeling Editor (SiOME) https://support.industry.siemens.com/cs/ww/en/view/109755133
\4\	TIA Portal Add-In Code2Docu for generating documentation https://support.industry.siemens.com/cs/ww/en/view/109809007
\5\	SIMATIC OMAC PackML V2022 Mode&State Management and Machine Data Interface https://support.industry.siemens.com/cs/ww/en/view/109821198
\6\	Siemens Information Model Kit for manufacturing use cases https://support.industry.siemens.com/cs/ww/en/view/109814835

Table 20-1: Links and literature

20.3. Change documentation

Version	Date	Modification
V1.0	2023/12/22	First version
V1.1	2024/06/28	N.a., as version V1.0 was not used.
V1.2	2024/10/15	• Hardware configuration: - ET200SP (-KE60) replaced by ET200SP with fail-safe I/O modules - ET200SP (-KE50) integrated into PROFINET IRT line topology - Additional G120, S120 and S200 drives added for tests and simulation - (010CV1-DU3, 020CV1-DU3, 020CV1-DU4, 050CV1-DU1, 060CV1-DU1, 060CV1-DU2) - Gaps between I/O addresses reduced to improve simulation performance - Standardized I/O tag names according to module designation - I/O tags organized in the tag table of the corresponding equipment module • LPD library for PROFIdrive telegrams added • PLC data types from LGF library for PLCopen diagnostics moved to LAF library • Safety Unit: - Data exchange between standard program and safety program: Update to the use of the Safety Unit, and new safety programming guidelines. - Updated structure of the safety program - New fail-safe I/O modules included into the safety program - Updated workflow for simulation of fail-safe I/Os • LBC 3.0 update • TmAspiration: - Changed reference designator from string to WString for FB and UDT - Added missing comments to interface - Removed unused "SinaSpeedScaleOut" - Removed not needed HMI/OPC UA accessibility flag • CentralFunctions: - Simulation functionality added. Central management of simulation tags. - Manual Operation Line faceplate fixed triggerDataSet call because of Tia V19 Upd3. Also fixed "unit" property not being shown. • Manual Operation Lines: - Block documentation correction. - Default tag connected for input "screenData" removed • LAF SelectJob: - Rework of the Code - Reset behaviour of the "jobCompleted" Bit adapted - Input, Output and InOut names changed • LAF SCLControl: Initialization behaviour adjusted • Call Environments: - "statReferenceDesignator" deleted and connections of the variable changed to the combined Reference Designator within the DB "Config" - Control Module calls seperated. Sensors at the beginning of the EM logic, Actuators are called after processing the sequence logic • Sequence blocks: - Stopped step has been deleted and the sequences only have an Immediate Stopped step that is activated when the command "immediateStop" of the sequence interface is triggered. - "LAF_UpdateJobHistory" call added to the Status update section

Table 20-2: Changelog

