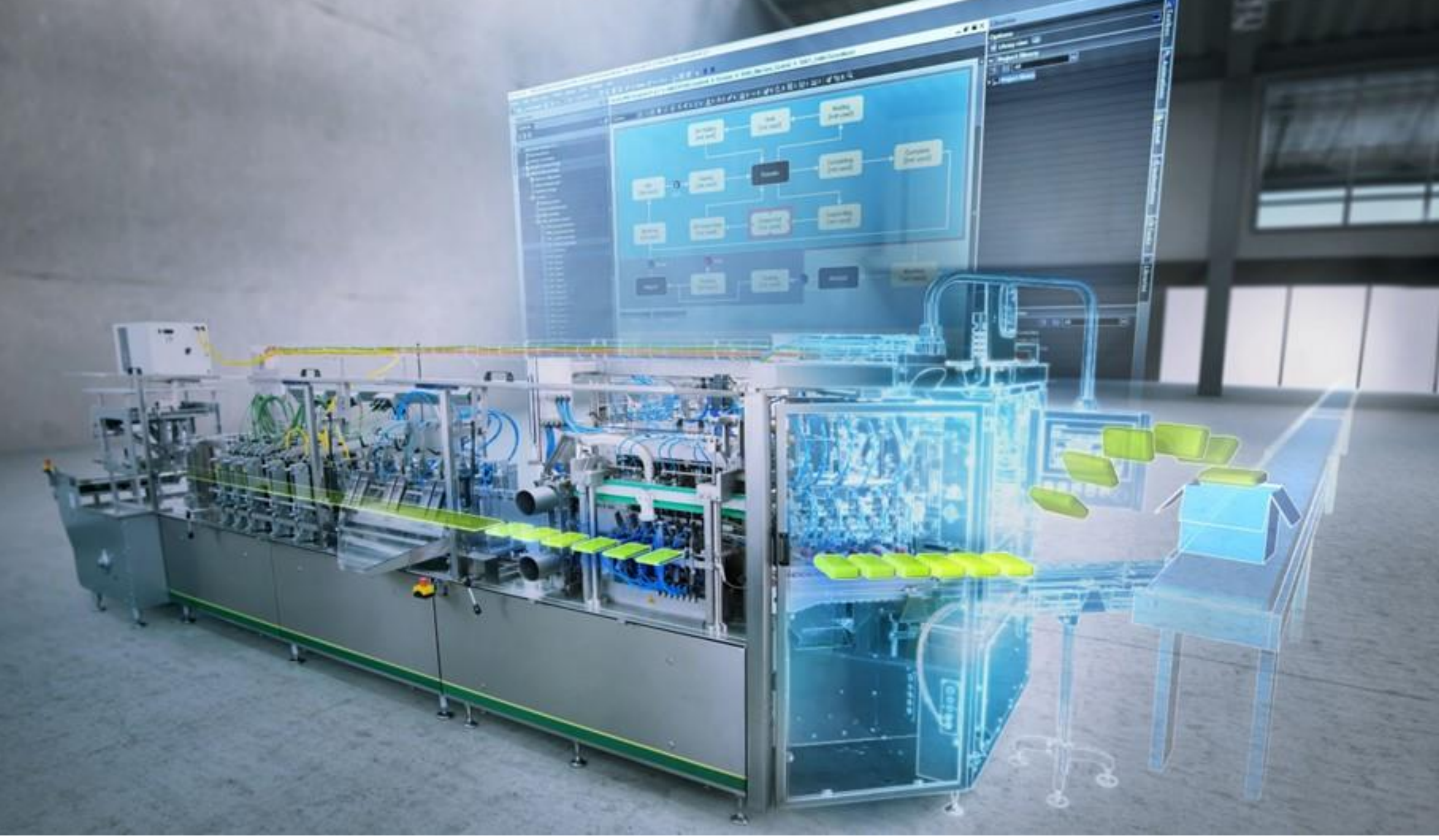


Siemens
Digital
Industries

Automation FrameworkBasic V2.2

SIEMENS

Table of Contents

1. What's new

1.1. New features

Issue	Date	Description
2.0	2025/04/04	First version
2.1	2025/05/14	Improvements, see change log
2.2	2025/06/30	• Central Functions: - Json-Files for manual operation lines will be downloaded to the Unified Panel via system functions (without an USB-Stick). - Multilingual support for "interface boolean" screen. - New "Enhanced interface" allows to show also non-Boolean signals like complex UDT's. • Unit - Unit detects automatically if SIMIT simulation is connected and running - Command interfaces extended - Commands can now be enables/disabled during operation - Mode and state history extended • Equipment modules - User can now configure in which modes and states the EM can be linked or unlinked from the unit - The user can now select which command interface is active when an EM is unlinked from unit. • HMI - New navigation concept : It is now much easier to add new units and equipment modules. New popup for special functions: It's now possible to clearly define what the special function should do for the selected units.

Table 1-1: New features

1.2. Change log

See Chapter Change Documentation

2. Introduction

2.1. Overview

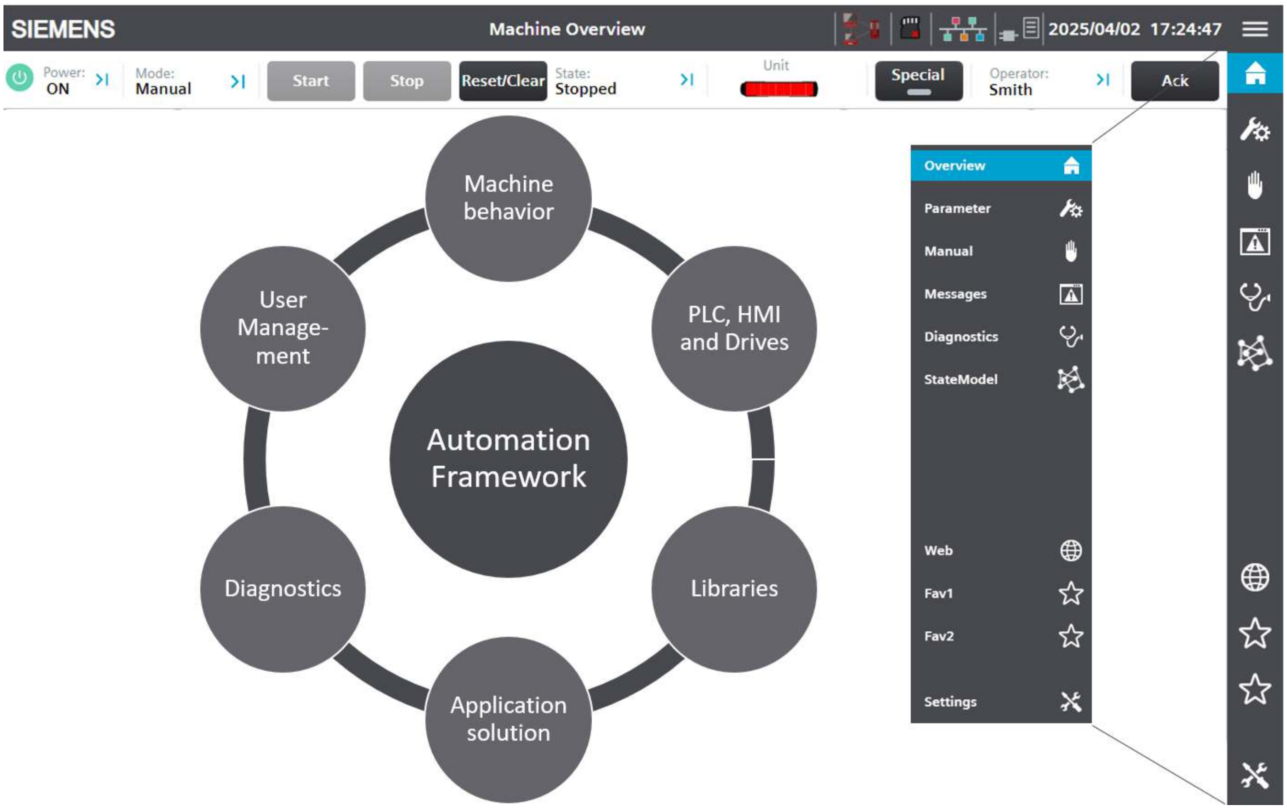


Figure 2-1: Siemens Automation Framework

A core offering of the Siemens Automation Framework (in following named as AF$_{Basic}$) 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 Automation Framework$_{Basic}$ 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)
- 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$_{\text{Basic}}$ 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)
  - SINAMICS Startdrive (TIA Portal)
- Machine and plant diagnostics
- Industrial Communication and Networks

NOTE

Please be aware that the usage of the TIA Portal project requires the latest Unified Panel image version (from V20.0.0.0).

2.2. Scope of Delivery

The Version 2.2.2 of Automation Framework$_{\text{Basic}}$ /AF$_{\text{Basic}}$ Library includes:

- This user documentation "109987391_AutomationFramework_Basic_DOC_V2_2_2_en.pdf"
- TIA Portal Project "109987391_AutomationFramework_Basic_PROJ_V2_2_2_V20.zip"

2.3. Used Libraries

The following TIA Portal libraries are used in the AF_{Basic} project:

AFBasic Global Library

The AFBasic global library contains all sub-libraries, master copies and user documentation recommended for the AFBasic. 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.	As part of Automation Framework
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
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 AFBasic project

2.4. Hardware and Software Requirements

The AFBasic project was developed based on the following components:

Hardware

- SIMATIC S7-1200 G2 with Firmware Version V1.0.0 or higher (SIOS-ID: 109972011)
- WinCC Unified Panel with display size 12" (from Panel Image V19.0.0.2, SIOS-ID:109825605)
- SINAMICS S200, S210 and G120
- ET 200SP

Software

- TIA Portal V20 or higher
- SIMATIC STEP 7 Basic V20 or higher for PLC engineering
- SIMATIC WinCC Unified 20 or higher for HMI engineering
- SINAMICS Startdrive Basic V20 or higher for configuration and commissioning of drives, PLC connection, diagnostic, optimization and parametrization
- TIA Portal Add-In Code2Docu \4\ for generating PLC-Block documentation

Optional products to be used

- S7-PLCSIM V20 or higher for PLC logic simulation
- SIMATIC WinCC Unified PC Runtime V20 or higher for simulation of the HMI application
- SIMATIC Visualization Architect (SiVarc) for automatic generation of "ready to use" screens
- SINAMICS Startdrive Advanced for safety acceptance and safety activation tests, measurement functions and long-term trace

Licenses

The following licenses are used in the AFB project:

Mandatory	See current AFB component release list for order and version information	License which must be delivered to the end user
Yes	SIMATIC STEP 7 Basic V20	
No	SIMATIC S7-PLCSIM V20	
Yes	SIMATIC WinCC Unified V20 Comfort Engineering	
No	SIMATIC WinCC Unified PC Runtime V20	1x Runtime License for SIMATIC WinCC Unified PC (10k) RT
Yes	SINAMICS Startdrive Basic V20	
No	SINAMICS Startdrive Advanced V20	1x Engineering-License for SINAMICS Startdrive Advanced

Table 2-2: Licenses for ABF project

NOTE

The HMI power tag license might vary with your specific application. Please ensure corresponding license size if needed.

2.5. User-defined documentation - Online Help

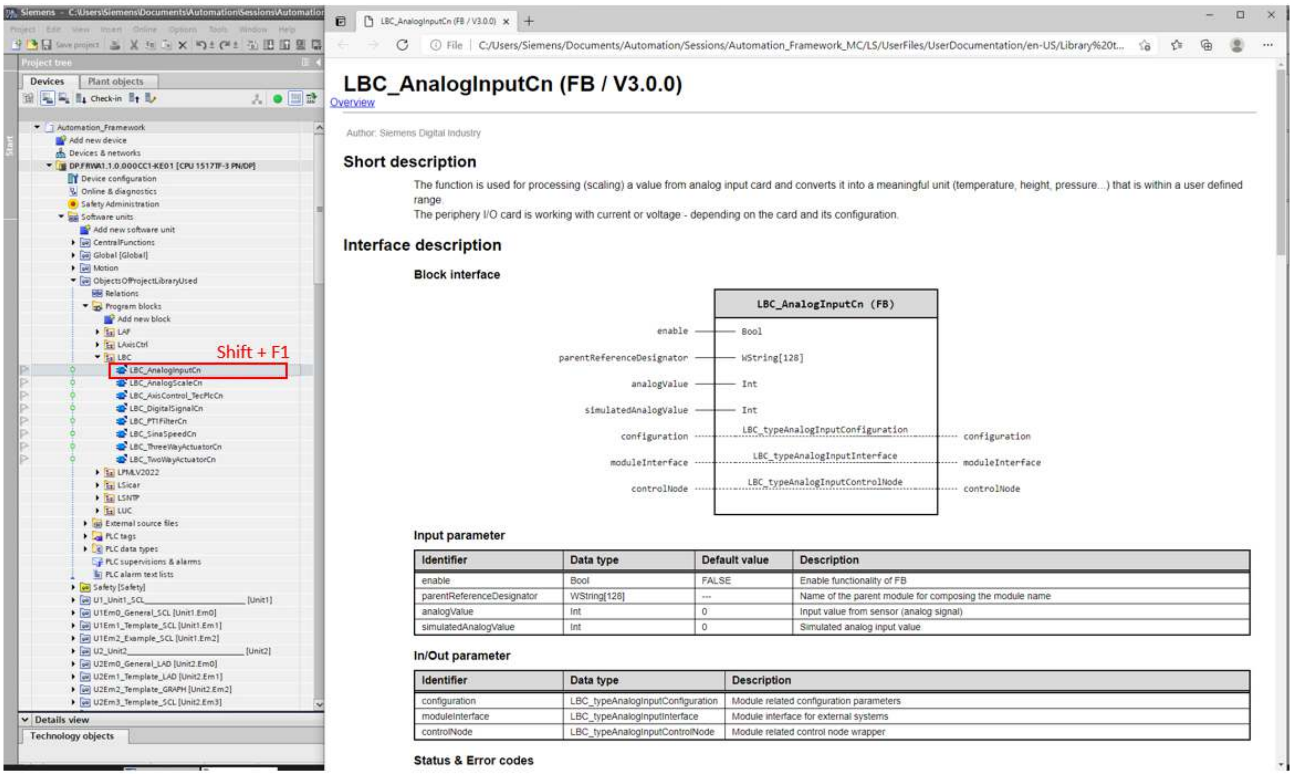
Over time, you create your own contents in your project or library, for example, blocks, tags, or library types. While the operating principle of the TIA Portal is described in the supplied help system (F1), there is no help for the contents you have created yourself.

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.


Shift + F1
LBC_AnalogInputCn
LBC_AnalogInputCn (FB / V3.0.0)
C:/Users/Siemens/Documents/Automation/Sessions/Automation_Framework_MC/LS/UserFiles/UserDocumentation/en-US/Library%20t...
Overview
LBC_AnalogInputCn (FB)
enable
Bool
parentReferenceDesignator
WString[128]
analogValue
Int
simulatedAnalogValue
Int
configuration
LBC_typeAnalogInputConfiguration
moduleInterface
LBC_typeAnalogInputInterface
controlNode
LBC_typeAnalogInputControlNode

LBC_AnalogInputCn (FB / V3.0.0)

Author: Siemens Digital Industry

Short description

The function is used for processing (scaling) a value from analog input card and converts it into a meaningful unit (temperature, height, pressure...) that is within a user defined range.
The periphery I/O card is working with current or voltage - depending on the card and its configuration.

Interface description

Block interface

Input parameter

Identifier	Data type	Default value	Description
enable	Bool	FALSE	Enable functionality of FB
parentReferenceDesignator	WString[128]	---	Name of the parent module for composing the module name
analogValue	Int	0	Input value from sensor (analog signal)
simulatedAnalogValue	Int	0	Simulated analog input value

In/Out parameter

Identifier	Data type	Description
configuration	LBC_typeAnalogInputConfiguration	Module related configuration parameters
moduleInterface	LBC_typeAnalogInputInterface	Module interface for external systems
controlNode	LBC_typeAnalogInputControlNode	Module related control node wrapper

Status & Error codes

Figure 2-2: How to open 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

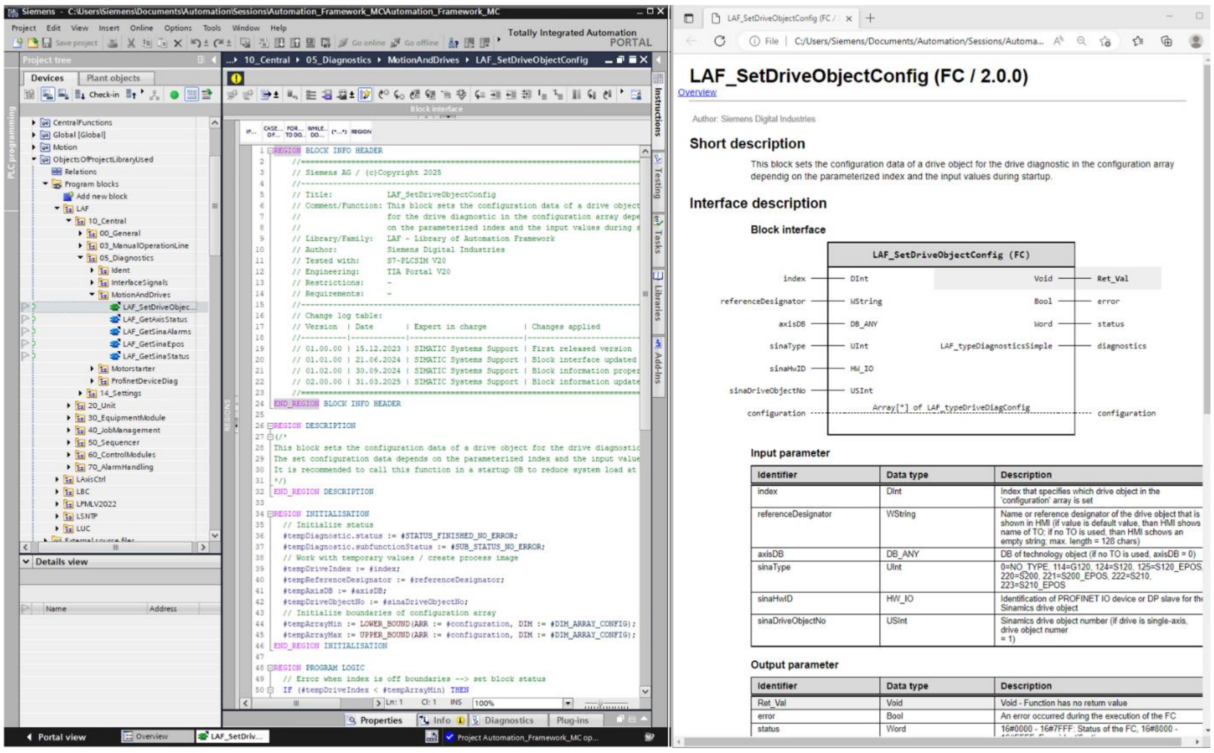
Save the documentation in the directory of the [global library]\UserFiles\UserDocumentation\[language]. This way the user-defined documentation can be forwarded with the global library.

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 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).



```
REGION BLOCK INFO HEADER
    //============================================================
    // Siemens AG / (c)Copyright 2025
    //------------------------------------------------------------
    // Title:            LAF_SetDriveObjectConfig
    // Comment/Function: This block sets the configuration data of a drive object
    //                   for the drive diagnostic in the configuration array depe
    //                   on the parameterized index and the input values during s
    // Library/Family:   LAF - Library of Automation Framework
    // Author:           Siemens Digital Industries
    // Tested with:      S7-PLCSIM V20
    // Engineering:      TIA Portal V20
    // Restrictions:     -
    // Requirements:     -
    //------------------------------------------------------------
    // Change log table:
    // Version  | Date       | Expert in charge        | Changes applied
    //----------|------------|-------------------------|----------------
    // 01.00.00 | 15.12.2023 | SIMATIC Systems Support | First released version
    // 01.01.00 | 21.06.2024 | SIMATIC Systems Support | Block interface updated
    // 01.02.00 | 30.09.2024 | SIMATIC Systems Support | Block information proper
    // 02.00.00 | 31.03.2025 | SIMATIC Systems Support | Block information update
    //============================================================
END_REGION BLOCK INFO HEADER

REGION DESCRIPTION
(/*
This block sets the configuration data of a drive object for the drive diagnostic
The set configuration data depends on the parameterized index and the input value
It is recommended to call this function in a startup OB to reduce system load at
*/)
END_REGION DESCRIPTION

REGION INITIALISATION
  // Initialize status
  #tempDiagnostic.status := #STATUS_FINISHED_NO_ERROR;
  #tempDiagnostic.subfunctionStatus := #SUB_STATUS_NO_ERROR;
  // Work with temporary values / create process image
  #tempDriveIndex := #index;
  #tempReferenceDesignator := #referenceDesignator;
  #tempAxisDB := #axisDB;
  #tempDriveObjectNo := #sinaDriveObjectNo;
  // Initialize boundaries of configuration array
  #tempArrayMin := LOWER_BOUND(ARR := #configuration, DIM := #DIM_ARRAY_CONFIG);
  #tempArrayMax := UPPER_BOUND(ARR := #configuration, DIM := #DIM_ARRAY_CONFIG);
END_REGION INITIALISATION

REGION PROGRAM LOGIC
  // Error when index is off boundaries --> set block status
  IF (#tempDriveIndex < #tempArrayMin) THEN
```

C:/Users/Siemens/Documents/Automation/Sessions/Automa...

LAF_SetDriveObjectConfig (FC / 2.0.0)

Overview

Author: Siemens Digital Industries

Short description

This block sets the configuration data of a drive object for the drive diagnostic in the configuration array dependig on the parameterized index and the input values during startup.

Interface description

Block interface

Input parameter

Identifier	Data type	Description
index	DInt	Index that specifies which drive object in the 'configuration' array is set
referenceDesignator	WString	Name or reference designator of the drive object that is shown in HMI (if value is default value, than HMI shows name of TO; if no TO is used, than HMI schows an empty string; max. length = 128 chars)
axisDB	DB_ANY	DB of technology object (if no TO is used, axisDB = 0)
sinaType	UInt	0=NO_TYPE, 114=G120, 124=S120, 125=S120_EPOS, 220=S200, 221=S200_EPOS, 222=S210, 223=S210_EPOS
sinaHwID	HW_IO	Identification of PROFINET IO device or DP slave for th Sinamics drive object
sinaDriveObjectNo	USInt	Sinamics drive object number (if drive is single-axis, drive object numer = 1)

Output parameter

Identifier	Data type	Description
Ret_Val	Void	Void - Function has no return value
error	Bool	An error occurred during the execution of the FC
status	Word	16#0000 - 16#7FFF: Status of the FC, 16#8000 -

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.

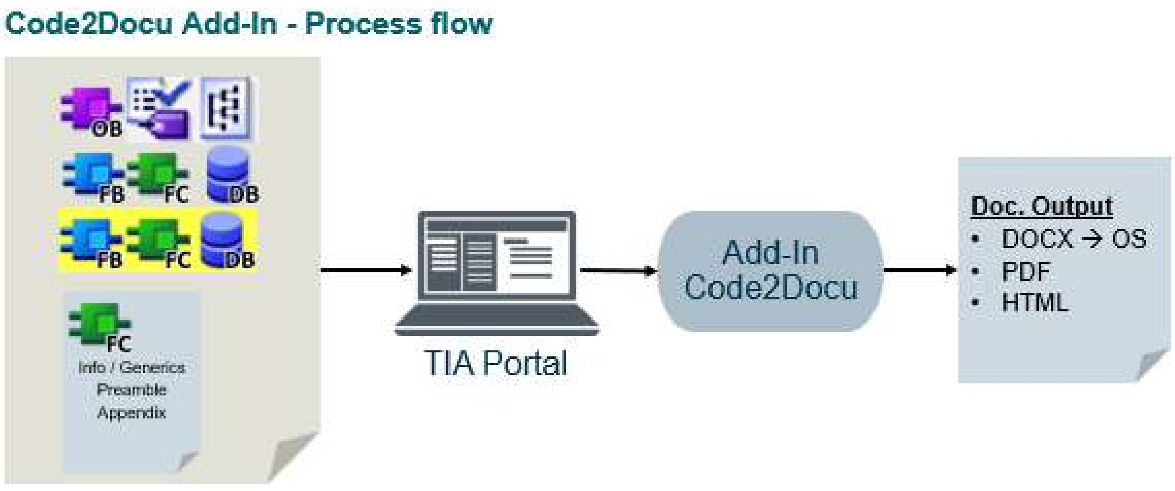


Figure 2-4: Code2Docu – Process flow.

2.6. Programming conventions

AF follows the conventions of the Siemens Programming Style guide 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 Style guide available in the following link:

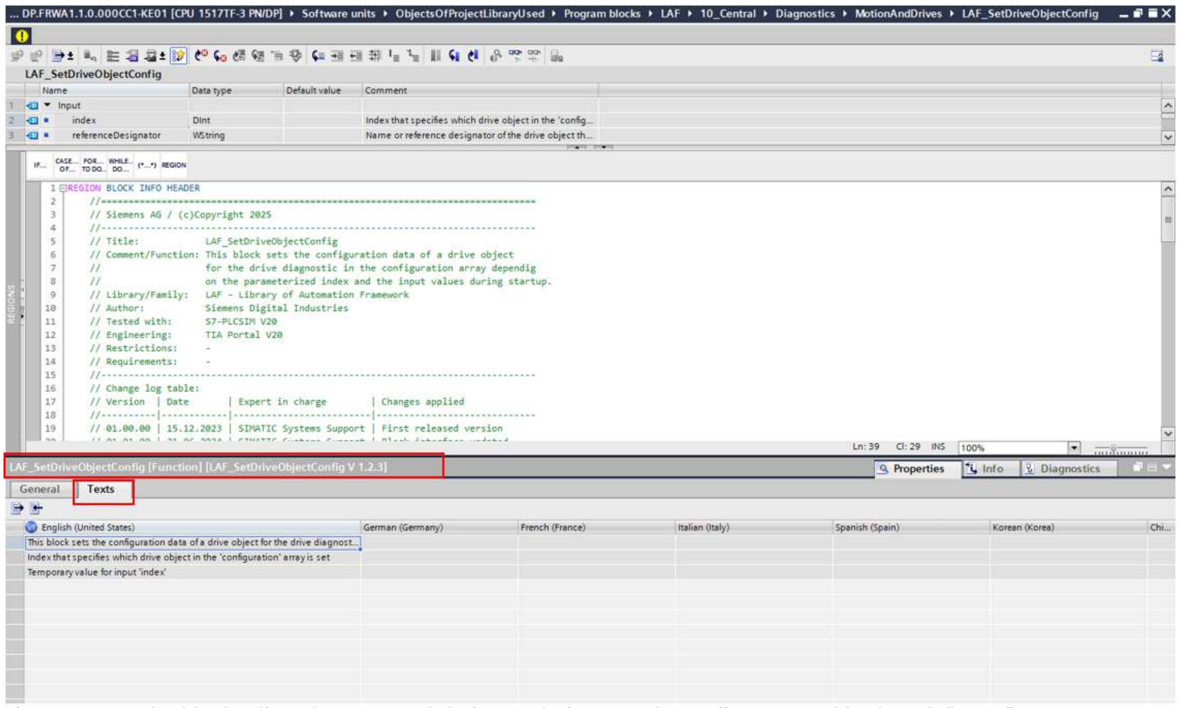
Siemens Guidelines

Programming Styleguide

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

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

  Project texts must always be translated into all active project languages. 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.


... DP.FRWA1.1.0.000CC1-KE01 [CPU 1517TF-3 PN/DP] ▸ Software units ▸ ObjectsOfProjectLibraryUsed ▸ Program blocks ▸ LAF ▸ 10_Central ▸ Diagnostics ▸ MotionAndDrives ▸ LAF_SetDriveObjectConfig
LAF_SetDriveObjectConfig
IF... CASE... OF... FOR... TO DO... WHILE... DO... (*...*) REGION
Ln: 39 Cl: 29 INS 100%
Properties
Info
Diagnostics
General
Texts

	Name	Data type	Default value	Comment
1	Input			
2	index	DInt		Index that specifies which drive object in the 'config...
3	referenceDesignator	WString		Name or reference designator of the drive object th...

```
REGION BLOCK INFO HEADER
    //==============================================================================
    // Siemens AG / (c)Copyright 2025
    //------------------------------------------------------------------------------
    // Title:            LAF_SetDriveObjectConfig
    // Comment/Function: This block sets the configuration data of a drive object
    //                   for the drive diagnostic in the configuration array dependig
    //                   on the parameterized index and the input values during startup.
    // Library/Family:   LAF - Library of Automation Framework
    // Author:           Siemens Digital Industries
    // Tested with:      S7-PLCSIM V20
    // Engineering:      TIA Portal V20
    // Restrictions:     -
    // Requirements:     -
    //------------------------------------------------------------------------------
    // Change log table:
    // Version  | Date       | Expert in charge       | Changes applied
    //----------|------------|------------------------|------------------------------
    // 01.00.00 | 15.12.2023 | SIMATIC Systems Support | First released version
```

LAF_SetDriveObjectConfig [Function] [LAF_SetDriveObjectConfig V 1.2.3]

English (United States)	German (Germany)	French (France)	Italian (Italy)	Spanish (Spain)	Korean (Korea)	Chi...
This block sets the configuration data of a drive object for the drive diagnost...						
Index that specifies which drive object in the 'configuration' array is set						
Temporary value for input 'index'						

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**

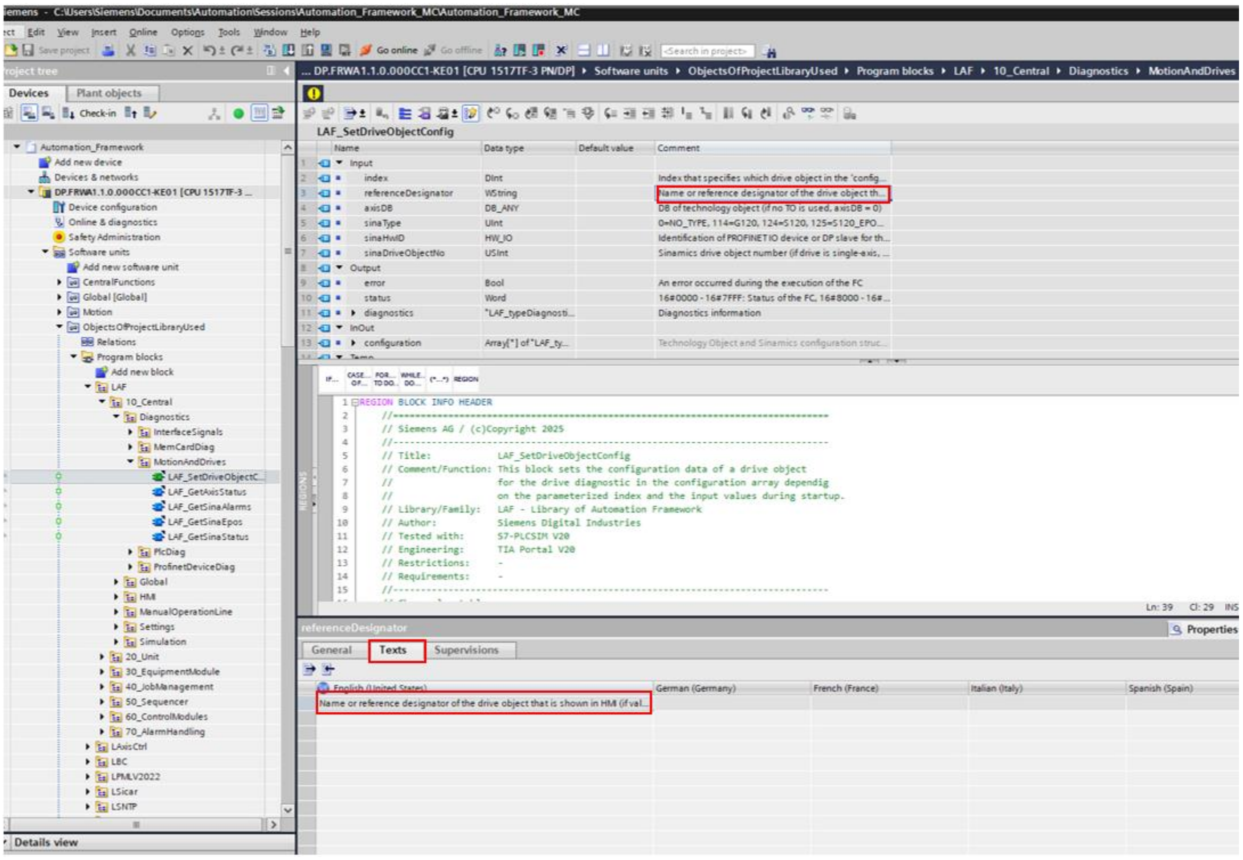
  Comment and property fields shall be used and filled with meaningful comments and information. This includes, for example

  - Block title and block comments (refer also to “NF003 Rule: Document developer information”)
  - Block interfaces (variables and constants)
  - Network title and network comments
  - PLC data types, Named value data types and their variables
  - PLC tag tables, PLC tags and user constants

- PLC alarm text lists, PLC supervision & alarms
- Library properties

Justification: Using this the user gets the most information and guidance in using the components, e.g. through tooltips.

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.


... DP.FRWA1.1.0.000CC1-KE01 [CPU 1517TF-3 PN/DP] ▸ Software units ▸ ObjectsOfProjectLibraryUsed ▸ Program blocks ▸ LAF ▸ 10_Central ▸ Diagnostics ▸ MotionAndDrives
Devices
Plant objects
Automation_Framework
Program blocks
LAF_SetDriveObjectConfig
Properties
Details view

LAF_SetDriveObjectConfig

	Name	Data type	Default value	Comment
1	Input			
2	index	DInt		Index that specifies which drive object in the 'config...
3	referenceDesignator	WString		Name or reference designator of the drive object th...
4	axisDB	DB_ANY		DB of technology object (if no TO is used, axisDB = 0)
5	sinaType	UInt		0=NO_TYPE, 114=G120, 124=S120, 125=S120_EPO...
6	sinaHwID	HW_IO		Identification of PROFINET IO device or DP slave for th...
7	sinaDriveObjectNo	USInt		Sinamics drive object number (if drive is single-axis, ...
8	Output			
9	error	Bool		An error occurred during the execution of the FC
10	status	Word		16#0000 - 16#7FFF: Status of the FC, 16#8000 - 16#...
11	diagnostics	"LAF_typeDiagnosti...		Diagnostics information
12	InOut			
13	configuration	Array[*] of "LAF_ty...		Technology Object and Sinamics configuration struc...

```
REGION BLOCK INFO HEADER
    //===============================================================================
    // Siemens AG / (c)Copyright 2025
    //-------------------------------------------------------------------------------
    // Title:            LAF_SetDriveObjectConfig
    // Comment/Function: This block sets the configuration data of a drive object
    //                   for the drive diagnostic in the configuration array dependig
    //                   on the parameterized index and the input values during startup.
    // Library/Family:   LAF - Library of Automation Framework
    // Author:           Siemens Digital Industries
    // Tested with:      S7-PLCSIM V20
    // Engineering:      TIA Portal V20
    // Restrictions:     -
    // Requirements:     -
    //-------------------------------------------------------------------------------
```

referenceDesignator

General | Texts | Supervisions

English (United States)	German (Germany)	French (France)	Italian (Italy)	Spanish (Spain)
Name or reference designator of the drive object that is shown in HMI (if val...				

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

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

  Identifiers for TIA Portal objects, such as:

  - Blocks, Software Units, Technology Objects, Libraries, Projects, Namespaces
  - PLC tag tables, PLC alarm text lists
  - Watch and force tables
  - Traces and measurements

  are written using UpperCamelCase (PascalCasing).

  The following rules apply for UpperCamelCase:

  - The first character is a capital letter.
  - If an identifier is assembled out of multiple words, then the first character of each word is a capital letter.
  - There are no separators (e.g. hyphen or underscore) used for the optical separation of the identifier. For structuring and specialization purposes the sparingly use of the underscore (not more than three) is permitted.

  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..Else..Elseif 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.

- **DA005 Rule : Exchange data only via formal parameters**

Data exchange within the PLC with FBs or FCs is always carried out via the block interface (input, output or in/out parameters).

If FBs or FCs are used as a call environment or for structuring the software and do not require an interface for reuse, formal parameters can be omitted. In this case, data is exchanged exclusively via direct access to global data. Mixing both types of access in one block is not permitted.

Justification: In terms of encapsulated modules, data is decoupled via the interface and dependencies are resolved. This ensures data consistency in the event of multiple calls (possibly at different program locations). If a call environment/structural element does not require an interface for reuse, the effort can be reduced by not using formal parameters.

The current version of the Automation Framework doesn't fully comply with this rule, as many users handle it differently today for various reasons. The Automation Framework targets a broad market and therefore contains a mix of direct access and access via a block interface in the current version.

Depending on user feedback and further developments of the TIA Portal, we reserve the right to exchange data between objects in future versions of the Automation Framework exclusively via the interface.

NOTE For further details, see also chapter Passing data via block interface vs. direct global access.

3. AF Guidelines

This document describes how to use the Automation Framework$_{Basic}$. Further guidelines used by the AF$_{Basic}$ are listed in the 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 "Versioned" 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.]()

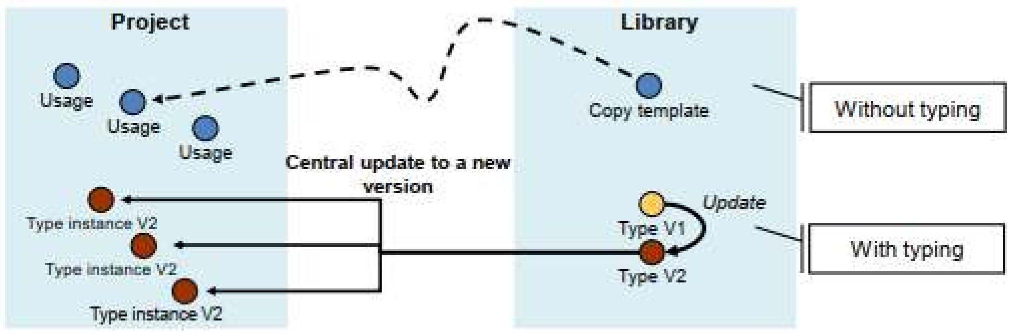


Figure 4-1: Differences using library copies vs library types

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.

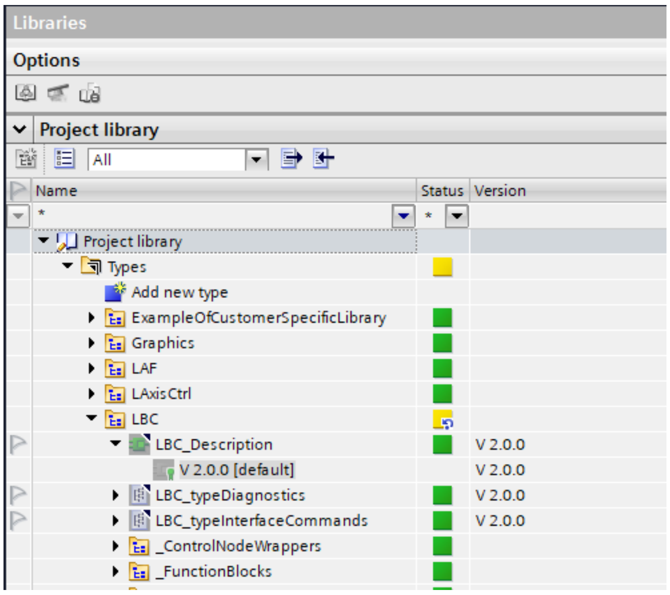
NOTE

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/de/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.



Libraries

Options

Project library

All

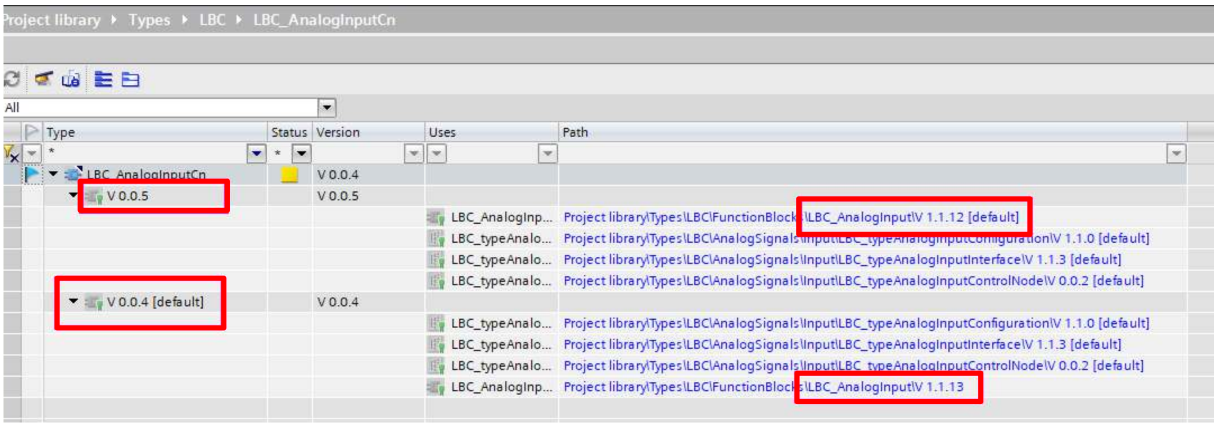
Name	Status	Version
Project library		
Types	🟨	
Add new type		
ExampleOfCustomerSpecificLibrary	🟩	
Graphics	🟩	
LAF	🟩	
LAxisCtrl	🟩	
LBC	🟨	
LBC_Description	🟩	V 2.0.0
V 2.0.0 [default]		V 2.0.0
LBC_typeDiagnostics	🟩	V 2.0.0
LBC_typeInterfaceCommands	🟩	V 2.0.0
_ControlNodeWrappers	🟩	
_FunctionBlocks	🟩	

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.



Project library ▸ Types ▸ LBC ▸ LBC_AnalogInputCn

All

Type	Status	Version	Uses	Path
LBC_AnalogInputCn	🟨	V 0.0.4		
V 0.0.5		V 0.0.5		
			LBC_AnalogInp...	Project library\Types\LBC\FunctionBlocks\LBC_AnalogInput\V 1.1.12 [default]
			LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\Input\LBC_typeAnalogInputConfiguration\V 1.1.0 [default]
			LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\Input\LBC_typeAnalogInputInterface\V 1.1.3 [default]
			LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\Input\LBC_typeAnalogInputControlNode\V 0.0.2 [default]
V 0.0.4 [default]		V 0.0.4		
			LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\Input\LBC_typeAnalogInputConfiguration\V 1.1.0 [default]
			LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\Input\LBC_typeAnalogInputInterface\V 1.1.3 [default]
			LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\Input\LBC_typeAnalogInputControlNode\V 0.0.2 [default]
			LBC_AnalogInp...	Project library\Types\LBC\FunctionBlocks\LBC_AnalogInput\V 1.1.13

Figure 4-3: Library management

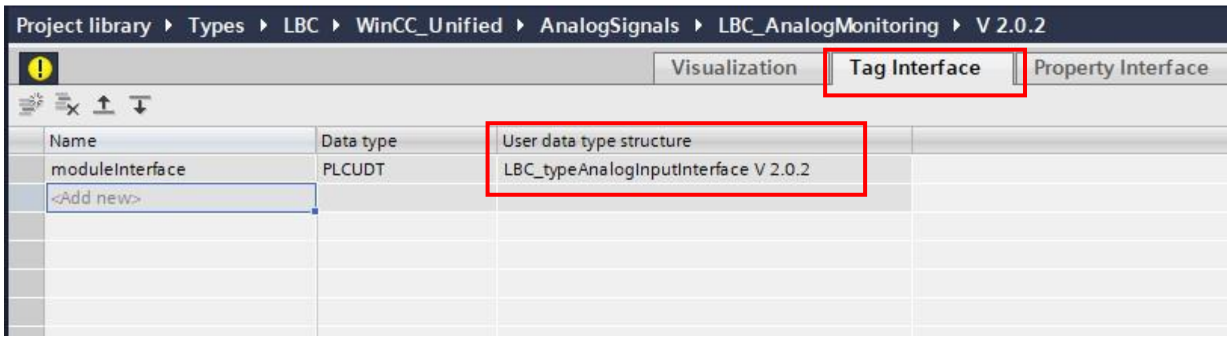
The displayed information helps understanding the current inconsistencies:

- 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.

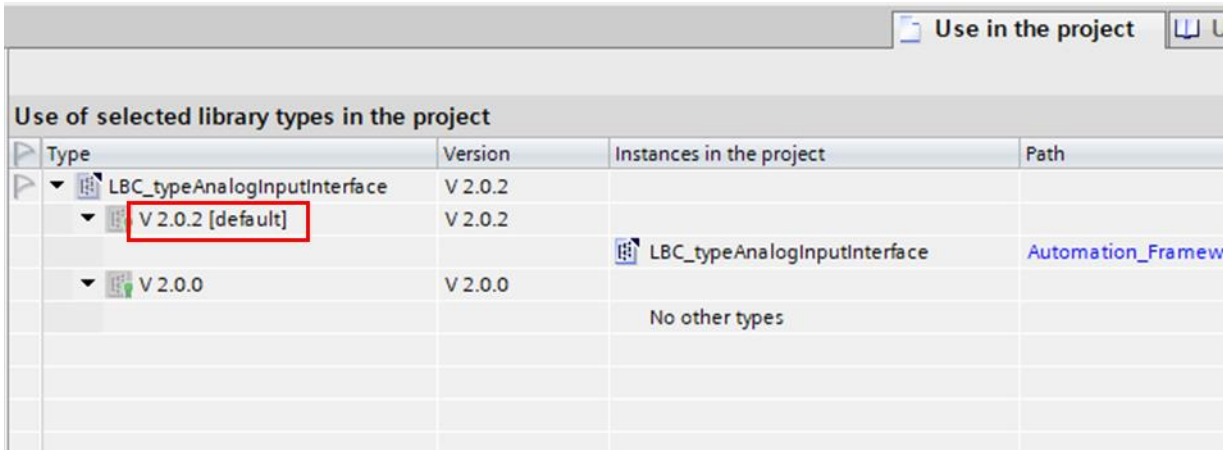

Project library ▸ Types ▸ LBC ▸ WinCC_Unified ▸ AnalogSignals ▸ LBC_AnalogMonitoring ▸ V 2.0.2

Visualization | Tag Interface | Property Interface

Name	Data type	User data type structure
moduleInterface	PLCUDT	LBC_typeAnalogInputInterface V 2.0.2
<Add new>		

Figure 4-4: Tag Interface

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

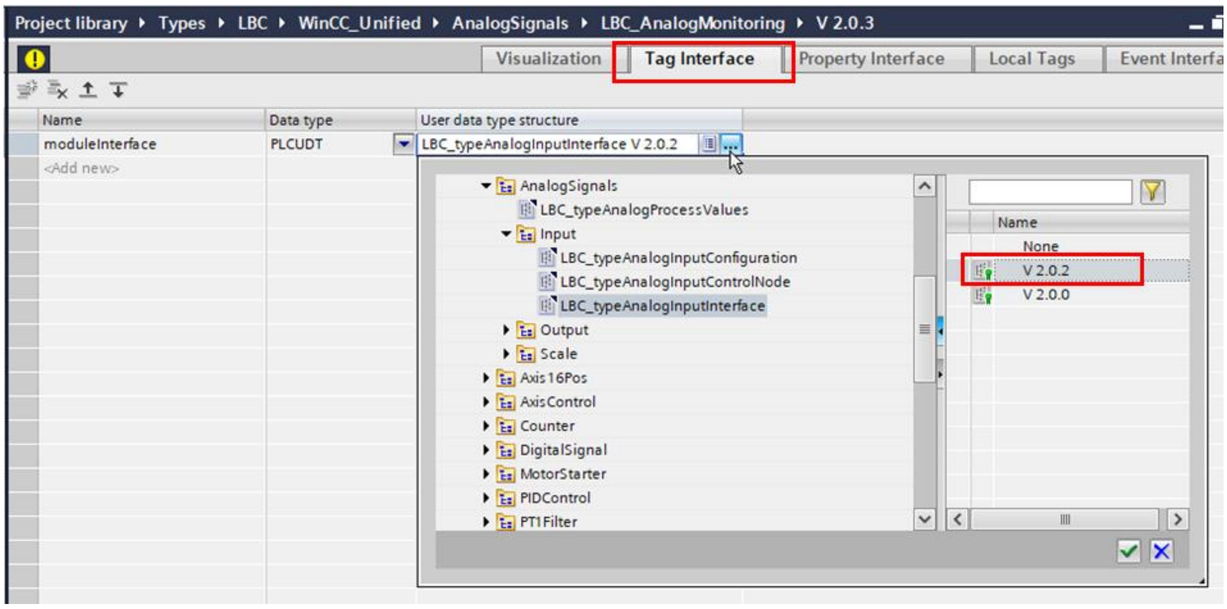

Use in the project

Use of selected library types in the project

Type	Version	Instances in the project	Path
LBC_typeAnalogInputInterface	V 2.0.2		
V 2.0.2 [default]	V 2.0.2		
		LBC_typeAnalogInputInterface	Automation_Framew
V 2.0.0	V 2.0.0		
		No other types	

Figure 4-5: Library management

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


Project library ▸ Types ▸ LBC ▸ WinCC_Unified ▸ AnalogSignals ▸ LBC_AnalogMonitoring ▸ V 2.0.3

Visualization | Tag Interface | Property Interface | Local Tags | Event Interfa

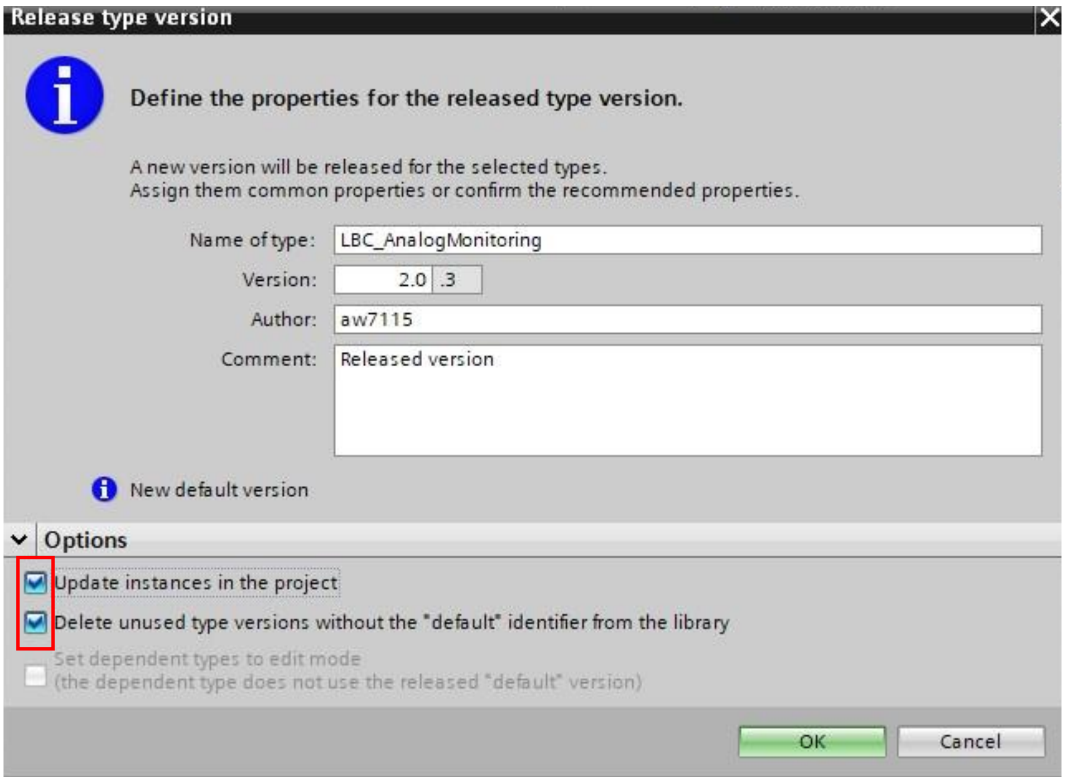
Name	Data type	User data type structure
moduleInterface	PLCUDT	LBC_typeAnalogInputInterface V 2.0.2
<Add new>		

```
AnalogSignals
    LBC_typeAnalogProcessValues
    Input
        LBC_typeAnalogInputConfiguration
        LBC_typeAnalogInputControlNode
        LBC_typeAnalogInputInterface
    Output
    Scale
Axis16Pos
AxisControl
Counter
DigitalSignal
MotorStarter
PIDControl
PT1Filter
```

Name
None
V 2.0.2
V 2.0.0

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.


Release type version

Define the properties for the released type version.

A new version will be released for the selected types.
Assign them common properties or confirm the recommended properties.

Name of type: LBC_AnalogMonitoring
Version: 2.0 .3
Author: aw7115
Comment: Released version

New default version

Options

[✓] Update instances in the project
[✓] Delete unused type versions without the "default" identifier from the library
[ ] Set dependent types to edit mode (the dependent type does not use the released "default" version)

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 versioned which is like a write protection.

If you see the need to create your own derived type from such a block you can follow the next steps in the following section.

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".

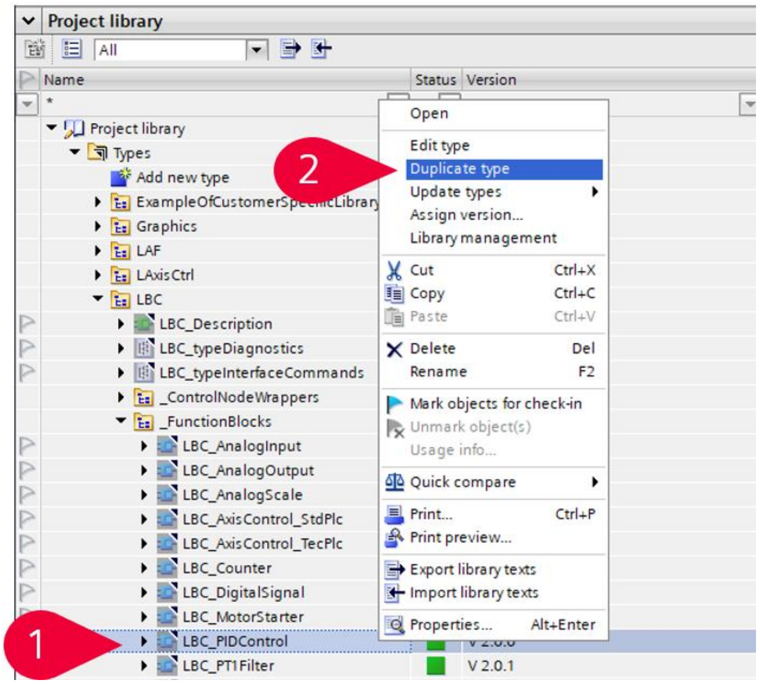
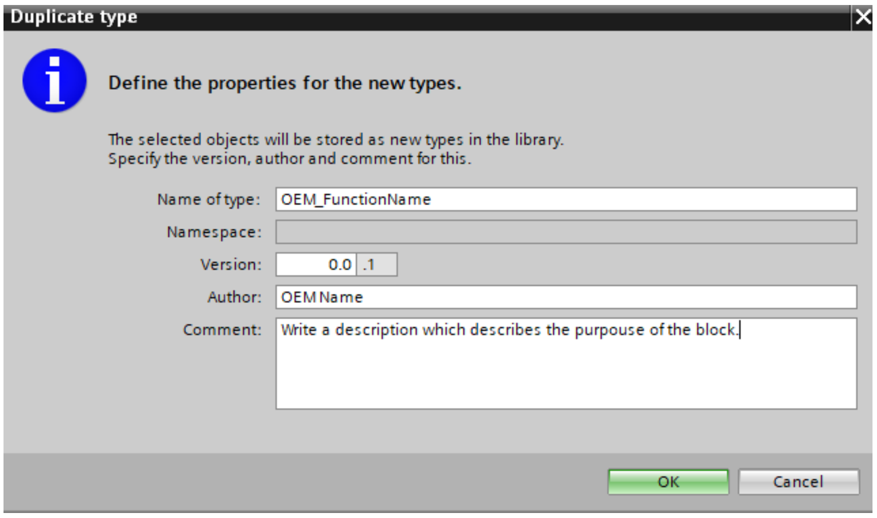


Figure 4-8: Duplicate a type

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


Duplicate type

Define the properties for the new types.

The selected objects will be stored as new types in the library.
Specify the version, author and comment for this.

Name of type:	OEM_FunctionName
Namespace:	
Version:	0.0 .1
Author:	OEM Name
Comment:	Write a description which describes the purpouse of the block.

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.

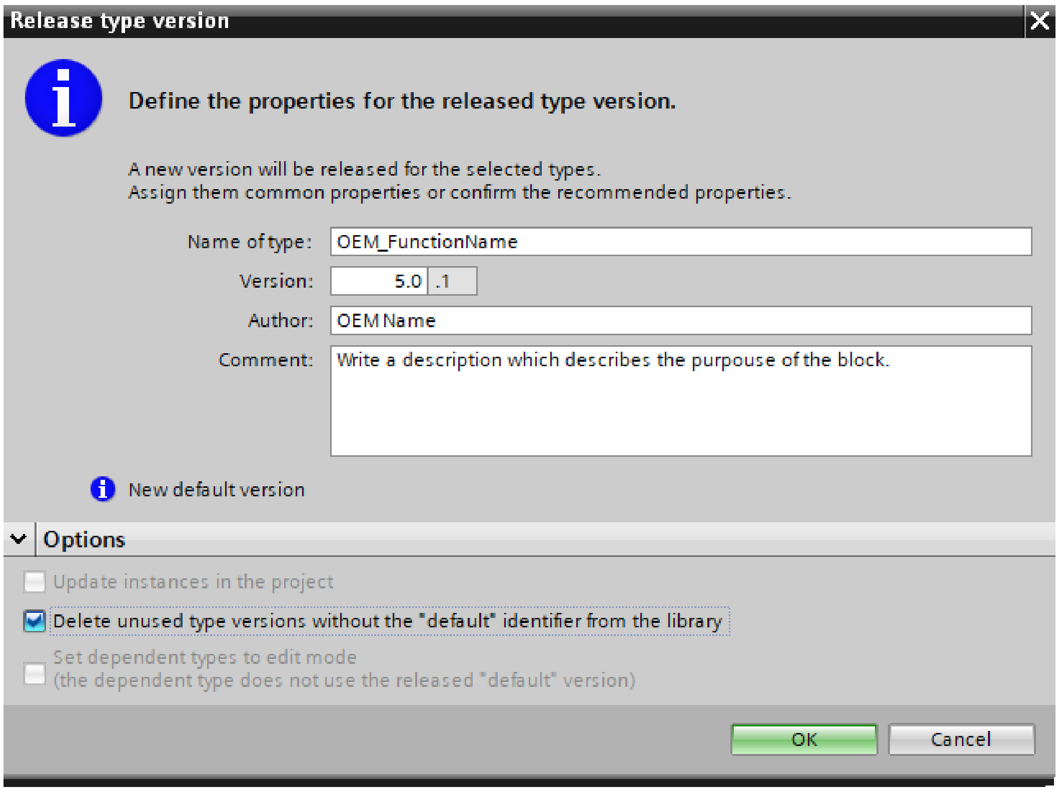
- Before you start the update, create a duplicate of the modified types as described above.
- Now you can run the library update without interference to your modification.

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.


Release type version

Define the properties for the released type version.

A new version will be released for the selected types.
Assign them common properties or confirm the recommended properties.

Name of type: OEM_FunctionName
Version: 5.0 .1
Author: OEM Name
Comment: Write a description which describes the purpouse of the block.

New default version

Options

[ ] Update instances in the project
[✓] Delete unused type versions without the "default" identifier from the library
[ ] Set dependent types to edit mode
(the dependent type does not use the released "default" version)

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.

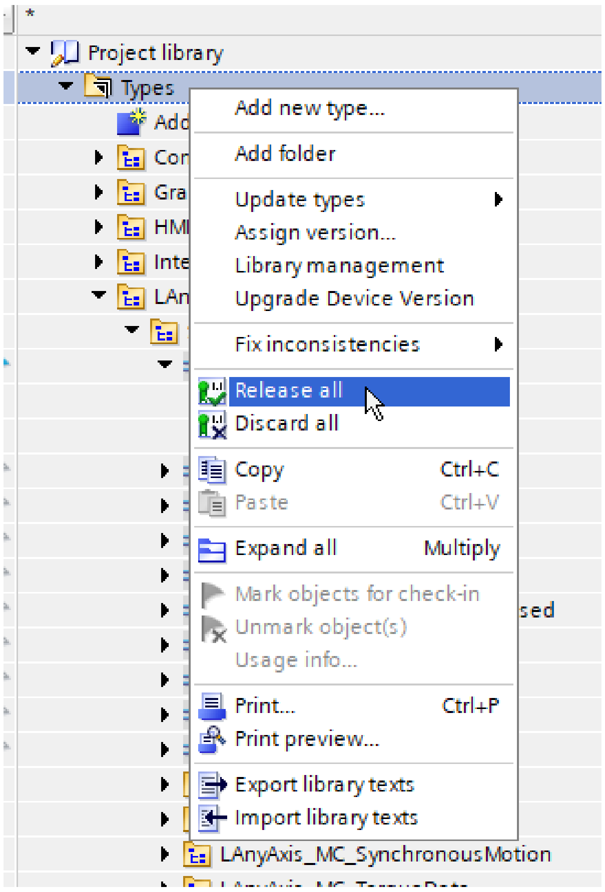


Figure 4-11: Release All

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.

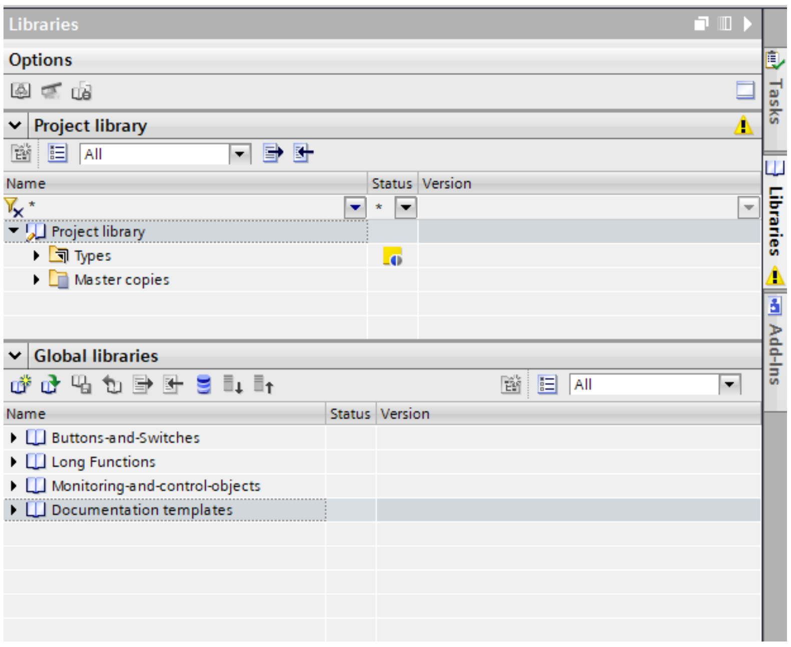
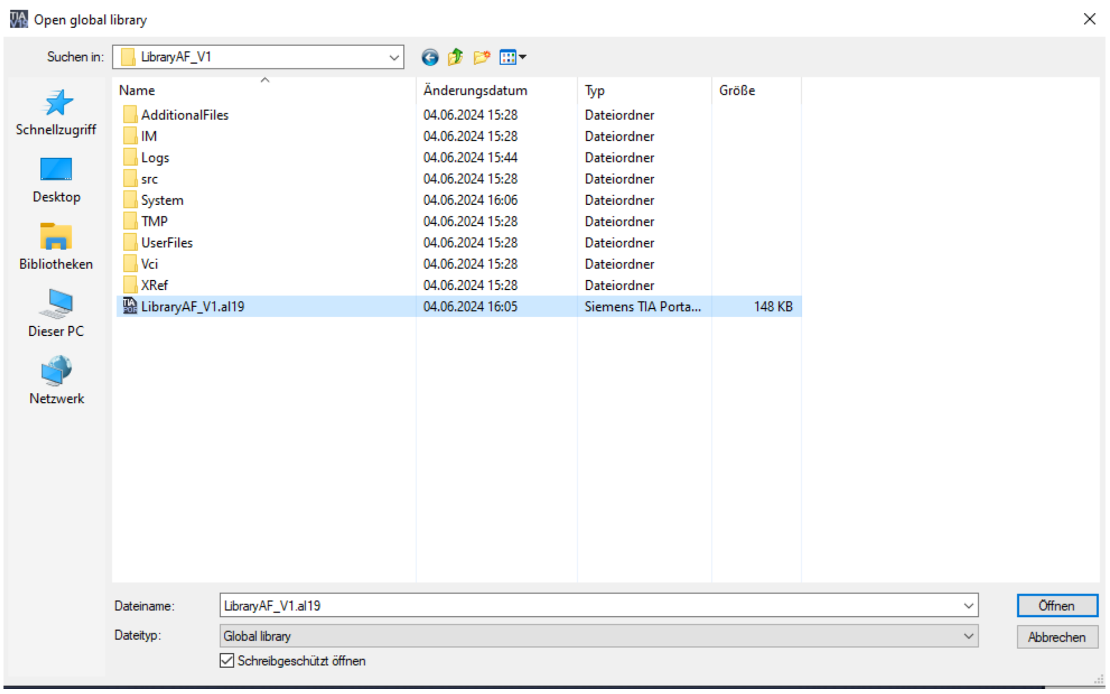


Figure 4-12: Open a global library

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



Name	Änderungsdatum	Typ	Größe
AdditionalFiles	04.06.2024 15:28	Dateiordner	
IM	04.06.2024 15:28	Dateiordner	
Logs	04.06.2024 15:44	Dateiordner	
src	04.06.2024 15:28	Dateiordner	
System	04.06.2024 16:06	Dateiordner	
TMP	04.06.2024 15:28	Dateiordner	
UserFiles	04.06.2024 15:28	Dateiordner	
Vci	04.06.2024 15:28	Dateiordner	
XRef	04.06.2024 15:28	Dateiordner	
LibraryAF_V1.al19	04.06.2024 16:05	Siemens TIA Porta...	148 KB

Dateiname: LibraryAF_V1.al19
Dateityp: Global library
☑ Schreibgeschützt öffnen

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 library will be updated.

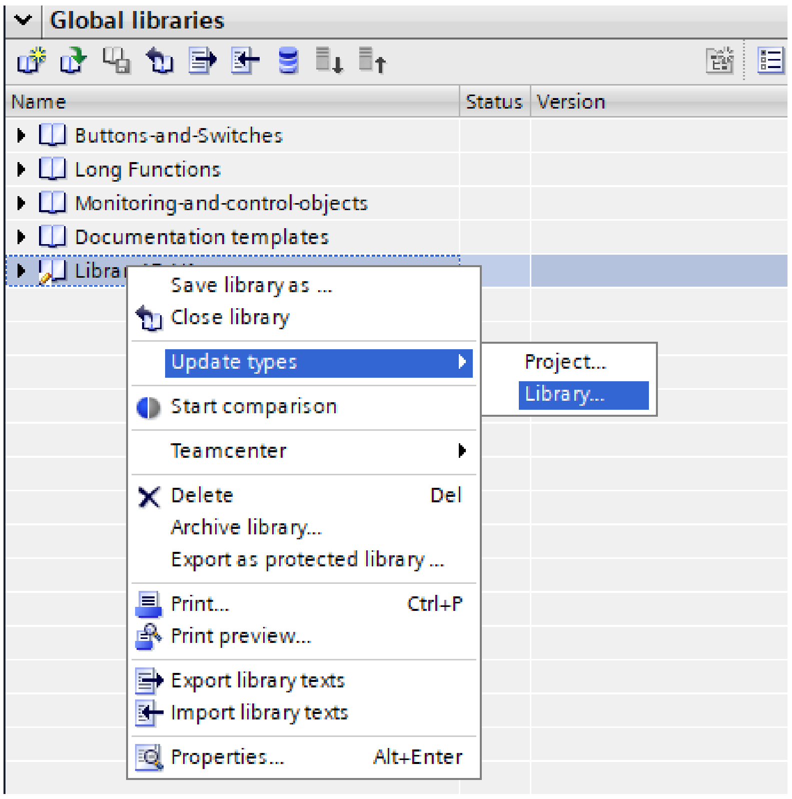
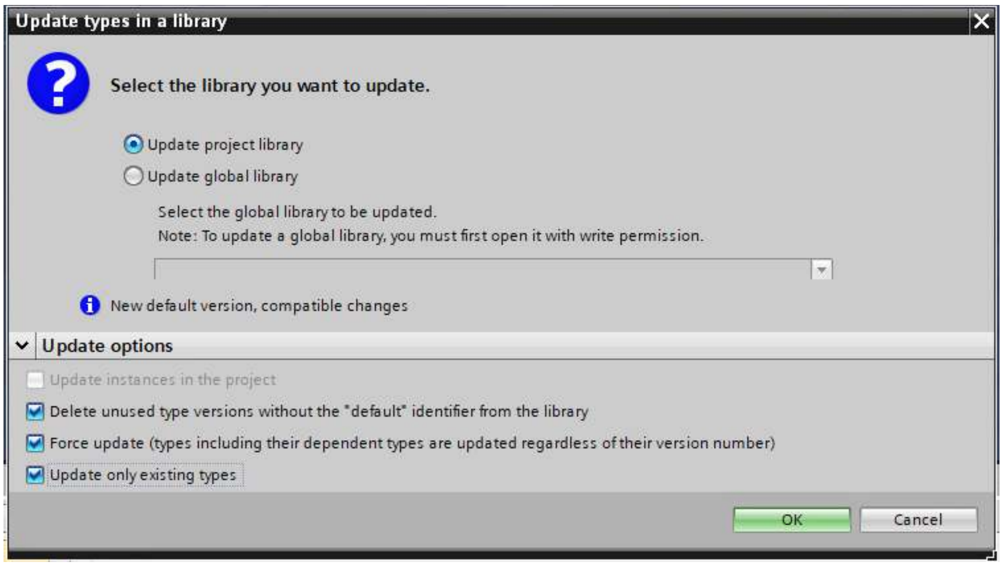


Figure 4-14: Update types from Global library


Update types in a library

Select the library you want to update.

[✓ Update project library]
Update global library

Select the global library to be updated.
Note: To update a global library, you must first open it with write permission.

New default version, compatible changes

Update options

☐ Update instances in the project
☑ Delete unused type versions without the "default" identifier from the library
☑ Force update (types including their dependent types are updated regardless of their version number)
☑ Update only existing types

OK Cancel

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)”.
- Activate “Update only existing types” to update only the types tha are in your project library

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.

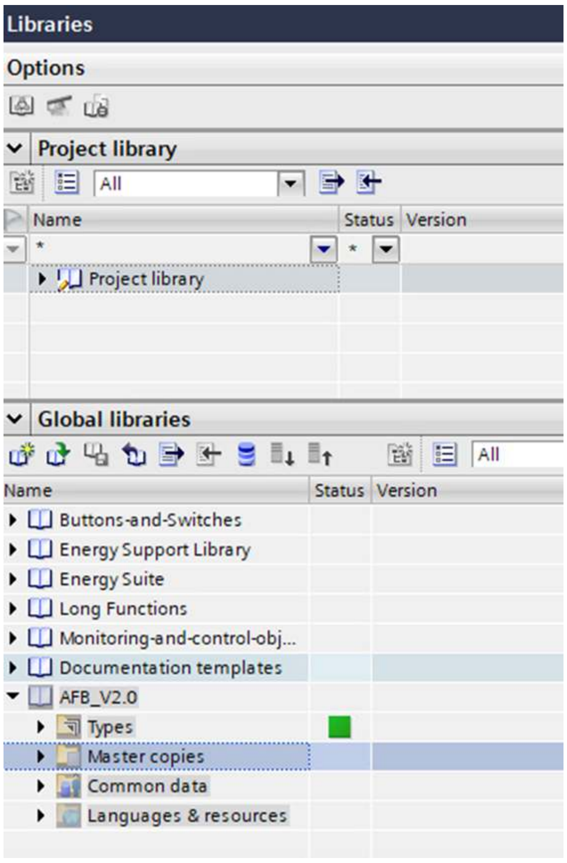
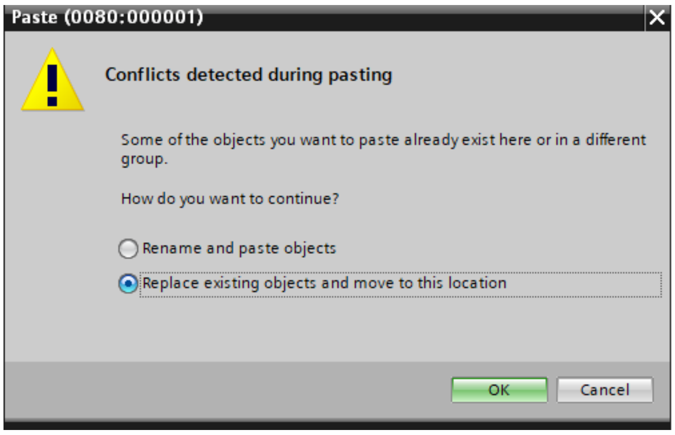


Figure 4-16: Import Master copies


Paste (0080:000001)

Conflicts detected during pasting

Some of the objects you want to paste already exist here or in a different group.

How do you want to continue?

Rename and paste objects [✓ Replace existing objects and move to this location]

Figure 4-17: Overwrite existing items

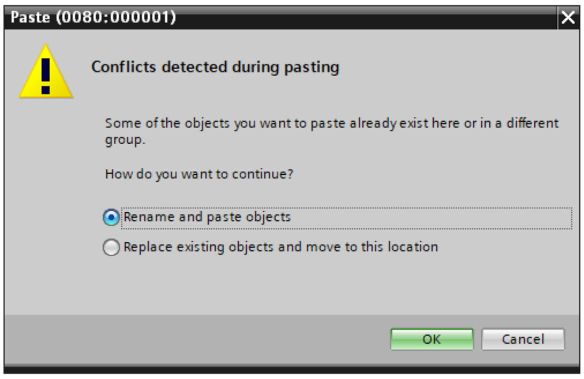
If this message appears, choose “Replace existing objects and move to this location".

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".


Paste (0080:000001)

Conflicts detected during pasting

Some of the objects you want to paste already exist here or in a different group.

How do you want to continue?

(●) Rename and paste objects

( ) Replace existing objects and move to this location

Figure 4-18: Rename and paste objects

The result will look like this:

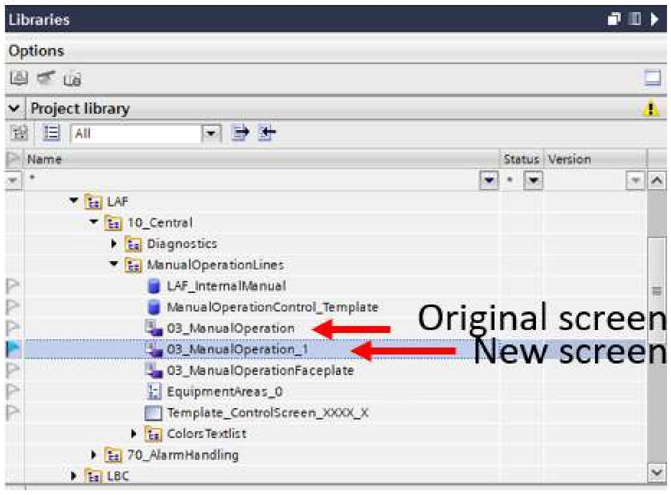


Figure 4-19: Renamed object

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

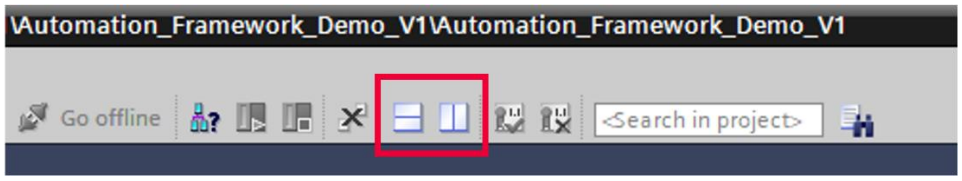


Figure 4-20: Split screen view

5. Hardware configuration

The AFBasic 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. Both types of PROFINET communication can be combined on the same network.

The AFBasic project introduces a network configuration with two drives using IRT (see highlighted in picture below).

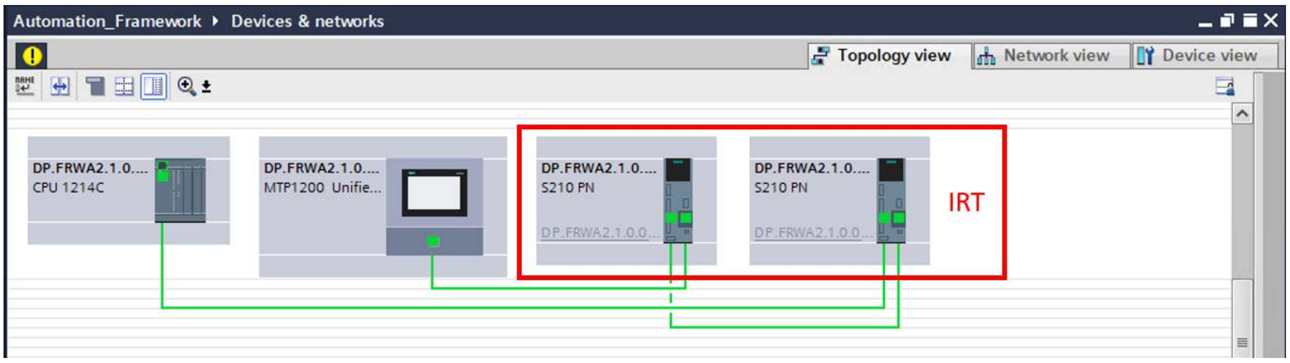


Figure 5-1: Topology-view with RT and IRT configuration

NOTE

By default, IRT supports up to 128 (64+64) network devices using X1 and X2 on innovated CPUs (with FW >= V4.0). 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 (384 (256+128) with CP1625 using X1 and X2 with FW >= V31.1).

- SIMATIC S7-1508S T/TF

- SIMATIC ET 200SP (HS)

- SIMATIC PN/PN Coupler

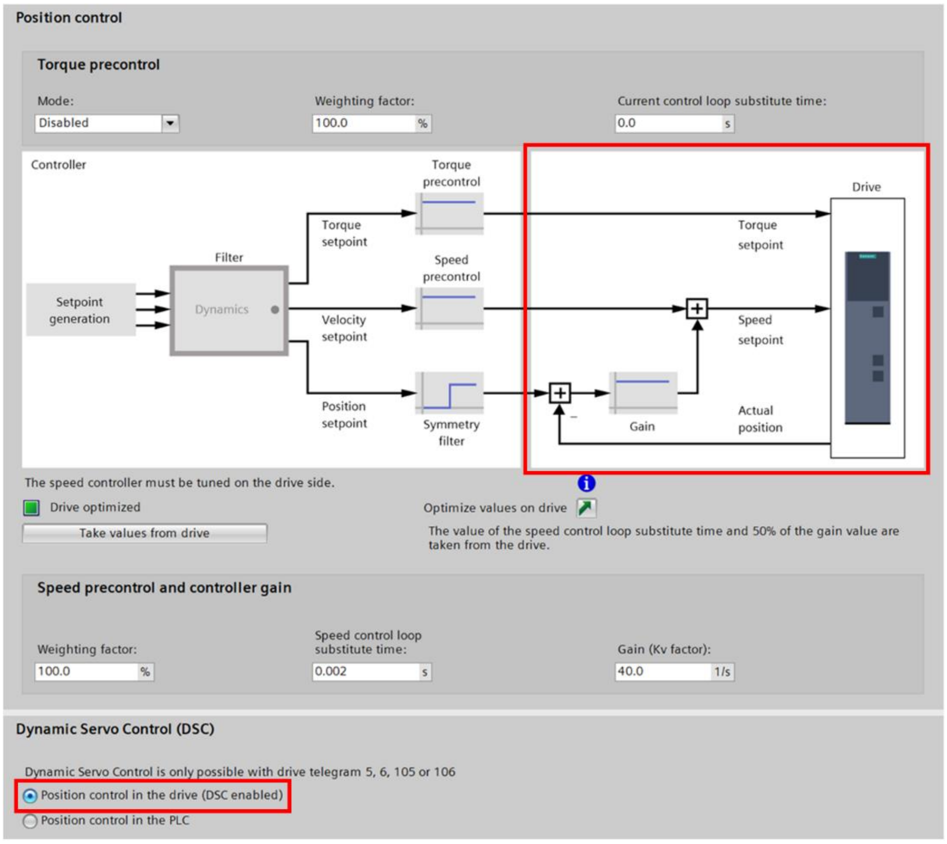
- SINAMICS S200

- SINAMICS S210 (6SL5310...)

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

SIOS-ID: 49948856

For drives using IRT, there is the option to configure “Dynamic Servo Control” (DSC). 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).


Position control

Torque precontrol

Mode:	Weighting factor:	Current control loop substitute time:
Disabled	100.0 %	0.0 s

The speed controller must be tuned on the drive side.

🟩 Drive optimized

Take values from drive

Optimize values on drive

The value of the speed control loop substitute time and 50% of the gain value are taken from the drive.

Speed precontrol and controller gain

Weighting factor:	Speed control loop substitute time:	Gain (Kv factor):
100.0 %	0.002 s	40.0 1/s

Dynamic Servo Control (DSC)

Dynamic Servo Control is only possible with drive telegram 5, 6, 105 or 106

[✓ Position control in the drive (DSC enabled)]

Position control in the PLC

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.

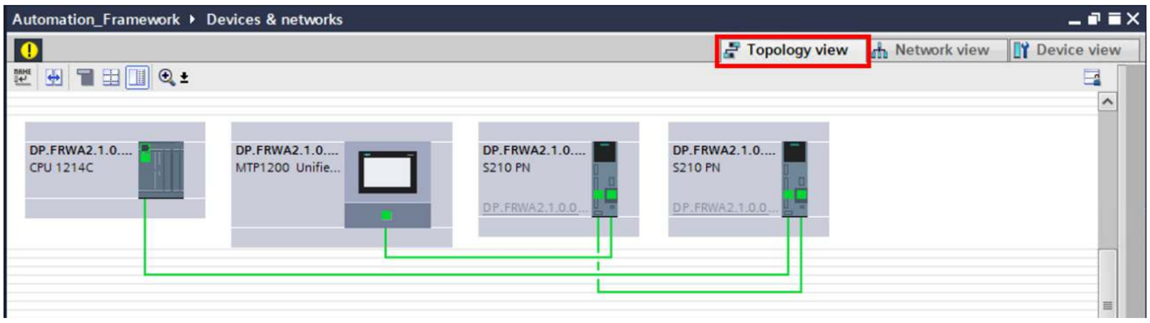


Figure 5-3: Topology view

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 in the function manual “PROFINET with STEP 7”: SIOS-ID: 49948856

5.3. Time synchronization and time settings

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 Time Protocol). The following chapter will introduce the necessary settings to enable a cascaded, system wide time synchronization using the LSNTP library.

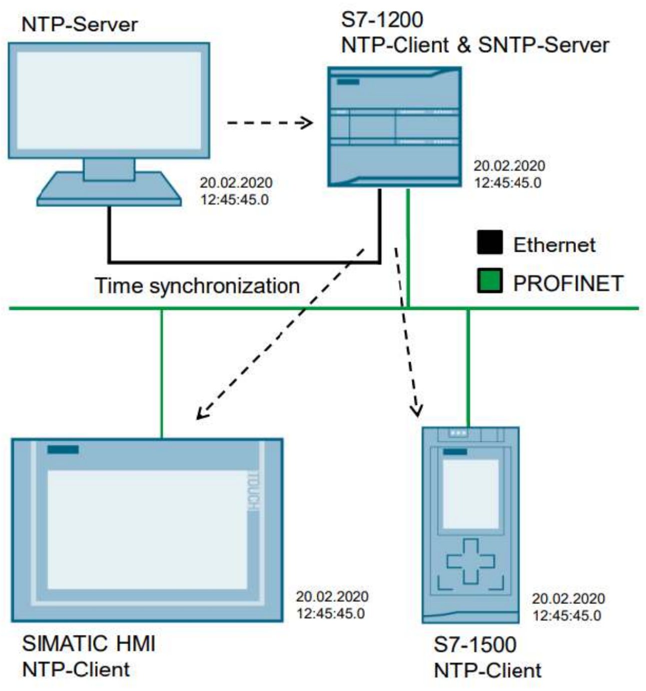
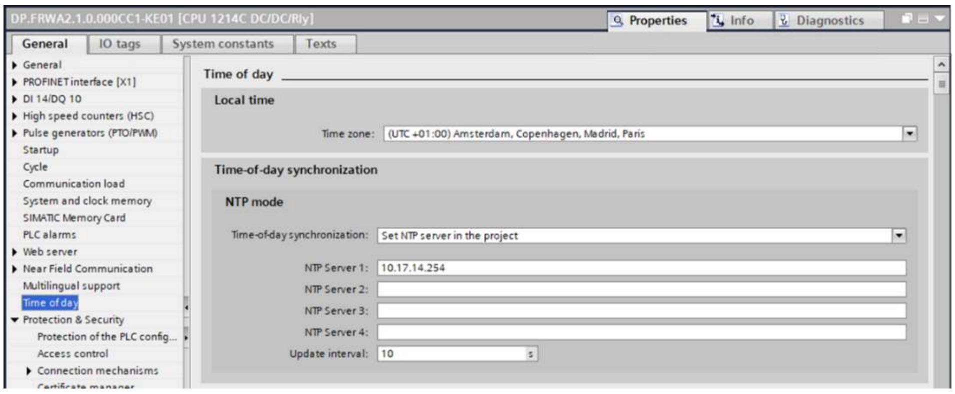


Figure 5-4: Time synchronization network overview

PLC date-time synchronization with a higher-level system

Before the PLC date-time is passed down to the different subsystems using the LSNP library, the PLC date-time can also be synchronized with an external higher-level system (NTP Server).

To do this, the NTP client of the PLC must be enabled. The following settings must be configured in TIA portal. To finish the configuration a download to the PLC is needed.



Time of day

Local time

Time zone: (UTC +01:00) Amsterdam, Copenhagen, Madrid, Paris

Time-of-day synchronization

NTP mode

Time-of-day synchronization: Set NTP server in the project

NTP Server 1: 10.17.14.254

NTP Server 2:

NTP Server 3:

NTP Server 4:

Update interval: 10 s

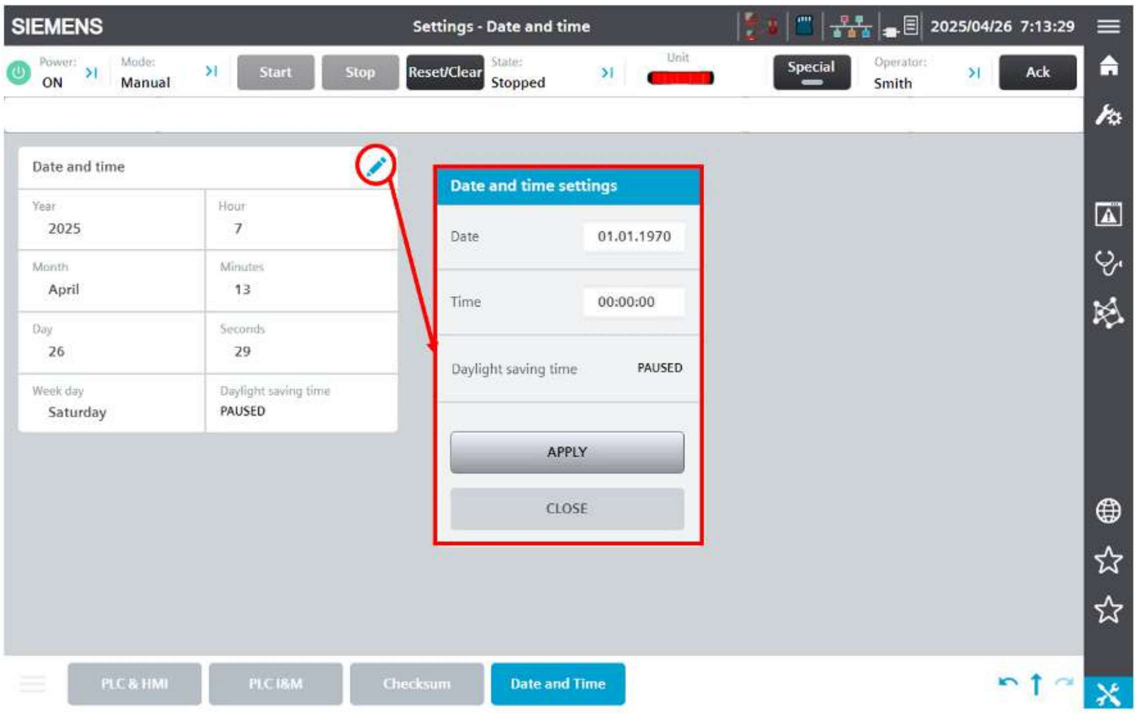
Figure 5-5: Configuration of NTP Client in the PLC

NOTE

The use of an external NTP server for time synchronization is optional.

PLC date-time manual configuration

In case date-time synchronization with an external NTP server is not used, the PLC date-time can be directly set via the AF HMI. The screen “Settings – Date and time” sends the new date and time through the variables in “Settings.setLocalPlcTime” to the instruction “WR_LOC_T” in 01_Central Functions – 14_Settings.



Date and time

Year: 2025	Hour: 7
Month: April	Minutes: 13
Day: 26	Seconds: 29
Week day: Saturday	Daylight saving time: PAUSED

Date and time settings

Date	01.01.1970
Time	00:00:00
Daylight saving time	PAUSED

APPLY
CLOSE

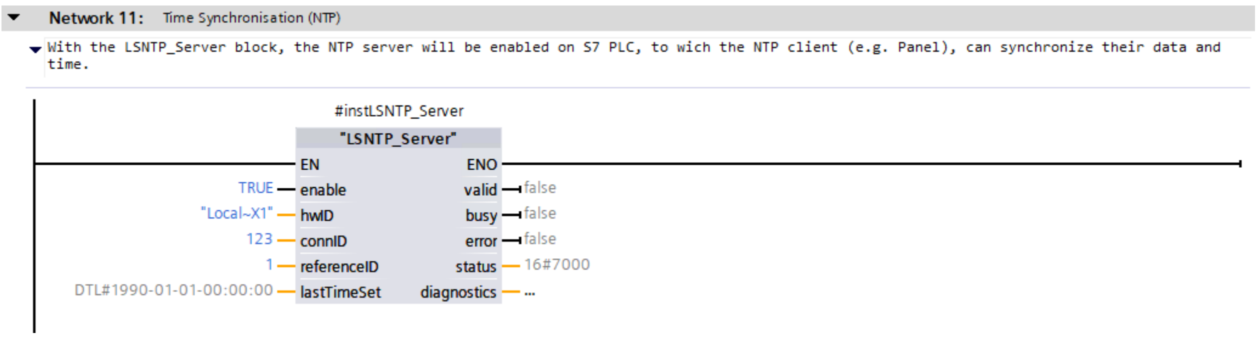
Figure 5-6: Manual configuration of PLC date-time via the AF Settings screen

Subsystem date-time synchronization. SNTP Technical concept

Once the PLC date-time has been configured, either manually via the HMI or automatically via an external system, the PLC can then act as a SNTP (Simple Network Time Protocol) server. This enables flexible and simple synchronization of all subsystems in the network (drives, IOs, HMI panels, etc.), in order to obtain meaningful timestamps for error messages, and logging data system wide. The library LSNTP provides a function block (LSNTP_Server) that performs the following functions:

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

The LSNTP_Server instruction is called from the 01_Central Functions – 00_General.



Network 11: Time Synchronisation (NTP)

With the LSNTP_Server block, the NTP server will be enabled on S7 PLC, to wich the NTP client (e.g. Panel), can synchronize their data and time.

```
                         #instLSNTP_Server
                          "LSNTP_Server"
                        EN              ENO
                TRUE — enable         valid — false
          "Local~X1" — hwID            busy — false
                 123 — connID         error — false
                   1 — referenceID   status — 16#7000
DTL#1990-01-01-00:00:00 — lastTimeSet diagnostics — ...
```

Figure 5-7: LSNTP_Server block activates the NTP server on the PLC

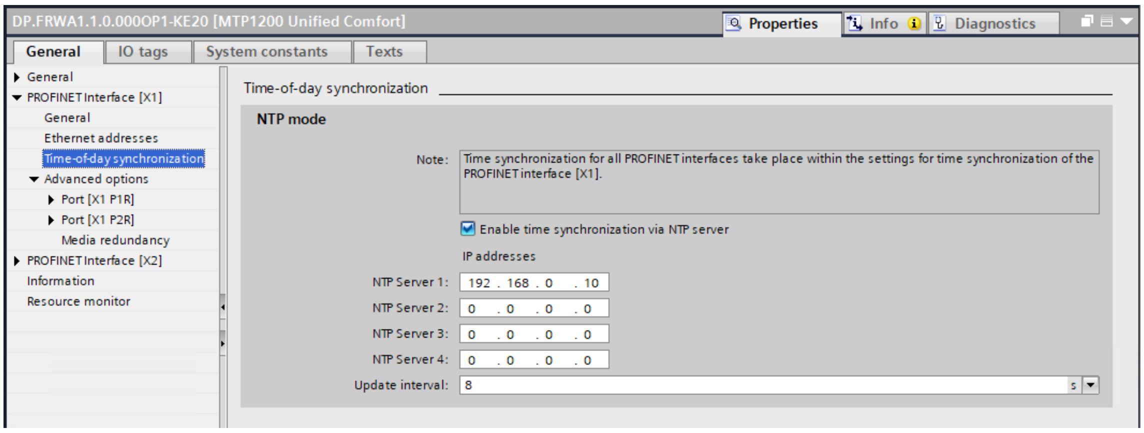
With the SNTP server running on the PLC, the network subsystems (NTP clients) can synchronize their date-time with the central PLC.

NOTE

NTP client function in variable frequency drives, is currently supported only by S120 and S210.

HMI date-time synchronization with central PLC

To configure a Unified Comfort panel as NTP client of the central PLC, the following settings must be configured in TIA portal. The IP address of the NTP Server must be that of the central PLC. To finish the configuration a download is needed.



Time-of-day synchronization

NTP mode

Note: Time synchronization for all PROFINET interfaces take place within the settings for time synchronization of the PROFINET interface [X1].

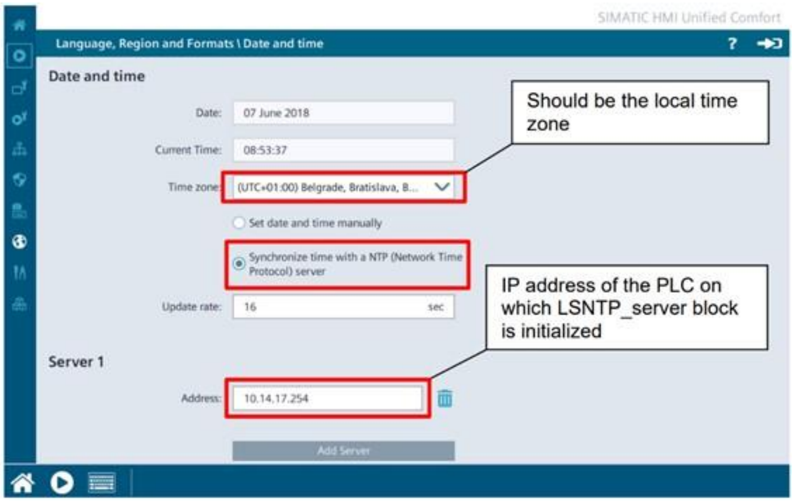
☑ Enable time synchronization via NTP server

IP addresses

NTP Server 1:	192 . 168 . 0 . 10
NTP Server 2:	0 . 0 . 0 . 0
NTP Server 3:	0 . 0 . 0 . 0
NTP Server 4:	0 . 0 . 0 . 0
Update interval:	8 s

Figure 5-8: NTP client setting in TIA Portal

Likewise, the configuration of the NTP client on the Unified Comfort panel can be done directly in the “Date and time” settings of the panel.


SIMATIC HMI Unified Comfort
Language, Region and Formats \ Date and time
Should be the local time zone
IP address of the PLC on which LSNTP_server block is initialized
Add Server

Date and time

Date:	07 June 2018
Current Time:	08:53:37
Time zone:	(UTC+01:00) Belgrade, Bratislava, B...

○ Set date and time manually

● Synchronize time with a NTP (Network Time Protocol) server

Update rate: 16 sec

Server 1

Address: 10.14.17.254

Figure 5-9: NTP client setting in HMI panel

6. PLC Software Architecture

6.1. Overview

The PLC program of the AF$_{\text{Basic}}$ is structured in the following major parts:

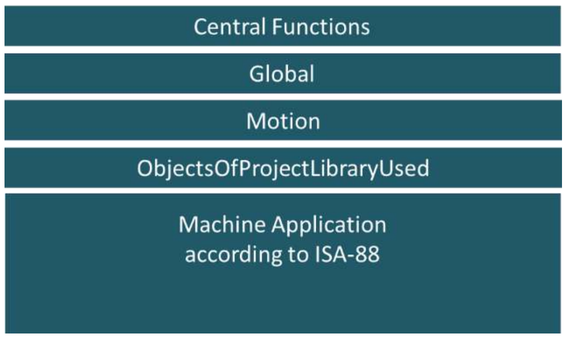

Central Functions
Global
Motion
ObjectsOfProjectLibraryUsed
Machine Application according to ISA-88

Figure 6-1: Main parts of the AF$_{\text{Basic}}$ PLC program

1. Central Functions: Contains general functionality like PLC setting, diagnostics or system features.
2. Global: Contains all data that is globally available in the PLC
3. Motion: Contains the Motion Control OBs
4. ObjectsOfProjectLibraryUsed: Contains the library elements (Blocks, PLC data types) from the project library, that are used in the program.
5. Machine Application: This is a major part of the program. It contains the actual machine application. It is structured according to ISA-88.

Part 1 – 4 are highly standardized and usually don't have to be adjusted by the user. Part 5 is machine and application specific and must be programmed by the user. The structure, interfaces, alarming concept and command and status flow for parts 5 is provided, the user must integrate the specific code.

6.2. 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.

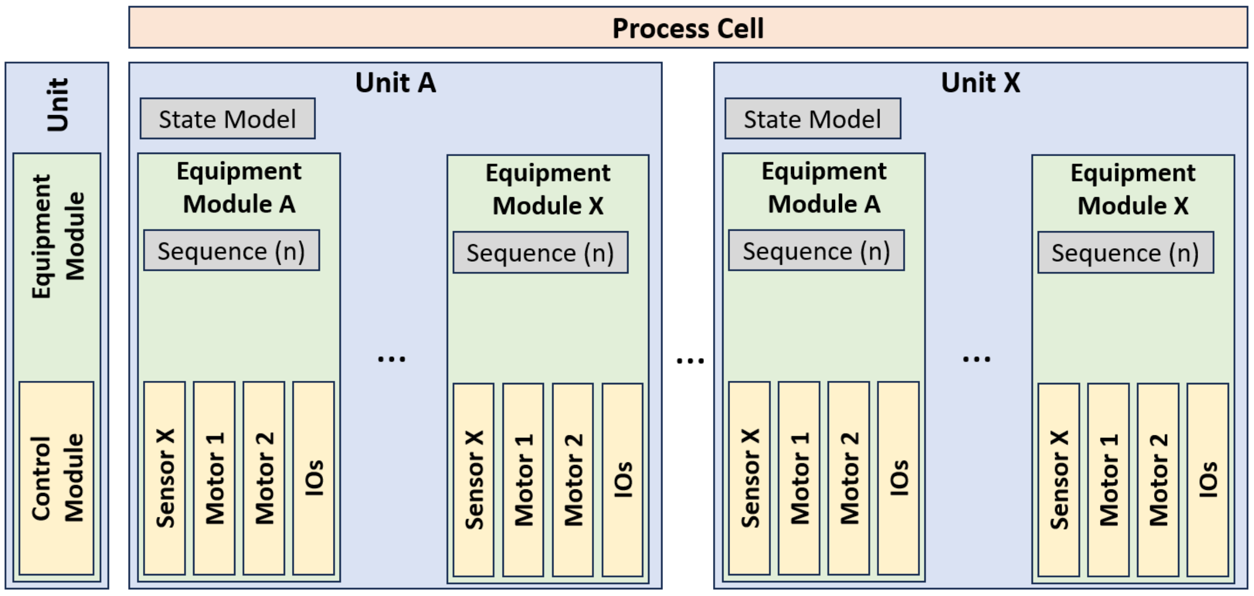


Figure 6-2: ISA-88 model

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.

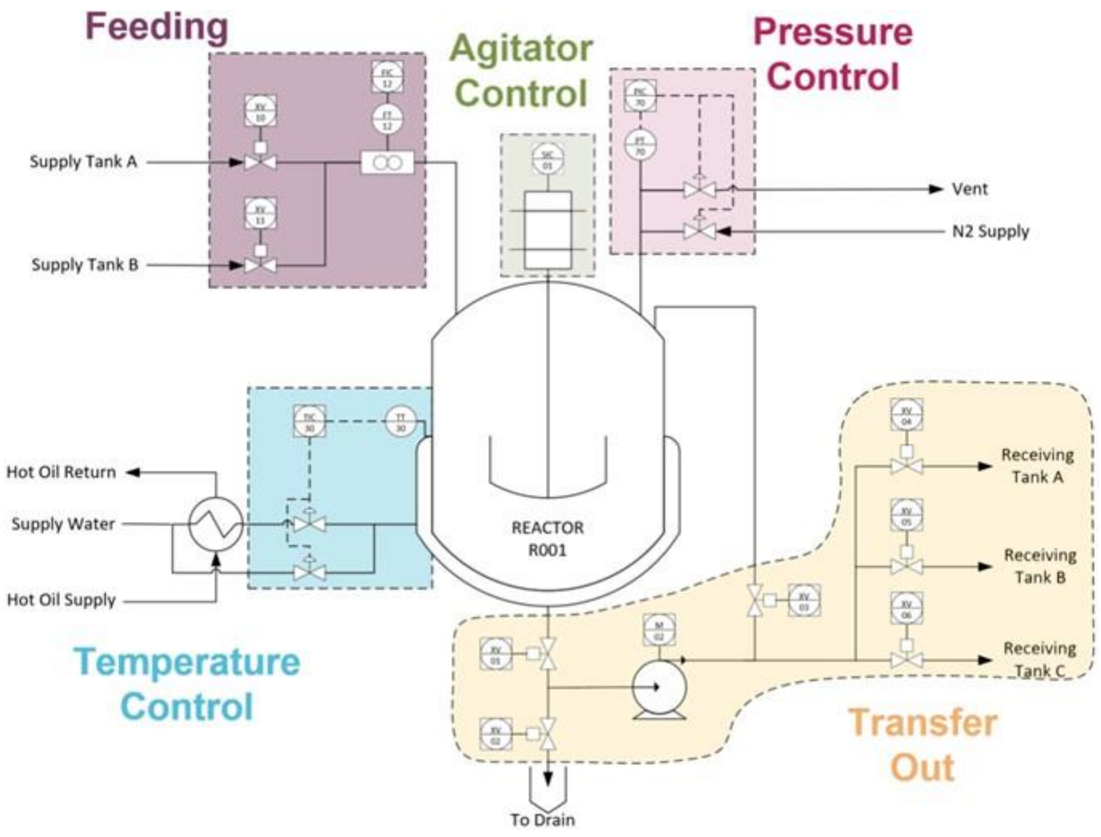


Figure 6-3: Example P&ID from process unit "Reactor" with its EM's (highlighted) and CM's

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$_{Basic}$

The basic implementation of an ISA-88 unit is shown in the following figure. The logic of one AF$_{Basic}$ unit is programmed within a dedicated folder. Each EM also has a dedicated folder. 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.

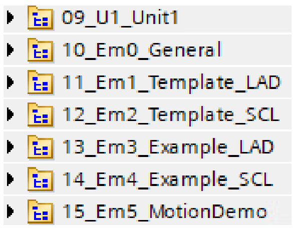

09_U1_Unit1
10_Em0_General
11_Em1_Template_LAD
12_Em2_Template_SCL
13_Em3_Example_LAD
14_Em4_Example_SCL
15_Em5_MotionDemo

Figure 6-4: ISA-88 unit example in the Automation Framework$_{Basic}$

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

6.3. Unit 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

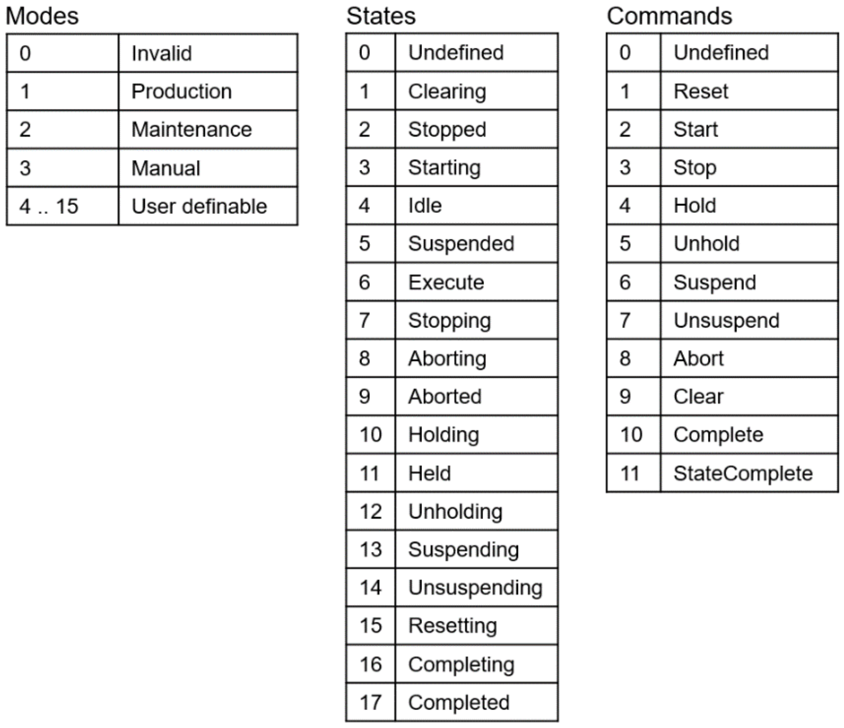
PackML is an automation standard by the OMAC and adopted by ISA as TR88.00.02 that makes it easier to transfer and retrieve consistent machine data. The primary goals of PackML are to encourage a common “look and feel” across a plant floor, and to enable and encourage industry innovation.

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_{Basic}, 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_{Basic} 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_{Basic}:



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 6-5: Modes, states and commands that can be used in the AF$_{Basic}$

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 6-7).

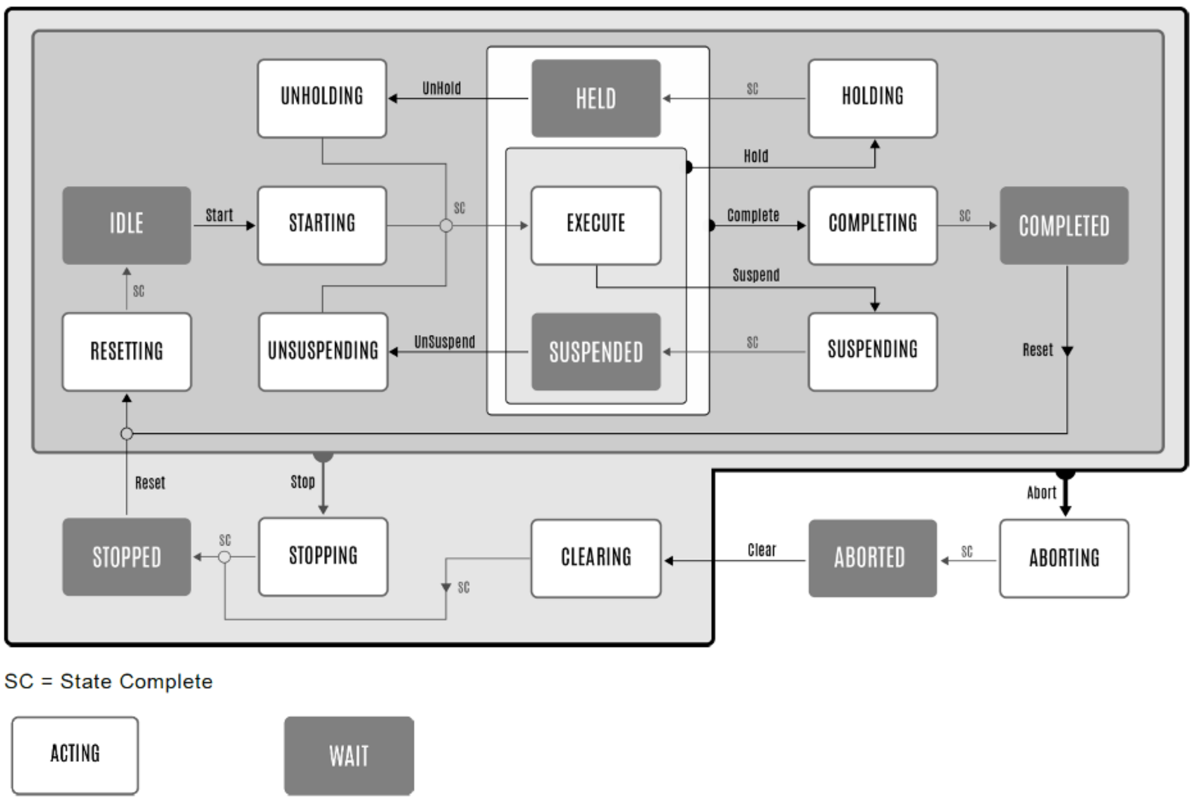


Figure 6-6: Base state model

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 6-7).

In the AF$_{Basic}$ 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).

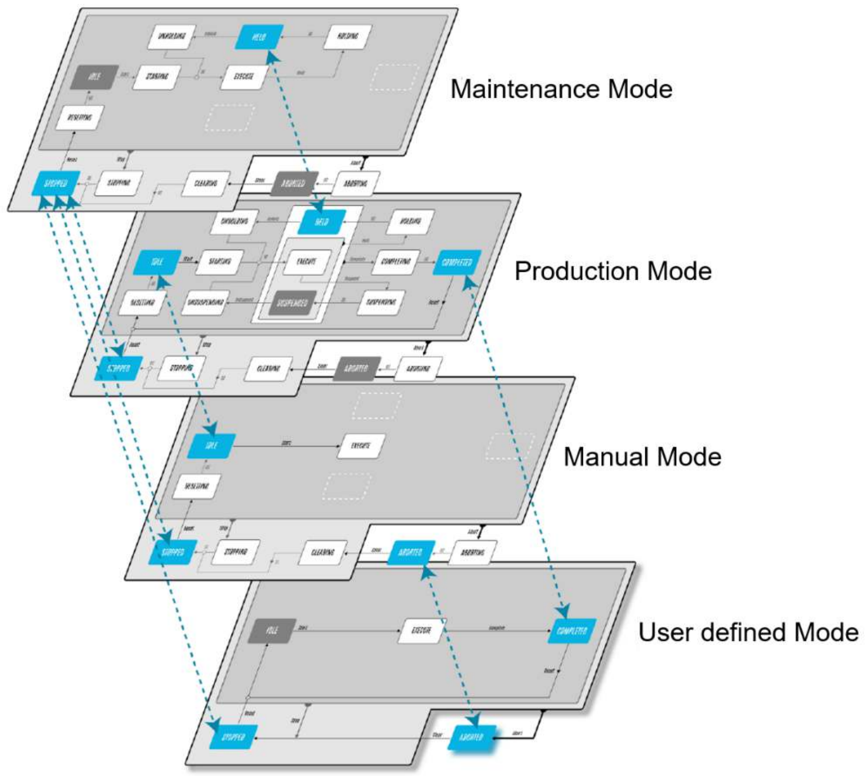


Figure 6-7: Example of different modes with the corresponding configured states and allowed mode changes

Unit States

The different states can be group 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 reach 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 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 library (LPML) mode and state manager \LPML\

6.4. Program structure

Main structure

The PLC program contains global data blocks, that are available for all other modules, for units, EMs, Central Functions and Motion:

- DB HmiControlInterface : Contains the control interface to the HMI.
- DB Data : Contains all PLC wide information
- DB Units : Contains the information that is provided by the individual units
- DB UnitxEms : Contains the information that is provided by the EMs. Each units has and own DB UnitxEms.

These DBs work as central data hubs between the modules. With that, the AF$_{Basic}$ is highly modular and scalable. I.e., an additional EM would be assigned to an existing unit by connecting it to DB "Units", and the EM itself would publish its own status to UnitsxEms.

The following figure shows, how units and equipment modules access the data blocks.

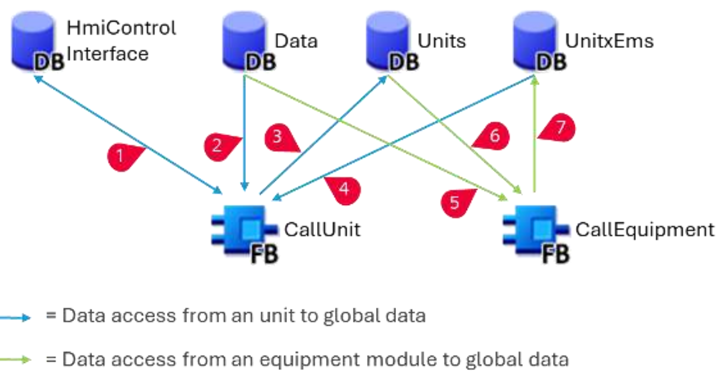

HmiControl Interface
DB
Data
DB
Units
DB
UnitxEms
DB
1
2
3
4
5
6
7
CallUnit
FB
CallEquipment
FB
= Data access from an unit to global data
= Data access from an equipment module to global data

Figure 6-8: Data access between modules and data blocks

Units and EMs access the data blocks for the following reasons (see red indicators in above figure):

1. Unit gets commands from the DB "HmiControlInterface". Additional DB "Interface" could be added, e.g. DB "RecipeControlInterface" and connected to FB "LUC_InterfaceManager" inside the Unit.
   Unit write status back to DB "HmiControlInterface".
2. Unit reads global data from DB "Data". This could be e.g. PLC time, general releases or clock bits.
3. Unit writes information and status to DB "Unit". From there, EMs and other Units can read that information. It can also be used for the HMI for visualization.
4. Unit read status of relevant EMs from DB "UnitxEms". This includes if the EM is linked, feedback of the EM if the execution of the code for the current state is complete (state complete) and commands derived from alarms, e.g. stop command due to a control module error.
5. EM reads global data from DB "Data". This could be e.g. PLC time, general releases or clock bits.
6. EM reads unit status from DB "Units", e.g. mode, state and unit releases.
7. EM writes status to DB "UnitxEms". This includes if the EM is linked, feedback of the EM if the execution of the code for the current state is complete (state complete) and commands derived from alarms, e.g. stop command due to a control module error.

Units and equipment modules consist of the following main parts:

- DBs for alarms and configuration data.
- OBs for execution of startup logic and main logic.
- FB "CallUnit" as call environment for all other library blocks, user defined blocks and direct logic.

Equipment modules additionally contain

- DBs for HMI data and command and status handling to job, sequences and control modules.

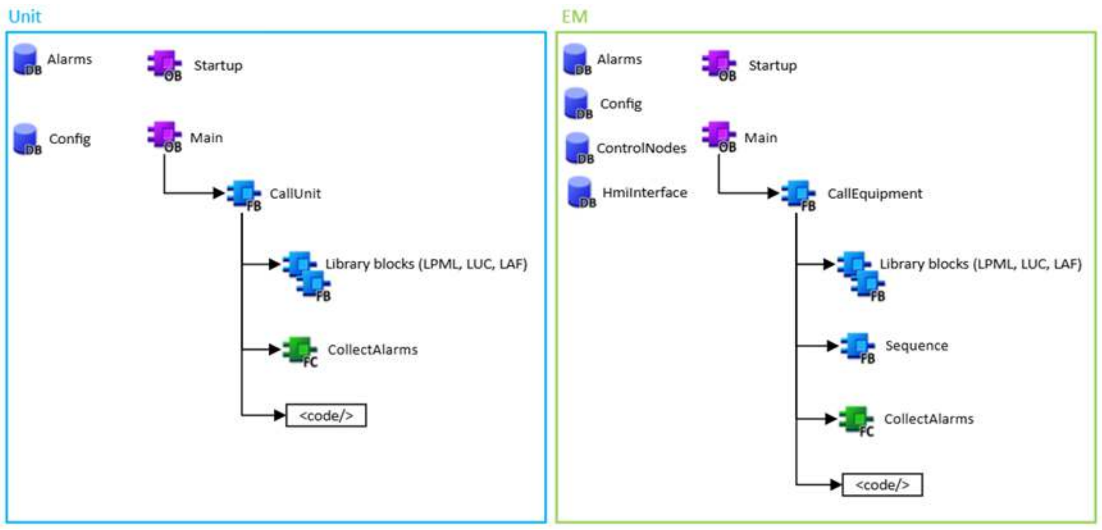


Figure 6-9: Main program blocks of a ISA-88 units and equipment modules

Details can be found in the dedicated chapter about Units and Equipment Modules.

Data handling of FB CallUnit and FB CallEquipment

Two different approaches how to handover data to FB "CallUnit" or FB "CallEquipment" are possible.

6. Handover data from outside the unit / equipment through the block interface and access data from own DBs directly.
7. Handover data exclusively through the block interface

The following figure shows the two solutions schematically for an FB "CallEquipment".

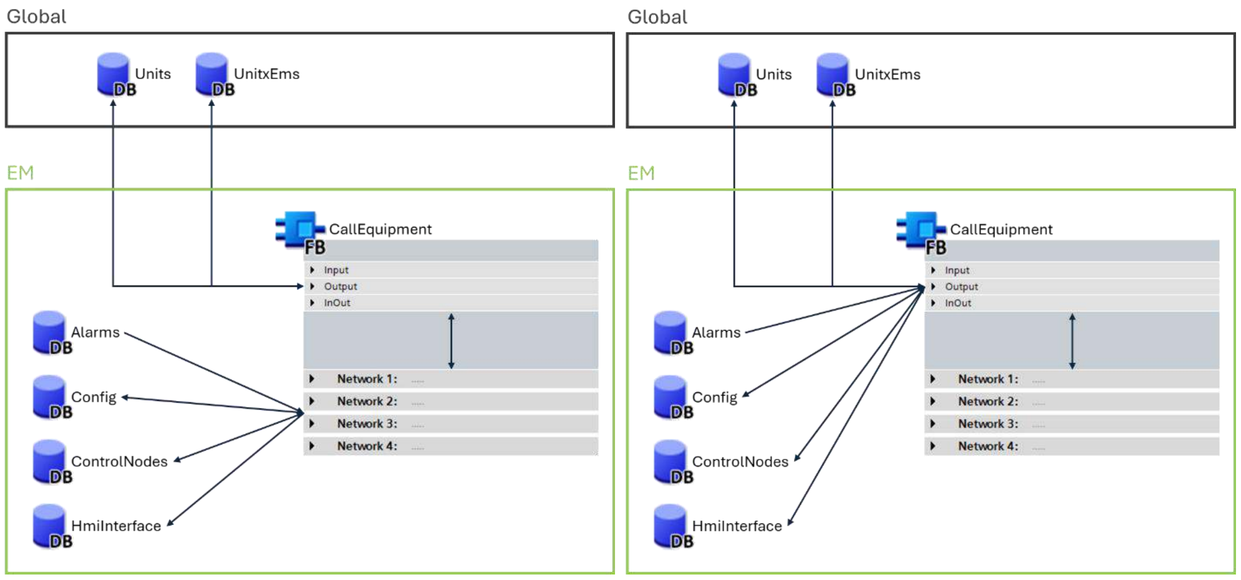


Figure 6-10: Different approaches for data handling. Left: Solution 1. Right: Solution 2

The AFBasic follows approach 1as the benefits outperform the drawbacks for most applications. For very high standardization however, approach 2 is more suitable. An implementation of that approach is shown in Unit 1 – EM 5.

A detailed comparison of both approaches is discussed in chapter 6.10.1 Passing data via block interface vs. direct global access.

Central Functions

Central functions are running outside of units or equipment modules centrally for the PLC. These could be:

- PLC system functionality like PLC clock and cycle time monitoring
- Central hardware diagnostics
- Handling of multiple HMI panles.

The program part "Central Functions" is allowed to access any data accross the PLC.

A central OB "Main" calls all central functions.

Central functions are primarily provided by the AF_{Basic}, hoewever, user could add their own central functions.

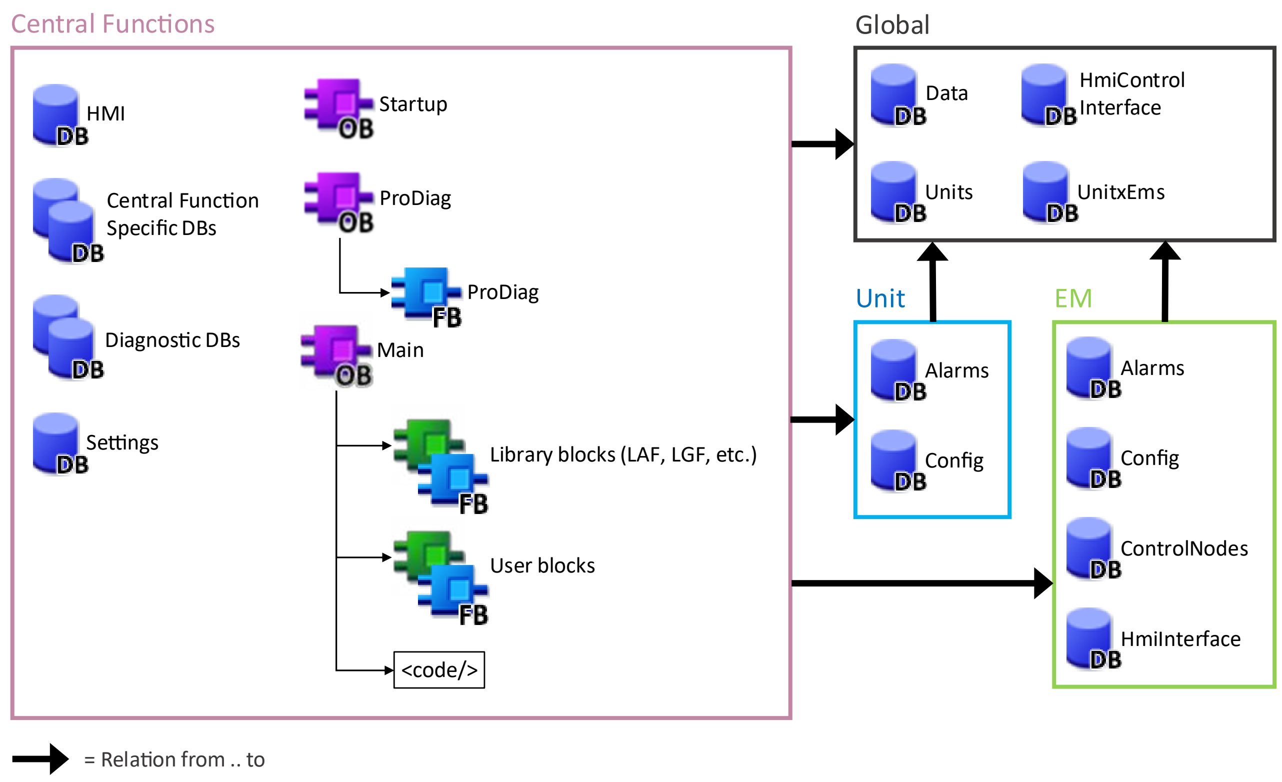


Figure 6-11: Content and interfaces of "Central Functions"

6.5. Program Folders

6.5.1. Overview

The Automation FrameworkBasic uses folders to structure the PLC program. The AFBasic project comes with the following folders for general functionality:

- "Central Functions" → Mandatory
- "Global " → Mandatory
- "Central Functions" → Mandatory
- "ObjectsOfLibraryUsed" → Mandatory
- "Motion" → Optional, if technology objects are used for motion control

6.5.2. Machine application structure

The machine application part of the PLC could be structured into software in multiple ways. By default, the standard structure is used. Each ISA-88 unit and equipment module is programmed in an own folder, including an own OB Main:

- "Ux_Unitx" → Contains the code for unit x.
- "UxEMy_EMx" → Contains the code for EM (y), that belongs to unit (x).

Alternative ways on how to structure the ISA-88 units and EMs into folders are possible. It is recommended to use the standard folder structure, except specific requirement or limits occur. Details are described further below.

Only the standard solution is provided in the Automation Framework Basic. The user could, however, change easily to another structure. The data flow (commands and status) is only affected slighly.

Standard structure

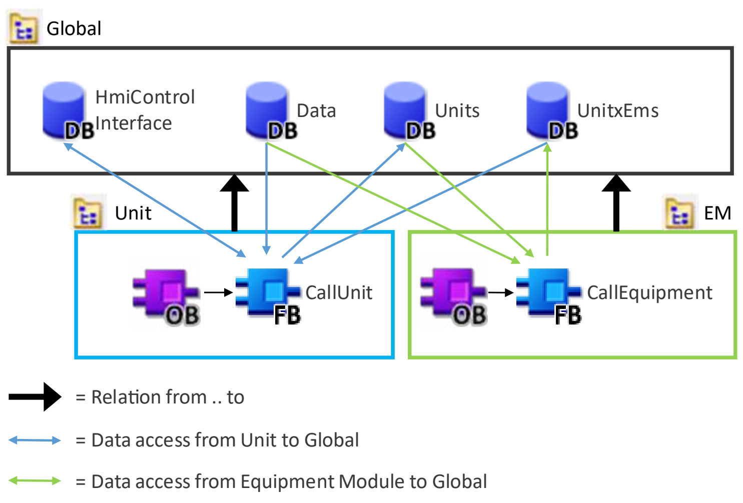


Figure 6-12: Standard folders structure of AFBasic for the ISA-88 machine application

Combined structure

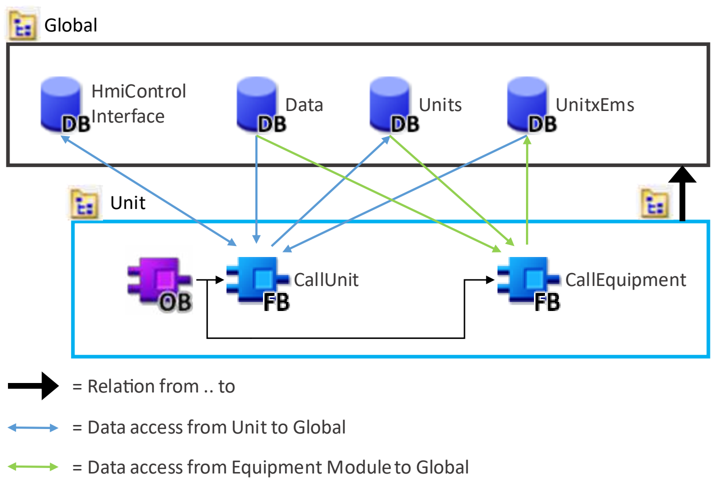


Figure 6-13: Combined folder structure of AF_{Basic} for the ISA-88 machine application

In the combined structure, no dedicated folders exist for the equipment modules. The ISA-88 unit and its underlaying folders are all programmed in one folder.

Hierarchical structure

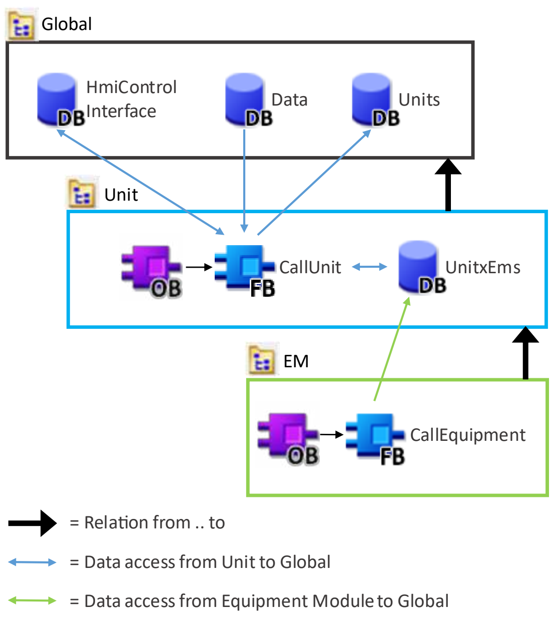


Figure 6-14: Hierarchical folders structure of AF$_{Basic}$ for the ISA-88 machine application

In the hierarchical structure, each ISA-88 unit and equipment module are programmed in an own individual folder including an own OB Main. DB “UnitxEMs” are located in folder “Unit” instead of “Global”.

Comparison of folders structures

	Standard structure	Combined structure	Hierarchical structure
Amount of folders	(o)	(++) Fewer folders	(o)
Amount of OBs	(o)	(++) Fewer OBs	(o)
Data exchange between EMs	(++) Flat hierarchy ensures data accessibility from all EMs to all other EMs.		If an EMs should access data from an EM of another unit, this might take an additional PLC cycle, as the unit has to access the data to global first, to share it with the other unit.
Differentiation between unit and EM	(+) Clear code differentiation between unit and EM.	(o) No code differentiation between unit and em on folder level.	(++) Very clear differentiation of code and data interfaced between unit and EM

Table 6-1: Comparison of the different folder structures

Decision guide

- The standard structure is the recommended solution, as it is most modular and scalable fullfilling all standard requirements.
- The combined structure should be used, if:
  a. a huge amount of EMS and/or units is used.
  b. the equipment modules are highly standardized an are instantiable FBs.
- The hierarchical structure shoud be used if the user puts a high emphasis on data seperation and clear definition of data flow. Due to the point that the EM can only access data from "Unit" and not from "Global", unwanted cross-access of data between units and EMs are restrained.

6.5.3. Content of program structure

The following figure shows the folders and blocks in the AF$_{\text{Basic}}$.

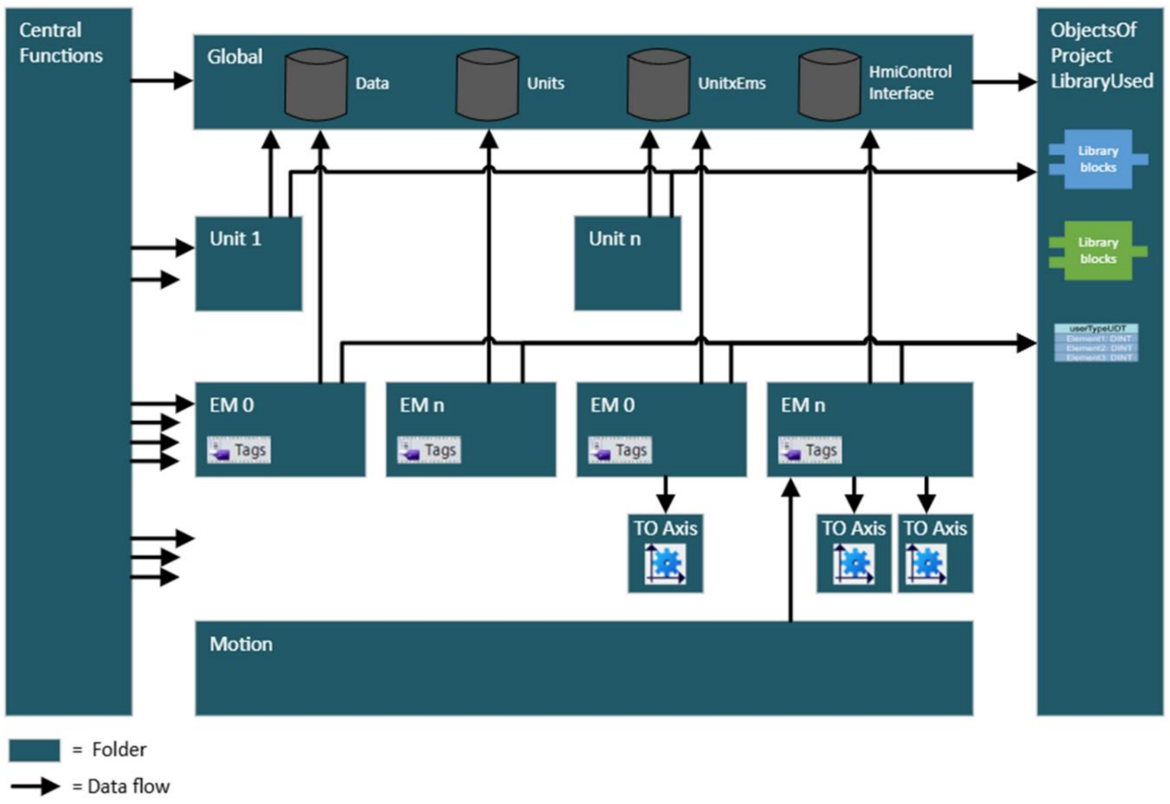


Figure 6-15: Content of program structure

The folders contain the following content:

- **CentralFunctions:** Includes sytem wide functions and diagnostics. Central Functions can access data from other folders, but if other folders want to acces data from Central Functions should be done through a DB.
- **Global:** Includes global data of all units and contains no logic, it is a purely passive folder. This folder accesses 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. 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 folder MotionDemo (see chapter EM Demo Implementation – Motion Demo).
- **ObjectsOfProjectLibraryUsed:** Local instances of the libraries that are used in the PLC program. It only contains blocks that are used by others.
- **Unit:** In the standard software structure, each ISA-88 unit is realized in a dedicated folder. The following relations exist:
  - Global: To read global status, commands, EM feedback and to share own status

  - ObjectsOfProjectLibraryUsed: To use elements from library

- **Equipment Modules:** In the standard software structure, each EM is realized in a dedicated folder. The following relations exist:
  - Global: To read global status, unit status and to share own feedback to the unit
  - ObjectsOfProjectLibraryUsed: To use elements from library
  - TOs: Only needed if TO Axes is used. EMs that execute MC commands need to have access the TOs via the relation.

6.6. Startup and cyclic program execution

The execution sequence of the user program depends on the initial event (PLC start or cyclic operation), the OB priority, and the OB number.

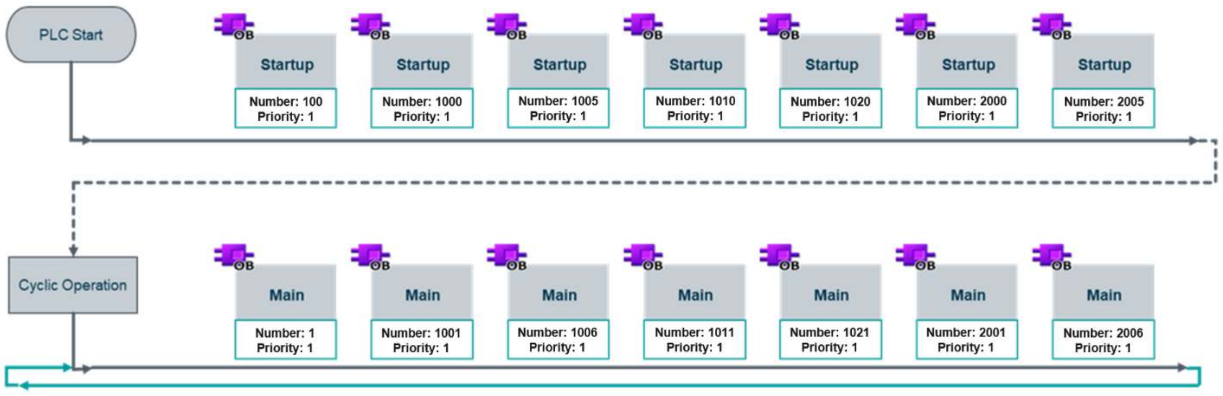


Figure 6-16: Call hierarchy of startup and cyclic OBs

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$_{\text{Basic}}$, each Unit has its own OB “Main” and OB “Startup” . To ensure the correct call hierarchy, the following numbering convention is used:

OB number = ueet where

u: Unit number from 1-9

ee: EM number from 0-99

t: 0 = Startup OB, 1 = Main OB, 3/4=empty, 5=StartupEM0, 6=MainEM0, 8/9=empty

Examples:

OB Number = 2001 : Unit 2, Main OB

OB Number = 3020 : Unit 3, EM 2, Startup OB

Unit Nr.	Unit Startup	Unit Main	EM 0 Startup	EM 0 Main	EM 1 Startup	EM 1 Main
1	1000	1001	1005	1006	1010	1011

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

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. Unit 1 OB Main
3. Unit 1, EM 0 OB Main
4. Unit 1, EM 1 OB Main
5. ...
6. Unit 2 OB Main
7. Unit 2, EM 0 OB Main
8. Unit 2, EM 1 OB Main
9. ...

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

6.7. Data flow

The following figure show the flow of the main command and status. It is focused merely on the main command and neglects all other functionality. The goal is to show a general overview of the core of the AF Basic.

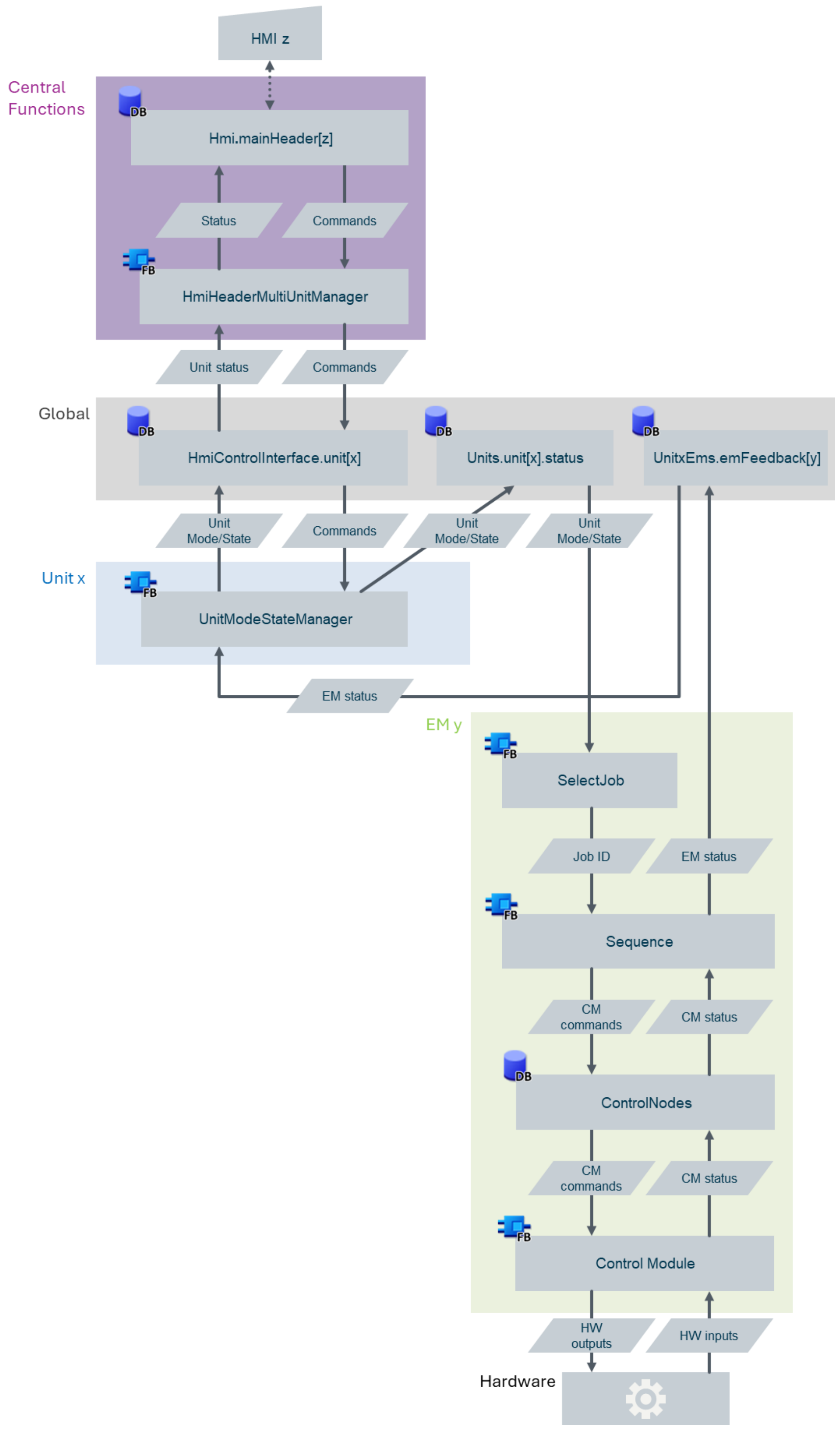


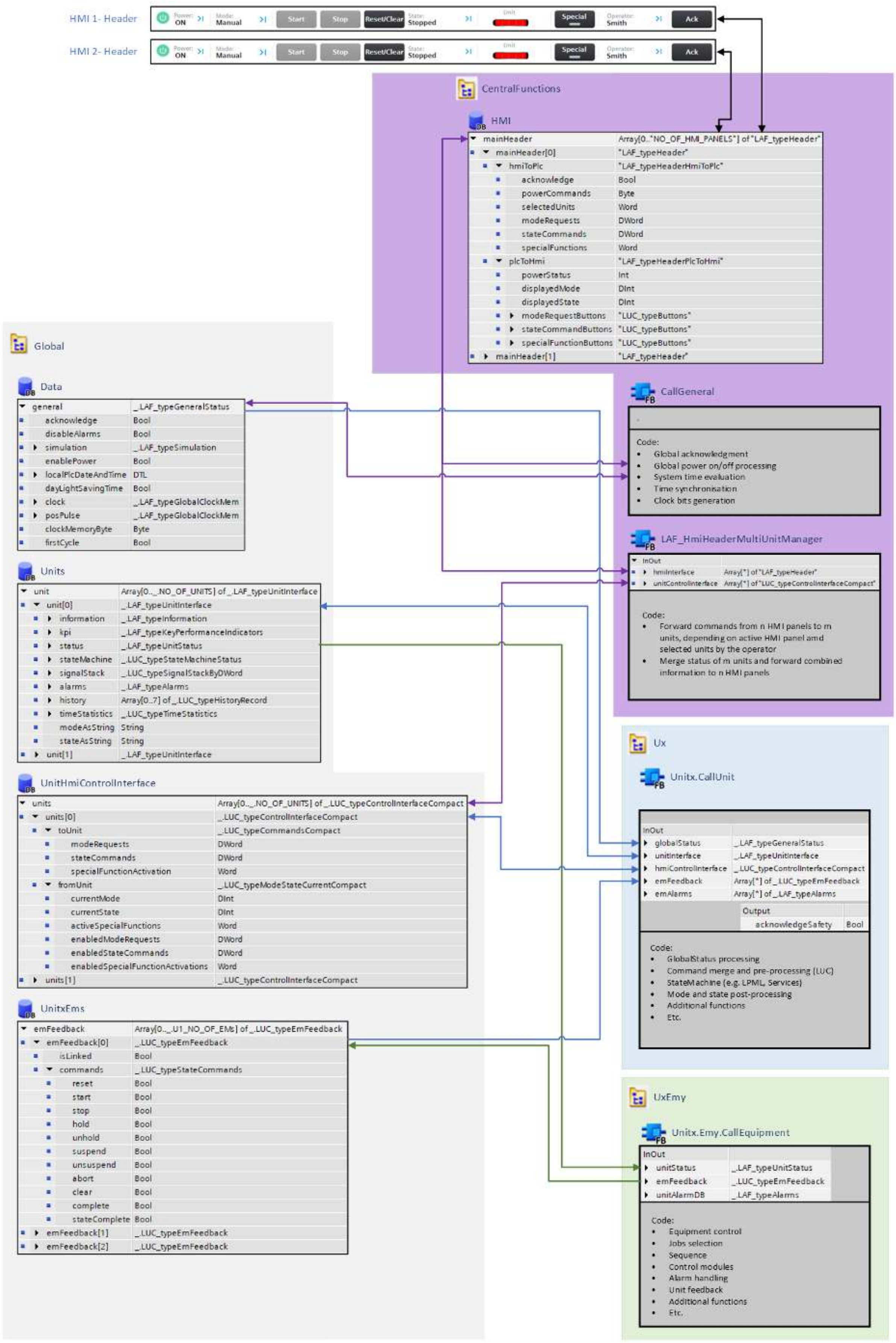
Figure 6-17: Data flow of main commands and status

Description of the above figure:

1. The HMI panel writes its operator commands to the DB “Hmi”.
2. FB “LAF_HmiHeaderMultiUnitManager” takes that information, evaluates the input “selectedUnits” and writes commands in DB “HmiControlInterface” for the selected units.
3. FB “LPMLV2022_UnitModeStateManager” inside FB “CallUnit” gets the commands and set the mode and state. The mode and state are written in DB “Units”.
4. FB “LAF_SelectJob” within FB “CallEquipment” selects the job depending on the programmed start conditions.
5. FB “Sequence” within FB “CallEquipment” reads the selected job and executes the respective logical steps. Inside the steps, control commands for the control modules are set and written on DB “ControlNodes”.
6. The FBs for the control modules (e.g. LBC blocks) read the commands and set HW output respectively.
7. The hardware receives commands an reacts accordingly.
8. Status of the hardware is received by the PLC through HW inputs.
9. The FBs for the control modules (e.g. LBC blocks) evaluate the hardware input and write the status in DB “ControlNodes”.
10. FB “Sequence” reads the status of the control modules and runs through the sequential steps. E.g. if a valve was supposed to open and the hardware reports a “valve-open” back, the sequence transitions to the next step. If the steps for the job are fulfilled, the EM evaluates if all jobs for the mode and state are fulfilled. If all tasks for the current mode and state are fulfilled, the EM writes a “State complete” to DB “UnitsxEms.emFeedback”.
11. FB “LPMLV2022_UnitModeStateManager” reads the feedback of the EMs and transitions to another state, if commanded or “State complete” of all EMs reported. The current mode and state are written to DB “HmiControlInterface”.
12. FB “LAF_HmiHeaderMultiUnitManager” reads the mode and state of the units and merges them to a combined status, depending on the selected units of the individual HMI panels. The summary is written to DB “Hmi”.
13. The HMI panel reads the status in DB “Hmi” provided that specific HMI panel and display the information in the Hmi header. (mode, state, operability of mode and state buttons).

The following figures shows the detailed variables that are transferred between the different blocks. It gives a high level overview of some of the most relevant variables. Like in the previous figures, its easiest to start from the HMI header and follow along the command flow.

- The commands that arrive in DB “Hmi” are distributed to DB “Data” and DB “HmiControlInterface” by FB “CallGeneral” and DB “LAF_HmiHeaderMultiUnitManager” respectively.
- The unit
  - reads the global status from DB “Data”,
  - reads and writes unitInterface in DB “Units”,
  - reads commands from DB “HmiControlInterface”,
  - and writes status back and reads the emFeedback from DB “UnitsxEms”. It additionally reads the safety reease information from DB “DataFromSafety”.
- The equipment module
  - reads the unit status from DB “Units”,
  - and writes its feedback to DB “UnitxEms”.
- The status that is shared by the unit in DB “HmiControlInterface” is evaluated by FB “LAF_HmiHeaderMultiUnitManager” and writte to DB “Hmi.”



HMI

mainHeader	Array[0.."NO_OF_HMI_PANELS"] of "LAF_typeHeader"
mainHeader[0]	"LAF_typeHeader"
hmiToPlc	"LAF_typeHeaderHmiToPlc"
acknowledge	Bool
powerCommands	Byte
selectedUnits	Word
modeRequests	DWord
stateCommands	DWord
specialFunctions	Word
plcToHmi	"LAF_typeHeaderPlcToHmi"
powerStatus	Int
displayedMode	DInt
displayedState	DInt
modeRequestButtons	"LUC_typeButtons"
stateCommandButtons	"LUC_typeButtons"
specialFunctionButtons	"LUC_typeButtons"
mainHeader[1]	"LAF_typeHeader"

Data

general	_LAF_typeGeneralStatus
acknowledge	Bool
disableAlarms	Bool
simulation	_LAF_typeSimulation
enablePower	Bool
localPlcDateAndTime	DTL
dayLightSavingTime	Bool
clock	_LAF_typeGlobalClockMem
posPulse	_LAF_typeGlobalClockMem
clockMemoryByte	Byte
firstCycle	Bool

Units

unit	Array[0.._NO_OF_UNITS] of _LAF_typeUnitInterface
unit[0]	_LAF_typeUnitInterface
information	_LAF_typeInformation
kpi	_LAF_typeKeyPerformanceIndicators
status	_LAF_typeUnitStatus
stateMachine	_LUC_typeStateMachineStatus
signalStack	_LUC_typeSignalStackByDWord
alarms	_LAF_typeAlarms
history	Array[0..7] of _LUC_typeHistoryRecord
timeStatistics	_LUC_typeTimeStatistics
modeAsString	String
stateAsString	String
unit[1]	_LAF_typeUnitInterface

UnitHmiControlInterface

units	Array[0.._NO_OF_UNITS] of _LUC_typeControlInterfaceCompact
units[0]	_LUC_typeControlInterfaceCompact
toUnit	_LUC_typeCommandsCompact
modeRequests	DWord
stateCommands	DWord
specialFunctionActivation	Word
fromUnit	_LUC_typeModeStateCurrentCompact
currentMode	DInt
currentState	DInt
activeSpecialFunctions	Word
enabledModeRequests	DWord
enabledStateCommands	DWord
enabledSpecialFunctionActivations	Word
units[1]	_LUC_typeControlInterfaceCompact

UnitxEms

emFeedback	Array[0.._U1_NO_OF_EMs] of _LUC_typeEmFeedback
emFeedback[0]	_LUC_typeEmFeedback
isLinked	Bool
commands	_LUC_typeStateCommands
reset	Bool
start	Bool
stop	Bool
hold	Bool
unhold	Bool
suspend	Bool
unsuspend	Bool
abort	Bool
clear	Bool
complete	Bool
stateComplete	Bool
emFeedback[1]	_LUC_typeEmFeedback
emFeedback[2]	_LUC_typeEmFeedback

CallGeneral

Code:
- Global acknowledgment
- Global power on/off processing
- System time evaluation
- Time synchronisation
- Clock bits generation

LAF_HmiHeaderMultiUnitManager

InOut	
hmiInterface	Array[*] of "LAF_typeHeader"
unitControlInterface	Array[*] of "LUC_typeControlInterfaceCompact"

Code:
- Forward commands from n HMI panels to m units, depending on active HMI panel amd selected units by the operator
- Merge status of m units and forward combined information to n HMI panels

Unitx.CallUnit

InOut	
globalStatus	_LAF_typeGeneralStatus
unitInterface	_LAF_typeUnitInterface
hmiControlInterface	_LUC_typeControlInterfaceCompact
emFeedback	Array[*] of _LUC_typeEmFeedback
emAlarms	Array[*] of _LAF_typeAlarms

Output	
acknowledgeSafety	Bool

Code:
- GlobalStatus processing
- Command merge and pre-processing (LUC)
- StateMachine (e.g. LPML, Services)
- Mode and state post-processing
- Additional functions
- Etc.

Unitx.Emy.CallEquipment

InOut	
unitStatus	_LAF_typeUnitStatus
emFeedback	_LUC_typeEmFeedback
unitAlarmDB	_LAF_typeAlarms

Code:
- Equipment control
- Jobs selection
- Sequence
- Control modules
- Alarm handling
- Unit feedback
- Additional functions
- Etc.

Figure 6-18: Detailed data flow

6.8. HMI interface

HMI header

The main header of the HMI provides enables the operator to send commands and shows the main status of the machine.

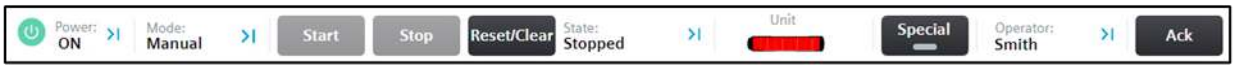


Figure 6-19: Main header of the HMI panel

The header is connected to the PLC through the date type LAF_typeHeader in DB “Hmi” in folder “Central Functions”.

The PLC can handle several HMI panels and several units, whereas only one HMI can be operable at a time. For details, please refer to the chapter HMI Software Architecture.

Two FBs are available to convert the information between the HMI header and the unit. This conversion is necessary, because the operator can select the unit, for which he wants to apply the commands and from which he wants to see the summarizes status:

- FB “LAF_HmiHeaderSingleUnitManager” : Can handle multiple HMI panles and one unit.
- FB “LAF_HmiHeaderMultiUnitManager”: Can handle multiple HMI panels and multiple unit.

The following figure show the data flow if multiple HMI panels and multiple units are configured.

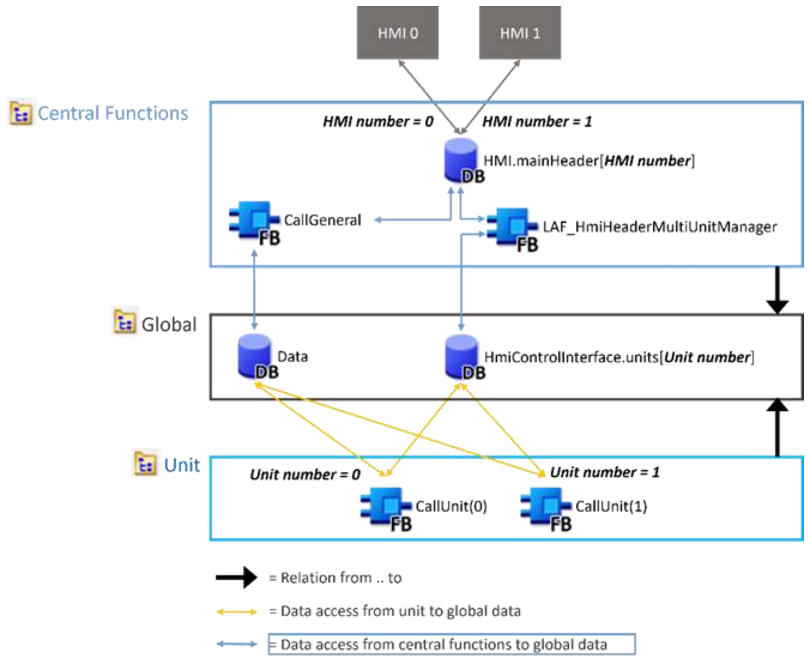

HMI 0
HMI 1
Central Functions
HMI number = 0
HMI number = 1
HMI.mainHeader[HMI number]
DB
CallGeneral
FB
LAF_HmiHeaderMultiUnitManager
FB
Global
Data
DB
HmiControlInterface.units[Unit number]
DB
Unit
Unit number = 0
Unit number = 1
CallUnit(0)
FB
CallUnit(1)
FB
= Relation from .. to
= Data access from unit to global data
= Data access from central functions to global data

Figure 6-20: Data flow, if multiple HMI panels multiple units are configured

The data is processed as follows:

From HMI to PLC: The commands from the HMI header (power, mode requests, state commands, special functions, acknowledge, selected units) are written into DB "Hmi". From there, FB "LAF_HmiHeaderMultiUnitManager" takes the information and distributes it to the selected units by writing the information in DB "HmiControlInterface".

From PLC to HMI: The units publish their information for the HMI in DB "HmiControlInterface". From there, FB "LAF_HmiHeaderMultiUnitManager" takes it, merges the mode, state and special function information to one displayed status and provides it to the HMI panels according to the selected units of the respective HMI panel.

Additionally, FB "CallGeneral" evaluates the power and acknowledge commands and writes that information in DB "Data" for general use.

From "Global" to "Unit": FB "CallUnit" reads the global general status from DB "Date" and the commands from DB "HmiControlInterface". It processes that information and writes relevant information in DB "Units" in and status back to DB "HmiControlInterface".

The following figure show the data flow, if only one HMI panel and one unit is configured.

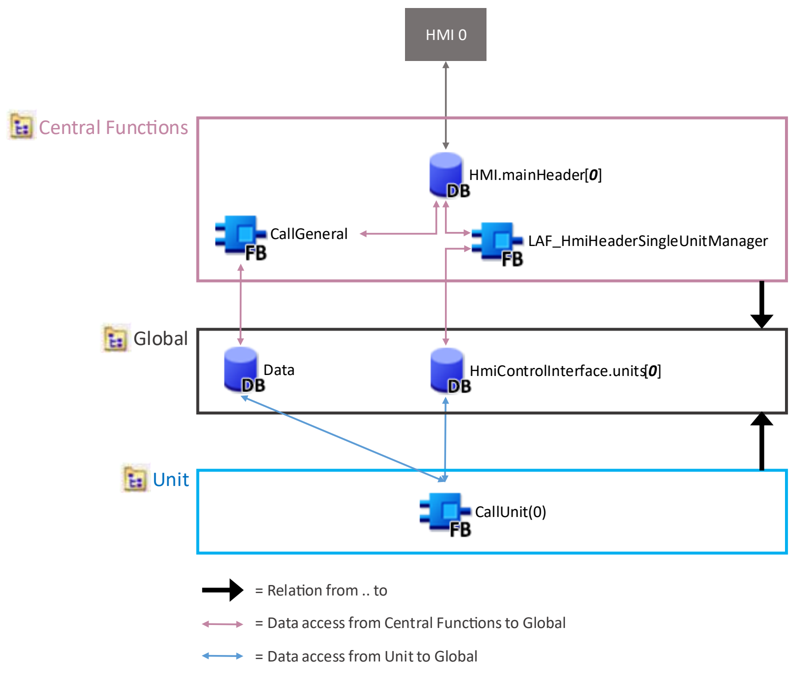
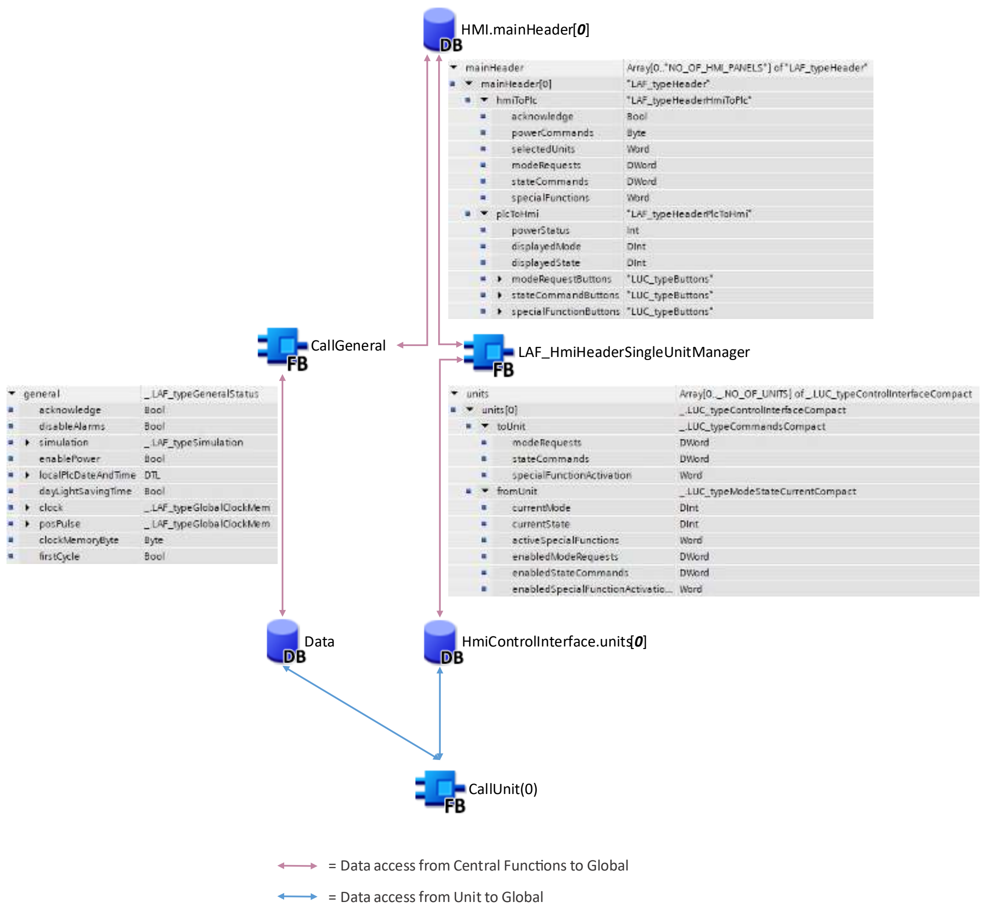

HMI 0
Central Functions
HMI.mainHeader[0]
DB
CallGeneral
FB
LAF_HmiHeaderSingleUnitManager
FB
Global
Data
DB
HmiControlInterface.units[0]
DB
Unit
CallUnit(0)
FB
= Relation from .. to
= Data access from Central Functions to Global
= Data access from Unit to Global

Figure 6-21: Data flow, if one HMI panel and one unit are configured

The following figure shows the detailed data that is transferred between the data blocks.


HMI.mainHeader[0]
DB
CallGeneral
FB
LAF_HmiHeaderSingleUnitManager
FB
Data
DB
HmiControlInterface.units[0]
DB
CallUnit(0)
FB
= Data access from Central Functions to Global
= Data access from Unit to Global

mainHeader	Array[0.."NO_OF_HMI_PANELS"] of "LAF_typeHeader"
mainHeader[0]	"LAF_typeHeader"
hmiToPlc	"LAF_typeHeaderHmiToPlc"
acknowledge	Bool
powerCommands	Byte
selectedUnits	Word
modeRequests	DWord
stateCommands	DWord
specialFunctions	Word
plcToHmi	"LAF_typeHeaderPlcToHmi"
powerStatus	Int
displayedMode	DInt
displayedState	DInt
modeRequestButtons	"LUC_typeButtons"
stateCommandButtons	"LUC_typeButtons"
specialFunctionButtons	"LUC_typeButtons"

general	_LAF_typeGeneralStatus
acknowledge	Bool
disableAlarms	Bool
simulation	_LAF_typeSimulation
enablePower	Bool
localPlcDateAndTime	DTL
dayLightSavingTime	Bool
clock	_LAF_typeGlobalClockMem
posPulse	_LAF_typeGlobalClockMem
clockMemoryByte	Byte
firstCycle	Bool

units	Array[0.._NO_OF_UNITS] of _LUC_typeControlInterfaceCompact
units[0]	_LUC_typeControlInterfaceCompact
toUnit	_LUC_typeCommandsCompact
modeRequests	DWord
stateCommands	DWord
specialFunctionActivation	Word
fromUnit	_LUC_typeModeStateCurrentCompact
currentMode	DInt
currentState	DInt
activeSpecialFunctions	Word
enabledModeRequests	DWord
enabledStateCommands	DWord
enabledSpecialFunctionActivatio...	Word

Figure 6-22: Detailed data transferred between the HMI header and the Unit

Further HMI interfaces

Besides the HMI header, further connections between the PLC and the HMI exist:

- The units share information about their detailed status including histotry and statistics through DB "Units"
- Each EM contains a DB "HmiInterface" to share status of the em, the jobs, seqeunces and control modules.
- Various data blocks from central function share information, e.g. system settings and diagnostics.

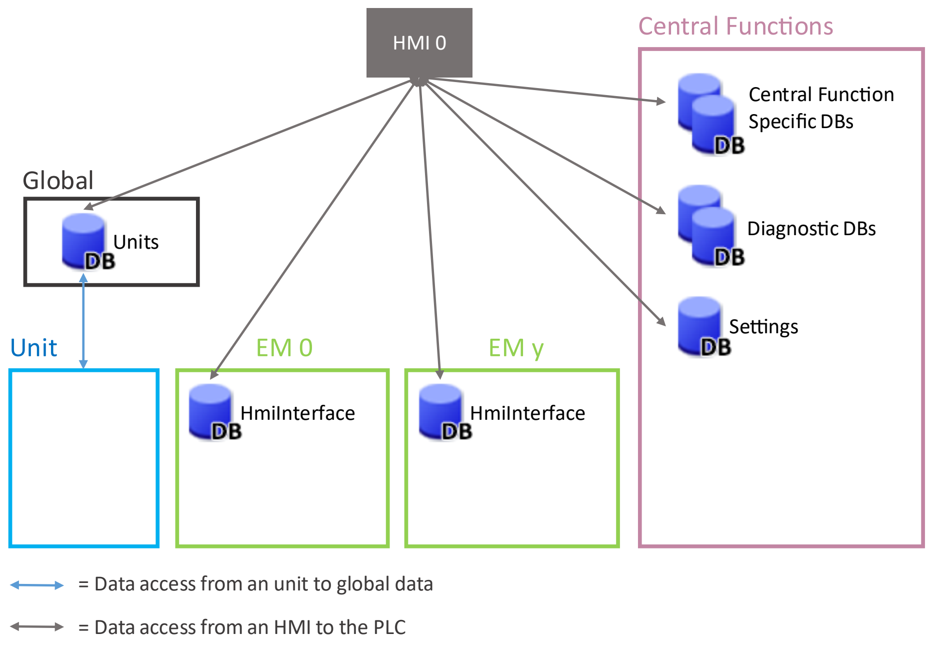
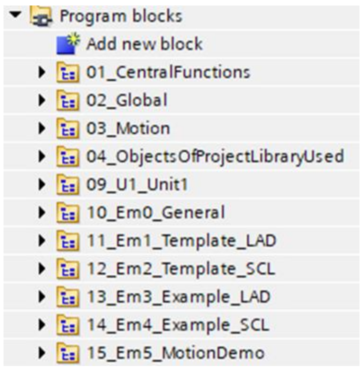

HMI 0
Central Functions
Central Function Specific DBs
DB
Diagnostic DBs
DB
Settings
DB
Global
Units
DB
Unit
EM 0
HmiInterface
DB
EM y
HmiInterface
DB
= Data access from an unit to global data
= Data access from an HMI to the PLC

Figure 6-23: Further HMI – PLC interfaces

6.9. TIA Portal implementation

The AFBasic contains two sections:

1. General functionality : Central Function, Global, Motion, ObjectsOfProjectLibraryUsed.
2. Unit 1 : ISA-88 unit and EM templates and examples programmed in LAD and SCL.



Program blocks
Add new block
01_CentralFunctions
02_Global
03_Motion
04_ObjectsOfProjectLibraryUsed
09_U1_Unit1
10_Em0_General
11_Em1_Template_LAD
12_Em2_Template_SCL
13_Em3_Example_LAD
14_Em4_Example_SCL
15_Em5_MotionDemo

Figure 6-24: Screenshot of PLC program structure in the TIA Portal project

Equipment module types

The AFBasic distinguishes between two different types of equipment modules:

- **Em_Template**: Contain only the mandatory blocks and calls. They are the base to start own EMs. Template for each programming languages are available.

- **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.

Programming languages

The AFBasic 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.

Folder	Description
U1_Unit1	Programmed in LAD
U1Em0_General	Programmed in LAD
U1Em1_Template_LAD	General logic programmed in LAD, sequencer programmed in LAD
U1Em2_Template_SCL	General logic programmed in LAD, sequencer programmed in SCL
U1Em3_Example_LAD	General logic programmed in LAD, sequencer programmed in LAD
U1Em4_Example_SCL	General logic programmed in LAD, sequencer programmed in SCL
U1Em5_MotionDemo	Example of how to split motion calls in standard cycle and motion cycle

Table 6-4: Overview of used programming languages

6.10. Software design principles

6.10.1. 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 6-5: 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 6-6: 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.

Realization in the AFBasic

The user can program using the AFBasic libraries following the one or the other schema. The AFBasic project was programmed according to schema 2 to keep the AFBasic 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.

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

The following figures the FB “CallEquipment” is connected to data from another folder. 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.

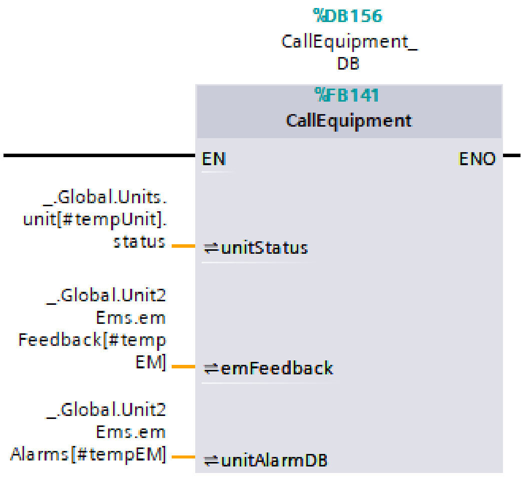


Figure 6-25: Call of FB “CallEquipment” in an OB “Main”

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 folder of 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. This direct variable link allows the user to jump to the program block directly via tooltip “Go to definition”.

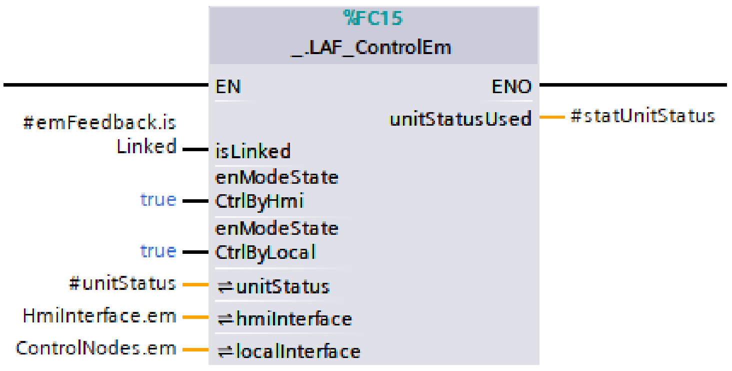


Figure 6-26: Call of FC “LAF_ControlEm” in a FB “CallEquipment”

6.10.2. 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 6-7: 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 6-8: 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$_{Basic}$

The AF$_{Basic}$ can be programmed in both architectures. However, the AF$_{Basic}$ 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 AFBasic project delivers a modular, customizable HMI Template for WinCC Unified, specifically designed to reduce programming effort and streamline development. This chapter introduces the HMI layout and highlights reusable elements such as the HMI header, intuitive page navigation, and ready-to-use plug-and-play application examples.

7.1. Screen Layout

A screen layout defines the arrangement and design of visual elements within a graphical user interface (GUI), determining how different process screens are positioned and structured within the HMI. Effective screen layouts significantly enhance usability, ensuring operators can quickly access and interact with relevant information and controls.

The starting screen layout is designed to offer an intuitive and user-friendly experience, considering visual hierarchy, logical grouping of related elements, optimal space utilization, and a consistent, aesthetically pleasing design.

Use case

In manufacturing environments, operators require intuitive and clear interfaces to efficiently monitor process parameters, control equipment, and quickly respond to alarms. An effective screen layout is essential to meet these operational demands.

The AF screen layout serves as the HMI's starting screen and is structured into six distinct sections, ensuring organized information display and easy operator interaction:

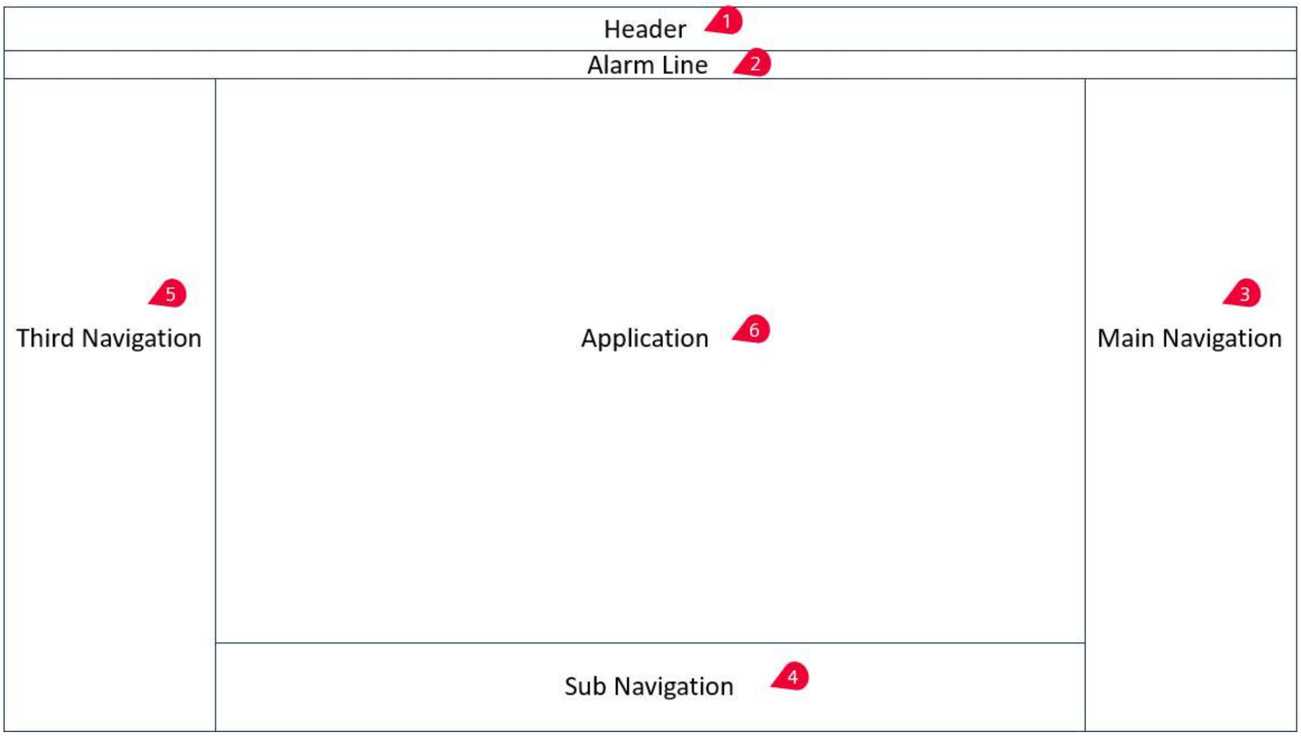
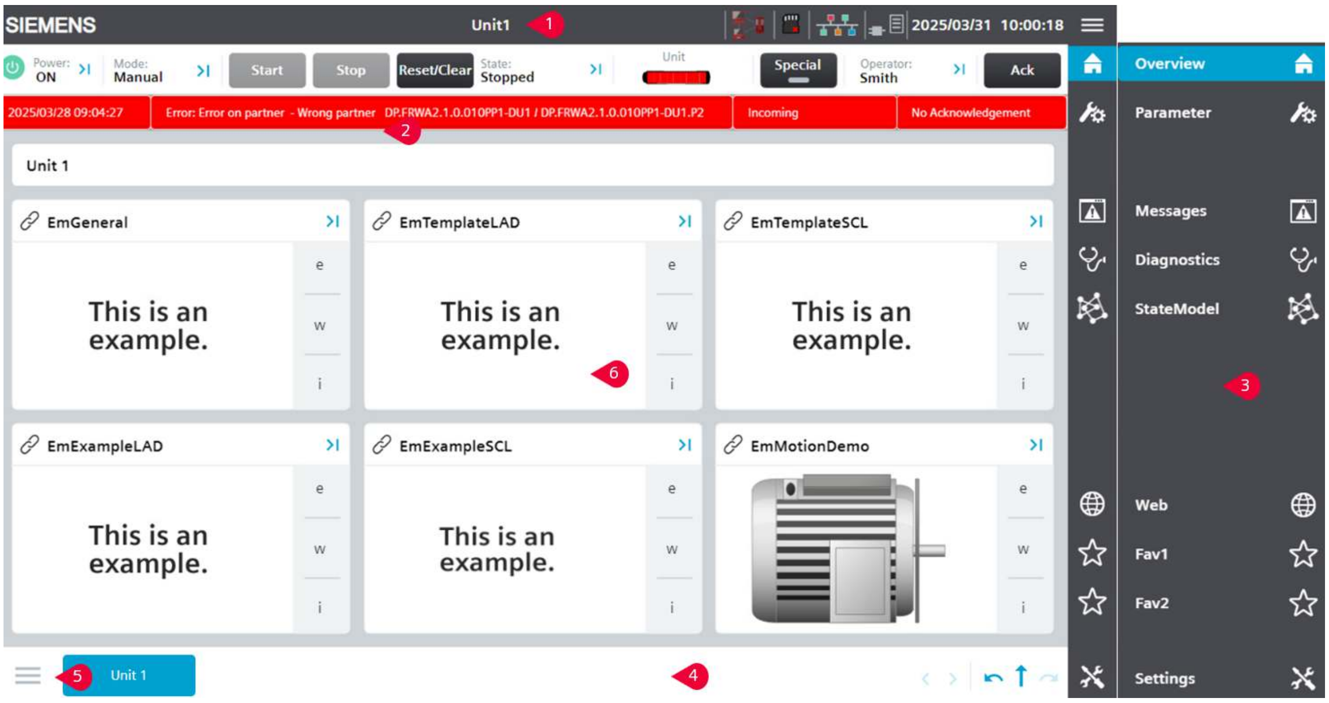


Figure 7-1: Screen layout

Section	Name	Description
1	Header	Permanently visible section containing key information: PLC name, current screen name, memory-card status, PN-connection, date/time, and PLC checksum.
2	AlarmLine	Displays the most recent active alarm, enabling operators to quickly identify and respond to issues.
3	MainNavigation	Provides access to 11 predefined navigation areas: Overview, State Model, Manual, Messages, Shift Model, Production Data, Energy, CM Info, Diagnostics, Web, Settings, and User Operation. (*For details, see Section "Navigation," Chapter 7.3*)
4	SubNavigation	An additional customizable navigation layer positioned at the bottom of the screen for quick access to specific functions.
5	ThirdNavigation	A customizable navigation layer located on the left side, offering supplementary options for enhanced usability.
6	Application	Central screen area displaying application-specific elements such as menus, I/Os, graphics, and control tags relevant to the process.

Table 7-1: Description screen layout



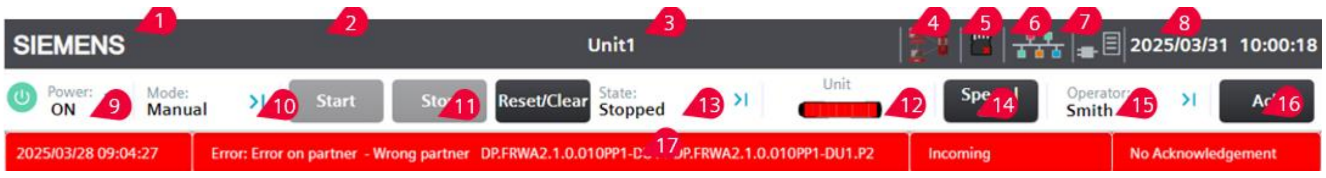
2025/03/28 09:04:27 | Error: Error on partner - Wrong partner DP.FRWA2.1.0.010PP1-DU1 / DP.FRWA2.1.0.010PP1-DU1.P2 | Incoming | No Acknowledgement

Figure 7-2: Screen layout of AF

Sections four, five, and six are implemented as dedicated WinCC Unified pages, loaded dynamically when the starting screen is opened. The remaining sections (one, two, and three) are integrated directly into the starting screen layout and programmed as part of its core structure.

7.2. Header

The header prominently displays essential information such as power status, current state, and operating mode of the selected units. Additionally, it provides quick access for operators to control and interact directly with these units.

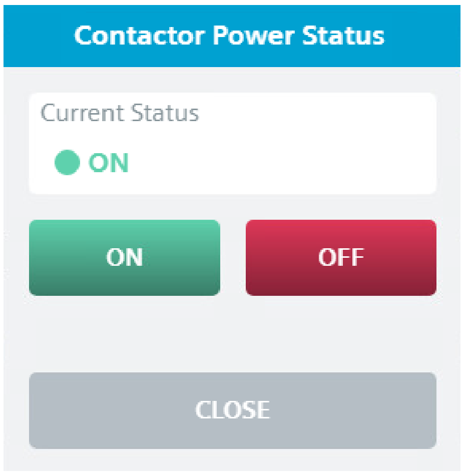
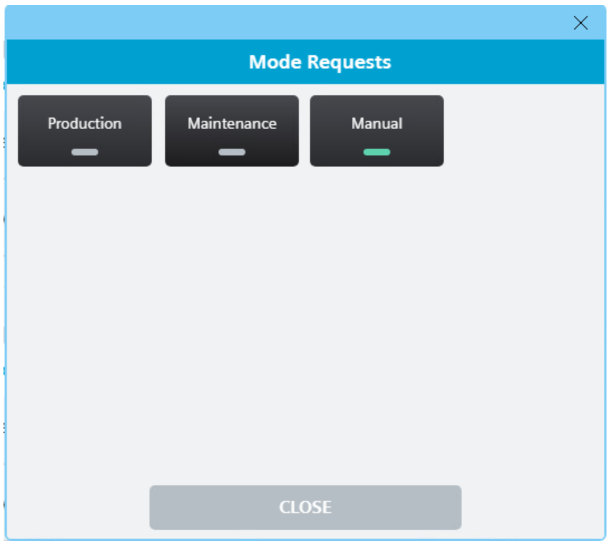


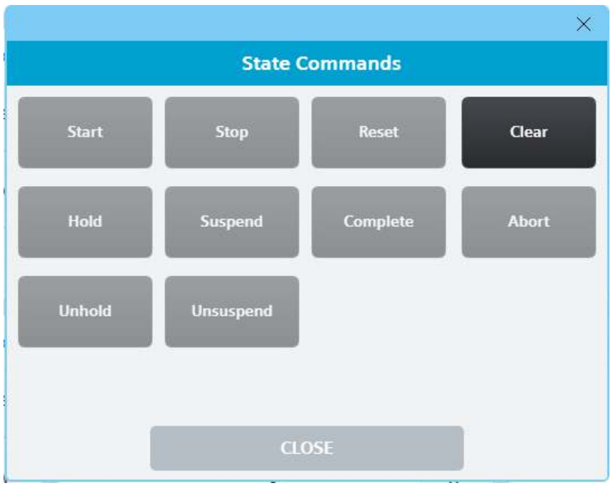
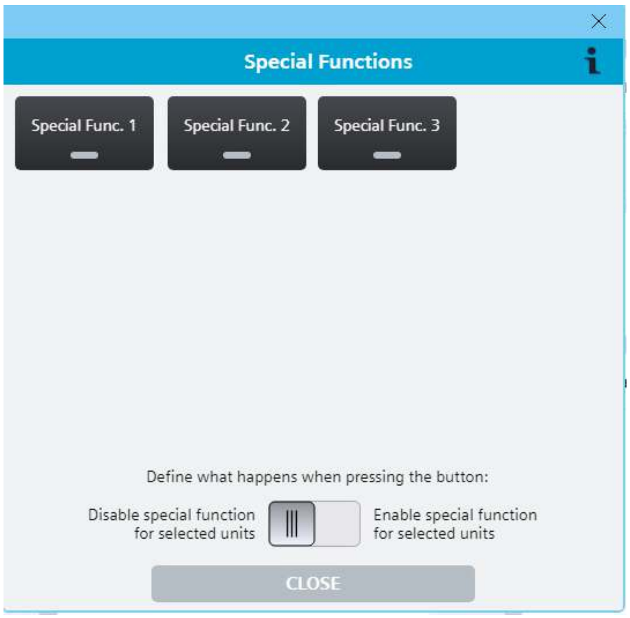
2025/03/28 09:04:27 | Error: Error on partner - Wrong partner DP.FRWA2.1.0.010PP1-D... P.FRWA2.1.0.010PP1-DU1.P2 | Incoming | No Acknowledgement

Figure 7-3: HMI layout

The following visual elements are in the "header":

Element	Name	Description
1	Company Name	Placeholder for inserting a dedicated company name.
2	PLC Name	Displays the PLC name derived from the PLC's Identification & Maintenance/Plant designation data.
3	Current Screen Name	Indicates the current screen name via the "Show Title" script.
4	HMI-PLC Connection Status / PLC Operation Mode	Visualizes PLC and HMI connection status, indicating conditions such as "PLC Stop" or "No Connection between PLC and HMI."
5	SIMATIC Memory Card Status	Provides detailed status of the SIMATIC memory card, including total size, used memory space, and lifetime utilization metrics (percentage of service life consumed).
6	PROFINET Diagnostic	Summarizes diagnostic information of connected PROFINET devices, enabling quick identification of network or device issues.
7	CRC Check	Checksums for standard PLC blocks, safety PLC blocks, and PLC text lists are monitored. Any detected changes from previous checksums trigger dynamic SVG visualization.
8	Current Date & Time	Displays current date and time settings. (Further details in the chapter "Settings.")

Element	Name	Description
9	Power On/Off	Allows centralized activation or deactivation of actuators via main power circuit breaker. Commands toggle a PLC bit (true/false), customizable by PLC programming.  Contactor Power Status Current Status ON ON OFF CLOSE Figure 7-4: Power On/Off
10	Mode and Mode Commands	Indicates the current operational mode of selected units. If multiple units with different modes are selected, "Various" is displayed. Modes can be changed via pop-up. Green indicators show active modes among selected units. Button properties (visibility, activation, control permissions) are managed by FB "HmiHeaderManager" in the PLC.  Mode Requests Production Maintenance Manual CLOSE Figure 7-5: Mode Commands

Element	Name	Description
11	State Command	Provides pop-up access for issuing state commands to selected units. Command availability depends on current mode, state, and state machine configuration, managed by FB "HmiHeaderManager" in the PLC.  State Commands Start Stop Reset Clear Hold Suspend Complete Abort Unhold Unsuspend CLOSE Figure 7-6: State Commands
12	Select Unit	Shows the current mode and state of units, allowing selection of individual or all units. Actions triggered via the header apply exclusively to selected units.
13	State	Displays the current operational state of selected units. If multiple units with varying states are selected, "Various" is displayed.
14	Special Functions	Enables additional customizable machine operations via operator-triggered requests. Button labels can be modified in WinCC text lists; visibility and control permissions configured via the "HMI" DB in the PLC.  Special Functions Special Func. 1 Special Func. 2 Special Func. 3 Define what happens when pressing the button: Disable special function for selected units Enable special function for selected units CLOSE Figure 7-7: Special Functions

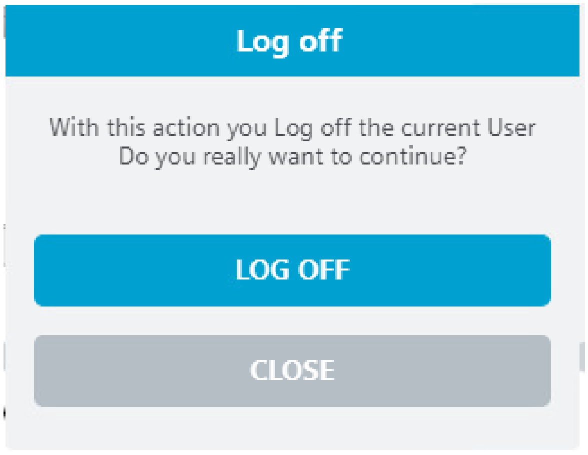
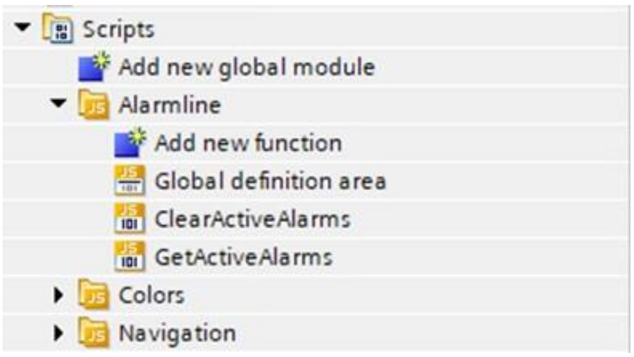
Element	Name	Description
15	Current Logged-In User	Displays currently logged-in user. Clicking the user icon opens a login/logout dialog depending on the current authentication status.  Log off With this action you Log off the current User Do you really want to continue? Figure 7-8: Log off.
16	Acknowledge	Button to acknowledges all pending alarms.
17	Alarm Line	Shows the most recent alarm event. Selecting the Alarm Line opens the Alarm Screen.  Scripts Add new global module Alarmline Add new function Global definition area ClearActiveAlarms GetActiveAlarms Colors Navigation Figure 7-9: Global Scripts – Alarms.

Table 7-2: Elements in the header

7.3. Navigation

Monitoring parameters, controlling actuators, and accessing machine data represent just some of the essential functions that operators perform through the HMI. To ensure user-friendly and efficient operation, a standardized navigation concept with clearly defined navigation levels has been developed, providing consistency across all screens.

Navigation within the HMI is structured into three distinct levels (see Figure 7-1).

- Main Navigation (AF screen section 3) → "00_ScreenLayout/ScreenLayout"
- Sub Navigation (AF screen section 4) → "00_ScreenLayout/SubNavigation"
- Third Navigation (AF screen section 5) → "00_ScreenLayout/ThirdNavigation"

The Main Navigation provides direct access to primary functional areas. If additional categorization or further details are necessary, the Sub Navigation layer is used. For scenarios requiring even deeper subdivisions, the Third Navigation layer can be implemented. While the main navigation is always visible, visibility for sub and third navigation layers is dynamically adjustable according to operator requirements.

7.3.1. Main navigation

The main navigation consists of two parts: a permanently visible icon bar, and a text-based menu accessible via the burger menu. Both navigation views provide direct access to the essential HMI features included by default in the AF HMI project.

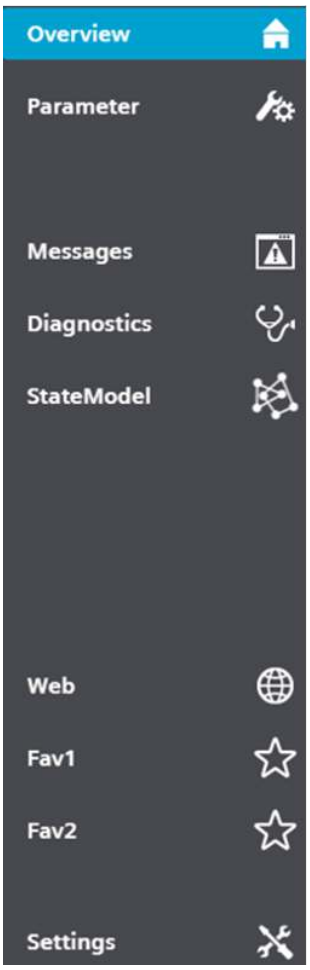


Figure 7-10: Main navigation

Technical description

The AF main navigation is pre-configured with default names and graphics corresponding to essential HMI features. Both the displayed texts and graphics, as well as the associated main screens shown in section 6 ("swContent" screen window), can be customized within Engineering System (ES). Each navigation element (icons and text boxes) triggers an event that loads the corresponding screen into section 6 when selected by the operator.

To modify or reassign the screen linked to a particular navigation item, simply adjust the corresponding event in the ES. The main navigation elements are defined in the screen 00_ScreenLayout/ScreenLayout: navigation icons reside on Layer 7, and navigation texts are placed on Layer 6 (see image below).

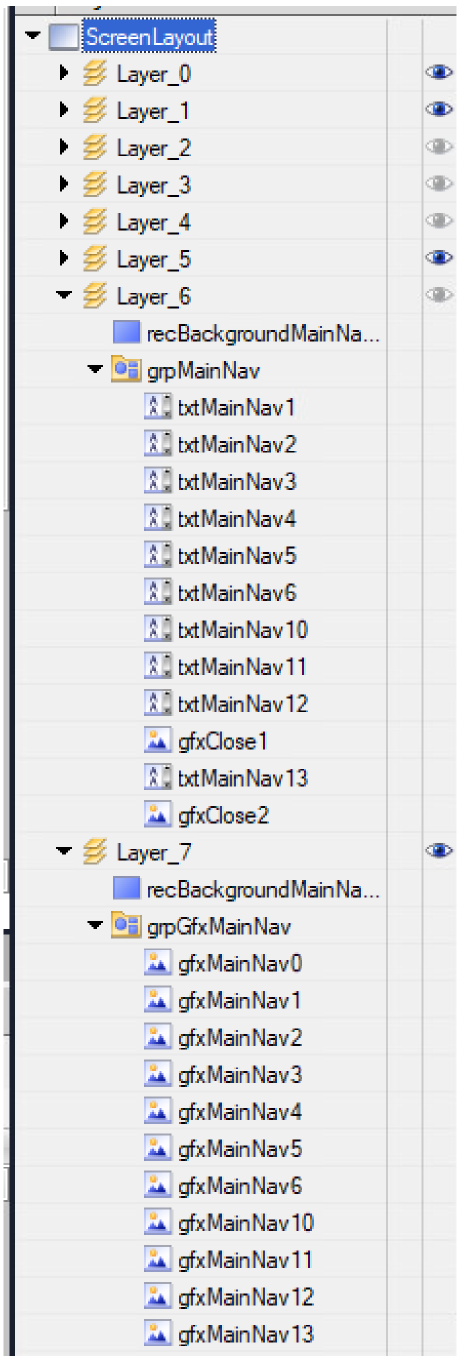

```
ScreenLayout
  Layer_0
  Layer_1
  Layer_2
  Layer_3
  Layer_4
  Layer_5
  Layer_6
    recBackgroundMainNa...
    grpMainNav
      txtMainNav1
      txtMainNav2
      txtMainNav3
      txtMainNav4
      txtMainNav5
      txtMainNav6
      txtMainNav10
      txtMainNav11
      txtMainNav12
      gfxClose1
      txtMainNav13
      gfxClose2
  Layer_7
    recBackgroundMainNa...
    grpGfxMainNav
      gfxMainNav0
      gfxMainNav1
      gfxMainNav2
      gfxMainNav3
      gfxMainNav4
      gfxMainNav5
      gfxMainNav6
      gfxMainNav10
      gfxMainNav11
      gfxMainNav12
      gfxMainNav13
```

Figure 7-11: Main Navigation – Screenlayout Layer 6 & Layer 7

7.3.2. Sub navigation

The sub-navigation extends the main navigation by providing additional navigation options within each central function. It enables operators to navigate efficiently through multiple screens associated with the same main navigation item. Each central function can have its dedicated sub-navigation configured accordingly.

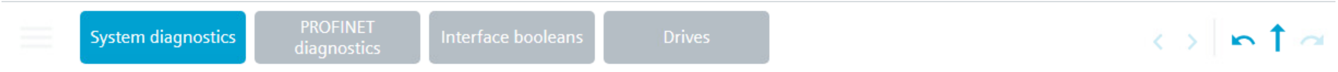


Figure 7-12: Sub navigation – Diagnostics example

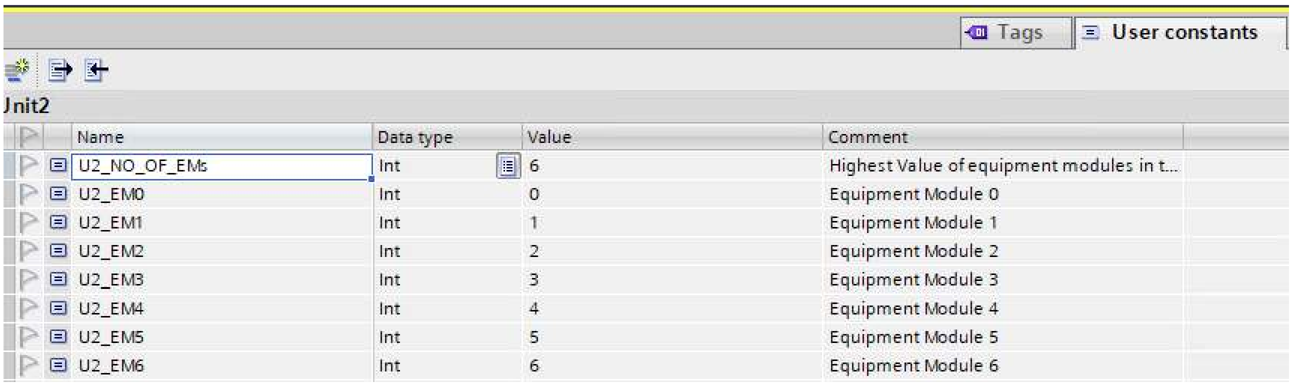
Technical description

The sub-navigation functionality addresses two distinct scenarios: Central Functions and Units/Ems:

- Central Functions: The sub-navigation items for central functions are pre-configured, with defined values and events that correspond directly to each main navigation category.
- Units and EMs: Sub-navigation entries for Units and EMs are dynamically generated during runtime (RT) according to specific project configurations, as the number of Units and EMs may vary between projects. Consequently, AF provides a single dynamic sub-navigation template for Units and EMs.

This dynamic behavior relies on PLC constants and HMI Text Lists, allowing the sub-navigation to update values automatically for each Unit and associated EM. Specifically, the "SubEm" navigation screen dynamically reflects the configured number of EMs per Unit through dedicated PLC constants linked to Unit Instance DBs (CallUnit_DB) and associated HMI tags (UnitXEmNum, such as Unit1EmNum, Unit2EmNum, etc.). These tags reside within their respective HMI Tag tables: U1, U2, U3, and U4.

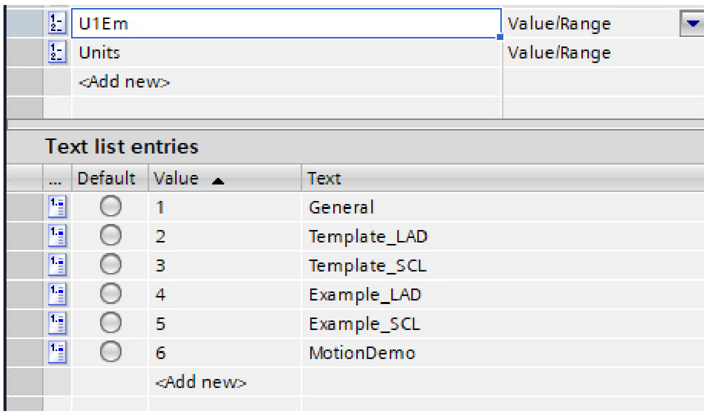
The configuration for these dynamic values takes place in the PLC constants, as illustrated in the following figure:



Name	Data type	Value	Comment
U2_NO_OF_EMs	Int	6	Highest Value of equipment modules in t...
U2_EM0	Int	0	Equipment Module 0
U2_EM1	Int	1	Equipment Module 1
U2_EM2	Int	2	Equipment Module 2
U2_EM3	Int	3	Equipment Module 3
U2_EM4	Int	4	Equipment Module 4
U2_EM5	Int	5	Equipment Module 5
U2_EM6	Int	6	Equipment Module 6

Figure 7-13 PLC Constants – Unit 2

The displayed names in the sub-navigation are dynamically configured using specific HMI Text Lists (U1Em, U2Em, U3Em, and U4Em). Similarly, the displayed values in the SubUnit navigation screen are dynamically managed through another dedicated HMI Text List (Units).



U1Em	Value/Range
Units	Value/Range
<Add new>	

Text list entries

Default	Value	Text
○	1	General
○	2	Template_LAD
○	3	Template_SCL
○	4	Example_LAD
○	5	Example_SCL
○	6	MotionDemo
	<Add new>	

Figure 7-14 HMI Textlists to dynamize SubUnit and SubEMs Screens

7.3.3. Third navigation

The third navigation level, located on the left side of the AF$_{Basic}$ screen layout, provides selection of application-specific content displayed in Section 6 ("Content Screen"). It follows the same conceptual approach as the sub-navigation, allowing dynamic configuration. Additionally, both sub-navigation and third navigation are synchronized when navigating among Units and EMs, ensuring consistent screen views regardless of the selected navigation method—whether via sub-navigation, third navigation, or graphical navigation within Section 6.

The third navigation can contain multiple navigation levels for specific screens, such as Home and Drives. Other screens typically utilize a single-level navigation structure.

These levels are the following:

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

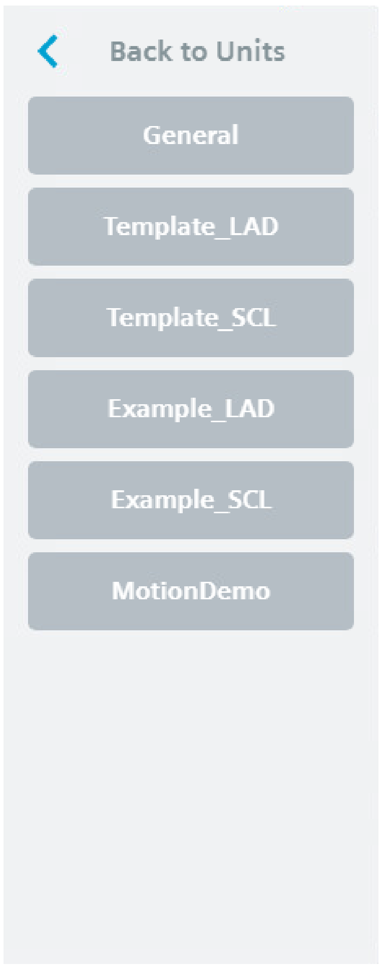


Figure 7-15: Third navigation

7.3.4. Unit navigation

The Unit Navigation provides operators with multiple methods to visualize and manage the various units within the system. Depending on user preferences, units can be navigated using:

- Application Section: Graphical navigation offering a comprehensive visual overview of all available units.
- Sub Navigation: Standardized navigation to easily switch between units.
- Third Navigation: Additional standardized navigation option for unit selection.

Application section – Unit Navigation

Within the Application section, operators can directly select a specific unit to view its associated EMs. Navigation across EMs and CMs is also accessible through this graphical interface, as illustrated in the following example images:

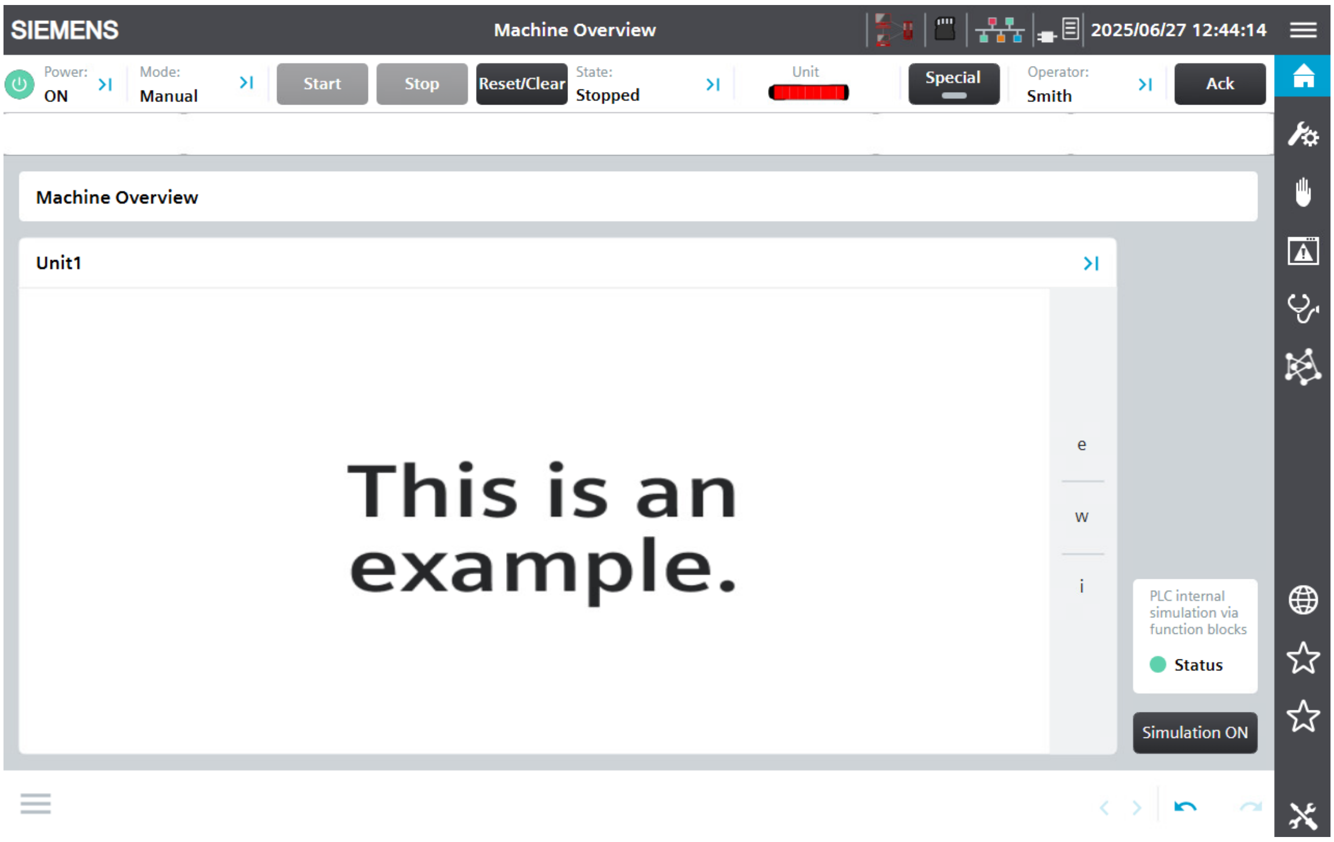
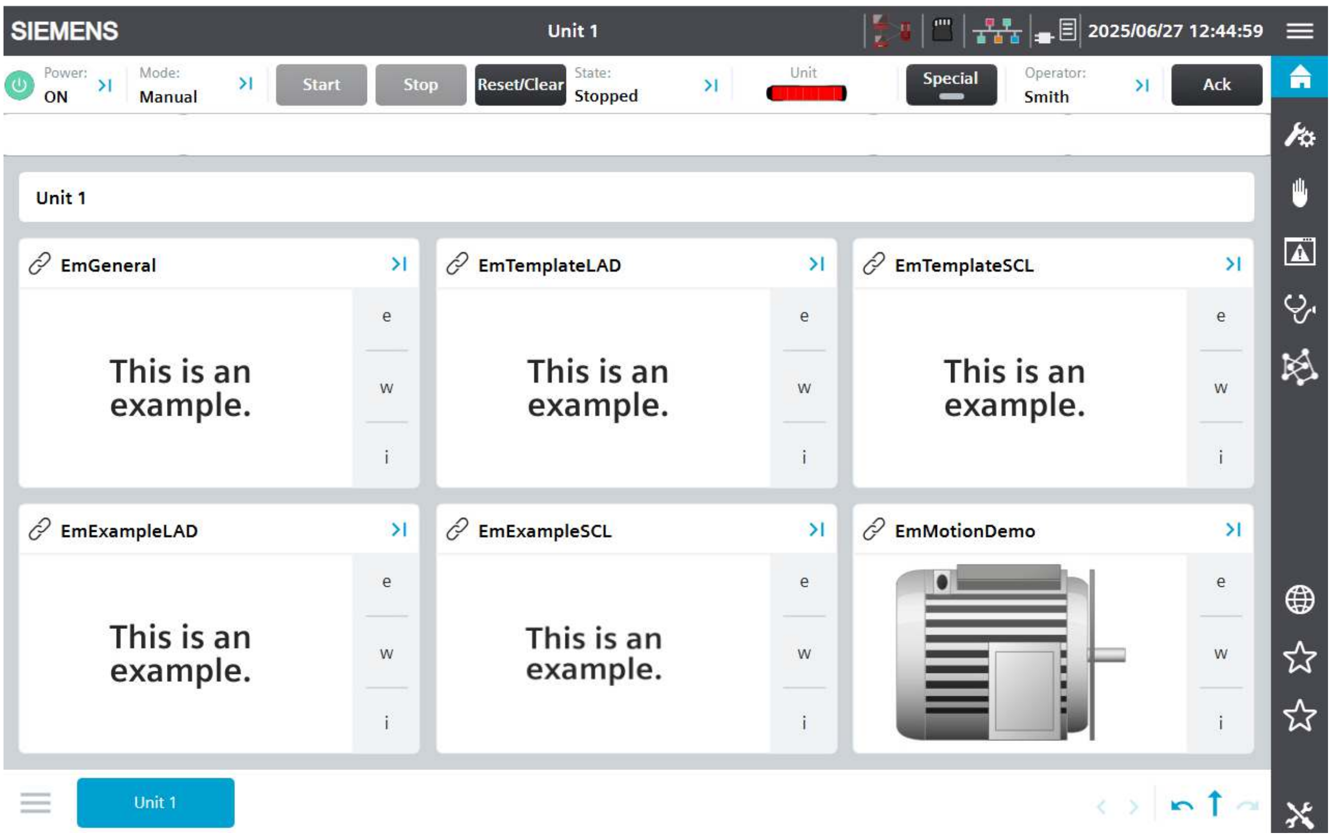
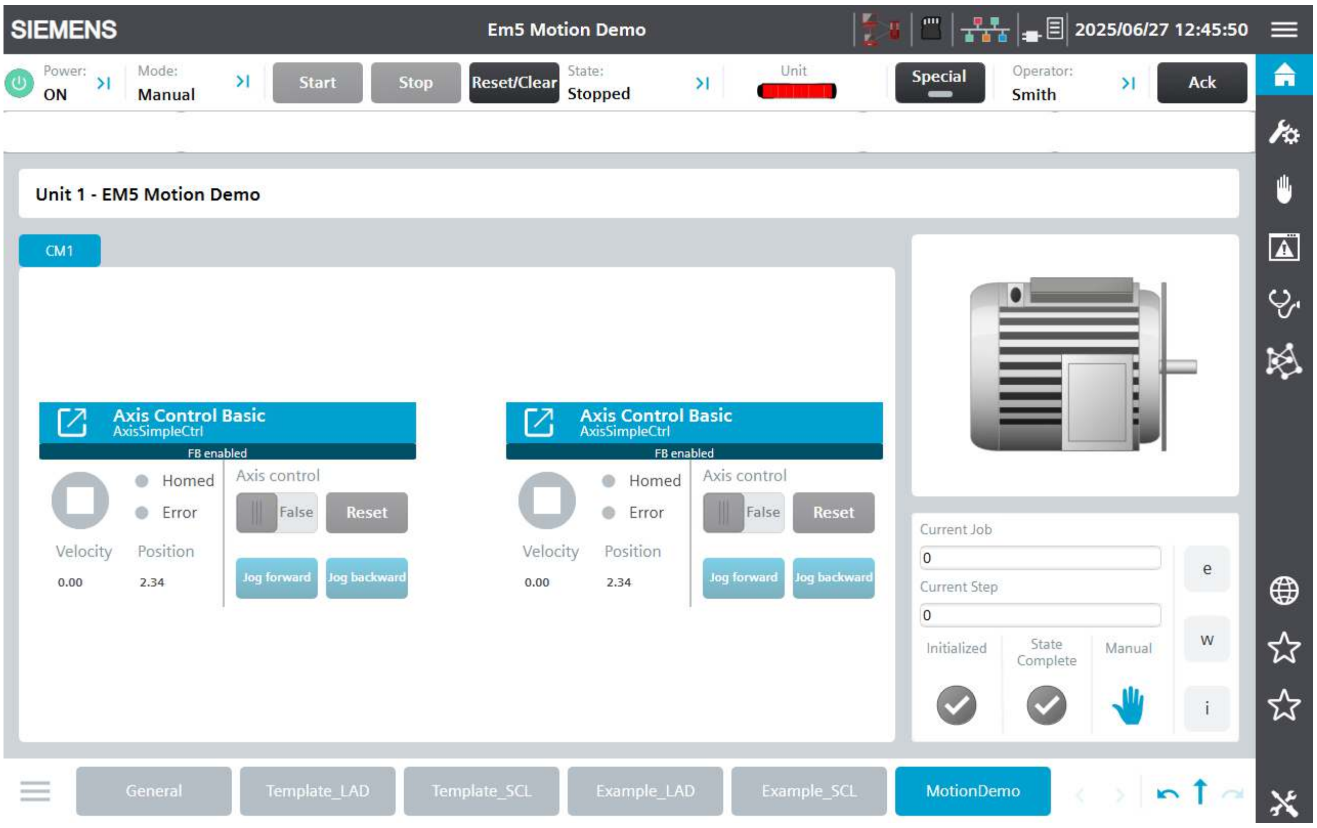


Figure 7-16: Unit Navigation – application section



Power: ON
Mode: Manual
State: Stopped
Operator: Smith

Figure 7-17: EM – application section



Power: ON
Mode: Manual
State: Stopped
Operator: Smith

Axis control: False
Velocity: 0.00
Position: 2.34

Current Job: 0
Current Step: 0

Figure 7-18: Control Module – application section

SubNavigation & ThirdNavigation – Unit Navigation

Sub Navigation and Third Navigation both offer a standardized method for navigating through Units, EMs, and CMs. Operators can easily switch between different units by selecting the appropriate navigation buttons. The navigation structure for Units, EMs, and CMs follows a consistent design concept across both levels.

Button labels displayed during runtime (RT) are dynamically generated using predefined HMI Text Lists (Units, U1Em, U2Em, U3Em, and U4Em) and PLC constants. This approach ensures that navigation labels and associated values adapt dynamically to the current configuration.

The following images illustrate how these navigational elements appear on the HMI panel:

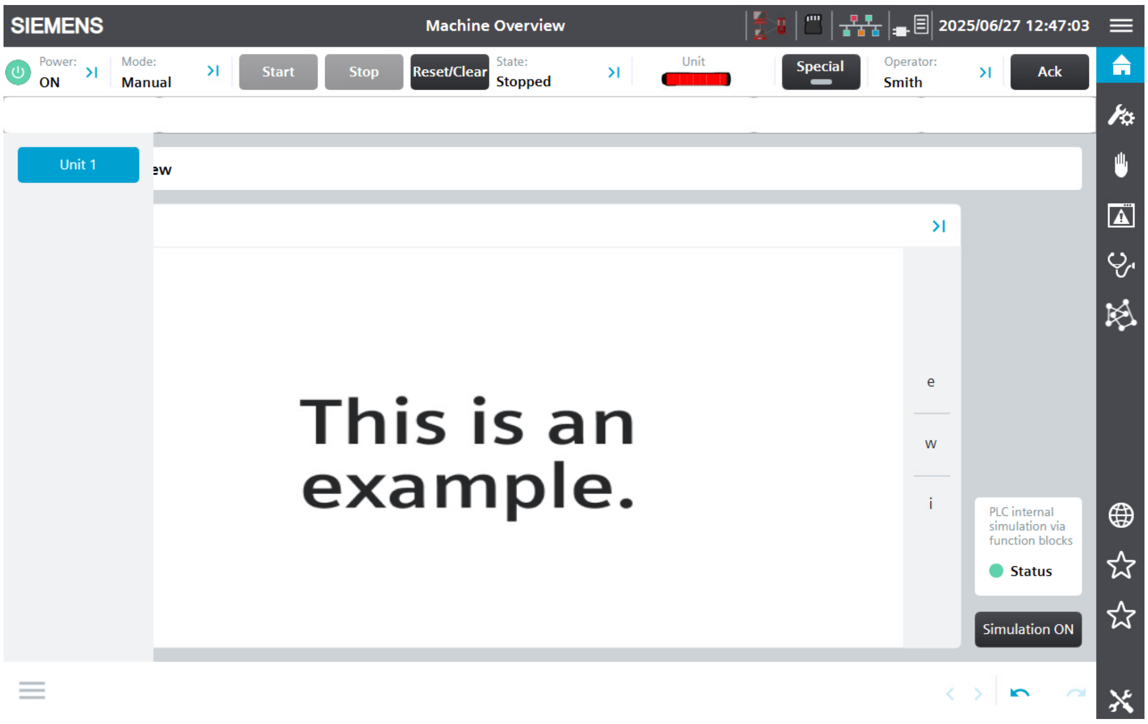


Figure 7-19: Unit SubNavigation and Unit ThirdNavigation

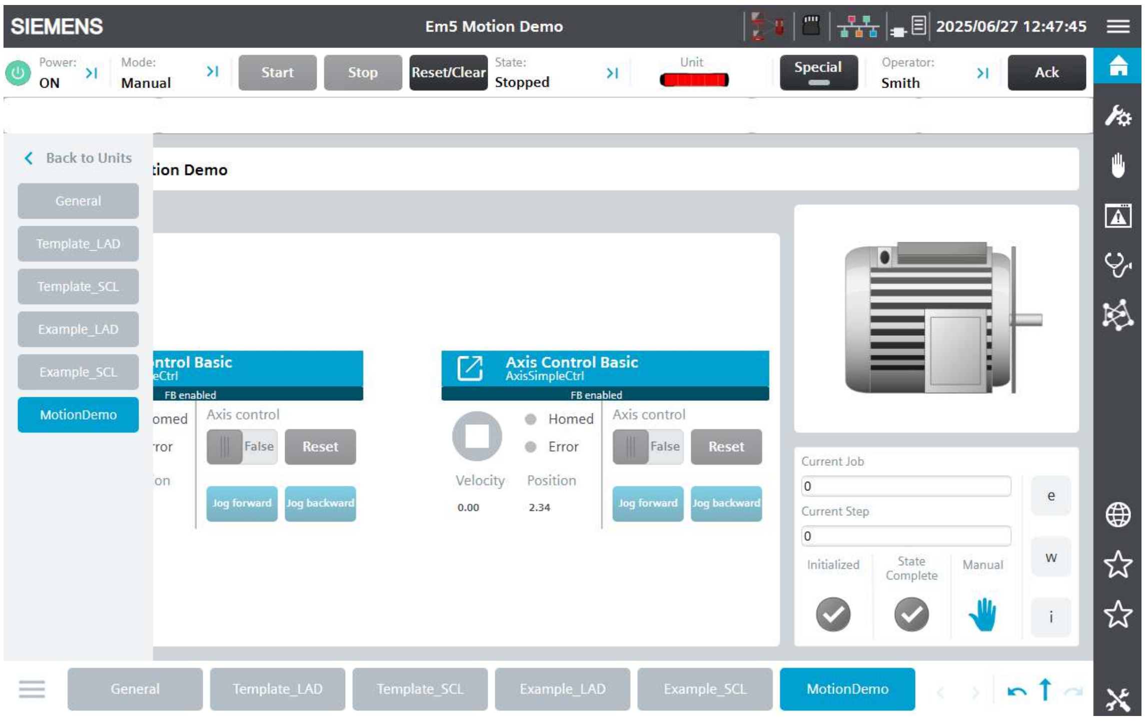


Figure 7-20: EM SubNavigation and EM ThirdNavigation

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: Navigates back to the previously displayed screen.
- Up: Moves up one navigation level, applicable specifically within Unit, EM, and CM screens (e.g., from CM to EM, or from EM to Unit).
- Next: Navigates forward to the next screen in the previously visited sequence. (Note: the Previous command must be used at least once beforehand).

Figure 7-21 HMI General Navigation Commands – Previous, Up and Next

7.3.6. HMI project tree navigation

The project tree navigation reflects the previously described navigation types, maintaining a consistent organizational structure across screens and HMI tags. The following image illustrates this structured approach:

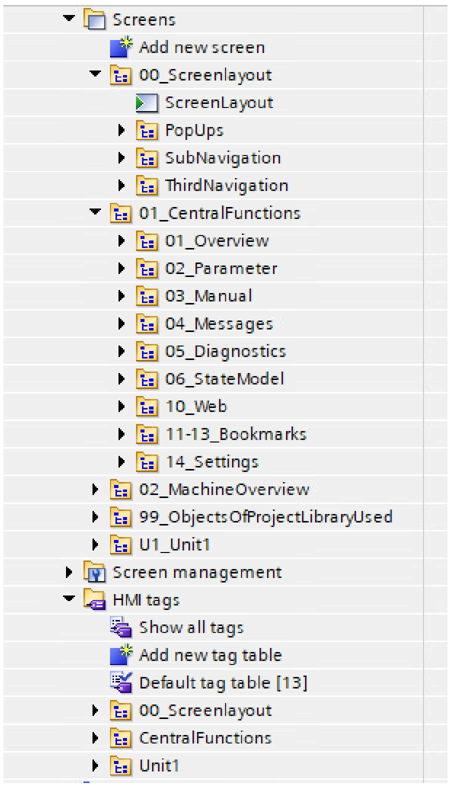

Screens
Add new screen
00_Screenlayout
ScreenLayout
PopUps
SubNavigation
ThirdNavigation
01_CentralFunctions
01_Overview
02_Parameter
03_Manual
04_Messages
05_Diagnostics
06_StateModel
10_Web
11-13_Bookmarks
14_Settings
02_MachineOverview
99_ObjectsOfProjectLibraryUsed
U1_Unit1
Screen management
HMI tags
Show all tags
Add new tag table
Default tag table [13]
00_Screenlayout
CentralFunctions
Unit1

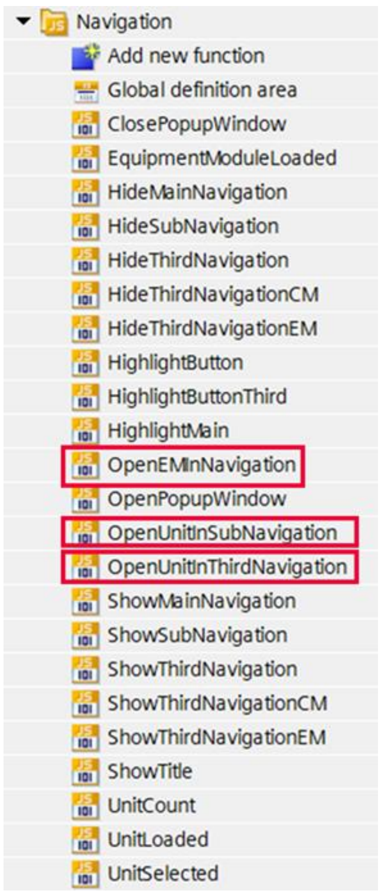
Figure 7-22: Project Tree structure – Screens and HMI tags

7.3.7. Scripts for generic Sub and Third Navigation

The main purpose of generic navigation scripts for Units and Equipment Modules is to simplify their management and accelerate their integration within the HMI project.

Navigation Scripts in the "Scripts Modules":

Previously "hard-coded" navigation scripts have been standardized as global scripts. This approach ensures simpler maintenance, improved debugging, and faster implementation in the project. These scripts are available in the global scripts directory under the Navigation folder.

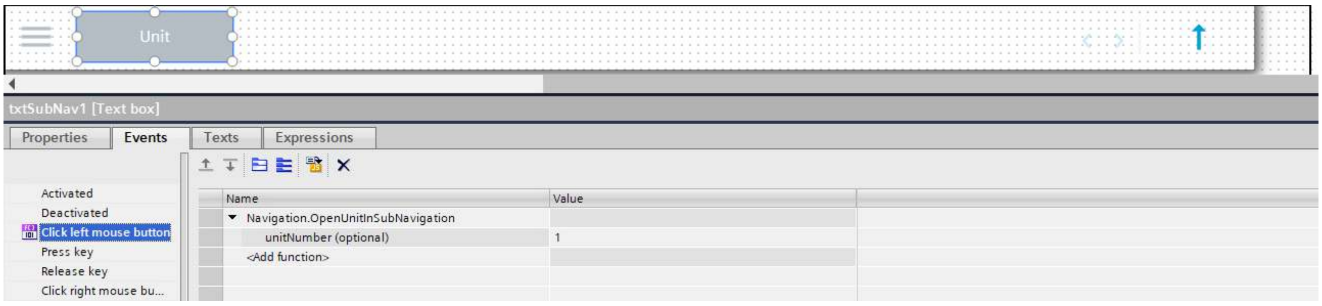


- Navigation
  - Add new function
  - Global definition area
  - ClosePopupWindow
  - EquipmentModuleLoaded
  - HideMainNavigation
  - HideSubNavigation
  - HideThirdNavigation
  - HideThirdNavigationCM
  - HideThirdNavigationEM
  - HighlightButton
  - HighlightButtonThird
  - HighlightMain
  - OpenEMInNavigation
  - OpenPopupWindow
  - OpenUnitInSubNavigation
  - OpenUnitInThirdNavigation
  - ShowMainNavigation
  - ShowSubNavigation
  - ShowThirdNavigation
  - ShowThirdNavigationCM
  - ShowThirdNavigationEM
  - ShowTitle
  - UnitCount
  - UnitLoaded
  - UnitSelected

Figure 7-23: Scripts for the navigation in the Scripts folder

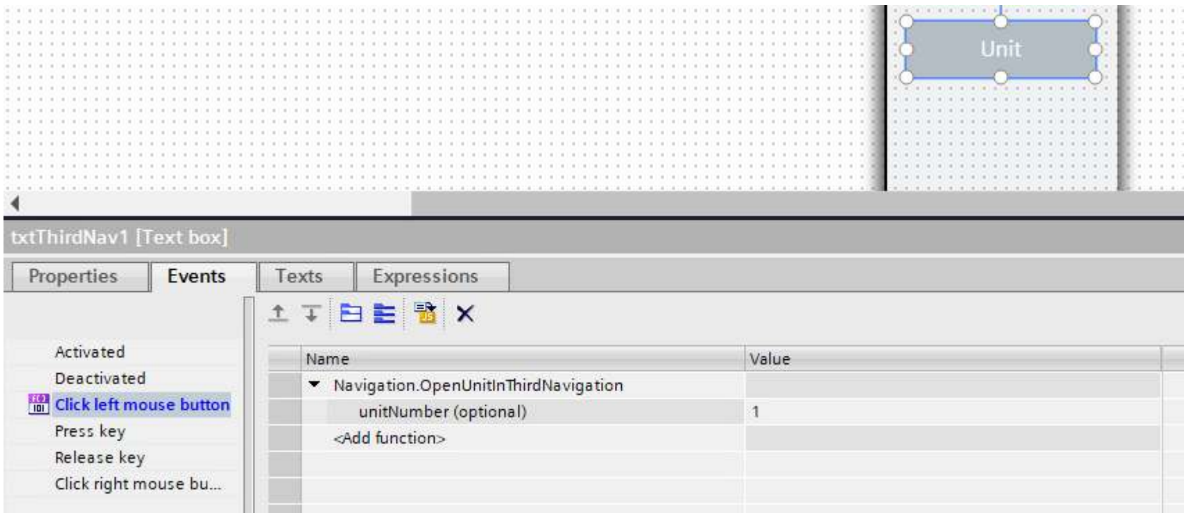
Using Navigation Scripts for Units

Navigation scripts are directly assigned to buttons associated with specific Units. To implement, simply import the script as a system function and assign the corresponding Unit number as a parameter:



Name	Value
Navigation.OpenUnitInSubNavigation	
unitNumber (optional)	1
<Add function>	

Figure 7-24: Script as system function for the Unit's Sub Navigation



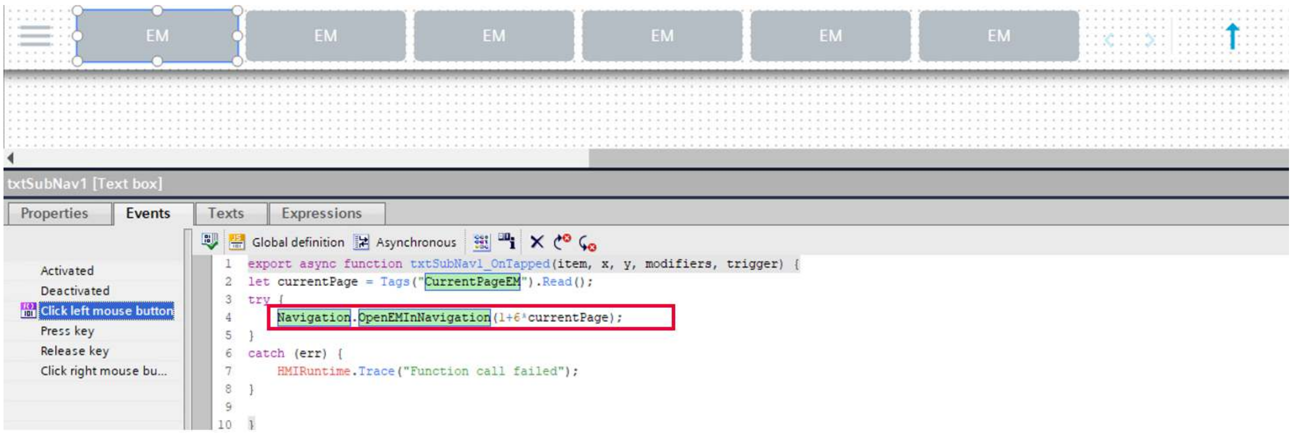
Name	Value
Navigation.OpenUnitInThirdNavigation	
unitNumber (optional)	1
<Add function>	

Figure 7-25: Script as system function import for the Unit Third Navigation

For detailed instructions on configuring and managing new Units, refer to Chapter 9.9.

Using Navigation Scripts for Equipment Modules

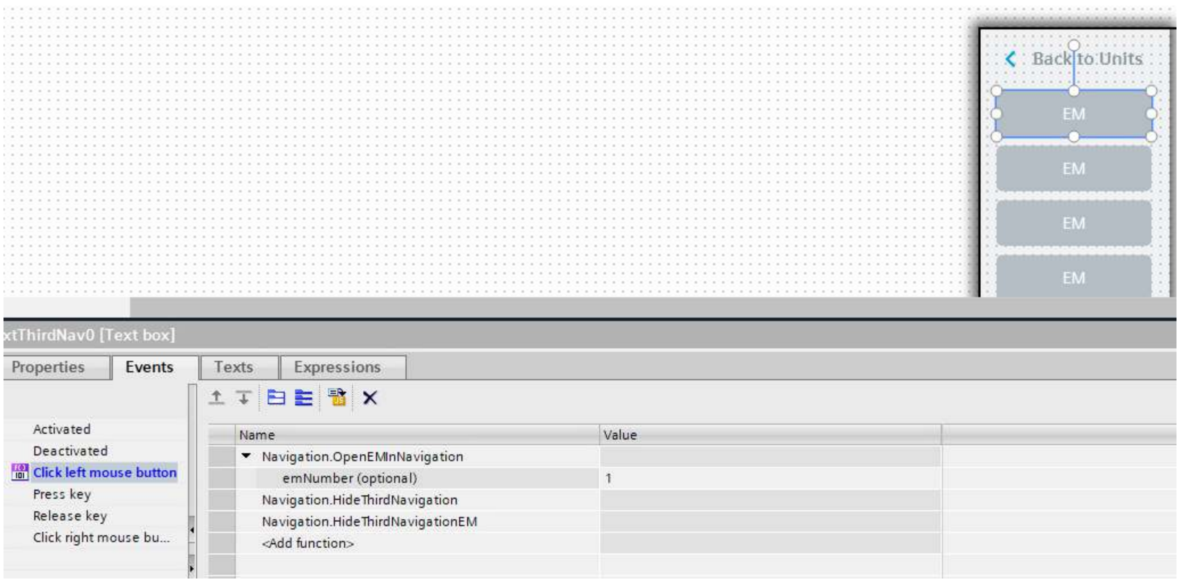
Navigation scripts are assigned directly to buttons associated with Equipment Modules. The script is being used and assigned to the corresponding EM number as a parameter (everything is prepared to not have to touch anything in the script. It dynamically changes depending on the number of EMs that we have in the proper Unit):



```
export async function txtSubNavl_OnTapped(item, x, y, modifiers, trigger) {
let currentPage = Tags("CurrentPageEM").Read();
try {
    Navigation.OpenEMInNavigation(1+6*currentPage);
}
catch (err) {
    HMIRuntime.Trace("Function call failed");
}

}
```

Figure 7-26: Script for the Equipment Module Sub Navigation



Activated
Deactivated
Click left mouse button
Press key
Release key
Click right mouse bu...

Name	Value
Navigation.OpenEMInNavigation	
emNumber (optional)	1
Navigation.HideThirdNavigation	
Navigation.HideThirdNavigationEM	
<Add function>	

Figure 7-27: Script import for the Equipment Module Third Navigation

For detailed instructions on configuring and managing Equipment modules, refer to chapter 10.15.

7.4. Machine Overview

The Machine Overview screen represents the highest-level visualization of the machine within the HMI. Because the AF$_{Basic}$ project supports multiple units, this overview provides operators with a clear summary, including status indicators for active errors (e), warnings (w), and informational messages (i) for each unit.

The HMI screen structure follows the ISA-88 standard, organized in a hierarchical, tree-based architecture. Clicking on any unit navigates down one level to display the detailed view of its associated Equipment Modules (EMs).

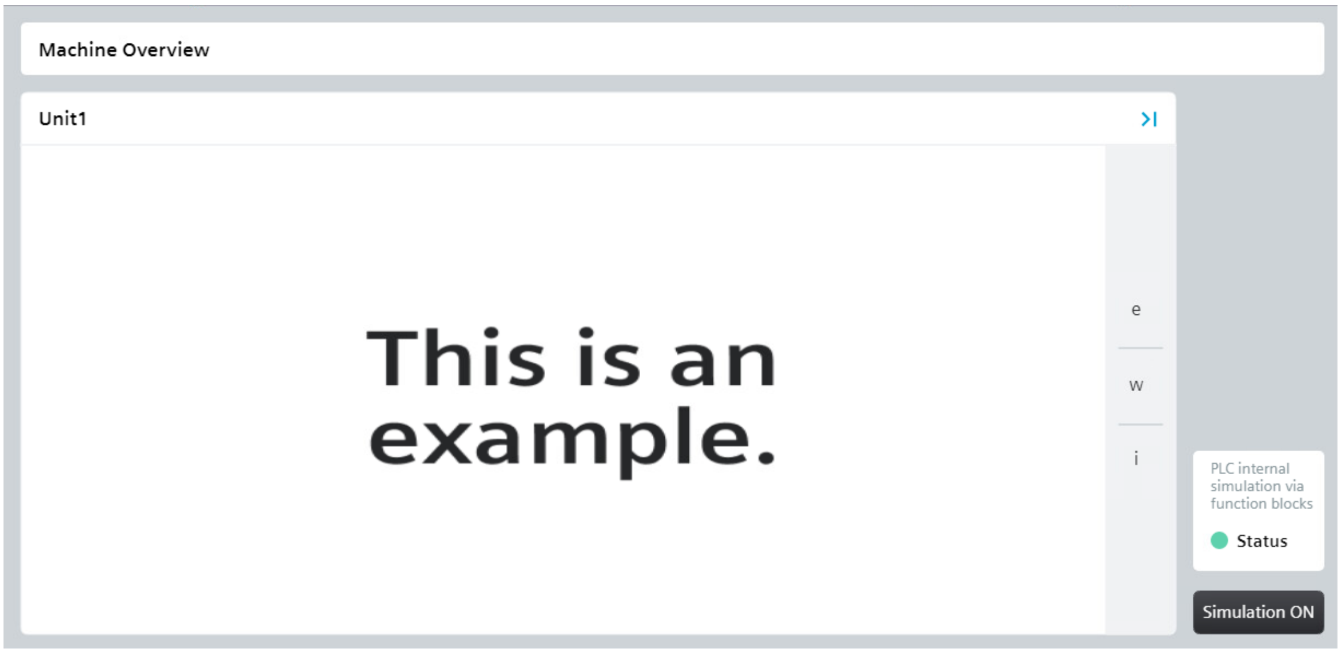


Figure 7-29: Example machine overview

7.5. Operation lock mechanism

When multiple HMIs are available to control the machine, it's essential to ensure that only one HMI can operate the machine at any given time. This functionality is implemented using the Operation Lock mechanism.

Concept

By default, all HMIs initialize in a locked state, displaying the Lock Screen until operational control is explicitly granted.

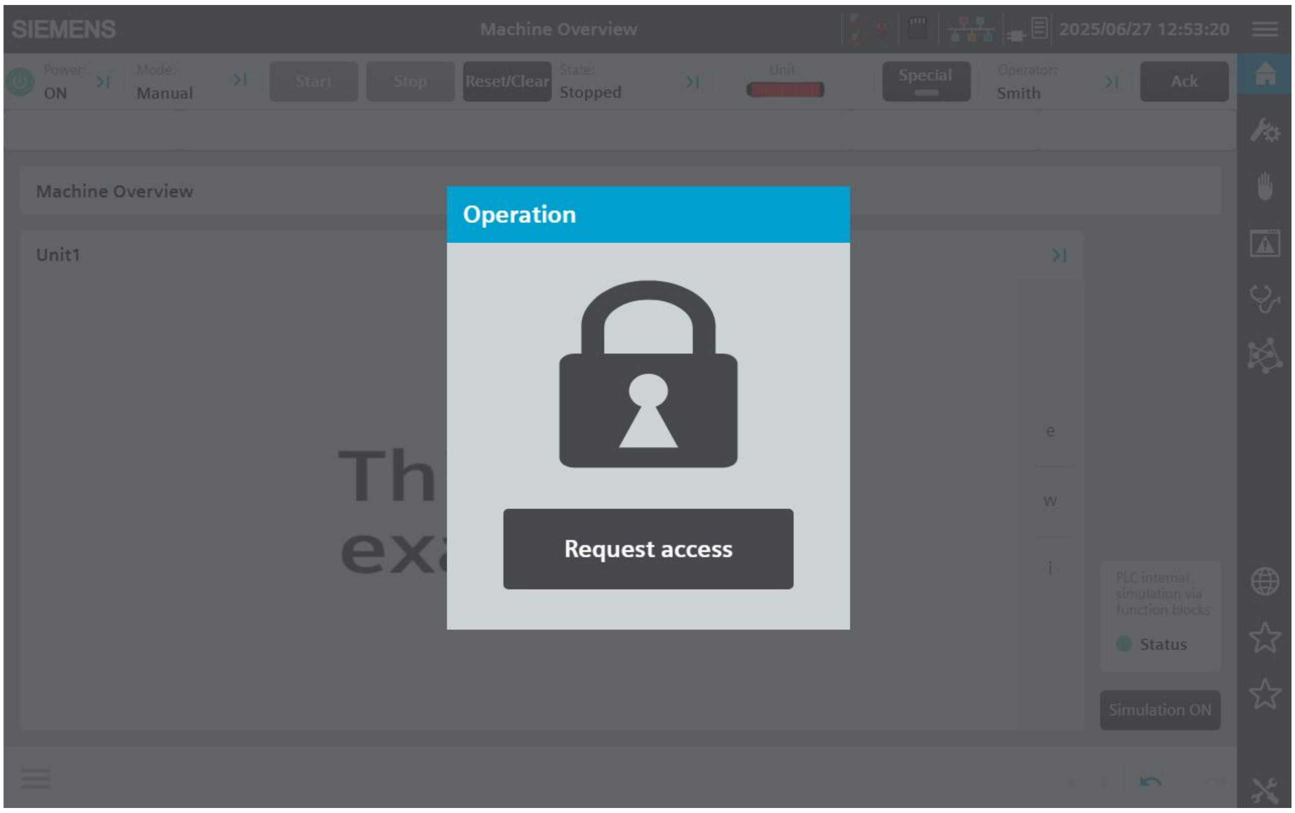


Figure 7-30: Operation lock

When the machine is powered on, all connected HMI devices display the Operation Lock sidebar window. To gain control of the machine, the operator must press the "Request access" button.

The Operation Lock indicates whether the current HMI has control rights. If no HMI or a different station is currently in control, the operator can request operation. When another HMI sends a new request, the previously active HMI is immediately interrupted and loses its control rights—though this behavior can be customized if needed.

Technical description

In the screen "00_ScreenLayout", the overlay screen "swLock" is configured with the highest priority (layer 31) to cover the entire display area when active.

When the "Request access" button is pressed, the following values are assigned to internal variables of the HMI:

- @ServerMachineName → 00_Screenlayout\Hmi_operationInterlock\commands\"ServerMachineName".
- @LocalMachineName → ' 00_Screenlayout \Hmi_operationInterlock\commands\"LocalMachineName".

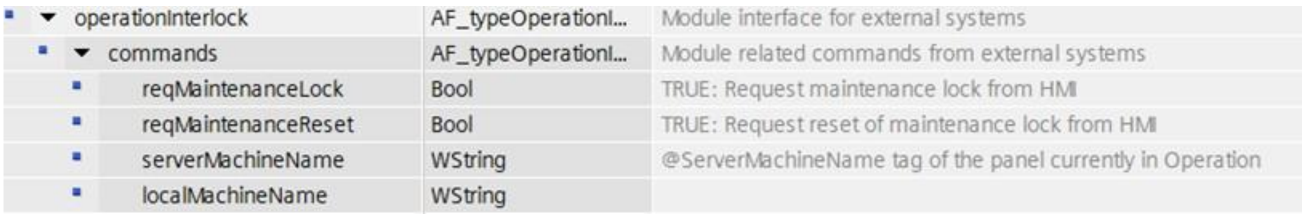
The visibility of "swLock" is controlled by a script triggered by PLC tags:

- " Hmi.operatorInterlock.commands.serverMachineName "
- " Hmi.operatorInterlock.commands.localMachineName "

  These PLC tags are linked to the HMI panel's internal tags @ServerMachineName and @LocalMachineName. When the operator requests control, the HMI updates these tags with its own identity, gaining exclusive operation access. All other HMIs are then locked, as their names no longer match the central PLC control values.

During testing or commissioning, 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 7-31: OperationInterlock data structure in the DB Hmi in the Folder Central Functions

8. Central Functions

Central Functions exist in the PLC and the HMI. In the PLC and HMI it is represented by a folder. The folder structure and folder numbering inside "General" 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.

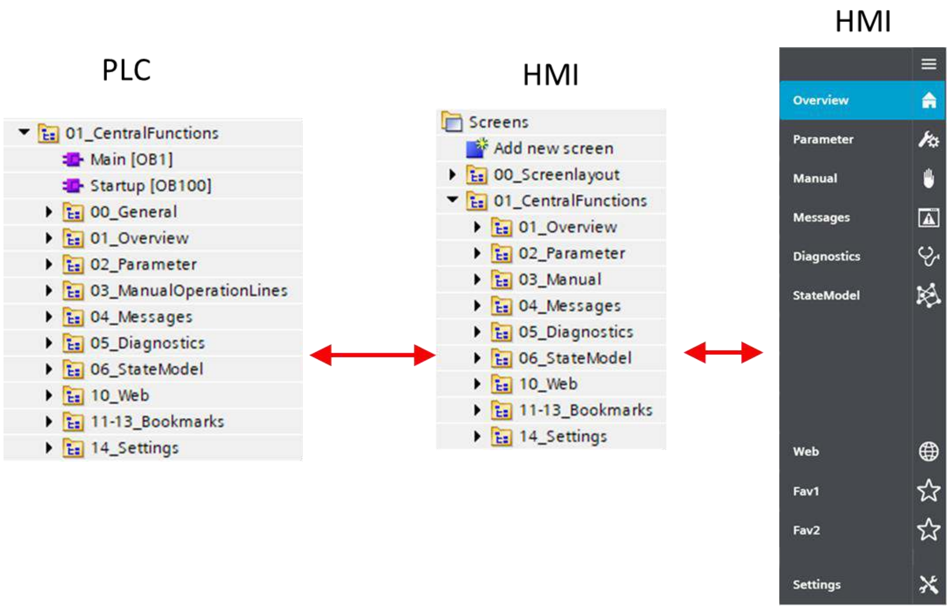


Figure 8-1: Overview of "Central Functions" in the PLC, HMI and side bar of the HMI

8.1. General

In the PLC, the folder "General" contains functionality that is system related and/or valid for the entire PLC. These are:

- Reaction to a connected SIMIT simulation
- Global Acknowledgement
- Global Power On/Off
- Connection to Main Header on the HMI
- Read local PLC date and time from CPU clock
- Clock bit generation
- Time synchronisation with NTP

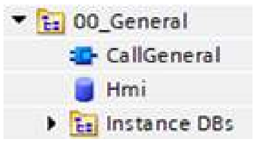


Figure 8-2: Content of folder "General" in the PLC

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

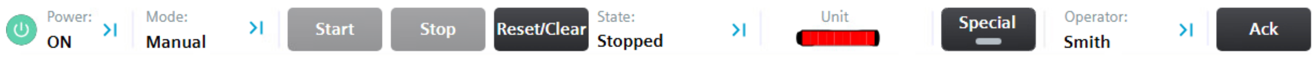


Figure 8-3: HMI main header

The HMI header displays the most global information: Power status, current mode and state of the unit. 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 main header in connected to the DB ”Hmi”. FB ”LAF_HmiHeaderSingleUnitManager” handles the data flow between the HMI and the unit. It can handle multiple panels and one single unit. Depending on the unit status (current mode and state) in the HMI, the button appearance and value displays (mode, state) are aggregated in the PLC from the unit and send to the HMI. FB ”LAF_HmiHeaderSingleManager” handles all that. It coordinates multiple HMIs with a single unit.

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 an equipment module from the unit.
- InterlinkControl_NonUnlinkable : Information box that the equipment module cannot be unlinked.

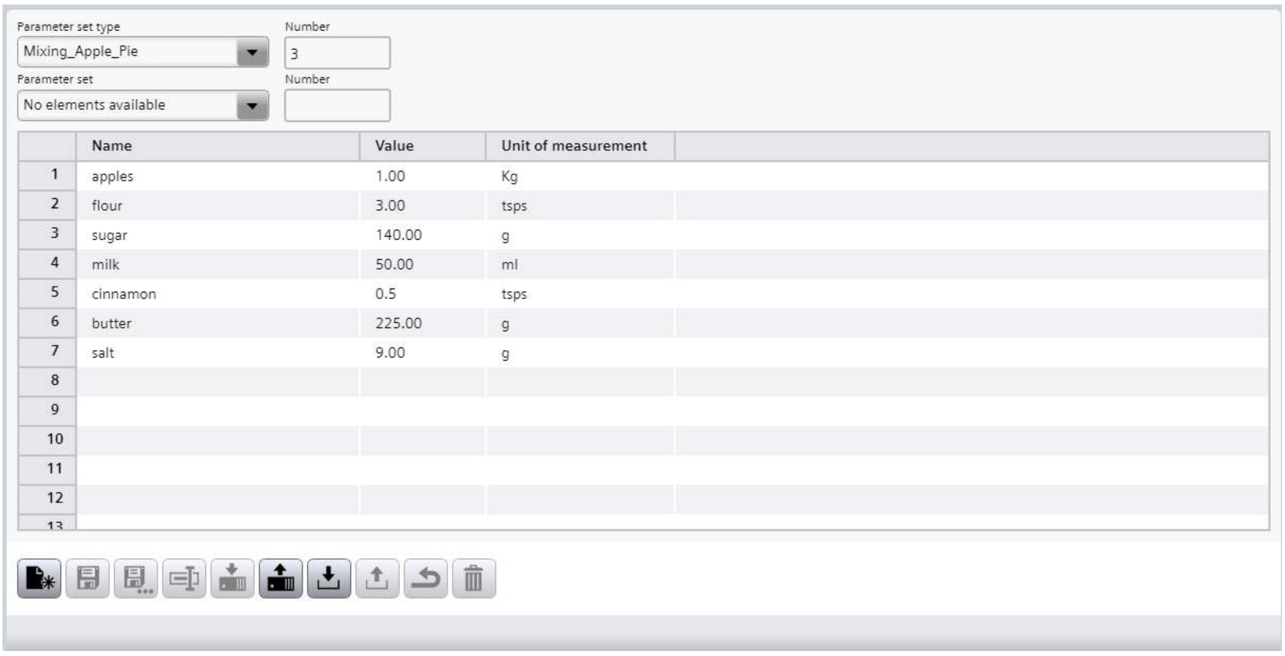
In the PLC, the functionality related to these pop-ups is handled within the FB “CallEquipment” of the equipment modules.

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.



Parameter set type	Mixing_Apple_Pie
Number	3
Parameter set	No elements available
Number	

	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/document/109821886

8.4. Manual Operation Lines

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:

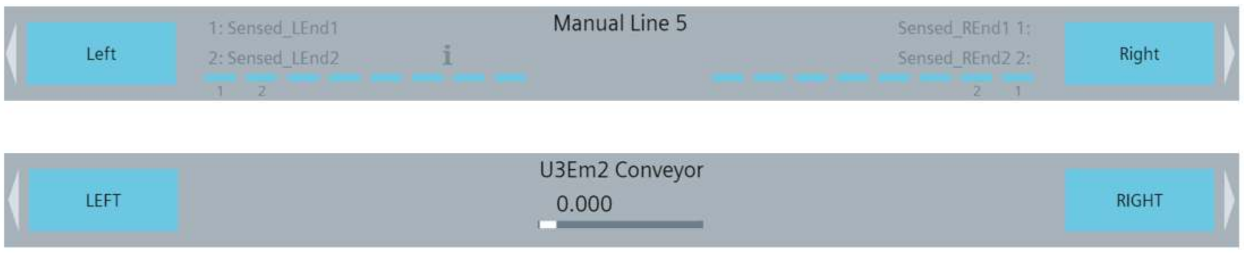


Figure 8-5: Example of Manual Operation Lines

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.

NOTICE

Json files on WinCC Unified Basic Panels

The Manual Operation Lines do need Json files to work. In the WinCC Unified Basic Panels there is no App to move the files from a USB Stick to the Panel. So in this case the chapter How to use Json files Add-In must be followed.

8.4.1. Main configuration concepts

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 How to implement Manual Operation Lines

Global DB for all Manual Operation Lines

For internal usage of Manual Operation Lines a global DB named "LAF_InternalManual" with an Array[0.."NO_OF_HMI_PANELS"] of "LAF_typeManHMIDisp", has to be created. A configuration is not necessary because the Automation Framework project works with one HMI: hmi[0] (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 Unit 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 are 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 combinations 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.

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:

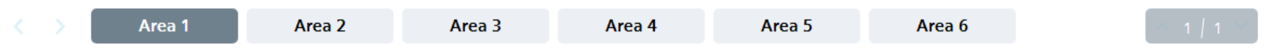


Figure 8-6: Area names for each EM

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 activation and deactivation colors, names for each end switch and name for the control screen (see Figure 8-10). Up to three different texts (symbolic, electrical or mechanical) can be configured in a Json file, allowing different texts to be displayed when a manual operation line is touched during the display of a message.

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 to the PC Runtime or panel and the path where it needs to be placed is configured in the HMI Tags as “manualOperationFilePath”.

- When using the manual operation lines for simulation and Unified PC runtime, the Json files must be manually copied into the configured path inside the HMI Tag “manualOperationFilePath”.

NOTE

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

- On the other hand, if the manual operation lines are to be used on a real Unified panel, the Json files need to be transferred using the Json files Add-In. The files will be automatically copied from the path <<locally saved TIA Project folder>>\UserFiles\jsonfiles to the configured path inside the HMI Tag "manualOperationFilePath" when the download is done. More information on chapter How to use Json files Add-In.

NOTE

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

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 structures 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.

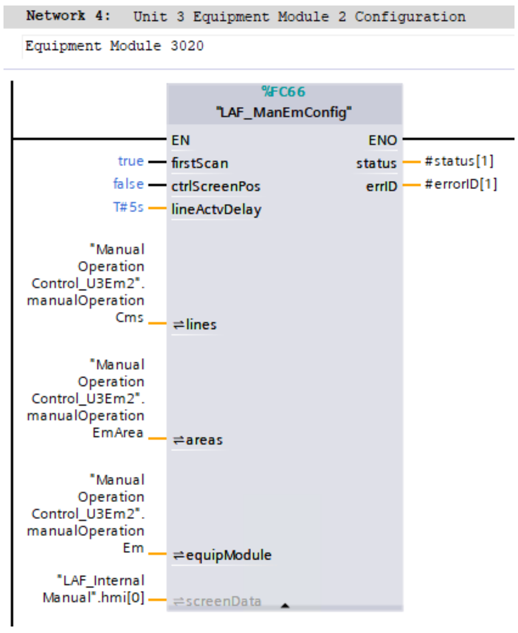

Network 4: Unit 3 Equipment Module 2 Configuration
Equipment Module 3020
%FC66
"LAF_ManEmConfig"
EN
ENO
true
firstScan
status
#status[1]
false
ctrlScreenPos
errID
#errorID[1]
T#5s
lineActvDelay
"Manual Operation Control_U3Em2".manualOperationCms
lines
"Manual Operation Control_U3Em2".manualOperationEmArea
areas
"Manual Operation Control_U3Em2".manualOperationEm
equipModule
"LAF_InternalManual".hmi[0]
screenData

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

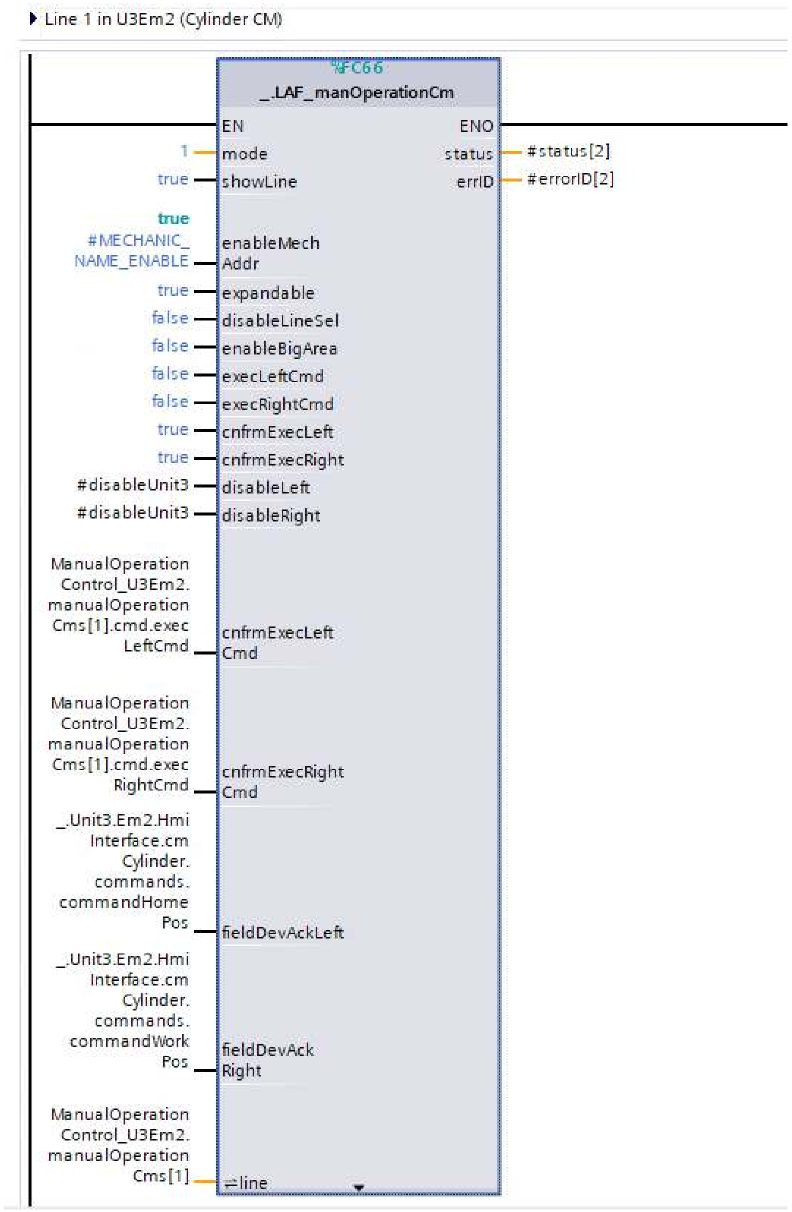
- LAF_manOperationCm

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



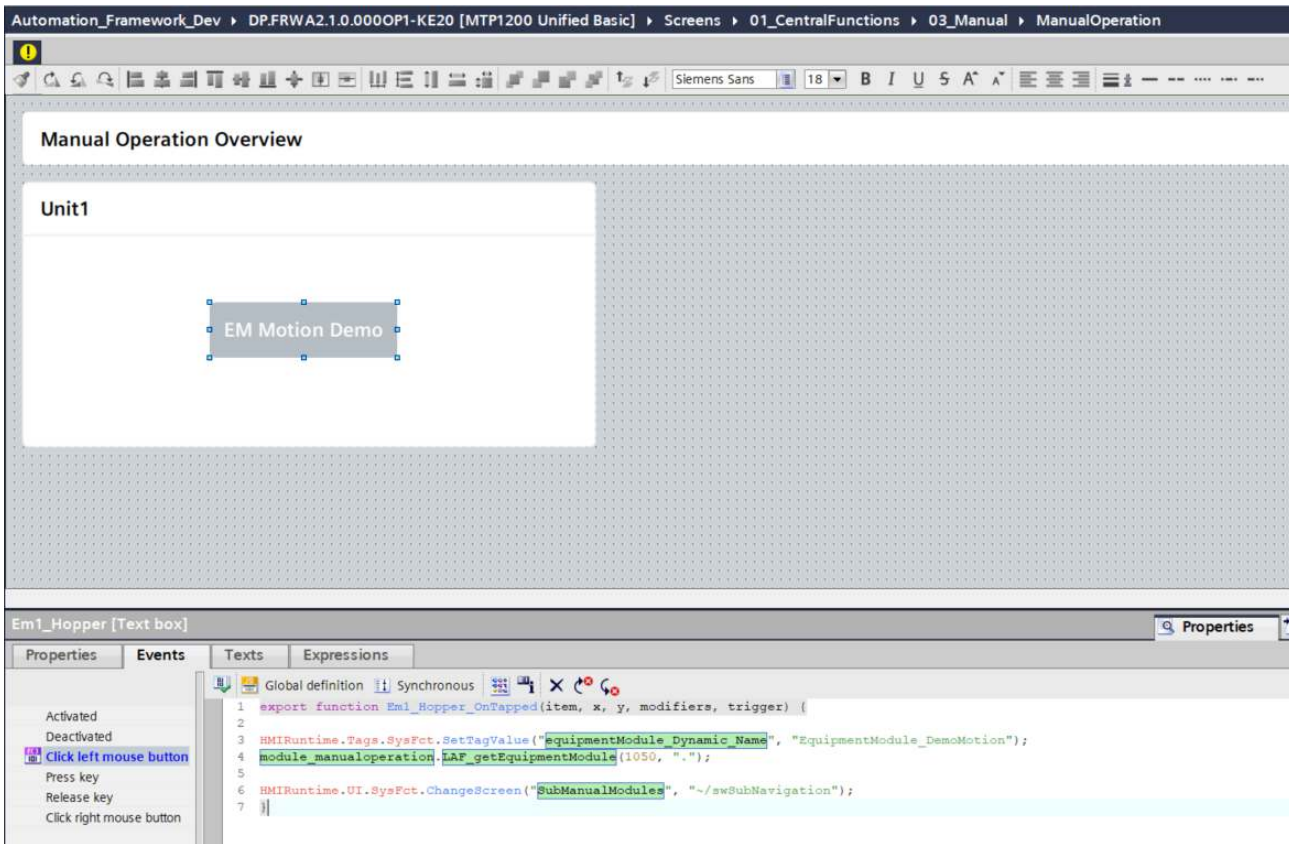
```
▶ Line 1 in U3Em2 (Cylinder CM)

                                  %FC66
                          _LAF_manOperationCm
                                  EN                 ENO
                         1 ── mode                status ── #status[2]
                      true ── showLine             errID ── #errorID[2]
                      true
       #MECHANIC_NAME_ENABLE ── enableMechAddr
                      true ── expandable
                     false ── disableLineSel
                     false ── enableBigArea
                     false ── execLeftCmd
                     false ── execRightCmd
                      true ── cnfrmExecLeft
                      true ── cnfrmExecRight
              #disableUnit3 ── disableLeft
              #disableUnit3 ── disableRight
ManualOperationControl_U3Em2.
 manualOperationCms[1].cmd.execLeftCmd ── cnfrmExecLeftCmd
ManualOperationControl_U3Em2.
 manualOperationCms[1].cmd.execRightCmd ── cnfrmExecRightCmd
_Unit3.Em2.HmiInterface.cmCylinder.
 commands.commandHomePos ── fieldDevAckLeft
_Unit3.Em2.HmiInterface.cmCylinder.
 commands.commandWorkPos ── fieldDevAckRight
ManualOperationControl_U3Em2.
 manualOperationCms[1] ── line
```

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

HMI Call for main view

The manual operation screen is generic so that the main view does not need additional screens to be created. To open the main screen "LAF_ManualOperation1_1280x800" with the correct equipment module, the global script "LAF_getEquipmentModule" under the "ManualOperation" needs to be used. As shown in the following image, the "LAF_getEquipmentModule" first parameter is the number of the equipment module, and the second parameter is the screen window path that should be used to open the screen.



```
export function Em1_Hopper_OnTapped(item, x, y, modifiers, trigger) {

HMIRuntime.Tags.SysFct.SetTagValue("equipmentModule_Dynamic_Name", "EquipmentModule_DemoMotion");
module_manualoperation.LAF_getEquipmentModule(1050, ".");

HMIRuntime.UI.SysFct.ChangeScreen("SubManualModules", "~/swSubNavigation");
}
```

Figure 8-9: Example HMI call for manual operation lines screen call

In the Figure ,the equipment module number 1050 is defined under the equipment modules DB “ManualOperationControl_U1Em1” with the parameter “equipmentModuleNum” from the variable “manualOperationEm”. The screen window path is set to “.” since the screen window opening the manual operation screen should be used, but in case a screen window in a layer higher or lower should be used, it is also possible to do. For Better visibility it is recommended to use the ScreenLayout’s screenWindow “swContent”.

HMI Line screens

Additional to the normal view, it is also possible to create specific expandable line screens to be open for specific lines as shown in the following image:

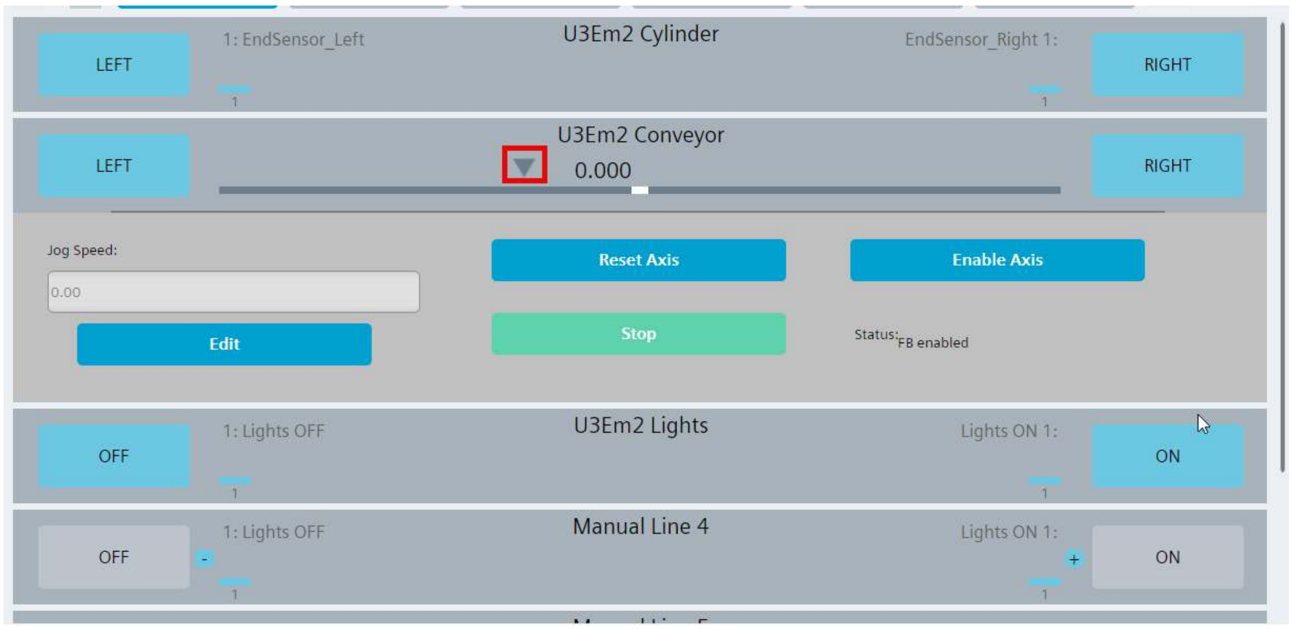


Figure 8-10: Example expandable manual operation line screen

To configure a screen for the line X of an equipment module YYYY, a screen located in the folder “LineScreenWindows” with the name “Template_ControlScreen_YYYY_X” is created and in the PLC, the "LAF_ManOperationCm" for line X has the input “expandable” set to true. To open the screens, the screen windows “controlX” in the screen “LAF_ActvInst1_1280x800” are used.

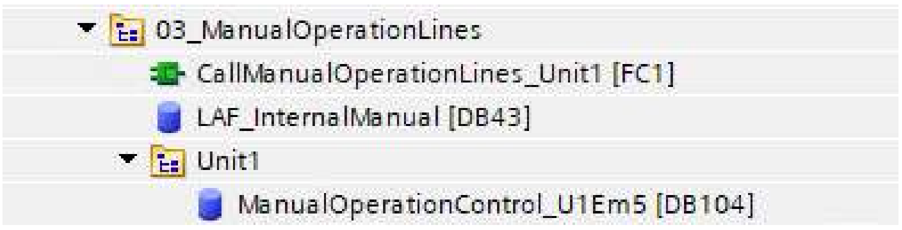
It is recommended for the line screen to be the same size as the screen window. For the 1280x800 resolution, the screen size 1194x180 is recommended.

8.4.2. How to implement Manual Operation Lines

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

In the AF project all library blocks for the Manual Operation Lines have been added into the folder “04_ObjectsOfProjectLibraryUsed - LAF - 10_Central – 03_ManualOperationLine - PLC”.

As they are part of the main navigation bar, in the AF Basic project, the Manual Operation Lines have been configured in the folder “01_CentralFunctions - 03_ManualOperationLines” where the lines are called.



03_ManualOperationLines
CallManualOperationLines_Unit1 [FC1]
LAF_InternalManual [DB43]
Unit1
ManualOperationControl_U1Em5 [DB104]

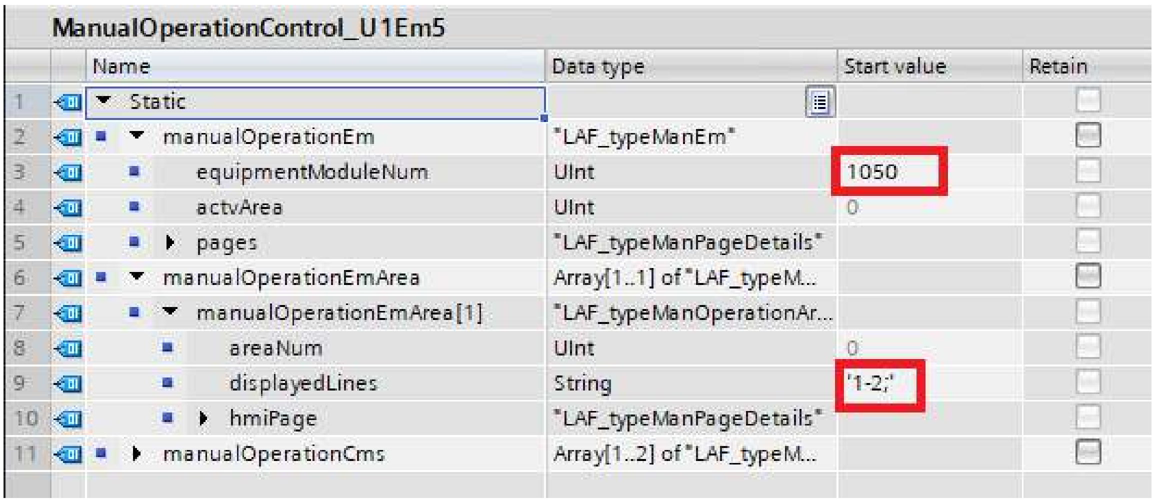
Figure 8-11: Blocks programmed for the Manual Operation Lines

Base configuration

The “CallManualOperationLines_UnitX” 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 chapter.

The connections for the FCs in the “CallManualOperationLines_UnitX” 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. The following image shows how one of the DBs, for EM5 of Unit1 was configured. The changes from the templated are highlighted in the red:

- number of the EM with the correct nomenclature (1050).
- number of lines to be displayed for each area, here two lines in the first area.



ManualOperationControl_U1Em5

	Name	Data type	Start value	Retain
1	Static			
2	manualOperationEm	"LAF_typeManEm"		
3	equipmentModuleNum	UInt	1050	
4	actvArea	UInt	0	
5	pages	"LAF_typeManPageDetails"		
6	manualOperationEmArea	Array[1..1] of "LAF_typeM...		
7	manualOperationEmArea[1]	"LAF_typeManOperationAr...		
8	areaNum	UInt	0	
9	displayedLines	String	'1-2;'	
10	hmiPage	"LAF_typeManPageDetails"		
11	manualOperationCms	Array[1..2] of "LAF_typeM...		

Figure 8-12: Manual operation DB configuration example U1 EM5

Connecting the CMs

After the configuration of the DB, the connections are added to the blocks inside “CallManualOperationLines” FC, connecting several of the inputs to the corresponding control node DB of the EMs. For example, in the case of Line1 of EM5, Unit1, 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 connect directly to the “jogForward” and “jogBackward” variables coming from the “commands” section of the HmiInterface.
- The inputs “fieldDevAckLeft“ and “fieldDevAckRight“ are the acknowledgement of execution from field device and it is connected to “Em5_HmiInterface.cmPosAxis.monitoring.jogBackward” which also comes from the HmiInterface DB of the EM5 cylinder when the monitoring to jog Backward has been activated.

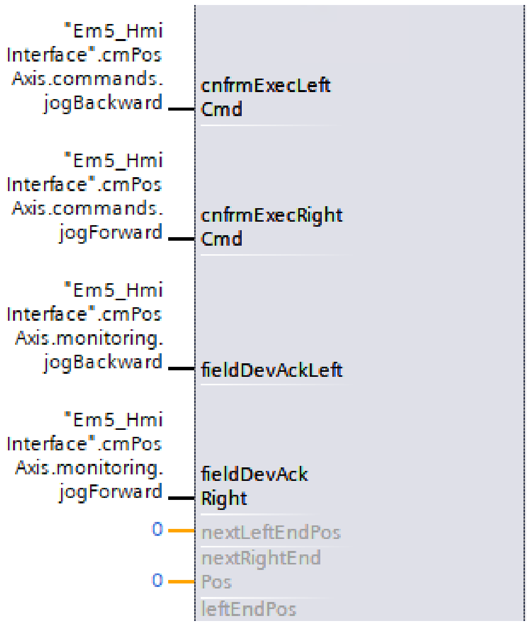

"Em5_HmiInterface".cmPosAxis.commands.jogBackward
cnfrmExecLeftCmd
"Em5_HmiInterface".cmPosAxis.commands.jogForward
cnfrmExecRightCmd
"Em5_HmiInterface".cmPosAxis.monitoring.jogBackward
fieldDevAckLeft
"Em5_HmiInterface".cmPosAxis.monitoring.jogForward
fieldDevAckRight
0
nextLeftEndPos
nextRightEndPos
0
leftEndPos

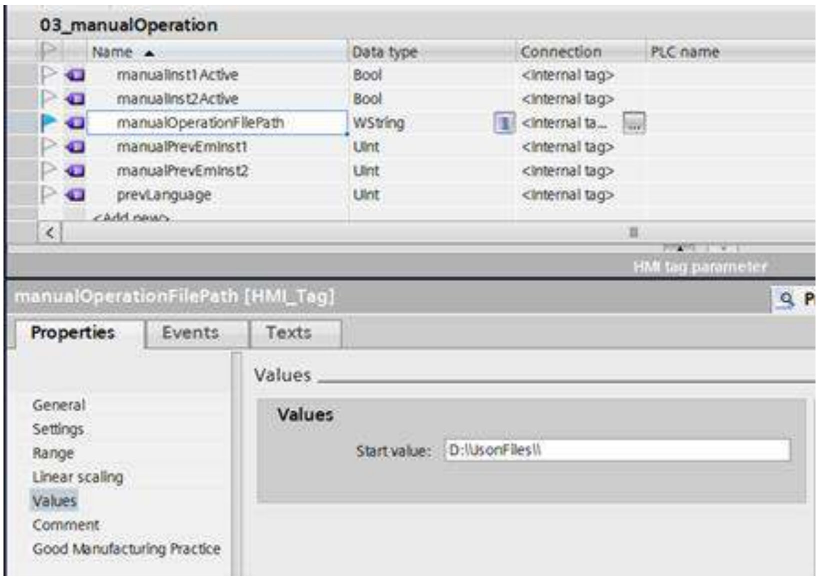
Figure 8-13: Closer look to inputs configured in “LAF_manOperationCm”

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. 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 8-15). 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/” to facilitate the testing.


03_manualOperation

Name	Data type	Connection	PLC name
manualInst1Active	Bool	<Internal tag>	
manualInst2Active	Bool	<Internal tag>	
manualOperationFilePath	WString	<Internal ta...	
manualPrevEmInst1	UInt	<Internal tag>	
manualPrevEmInst2	UInt	<Internal tag>	
prevLanguage	UInt	<Internal tag>	

manualOperationFilePath [HMI_Tag]

Properties | Events | Texts

General, Settings, Range, Linear scaling, Values, Comment, Good Manufacturing Practice

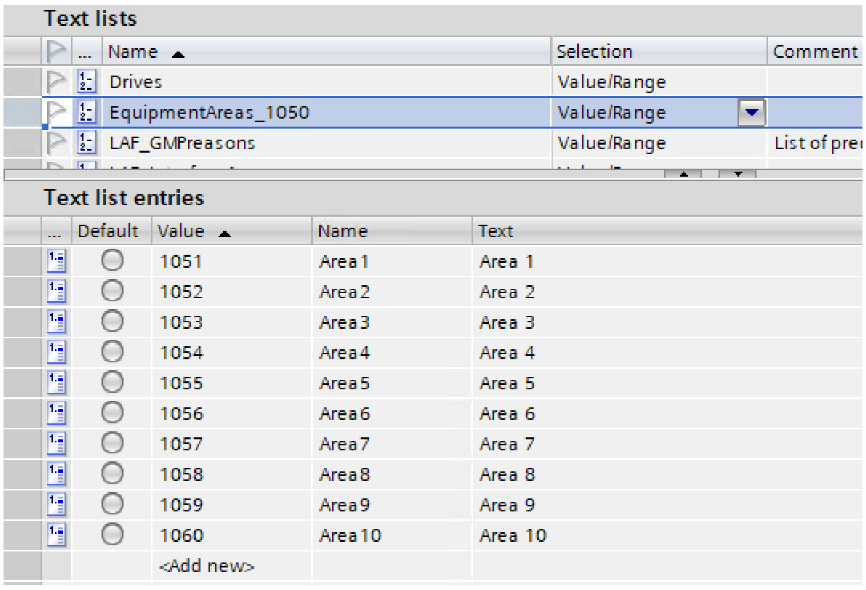
Values

Start value: D:\\JsonFiles\\

Figure 8-14: 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 Basic project:

- “EquipmentAreas” text lists : need to be configured for each EM with the corresponding name. For example, in the “EquipmentAreas_1050” text list the areas are configured as in the image below:


Text lists

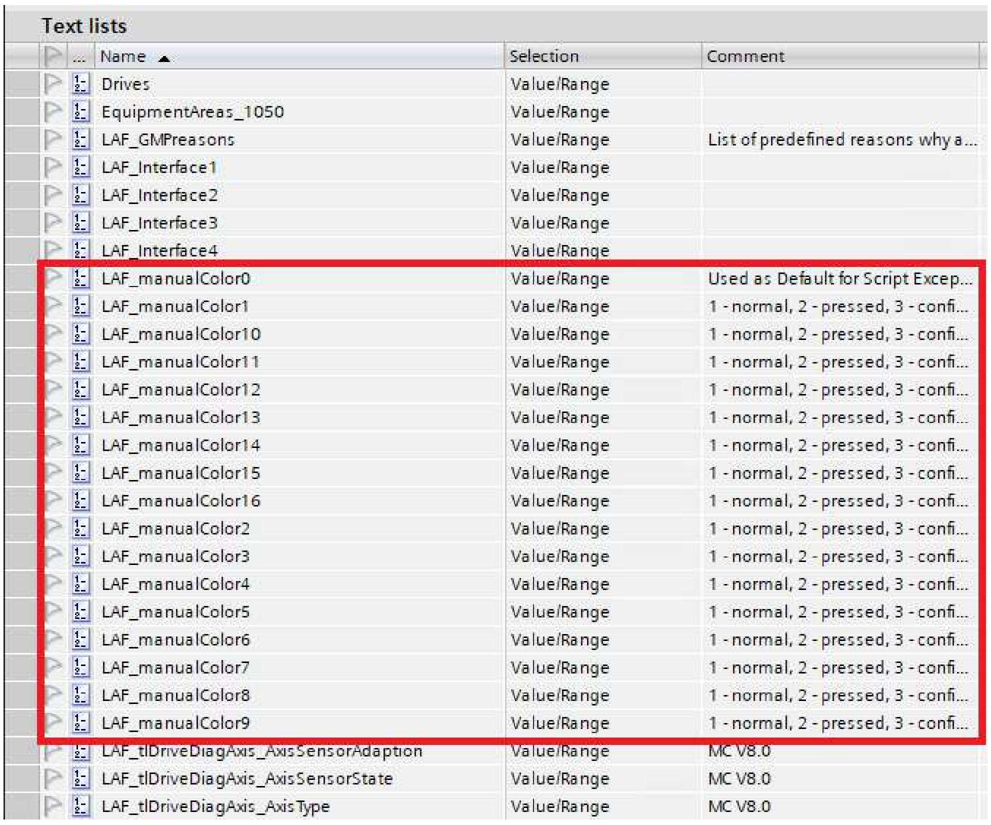
Name	Selection	Comment
Drives	Value/Range	
EquipmentAreas_1050	Value/Range	
LAF_GMPreasons	Value/Range	List of pre

Text list entries

Default	Value	Name	Text
○	1051	Area1	Area 1
○	1052	Area2	Area 2
○	1053	Area3	Area 3
○	1054	Area4	Area 4
○	1055	Area5	Area 5
○	1056	Area6	Area 6
○	1057	Area7	Area 7
○	1058	Area8	Area 8
○	1059	Area9	Area 9
○	1060	Area10	Area 10
	<Add new>		

Figure 8-15: “EquipmentAreas_1050” 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 Lines faceplates as a hexadecimal value. These text lists are needed to transform the colors that are read from the json files.



Text lists

Name	Selection	Comment
Drives	Value/Range	
EquipmentAreas_1050	Value/Range	
LAF_GMPreasons	Value/Range	List of predefined reasons why a...
LAF_Interface1	Value/Range	
LAF_Interface2	Value/Range	
LAF_Interface3	Value/Range	
LAF_Interface4	Value/Range	
LAF_manualColor0	Value/Range	Used as Default for Script Excep...
LAF_manualColor1	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor10	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor11	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor12	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor13	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor14	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor15	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor16	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor2	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor3	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor4	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor5	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor6	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor7	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor8	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_manualColor9	Value/Range	1 - normal, 2 - pressed, 3 - confi...
LAF_tlDriveDiagAxis_AxisSensorAdaption	Value/Range	MC V8.0
LAF_tlDriveDiagAxis_AxisSensorState	Value/Range	MC V8.0
LAF_tlDriveDiagAxis_AxisType	Value/Range	MC V8.0

Figure 8-16: Manual Operation Lines text lists needed in the AF Basic project

In the following step, the scripts for the Manual Operation Lines must be added to the HMI project. There is a folder named 'ManualOperation' that contains the main scripts used within the project screens. The remaining scripts are typified scripts that will be automatically included or updated when the LAF project library is updated.

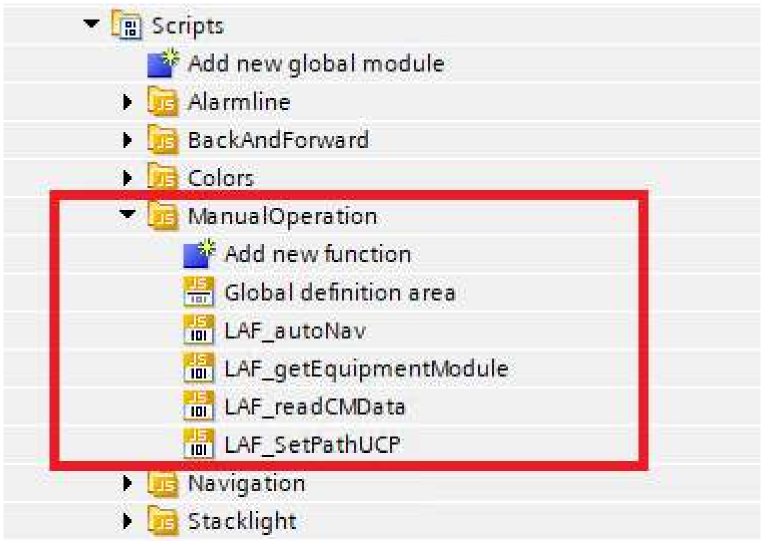


Figure 8-17: Manual Operation Lines scripts

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.

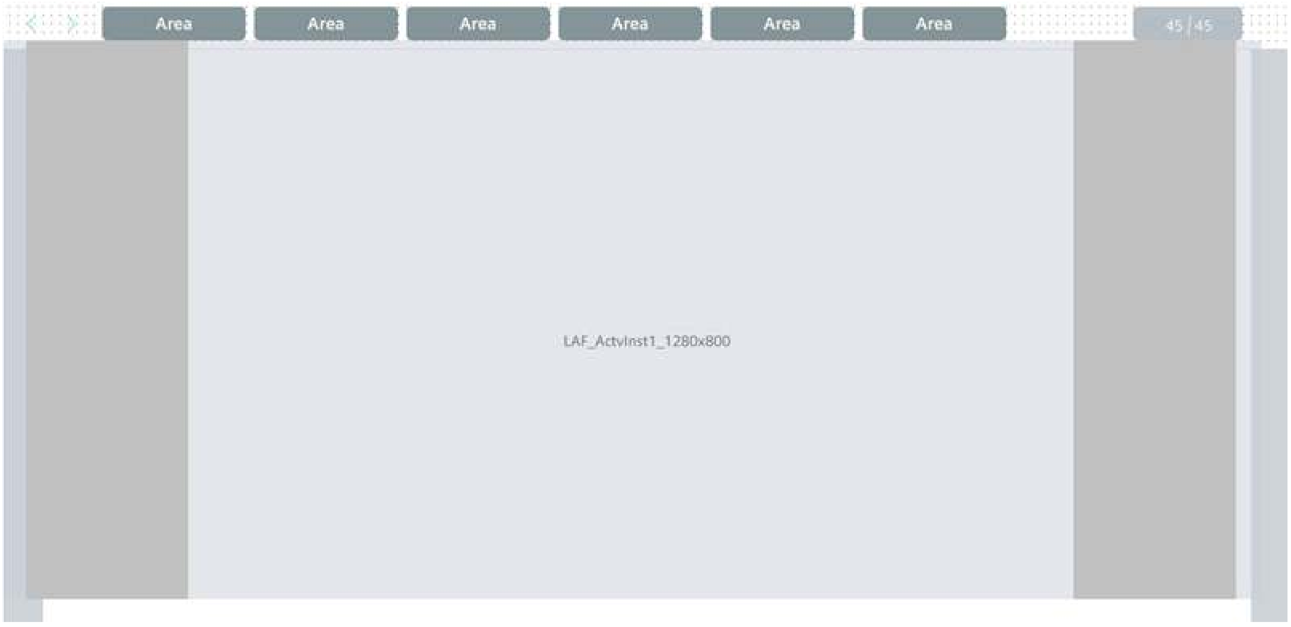
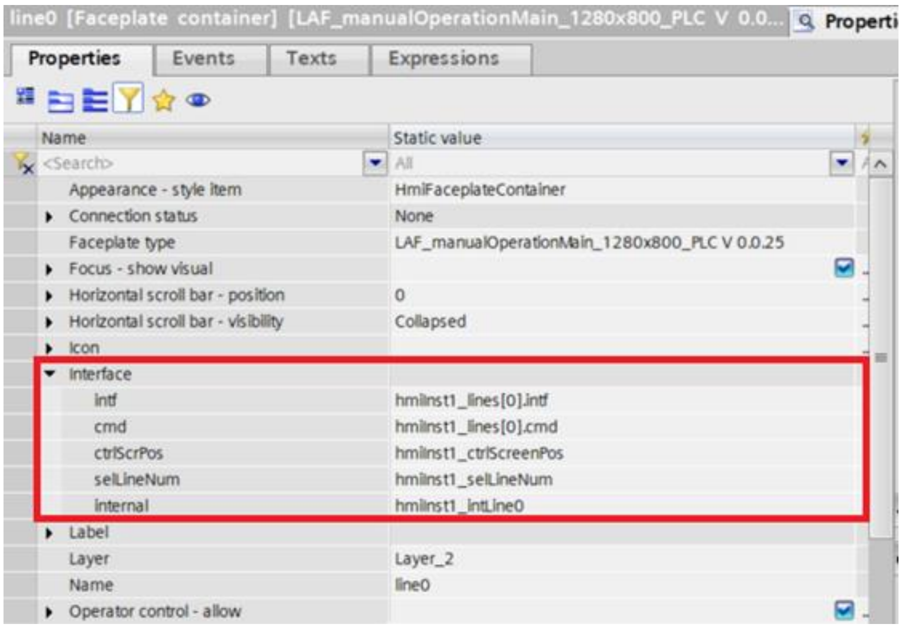


Figure 8-18: "LAF_ManualOperation1_1280x800" screen

• "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):

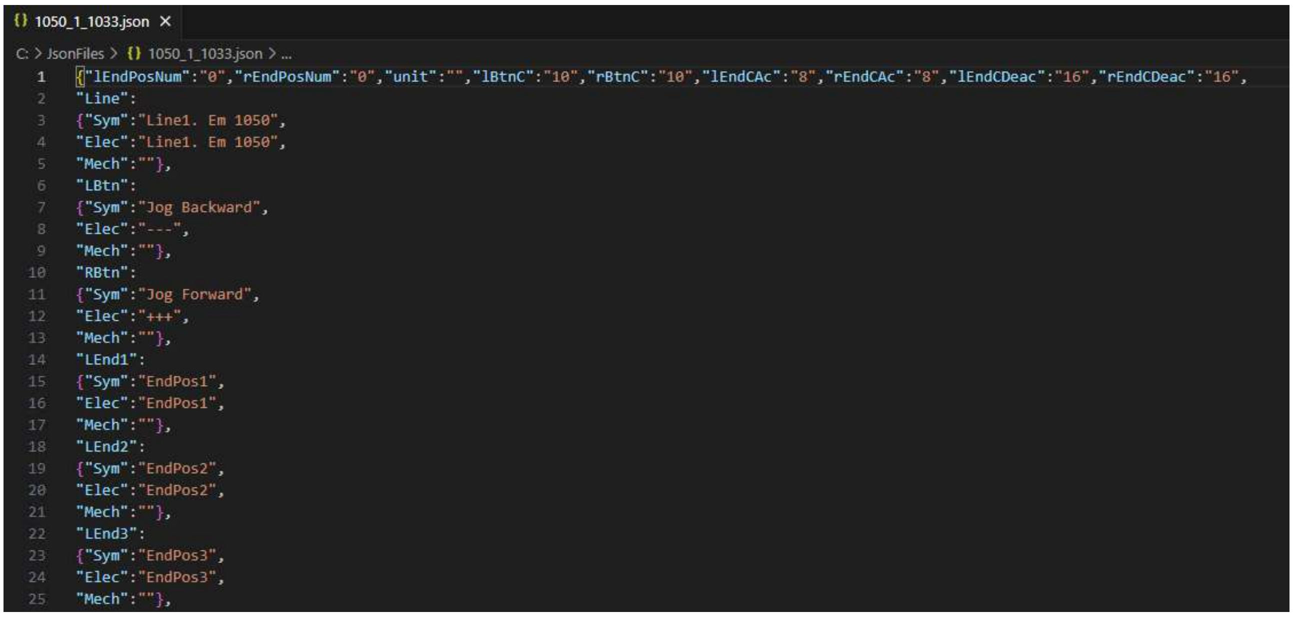

line0 [Faceplate container] [LAF_manualOperationMain_1280x800_PLC V 0.0...
Properti
Properties
Events
Texts
Expressions

Name	Static value
<Search>	All
Appearance - style item	HmiFaceplateContainer
Connection status	None
Faceplate type	LAF_manualOperationMain_1280x800_PLC V 0.0.25
Focus - show visual	✓
Horizontal scroll bar - position	0
Horizontal scroll bar - visibility	Collapsed
Icon	
Interface	
intf	hmiInst1_lines[0].intf
cmd	hmiInst1_lines[0].cmd
ctrlScrPos	hmiInst1_ctrlScreenPos
selLineNum	hmiInst1_selLineNum
internal	hmiInst1_intLine0
Label	
Layer	Layer_2
Name	line0
Operator control - allow	✓

Figure 8-19: 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, the control screen name 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 “C:\\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):



1050_1_1033.json

C: > JsonFiles > 1050_1_1033.json > ...

```
{"lEndPosNum":"0","rEndPosNum":"0","unit":"","lBtnC":"10","rBtnC":"10","lEndCAc":"8","rEndCAc":"8","lEndCDeac":"16","rEndCDeac":"16",
"Line":
{"Sym":"Line1. Em 1050",
"Elec":"Line1. Em 1050",
"Mech":""},
"LBtn":
{"Sym":"Jog Backward",
"Elec":"---",
"Mech":""},
"RBtn":
{"Sym":"Jog Forward",
"Elec":"+++",
"Mech":""},
"LEnd1":
{"Sym":"EndPos1",
"Elec":"EndPos1",
"Mech":""},
"LEnd2":
{"Sym":"EndPos2",
"Elec":"EndPos2",
"Mech":""},
"LEnd3":
{"Sym":"EndPos3",
"Elec":"EndPos3",
"Mech":""},
```

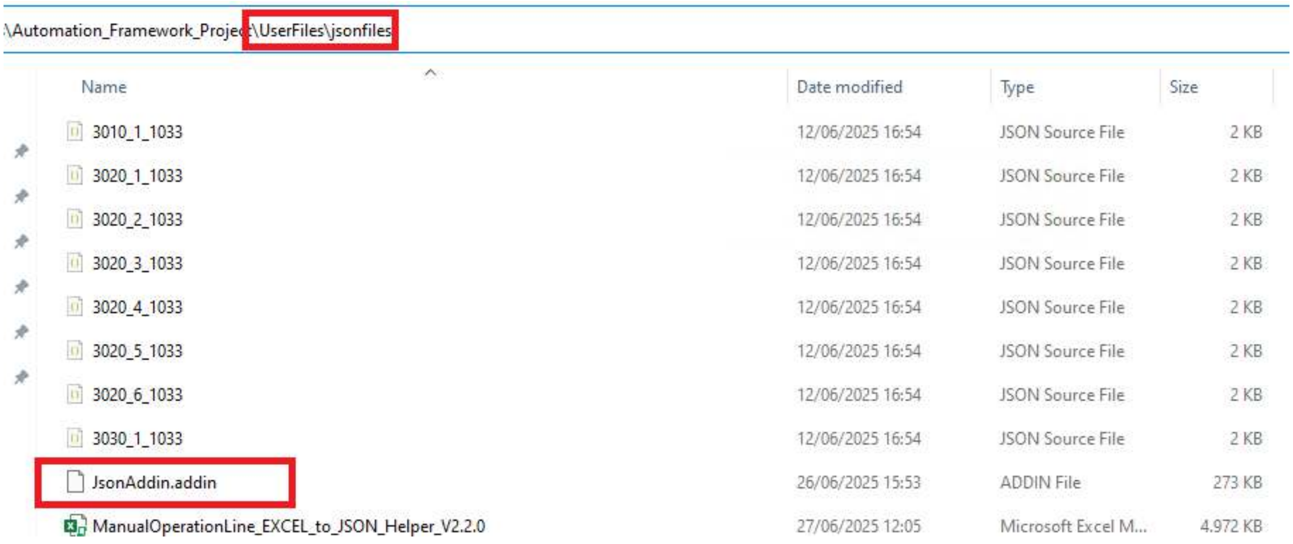
Figure 8-20: Example of Json File structure for Line 1 of the EM5 in the Unit 1 (1050_1_1033.json)

NOTE

The TIA Portal project contains a *.json-file helper (Excel-Sheet called ManualOperationLine_EXCEL_to_JSON_Helper) to generate the json-files automatically in the path. For more details see folder <<TIAPortalProjectPath>>\UserFiles\jsonfiles

8.4.3. How to use Json files Add-In

As mentioned earlier in Chapter 8.4.1, there is an add-in called "JsonAddin.addin" available at the path <locally saved TIA Project folder>\UserFiles\jsonfiles, which allows the necessary Json files to be directly copied to a real Unified panel so that the manual operation lines in the Automation Framework Basic can function correctly.



\Automation_Framework_Project\UserFiles\jsonfiles

Name	Date modified	Type	Size
3010_1_1033	12/06/2025 16:54	JSON Source File	2 KB
3020_1_1033	12/06/2025 16:54	JSON Source File	2 KB
3020_2_1033	12/06/2025 16:54	JSON Source File	2 KB
3020_3_1033	12/06/2025 16:54	JSON Source File	2 KB
3020_4_1033	12/06/2025 16:54	JSON Source File	2 KB
3020_5_1033	12/06/2025 16:54	JSON Source File	2 KB
3020_6_1033	12/06/2025 16:54	JSON Source File	2 KB
3030_1_1033	12/06/2025 16:54	JSON Source File	2 KB
JsonAddin.addin	26/06/2025 15:53	ADDIN File	273 KB
ManualOperationLine_EXCEL_to_JSON_Helper_V2.2.0	27/06/2025 12:05	Microsoft Excel M...	4.972 KB

Figure 8-22: Location of Json files Addin

NOTE

In case support is needed on how to use an Add-in in TIA Portal, here is the Siemens support link for Add-ins: SIOS-ID:109773999

First, the Addin must be activated in the section of TIA Portal for Addins as it can be seen in the next figure:

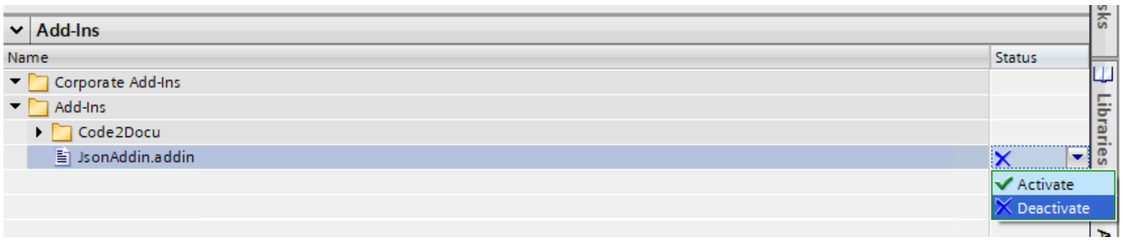


Figure 8-23: Json Addin activation requirement

Once Addin is activated, a full compilation of the software in the Unified Comfort Panel is needed to start with the process (the Json files will be copied after the compilation but if the user re-compiles again without launching the Addin the Json files will not be copied anymore).

After the compilation, select the Addin with right click on the Unified Comfort Panel: Json Addin>Manual operation>Copy Json files.

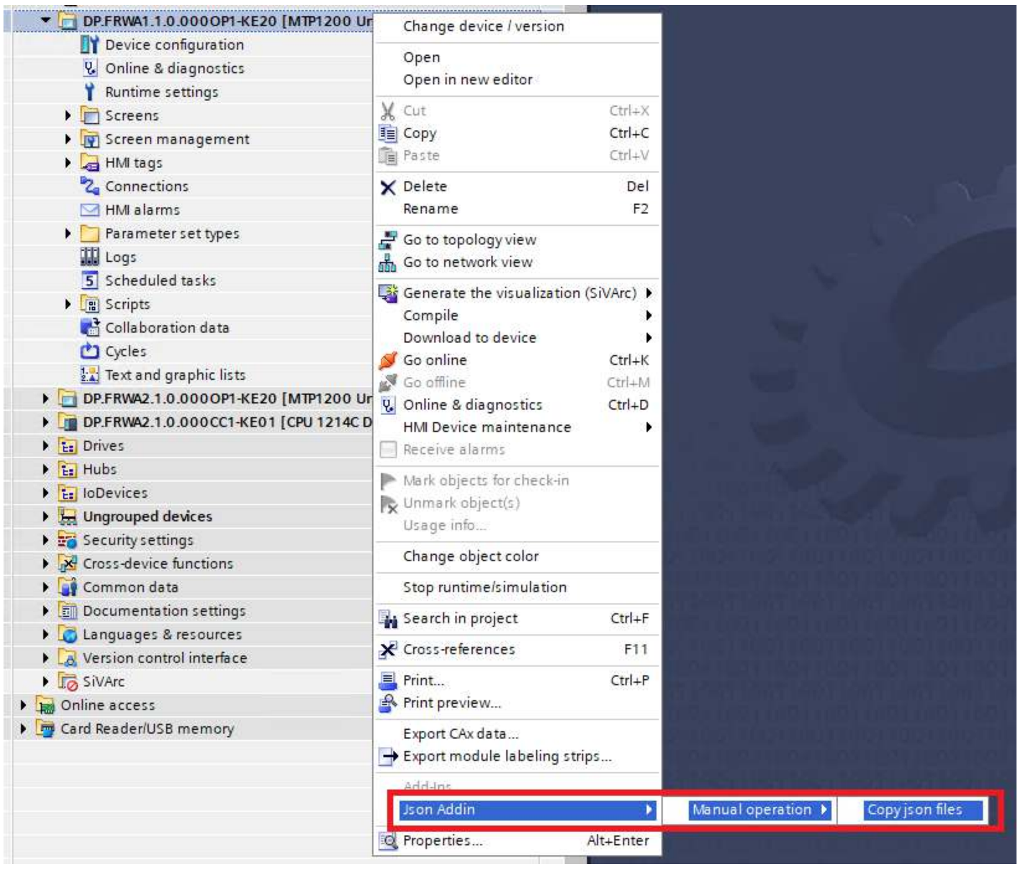


Figure 8-24: Addin execution steps

After launching the Addin, a message will appear reminding that a compilation of the project is required before executing the Addin. As the first step has been doing a full compilation, it can be clicked on “OK”. If everything has gone correct, another message should appear announcing “Copy of Json files completed”.

Last step would be downloading to device where the manual operation files should already be visible. It can also be check if the Json files have been copied correctly inside the panel by going to the path “/home/industrial/JsonFiles/”.

The copying of the JsonFiles is done in the Loaded event of “ScreenLayout” with the script “LAF_Copy”.

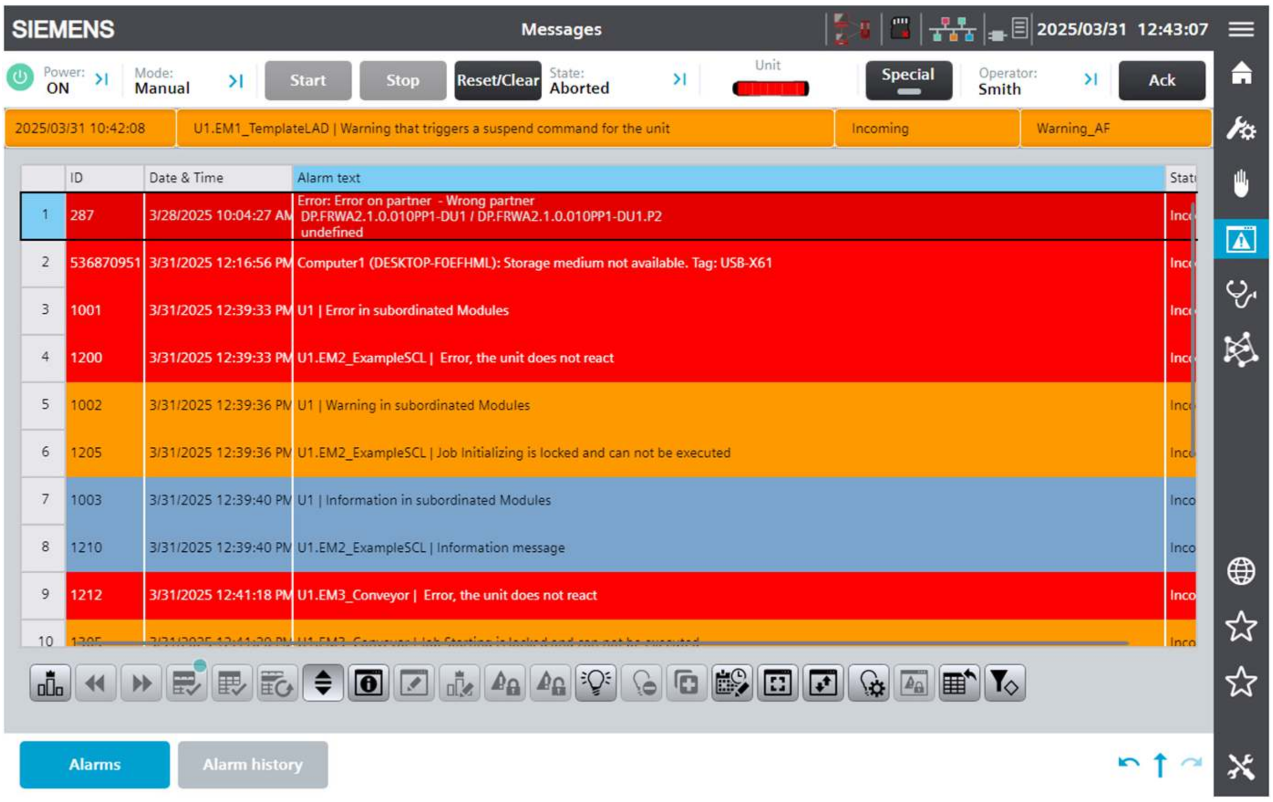
8.5. Messages

The “Messages” has two sub menus. The first one handle current alarm/messages viewer and the second opens the historical alarm viewer. This last one shows the alarms saved in the “Alarm History” log.

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 diagnostics messages are also displayed.



2025/03/31 10:42:08	U1.EM1_TemplateLAD | Warning that triggers a suspend command for the unit	Incoming	Warning_AF

	ID	Date & Time	Alarm text	Stat
1	287	3/28/2025 10:04:27 AM	Error: Error on partner - Wrong partner DP.FRWA2.1.0.010PP1-DU1 / DP.FRWA2.1.0.010PP1-DU1.P2 undefined	Inc
2	536870951	3/31/2025 12:16:56 PM	Computer1 (DESKTOP-F0EFHML): Storage medium not available. Tag: USB-X61	Inc
3	1001	3/31/2025 12:39:33 PM	U1 | Error in subordinated Modules	Inc
4	1200	3/31/2025 12:39:33 PM	U1.EM2_ExampleSCL | Error, the unit does not react	Inc
5	1002	3/31/2025 12:39:36 PM	U1 | Warning in subordinated Modules	Inc
6	1205	3/31/2025 12:39:36 PM	U1.EM2_ExampleSCL | Job Initializing is locked and can not be executed	Inc
7	1003	3/31/2025 12:39:40 PM	U1 | Information in subordinated Modules	Inco
8	1210	3/31/2025 12:39:40 PM	U1.EM2_ExampleSCL | Information message	Inco
9	1212	3/31/2025 12:41:18 PM	U1.EM3_Conveyor | Error, the unit does not react	Inco
10	1305	3/31/2025 12:41:20 PM	U1.EM3_Conveyor | Job Starting is locked and can not be executed	Inco

Figure 8-11: Alarms in Menu “Messages”

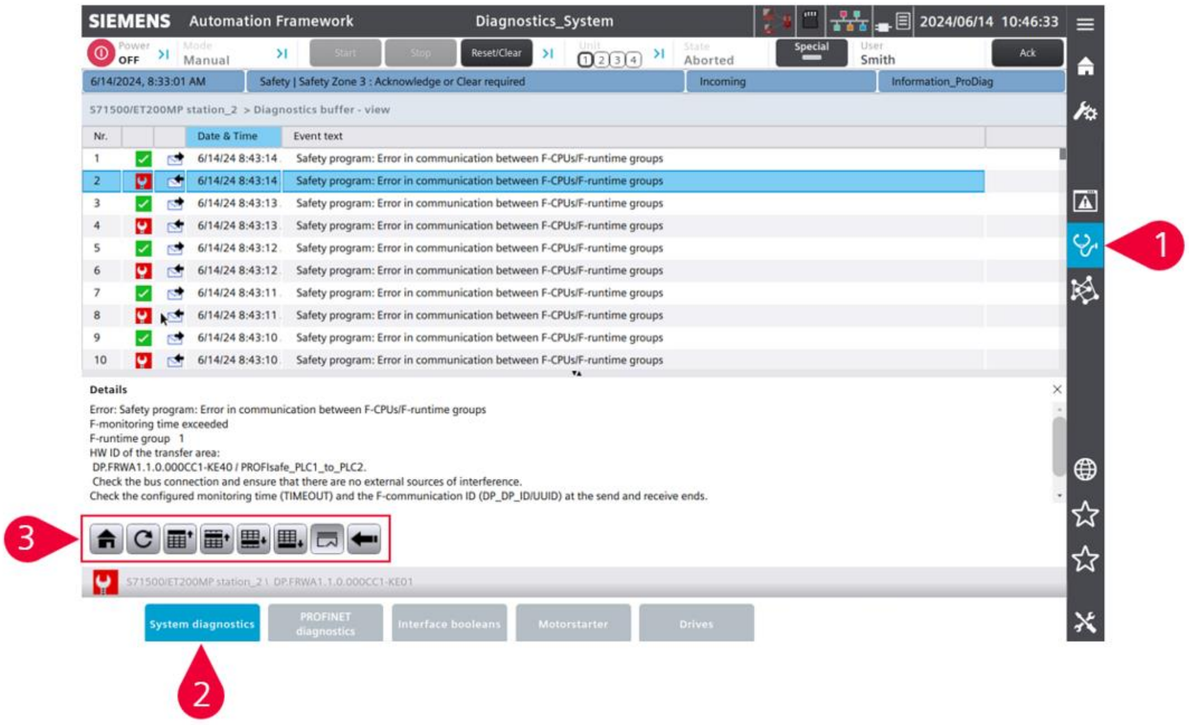
8.6. Diagnostics

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

8.6.1. System diagnostics

The AFBasic 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 IO-Systems.

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.



Nr.			Date & Time	Event text
1	✓		6/14/24 8:43:14	Safety program: Error in communication between F-CPUs/F-runtime groups
2	🟥		6/14/24 8:43:14	Safety program: Error in communication between F-CPUs/F-runtime groups
3	✓		6/14/24 8:43:13	Safety program: Error in communication between F-CPUs/F-runtime groups
4	🟥		6/14/24 8:43:13	Safety program: Error in communication between F-CPUs/F-runtime groups
5	✓		6/14/24 8:43:12	Safety program: Error in communication between F-CPUs/F-runtime groups
6	🟥		6/14/24 8:43:12	Safety program: Error in communication between F-CPUs/F-runtime groups
7	✓		6/14/24 8:43:11	Safety program: Error in communication between F-CPUs/F-runtime groups
8	🟥		6/14/24 8:43:11	Safety program: Error in communication between F-CPUs/F-runtime groups
9	✓		6/14/24 8:43:10	Safety program: Error in communication between F-CPUs/F-runtime groups
10	🟥		6/14/24 8:43:10	Safety program: Error in communication between F-CPUs/F-runtime groups

Details

Error: Safety program: Error in communication between F-CPUs/F-runtime groups
F-monitoring time exceeded
F-runtime group 1
HW ID of the transfer area:
DP.FRWA1.1.0.000CC1-KE40 / PROFIsafe_PLC1_to_PLC2.
Check the bus connection and ensure that there are no external sources of interference.
Check the configured monitoring time (TIMEOUT) and the F-communication ID (DP_DP_ID/UUID) at the send and receive ends.

Figure 8-12: 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” (for 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 AFBasic provides a detailed status diagnostic information of the PLC and their lower-level hardware components of the PROFINET IO-System 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.

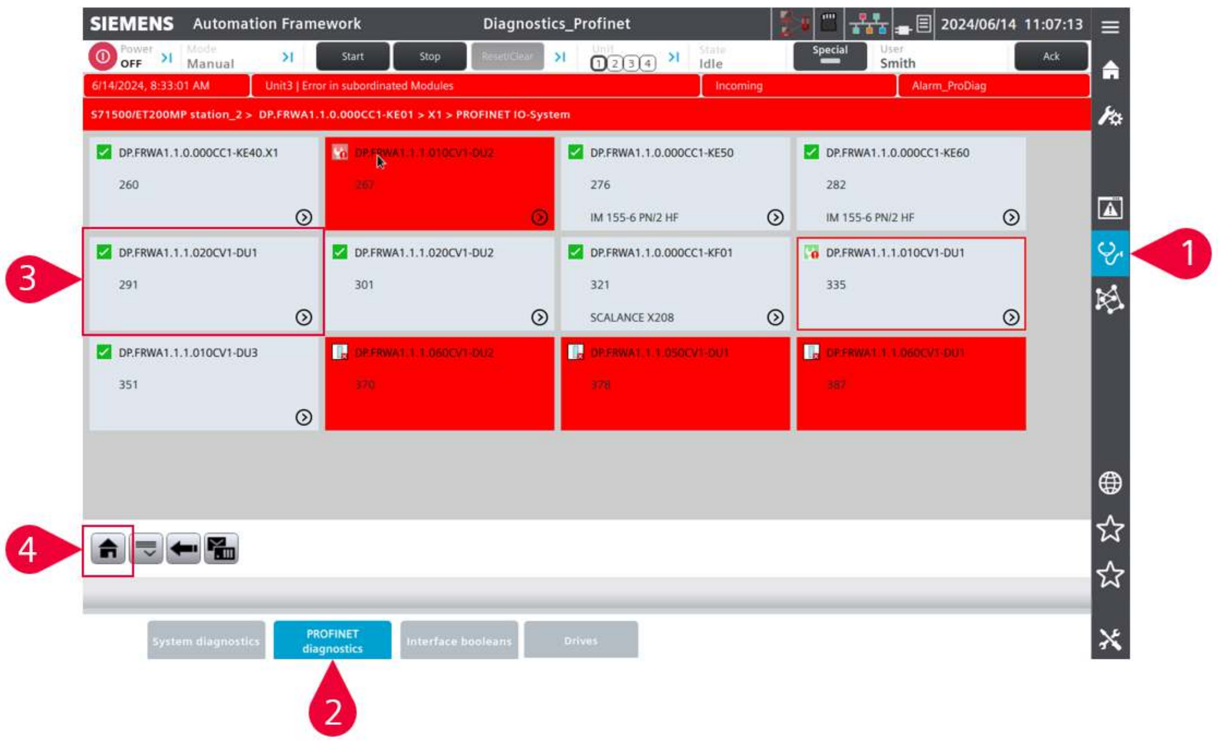


Figure 8-13: PROFINET diagnostics (Distributed IO view)

8.6.3. Interface Signals

8.6.3.1. Overview

Industrial automation systems frequently require monitoring numerous discrete (Boolean) process signals for diagnostics, status indication, and operational feedback. These signals originate from devices such as light barriers, limit switches, pressure sensors, and any other process signals. Within the AFBasic environment, a specific mechanism exists to facilitate the monitoring of such signals on an HMI. This mechanism involves selecting relevant signals during the engineering phase, configuring them within the PLC, and visualizing their status on the HMI during runtime.

The LAF_ReadBoolSignals and LAF_ReadWordSignals FC blocks serve as the core component enabling this functionality. These blocks are called within CallInterfaceSignals FB as shown on Figure 8-16: Call LAF_InterfaceSignals under Engineering section. The blocks read signals connected as inputs (signalLeft00 through signalLeft15 and signalRight00 through signalRight15 for the LAF_readBoolSignals) and (signalsLeft and SignalsRight as WORDs for the LAF_ReadWordSignals). Each interface block represents 32 Boolean signals which can be monitored on a single screen.

To efficiently manage multiple signals and optimize HMI visualization, each logical "interface" representing 32 Boolean signals is implemented using two consecutive WORD elements in the input interface for the LAF_ReadWordSignals and 32 input Booleans for the LAF_ReadBoolSignals, see Architecture. The LAF_Interface blocks process these inputs and produces two separate WORDs, designated as "signalsLeft" and "signalsRight" HMI interfaces so the HMI can display the 32 signals on two separate windows "left" and "right" per page.

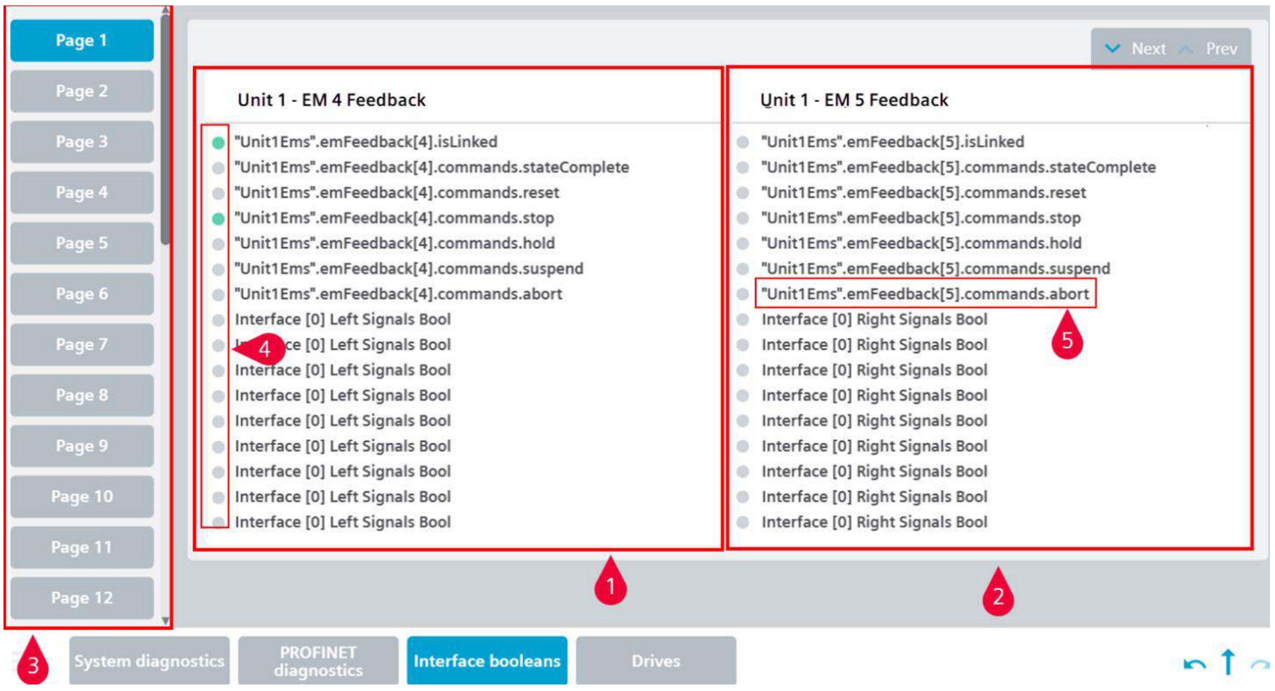
The corresponding WinCC screen is prepared for 16 datasets that can be navigated from "Page 1" to "Page 16", the number of pages can be configured as per the user requirements, see Engineering on how to configure the amount of pages to display. Each dataset contains 32 Boolean signals represented on a page split into two sides, see the figure bellow, "Left Page side"(1) and "Right Page side" (2), 16 Boolean signals are displayed on either side.

The displayed variables page is responsive to the selected page in the third navigation (3), allows the user to have a better overview of all configured 16 pages

The Status of each signal Bool is written cyclically on the PLC and displayed on the pages with an indicator light (4).

- Green indicator light for an active/ON signal.
- Lite grey for inactive /OFF signal.

The Symbolic names are pre-configured through Text lists on WinCC (see Engineering on how to configure the text lists) and are displayed on the pages with texts (5).



Unit 1 - EM 4 Feedback

"Unit1Ems".emFeedback[4].isLinked
"Unit1Ems".emFeedback[4].commands.stateComplete
"Unit1Ems".emFeedback[4].commands.reset
"Unit1Ems".emFeedback[4].commands.stop
"Unit1Ems".emFeedback[4].commands.hold
"Unit1Ems".emFeedback[4].commands.suspend
"Unit1Ems".emFeedback[4].commands.abort
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool
Interface [0] Left Signals Bool

Unit 1 - EM 5 Feedback

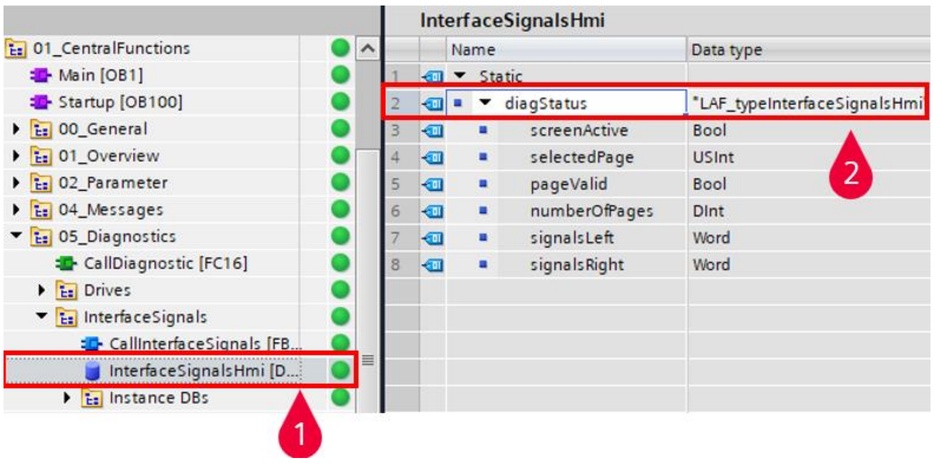
"Unit1Ems".emFeedback[5].isLinked
"Unit1Ems".emFeedback[5].commands.stateComplete
"Unit1Ems".emFeedback[5].commands.reset
"Unit1Ems".emFeedback[5].commands.stop
"Unit1Ems".emFeedback[5].commands.hold
"Unit1Ems".emFeedback[5].commands.suspend
"Unit1Ems".emFeedback[5].commands.abort
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool
Interface [0] Right Signals Bool

Figure 8-14: Diagnostic screen for the interface signals

8.6.3.2. Architecture

The monitoring system architecture is designed to convert individual Boolean signals into a compact WORD-based format for processing by the LAF_ReadWordSignals and LAF_ReadBoolSignals blocks. These WORD values are later unpacked back into Boolean signals for representation on the HMI.

The structure defined in the global DB InterfaceSignalHmi (see the figure bellow), located in the Diagnostics folder under the InterfaceSignals subfolder(1), serves as interface to the HMI and contains a primary member: "diagStatus" (2).



- 01_CentralFunctions
  - Main [OB1]
  - Startup [OB100]
  - 00_General
  - 01_Overview
  - 02_Parameter
  - 04_Messages
  - 05_Diagnostics
    - CallDiagnostic [FC16]
    - Drives
    - InterfaceSignals
      - CallInterfaceSignals [FB...
      - InterfaceSignalsHmi [D...
      - Instance DBs

InterfaceSignalsHmi

	Name	Data type
1	Static	
2	diagStatus	"LAF_typeInterfaceSignalsHmi
3	screenActive	Bool
4	selectedPage	USInt
5	pageValid	Bool
6	numberOfPages	DInt
7	signalsLeft	Word
8	signalsRight	Word

Figure 8-15: Interface Booleans Structure

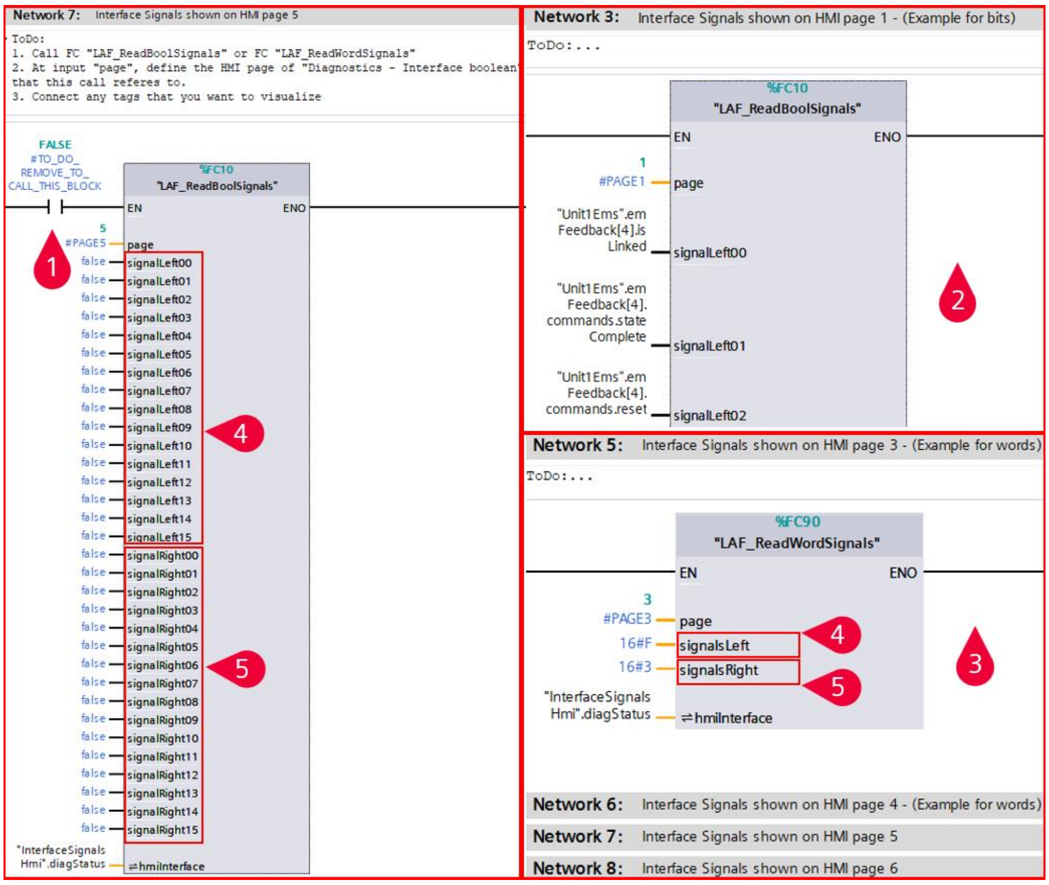
"diagStatus" is of type LAF_typeInterfaceSignalsHmi, containing six members: screenActive, selectedPage, pageValid, numberOfPages, signalsLeft, and signalsRight. Each serves a specific function in the operation and visualization of the interface signals:

- screenActive (BOOL): Enables the LAF_ReadBoolSignals and LAF_ReadWordSignals blocks to start processing the configured signals only when the "Interface Boolean" window on the HMI is active.
- selectedPage (USINT): Specifies which configured "page" is selected by the operator, allowing the system to process and display the correct set of signals for the page.
- pageValid (BOOL): Tracks if the selected page on the HMI is also configured on the PLC. True = Valid, False = Invalid
- numberOfPages (DINT): Stores the count of configured pages. The user configure the value of this tag as per Figure 8-17: Configuration of amounting pages(2) under Engineering subsection. It is then sent to the HMI to indicate how many interface pages are available for visualization.
- signalsLeft and signalsRight (WORD): Each WORD represents a set of 16 Booleans that can be displayed on either the left or right side of the active page on the HMI respectively.

8.6.3.3. Engineering

To configure the monitoring of interface signals on HMI, the following procedures need to be followed.

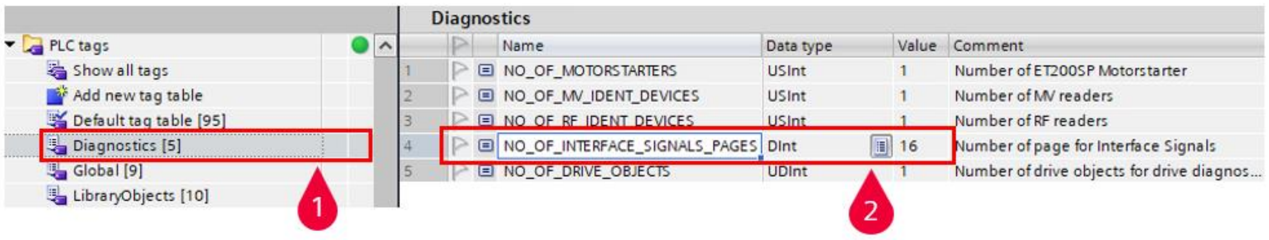
- Call FC "LAF_ReadBoolSignals" or FC "LAF_ReadWordSignals", depending on the signals format of the user. The AFB “CallInterfaceSignals” FB already contains 16 Networks, each with either FC "LAF_ReadBoolSignals" or FC "LAF_ReadWordSignals", the user then have to call their prefered ‘Read Signals’ block by enabling the block by removing the coil “TO_DO_REMOVE_TO_CALL_THIS_BLOCK” on the “EN” of the block interface, see the figure bellow (1).
- Connect the interface Signals selected tobe displaced on the HMI, signals to be displayed on the left screen of the page are connected on the “signalLeft#” block interface (4) and signals to be displayed on the right screen of the page are connected on the “signalRight#” block interface (5) . If the signals are Booleans, then FC "LAF_ReadBoolSignals" is called (2) and if the signals are WORD formatted, then FC “LAF_ReadWordSignals” is called (3).



```
ToDo:
1. Call FC "LAF_ReadBoolSignals" or FC "LAF_ReadWordSignals"
2. At input "page", define the HMI page of "Diagnostics - Interface boolean"
that this call referes to.
3. Connect any tags that you want to visualize
```

Figure 8-16: Call LAF_InterfaceSignals

- On the user constants of the “Diagnostics” under PLC tags, see the figure bellow (1). configure the amount of configured datasets / pages by entering the correct configured amount (2). This amount determines the number of pages can be navigated on the HMI Interface boolean window.



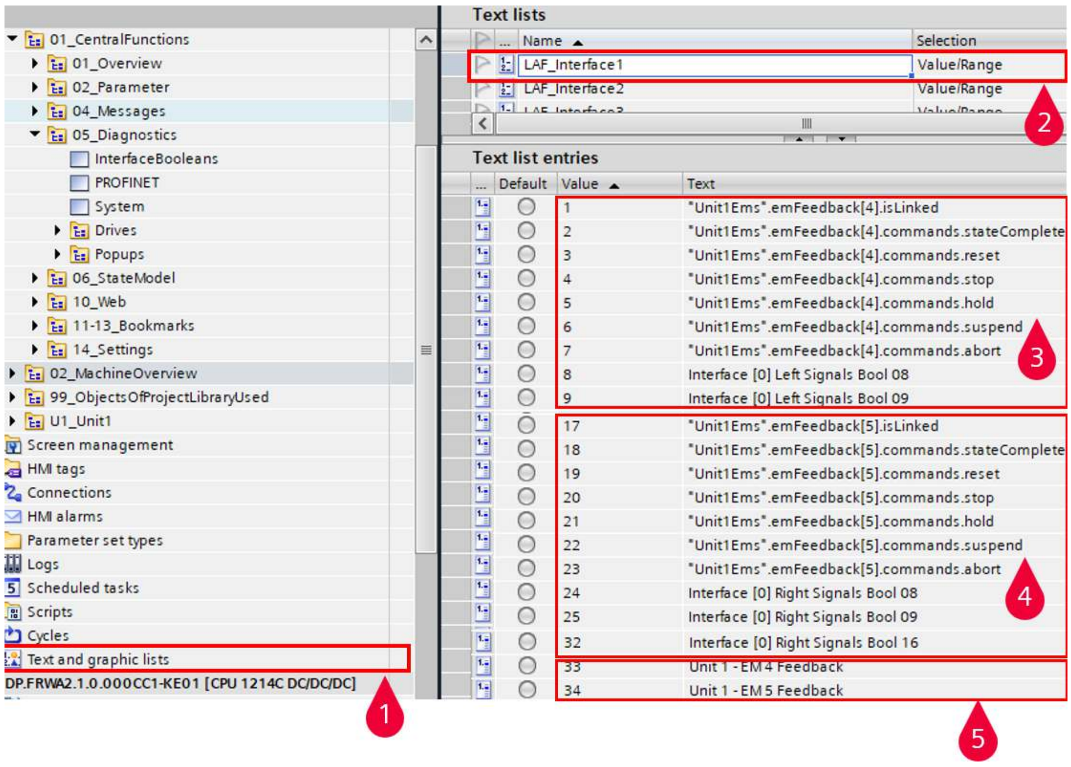
- PLC tags
  - Show all tags
  - Add new tag table
  - Default tag table [95]
  - Diagnostics [5]
  - Global [9]
  - LibraryObjects [10]

Diagnostics

	Name	Data type	Value	Comment
1	NO_OF_MOTORSTARTERS	USInt	1	Number of ET200SP Motorstarter
2	NO_OF_MV_IDENT_DEVICES	USInt	1	Number of MV readers
3	NO_OF_RF_IDENT_DEVICES	USInt	1	Number of RF readers
4	NO_OF_INTERFACE_SIGNALS_PAGES	DInt	16	Number of page for Interface Signals
5	NO_OF_DRIVE_OBJECTS	UDInt	1	Number of drive objects for drive diagnos...

Figure 8-17: Configuration of amounting pages

- To configure the symbolic names of the interface signals to be displayed with the status as mentioned on Overview, the user needs to navigate to “LAF_Interface#” Text list for the page they want to display the configured Boolean signals, under “Text and graphic lidts”, (see the figure bellow (1)), on the WinCC, for each page (Represented by “LAF_Interface#” (2), there are 34 text list entries, with entry 1 to 16 reserved for Left signal Bools (3), entry 17 to 32 reserved for Right signals (4) , and entry 33 and 34 reserved for for Left and Right dataset descriptions respectively (5). The configuration example on the bellow figure, produces the results displyed on Figure 8-14: Diagnostic screen for the interface signals under the subsection Overview.



- 01_CentralFunctions
  - 01_Overview
  - 02_Parameter
  - 04_Messages
  - 05_Diagnostics
    - InterfaceBooleans
    - PROFINET
    - System
    - Drives
    - Popups
  - 06_StateModel
  - 10_Web
  - 11-13_Bookmarks
  - 14_Settings
- 02_MachineOverview
- 99_ObjectsOfProjectLibraryUsed
- U1_Unit1
- Screen management
- HMI tags
- Connections
- HMI alarms
- Parameter set types
- Logs
- Scheduled tasks
- Scripts
- Cycles
- Text and graphic lists

DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC]

Text lists

Name	Selection
LAF_Interface1	Value/Range
LAF_Interface2	Value/Range
LAF_Interface3	Value/Range

Text list entries

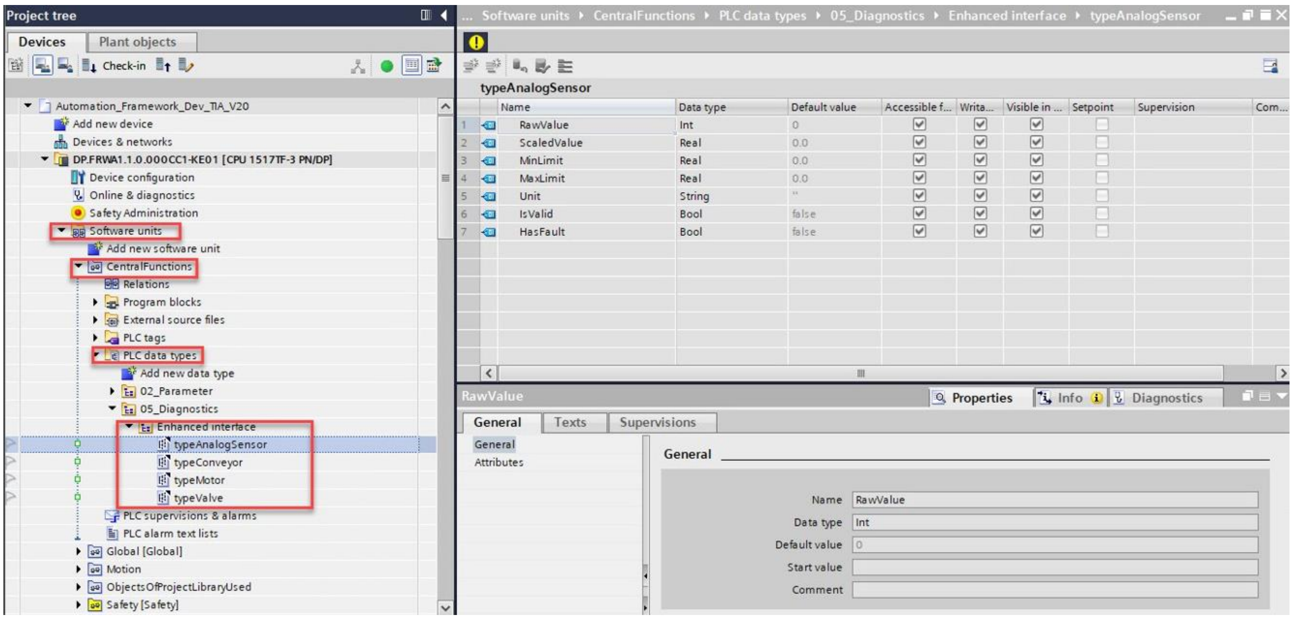
Default	Value	Text
○	1	"Unit1Ems".emFeedback[4].isLinked
○	2	"Unit1Ems".emFeedback[4].commands.stateComplete
○	3	"Unit1Ems".emFeedback[4].commands.reset
○	4	"Unit1Ems".emFeedback[4].commands.stop
○	5	"Unit1Ems".emFeedback[4].commands.hold
○	6	"Unit1Ems".emFeedback[4].commands.suspend
○	7	"Unit1Ems".emFeedback[4].commands.abort
○	8	Interface [0] Left Signals Bool 08
○	9	Interface [0] Left Signals Bool 09
○	17	"Unit1Ems".emFeedback[5].isLinked
○	18	"Unit1Ems".emFeedback[5].commands.stateComplete
○	19	"Unit1Ems".emFeedback[5].commands.reset
○	20	"Unit1Ems".emFeedback[5].commands.stop
○	21	"Unit1Ems".emFeedback[5].commands.hold
○	22	"Unit1Ems".emFeedback[5].commands.suspend
○	23	"Unit1Ems".emFeedback[5].commands.abort
○	24	Interface [0] Right Signals Bool 08
○	25	Interface [0] Right Signals Bool 09
○	32	Interface [0] Right Signals Bool 16
○	33	Unit 1 - EM4 Feedback
○	34	Unit 1 - EM5 Feedback

Figure 8-18: Interface Boolean text configuration

8.6.4. Enhanced Interface

The Enhanced Interfaces are set up in the project to deliver to the final user another option of having in a dynamic environment the parameter control to be more flexible. Here is how they are implemented in the project.

- The UDTs are created for the different things that you want to display (PLC data types). Remember to make them HMI accessible, writable and visible.



... Software units ▸ CentralFunctions ▸ PLC data types ▸ 05_Diagnostics ▸ Enhanced interface ▸ typeAnalogSensor

typeAnalogSensor

	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Com...
1	RawValue	Int	0	✓	✓	✓	☐		
2	ScaledValue	Real	0.0	✓	✓	✓	☐		
3	MinLimit	Real	0.0	✓	✓	✓	☐		
4	MaxLimit	Real	0.0	✓	✓	✓	☐		
5	Unit	String	''	✓	✓	✓	☐		
6	IsValid	Bool	false	✓	✓	✓	☐		
7	HasFault	Bool	false	✓	✓	✓	☐		

RawValue — General

Name	RawValue
Data type	Int
Default value	0
Start value	
Comment	

Figure 8-19: Enchanced Udts

- Types are created in the library out of the PLC data type:

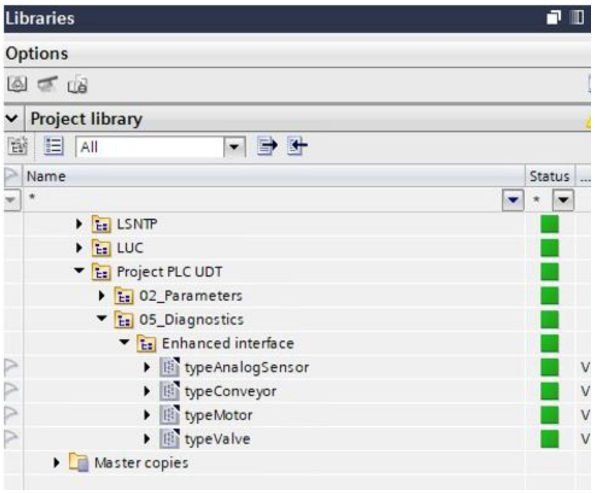
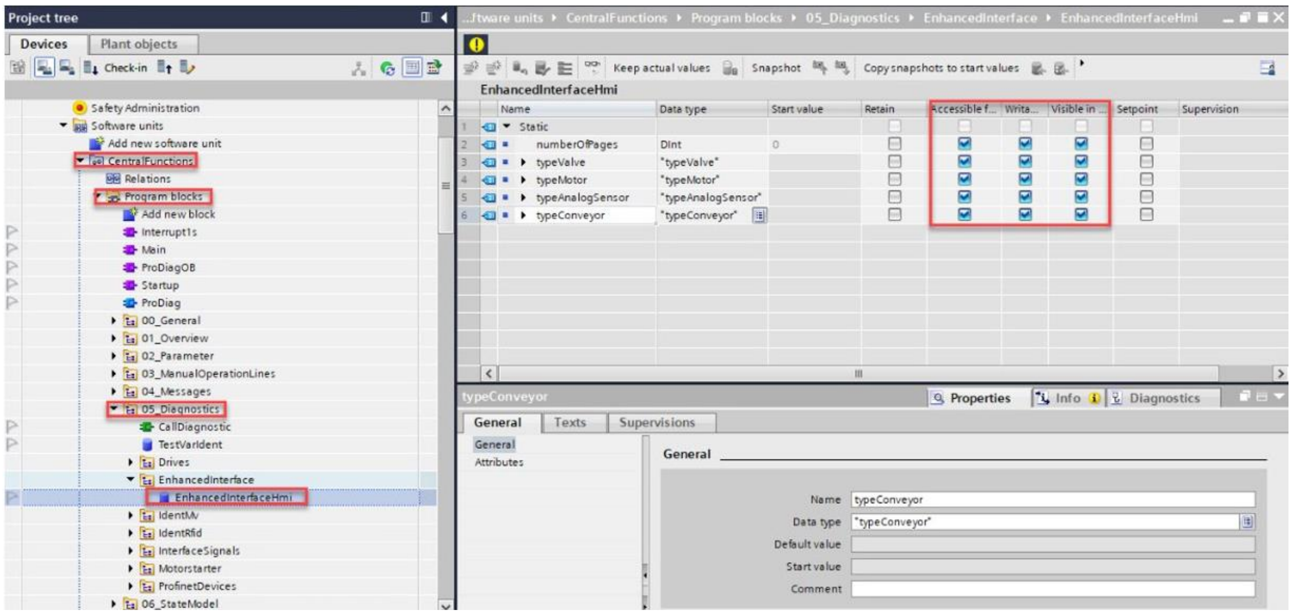


Figure 8-20: Library Udts for the Enhanced

- The corresponding DBs are created. Remember to make them HMI accessible, writable, visible.



...ftware units ▸ CentralFunctions ▸ Program blocks ▸ 05_Diagnostics ▸ EnhancedInterface ▸ EnhancedInterfaceHmi

EnhancedInterfaceHmi

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision
1	▾ Static			☐	☐	☐	☐	☐	
2	numberOfPages	DInt	0	☐	✓	✓	✓	☐	
3	▸ typeValve	"typeValve"		☐	✓	✓	✓	☐	
4	▸ typeMotor	"typeMotor"		☐	✓	✓	✓	☐	
5	▸ typeAnalogSensor	"typeAnalogSensor"		☐	✓	✓	✓	☐	
6	▸ typeConveyor	"typeConveyor"		☐	✓	✓	✓	☐	

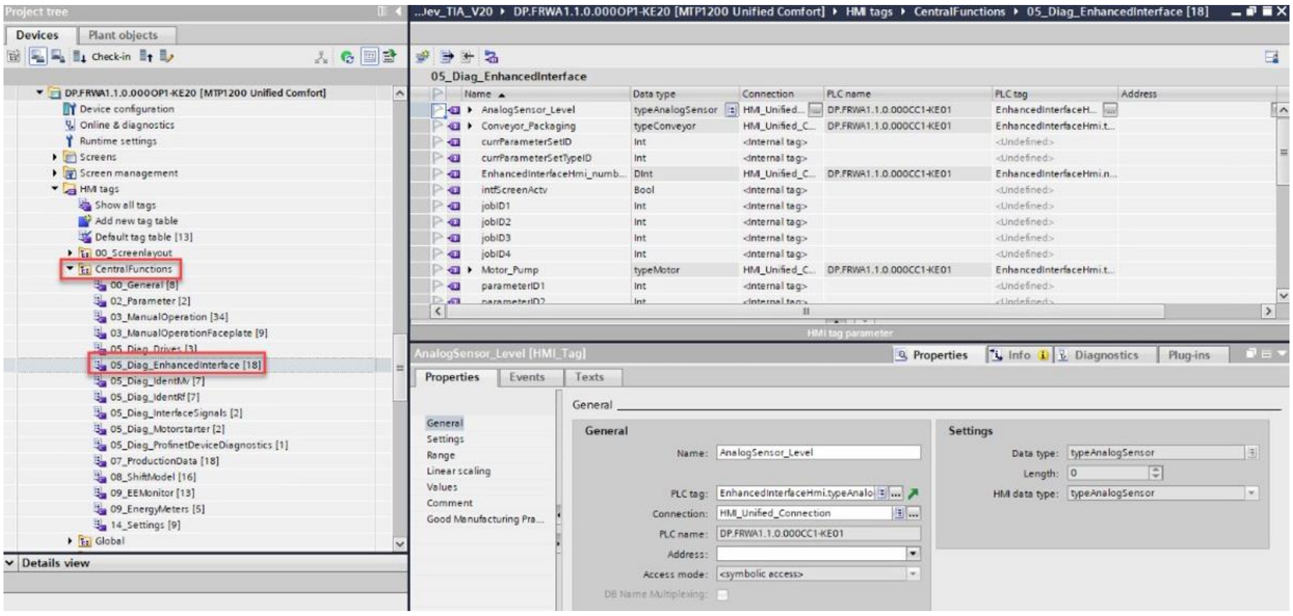
typeConveyor

General

Name	typeConveyor
Data type	"typeConveyor"
Default value	
Start value	
Comment	

Figure 8-21: DBs for Enhanced

- The corresponding tags in the HMI are created:



...Jev_TIA_V20 ▸ DP.FRWA1.1.0.000OP1-KE20 [MTP1200 Unified Comfort] ▸ HMI tags ▸ CentralFunctions ▸ 05_Diag_EnhancedInterface [18]

05_Diag_EnhancedInterface

Name	Data type	Connection	PLC name	PLC tag	Address
▸ AnalogSensor_Level	typeAnalogSensor	HMI_Unified...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceH...	
▸ Conveyor_Packaging	typeConveyor	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.t...	
currParameterSetID	Int	<Internal tag>		<Undefined>	
currParameterSetTypeID	Int	<Internal tag>		<Undefined>	
EnhancedInterfaceHmi_numb...	DInt	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.n...	
intfScreenActv	Bool	<Internal tag>		<Undefined>	
jobID1	Int	<Internal tag>		<Undefined>	
jobID2	Int	<Internal tag>		<Undefined>	
jobID3	Int	<Internal tag>		<Undefined>	
jobID4	Int	<Internal tag>		<Undefined>	
▸ Motor_Pump	typeMotor	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.t...	
parameterID1	Int	<Internal tag>		<Undefined>	
parameterID2	Int	<Internal tag>		<Undefined>	

AnalogSensor_Level [HMI_Tag]

General

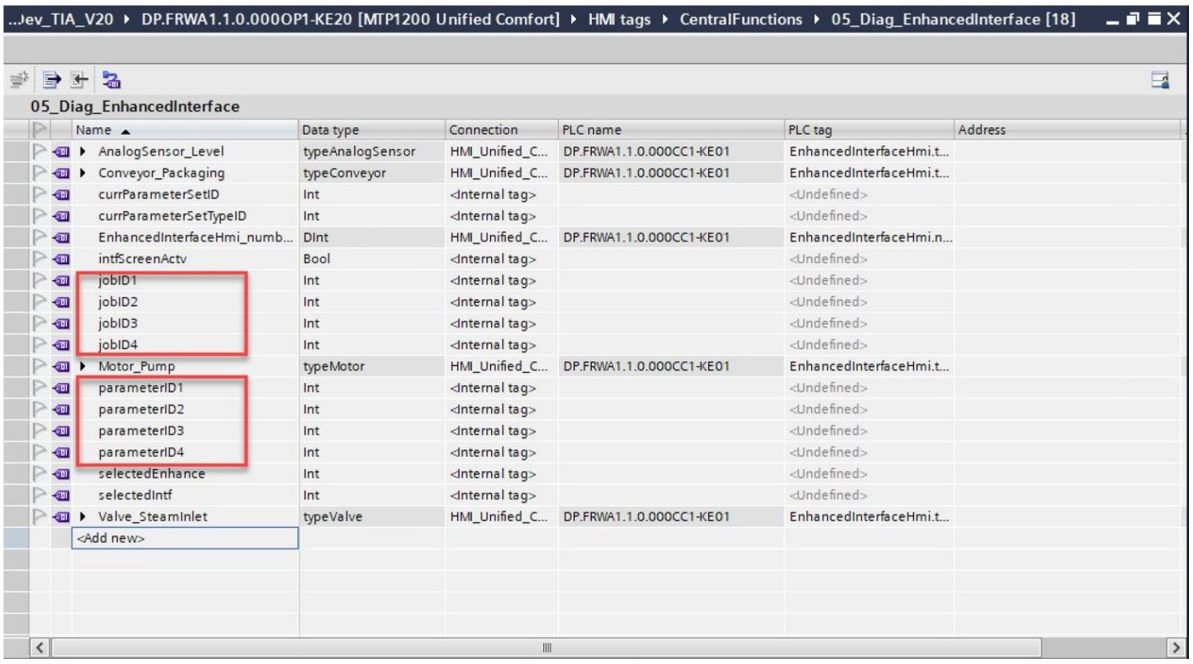
Name:	AnalogSensor_Level
PLC tag:	EnhancedInterfaceHmi.typeAnalo
Connection:	HMI_Unified_Connection
PLC name:	DP.FRWA1.1.0.000CC1-KE01
Address:	
Access mode:	<symbolic access>

Settings

Data type:	typeAnalogSensor
Length:	0
HMI data type:	typeAnalogSensor

Figure 8-22: Tags for the HMI Enchanced

- Tags [int] are created and called "JobID+n" and "ParameterID+n" for each of the displayed UDTs.



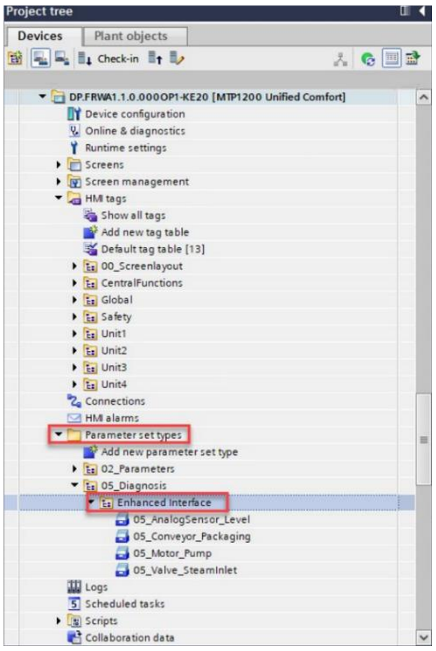
..Jev_TIA_V20 ▸ DP.FRWA1.1.0.000OP1-KE20 [MTP1200 Unified Comfort] ▸ HMI tags ▸ CentralFunctions ▸ 05_Diag_EnhancedInterface [18]

05_Diag_EnhancedInterface

Name	Data type	Connection	PLC name	PLC tag	Address
AnalogSensor_Level	typeAnalogSensor	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.t...	
Conveyor_Packaging	typeConveyor	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.t...	
currParameterSetID	Int	<Internal tag>		<Undefined>	
currParameterSetTypeID	Int	<Internal tag>		<Undefined>	
EnhancedInterfaceHmi_numb...	DInt	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.n...	
intfScreenActv	Bool	<Internal tag>		<Undefined>	
jobID1	Int	<Internal tag>		<Undefined>	
jobID2	Int	<Internal tag>		<Undefined>	
jobID3	Int	<Internal tag>		<Undefined>	
jobID4	Int	<Internal tag>		<Undefined>	
Motor_Pump	typeMotor	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.t...	
parameterID1	Int	<Internal tag>		<Undefined>	
parameterID2	Int	<Internal tag>		<Undefined>	
parameterID3	Int	<Internal tag>		<Undefined>	
parameterID4	Int	<Internal tag>		<Undefined>	
selectedEnhance	Int	<Internal tag>		<Undefined>	
selectedIntf	Int	<Internal tag>		<Undefined>	
Valve_SteamInlet	typeValve	HMI_Unified_C...	DP.FRWA1.1.0.000CC1-KE01	EnhancedInterfaceHmi.t...	
<Add new>					

Figure 8-23: Names for the Enhanced jobs

- Parameter set type is created:
  - Select the UDT
  - Select the tag
  - Select the control tags
  - JobID+n
  - ParameterID+n



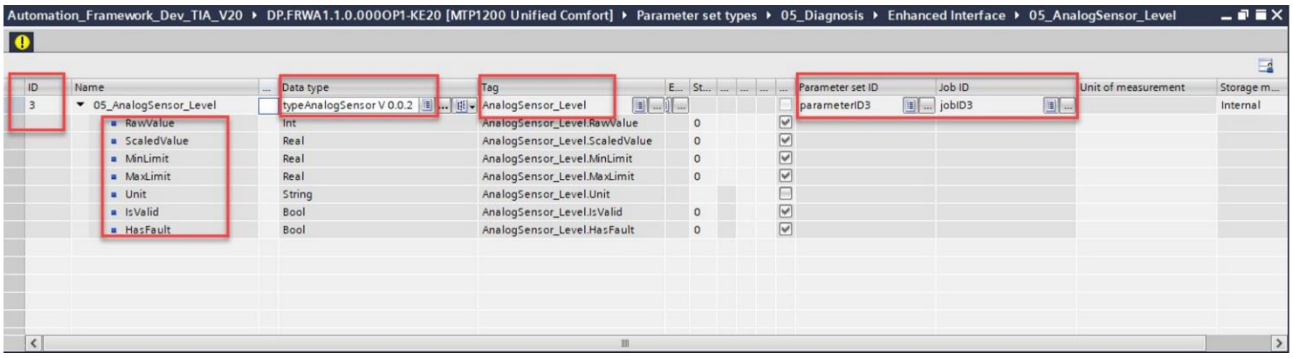
Project tree

Devices | Plant objects

Check-in

```
▾ DP.FRWA1.1.0.000OP1-KE20 [MTP1200 Unified Comfort]
    Device configuration
    Online & diagnostics
    Runtime settings
  ▸ Screens
  ▸ Screen management
  ▾ HMI tags
      Show all tags
      Add new tag table
      Default tag table [13]
    ▸ 00_Screenlayout
    ▸ CentralFunctions
    ▸ Global
    ▸ Safety
    ▸ Unit1
    ▸ Unit2
    ▸ Unit3
    ▸ Unit4
    Connections
    HMI alarms
  ▾ Parameter set types
      Add new parameter set type
    ▸ 02_Parameters
    ▾ 05_Diagnosis
      ▾ Enhanced Interface
          05_AnalogSensor_Level
          05_Conveyor_Packaging
          05_Motor_Pump
          05_Valve_SteamInlet
    Logs
    Scheduled tasks
  ▸ Scripts
    Collaboration data
```

Figure 8-24: Parameter set for the Enchanced

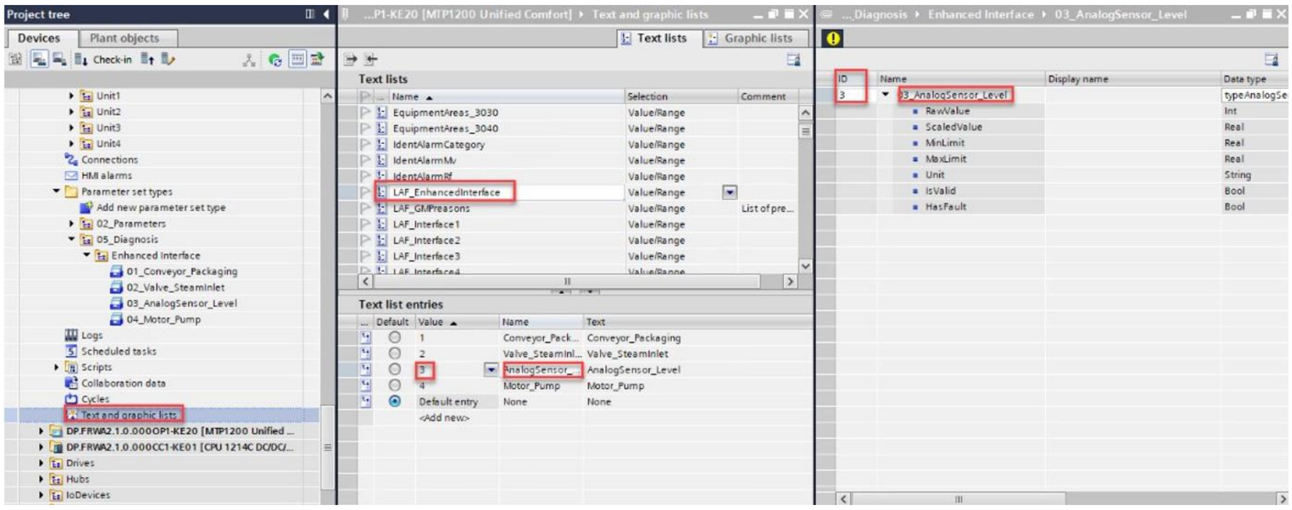


Automation_Framework_Dev_TIA_V20 ▸ DP.FRWA1.1.0.000OP1-KE20 [MTP1200 Unified Comfort] ▸ Parameter set types ▸ 05_Diagnosis ▸ Enhanced Interface ▸ 05_AnalogSensor_Level

ID	Name	Data type	Tag	St...		Parameter set ID	Job ID	Unit of measurement	Storage m...
3	05_AnalogSensor_Level	typeAnalogSensor V 0.0.2	AnalogSensor_Level		☐	parameterID3	jobID3		Internal
	RawValue	Int	AnalogSensor_Level.RawValue	0	✓				
	ScaledValue	Real	AnalogSensor_Level.ScaledValue	0	✓				
	MinLimit	Real	AnalogSensor_Level.MinLimit	0	✓				
	MaxLimit	Real	AnalogSensor_Level.MaxLimit	0	✓				
	Unit	String	AnalogSensor_Level.Unit		☐				
	IsValid	Bool	AnalogSensor_Level.IsValid	0	✓				
	HasFault	Bool	AnalogSensor_Level.HasFault	0	✓				

Figure 8-25: Parameter set settings for the Enhanced

- Textlists are created for the resource list (IMPORTANT: Number in the textlist = ID Parameter Set Type)



...P1-KE20 [MTP1200 Unified Comfort] ▸ Text and graphic lists

Text lists

Name	Selection	Comment
EquipmentAreas_3030	Value/Range	
EquipmentAreas_3040	Value/Range	
IdentAlarmCategory	Value/Range	
IdentAlarmMv	Value/Range	
IdentAlarmRf	Value/Range	
LAF_EnhancedInterface	Value/Range	
LAF_GMPreasons	Value/Range	List of pre...
LAF_Interface1	Value/Range	
LAF_Interface2	Value/Range	
LAF_Interface3	Value/Range	
LAF_Interface4	Value/Range	

Text list entries

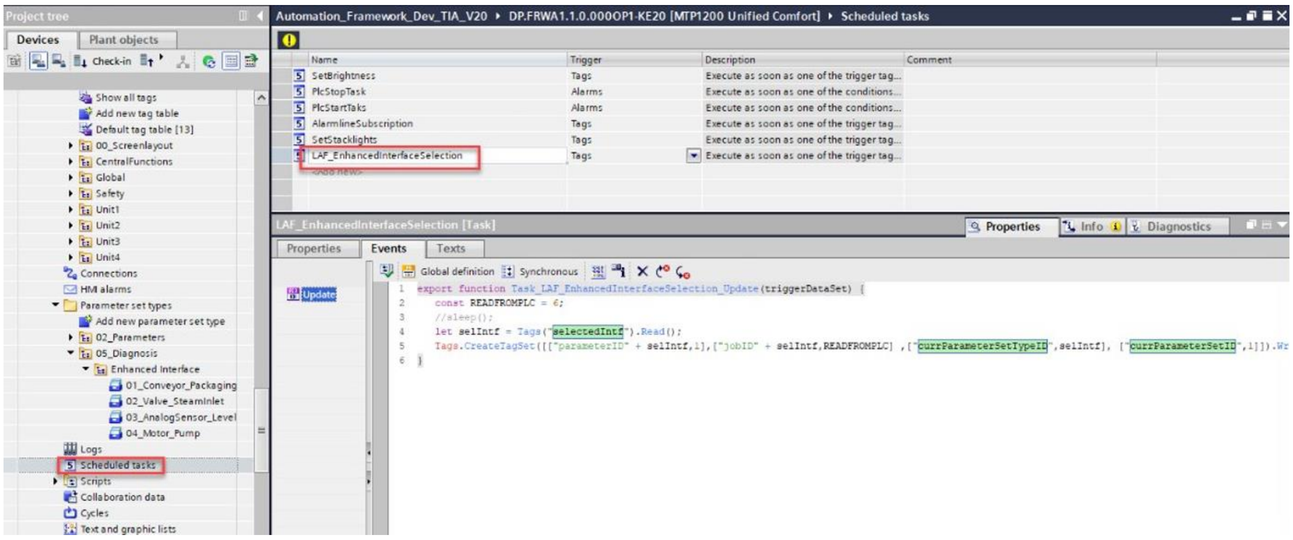
Default	Value	Name	Text
○	1	Conveyor_Pack...	Conveyor_Packaging
○	2	Valve_SteamInl...	Valve_SteamInlet
○	3	AnalogSensor_...	AnalogSensor_Level
○	4	Motor_Pump	Motor_Pump
●	Default entry	None	None
	<Add new>		

...Diagnosis ▸ Enhanced Interface ▸ 03_AnalogSensor_Level

ID	Name	Display name	Data type
3	03_AnalogSensor_Level		typeAnalogSe
	RawValue		Int
	ScaledValue		Real
	MinLimit		Real
	MaxLimit		Real
	Unit		String
	IsValid		Bool
	HasFault		Bool

Figure 8-26: Text lists for the Enhanced

- The screen and the schedule task are implemented already in the project.



Automation_Framework_Dev_TIA_V20 ▸ DP.FRWA1.1.0.000OP1-KE20 [MTP1200 Unified Comfort] ▸ Scheduled tasks

Name	Trigger	Description	Comment
SetBrightness	Tags	Execute as soon as one of the trigger tag...	
PlcStopTask	Alarms	Execute as soon as one of the conditions...	
PlcStartTaks	Alarms	Execute as soon as one of the conditions...	
AlarmlineSubscription	Tags	Execute as soon as one of the trigger tag...	
SetStacklights	Tags	Execute as soon as one of the trigger tag...	
LAF_EnhancedInterfaceSelection	Tags	Execute as soon as one of the trigger tag...	

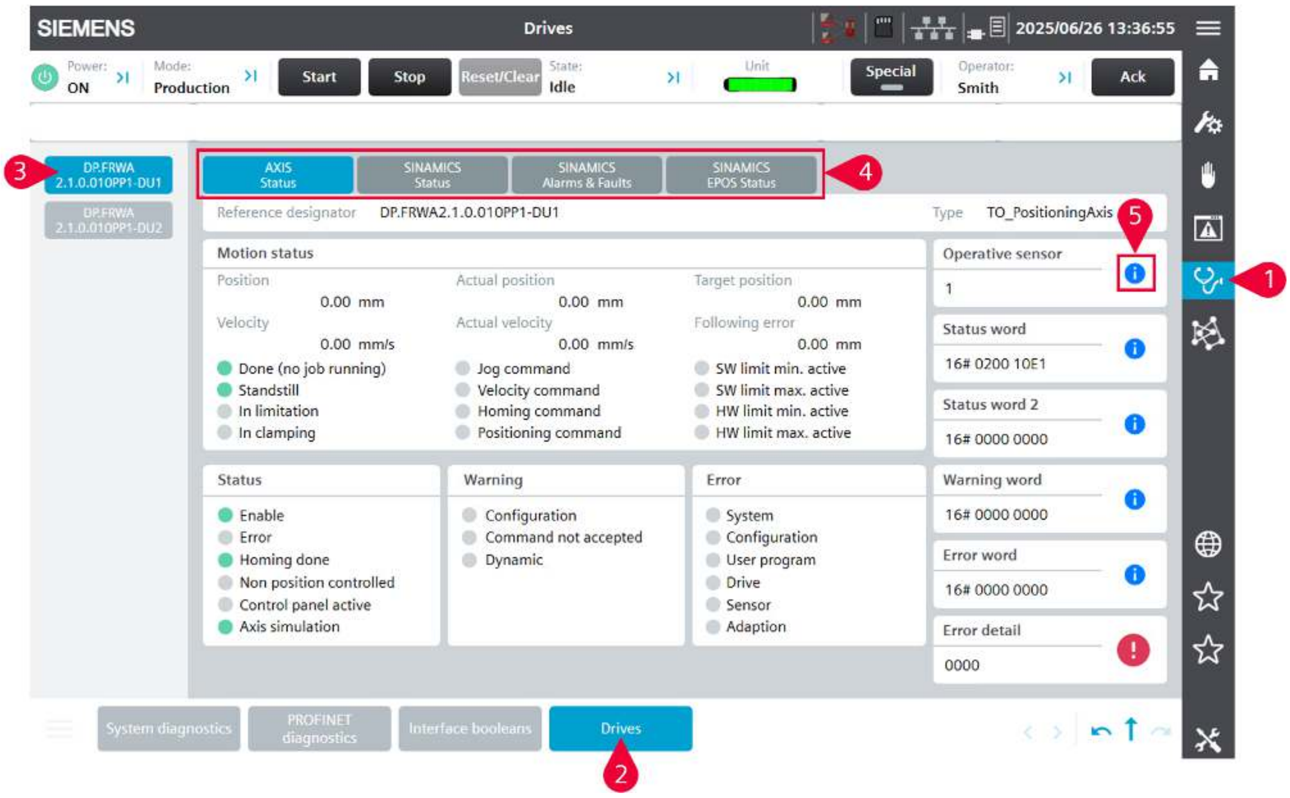
LAF_EnhancedInterfaceSelection [Task]

```
export function Task_LAF_EnhancedInterfaceSelection_Update(triggerDataSet) {
   const READFROMPLC = 6;
   //sleep();
   let selIntf = Tags("selectedIntf").Read();
   Tags.CreateTagSet([["parameterID" + selIntf,1],["jobID" + selIntf,READFROMPLC] ,["currParameterSetTypeID",selIntf], ["currParameterSetID",1]]).Wr
}
```

Figure 8-27: Schedule Tasks for the Enchanced

8.6.5. 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.



Power: ON
Mode: Production
State: Idle
Unit
Operator: Smith

Reference designator DP.FRWA2.1.0.010PP1-DU1
Type TO_PositioningAxis

Motion status

Position	Actual position	Target position
0.00 mm	0.00 mm	0.00 mm
Velocity	**Actual velocity**	**Following error**
0.00 mm/s	0.00 mm/s	0.00 mm
● Done (no job running)	○ Jog command	○ SW limit min. active
● Standstill	○ Velocity command	○ SW limit max. active
○ In limitation	○ Homing command	○ HW limit min. active
○ In clamping	○ Positioning command	○ HW limit max. active

Status	Warning	Error
● Enable	○ Configuration	○ System
○ Error	○ Command not accepted	○ Configuration
● Homing done	○ Dynamic	○ User program
○ Non position controlled		○ Drive
○ Control panel active		○ Sensor
● Axis simulation		○ Adaption

Operative sensor	1
Status word	16# 0200 10E1
Status word 2	16# 0000 0000
Warning word	16# 0000 0000
Error word	16# 0000 0000
Error detail	0000

Figure 8-28: 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).

8.6.5.1. 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 “DriveDiagHmi”	HMI interface for drive diagnostics.	-
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.	-

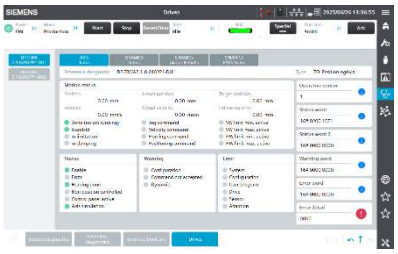
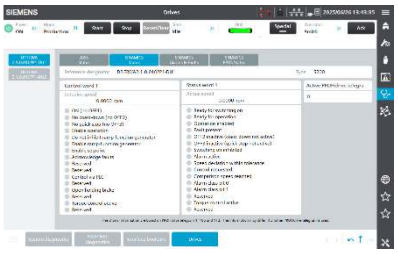
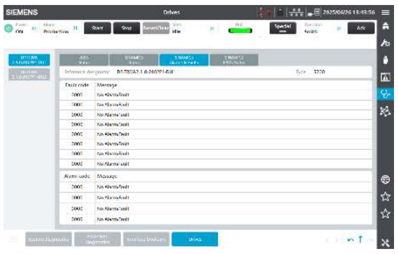
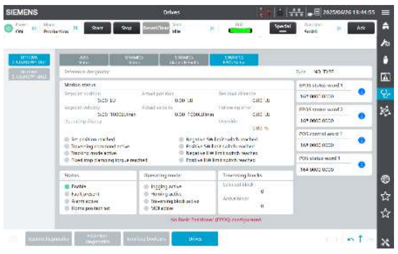
Block	Functionality	Screen
FB “LAF_GetAxisStatus”	This block reads diagnostic information of an axis depending on the configured technology object and the selected drive object index in the HMI by the operator.	“AxisStatus” 
FB “LAF_GetSinaStatus”	This block reads PROFIdrive telegram status word 1 (ZSW1) 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

8.6.5.2. 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 data block “DriveDiagHmi” (2). The function block “LAF_GetSinaEpos” (3) is enabled by the active “SinamicsEpos” screen. This function block reads the configuration from the data block “DriveDiagConfig” (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 function “LAF_SetDriveObjectConfig” (A) sets the configuration for one drive diagnostic object in the data block “DriveDiagConfig” during PLC startup.

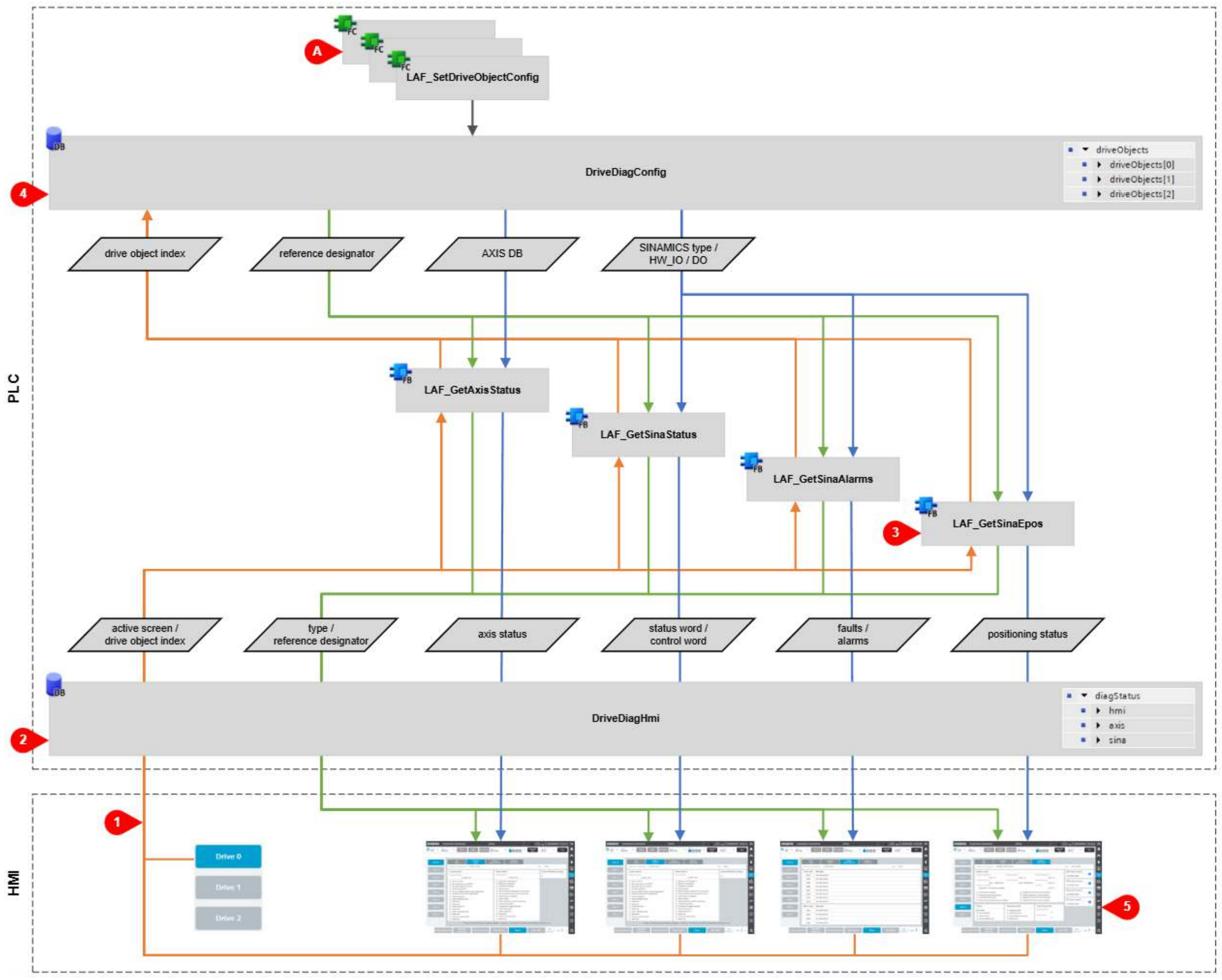
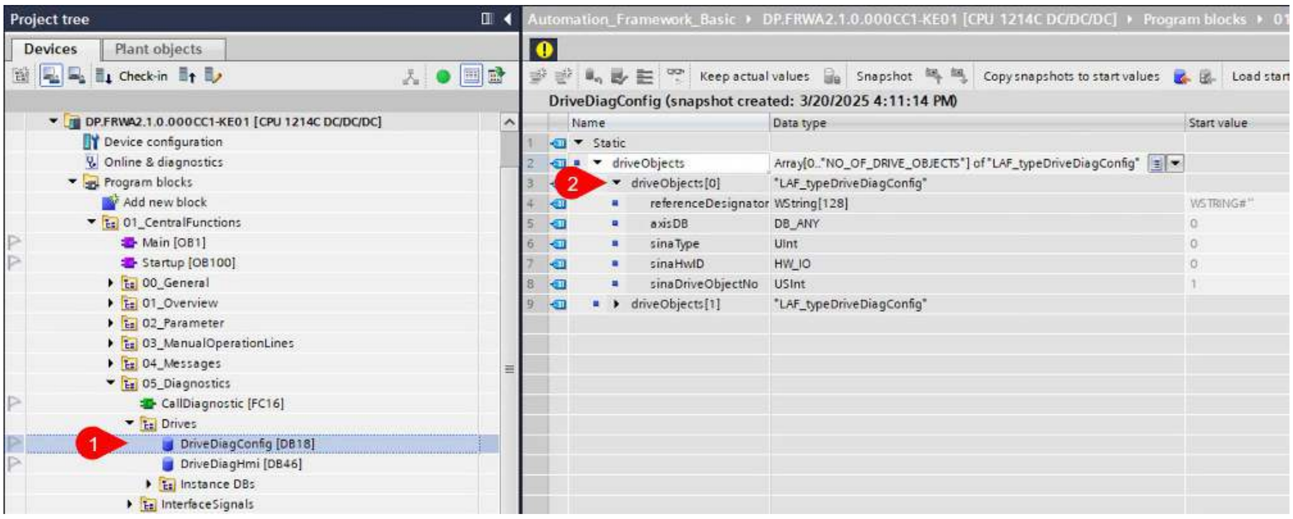


Figure 8-29: Big picture of the drive diagnostics architecture

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” (1). Each drive diagnostics object has its own array element (2).



DriveDiagConfig (snapshot created: 3/20/2025 4:11:14 PM)

	Name	Data type	Start value
1	Static		
2	driveObjects	Array[0.."NO_OF_DRIVE_OBJECTS"] of "LAF_typeDriveDiagConfig"	
3	driveObjects[0]	"LAF_typeDriveDiagConfig"	
4	referenceDesignator	WString[128]	WSTRING#''
5	axisDB	DB_ANY	0
6	sinaType	UInt	0
7	sinaHwID	HW_IO	0
8	sinaDriveObjectNo	USInt	1
9	driveObjects[1]	"LAF_typeDriveDiagConfig"	

Figure 8-30: Structure of the configuration data block “DriveDiagConfig”

NOTE

To reduce system load, it is recommended to set the configuration of each drive diagnostics object with the “LAF_SetDriveObjectConfig” function during PLC startup.

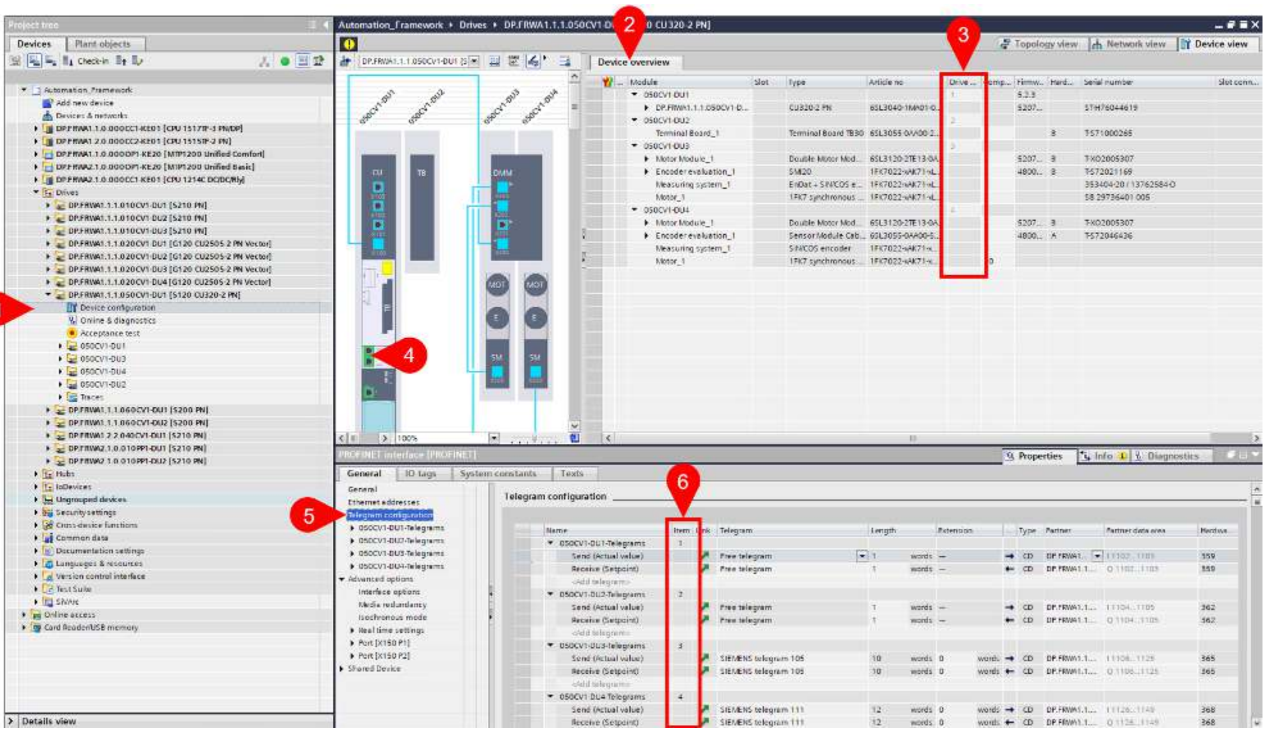
Each drive diagnostics object consists of the following tags:

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
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 S120. 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.



Project tree:

- Automation_Framework
  - Add new device
  - Devices & networks
  - DP.FRWA1.1.0.000CC1-KE01 [CPU 1517TF-3 PN/DP]
  - DP.FRWA1.2.0.000CC2-KE01 [CPU 1515TF-2 PN]
  - DP.FRWA1.1.0.000OP1-KE20 [MTP1200 Unified Comfort]
  - DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic]
  - DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/Rly]
  - Drives
    - DP.FRWA1.1.1.010CV1-DU1 [S210 PN]
    - DP.FRWA1.1.1.010CV1-DU2 [S210 PN]
    - DP.FRWA1.1.1.010CV1-DU3 [S210 PN]
    - DP.FRWA1.1.1.020CV1-DU1 [G120 CU250S-2 PN Vector]
    - DP.FRWA1.1.1.020CV1-DU2 [G120 CU250S-2 PN Vector]
    - DP.FRWA1.1.1.020CV1-DU3 [G120 CU250S-2 PN Vector]
    - DP.FRWA1.1.1.020CV1-DU4 [G120 CU250S-2 PN Vector]
    - DP.FRWA1.1.1.050CV1-DU1 [S120 CU320-2 PN]
      - Device configuration
      - Online & diagnostics
      - Acceptance test
      - 050CV1-DU1
      - 050CV1-DU3
      - 050CV1-DU4
      - 050CV1-DU2
      - Traces
    - DP.FRWA1.1.1.060CV1-DU1 [S200 PN]
    - DP.FRWA1.1.1.060CV1-DU2 [S200 PN]
    - DP.FRWA1.2.2.040CV1-DU1 [S210 PN]
    - DP.FRWA2.1.0.010PP1-DU1 [S210 PN]
    - DP.FRWA2.1.0.010PP1-DU2 [S210 PN]
  - Hubs
  - IoDevices
  - Ungrouped devices
  - Security settings
  - Cross-device functions
  - Common data
  - Documentation settings
  - Languages & resources
  - Version control interface
  - Test Suite
  - SIVArc
- Online access
- Card Reader/USB memory

Device overview:

Module	Slot	Type	Article no	Drive ...	Comp...	Firmw...	Hard...	Serial number	Slot conn...
050CV1-DU1				1		5.2.3			
DP.FRWA1.1.1.050CV1-D...		CU320-2 PN	6SL3040-1MA01-0...			5207...		ST-H76044619	
050CV1-DU2				2					
Terminal Board_1		Terminal Board TB30	6SL3055-0AA00-2...				B	T-S71000265	
050CV1-DU3				3					
Motor Module_1		Double Motor Mod...	6SL3120-2TE13-0A...			5207...	B	T-XO2005307	
Encoder evaluation_1		SMI20	1FK7022-xAK71-xL...			4800...	B	T-S72021169	
Measuring system_1		EnDat + SIN/COS e...	1FK7022-xAK71-xL...					353404-20 / 13762584-O	
Motor_1		1FK7 synchronous ...	1FK7022-xAK71-xL...					58 29736401 005	
050CV1-DU4				4					
Motor Module_1		Double Motor Mod...	6SL3120-2TE13-0A...			5207...	B	T-XO2005307	
Encoder evaluation_1		Sensor Module Cab...	6SL3055-0AA00-5...			4800...	A	T-S72046436	
Measuring system_1		SIN/COS encoder	1FK7022-xAK71-x...						
Motor_1		1FK7 synchronous ...	1FK7022-xAK71-x...		0				

PROFINET interface properties:

- General
- Ethernet addresses
- Telegram configuration
  - 050CV1-DU1-Telegrams
  - 050CV1-DU2-Telegrams
  - 050CV1-DU3-Telegrams
  - 050CV1-DU4-Telegrams
- Advanced options
  - Interface options
  - Media redundancy
  - Isochronous mode
  - Real time settings
  - Port [X150 P1]
  - Port [X150 P2]
- Shared Device

Telegram configuration

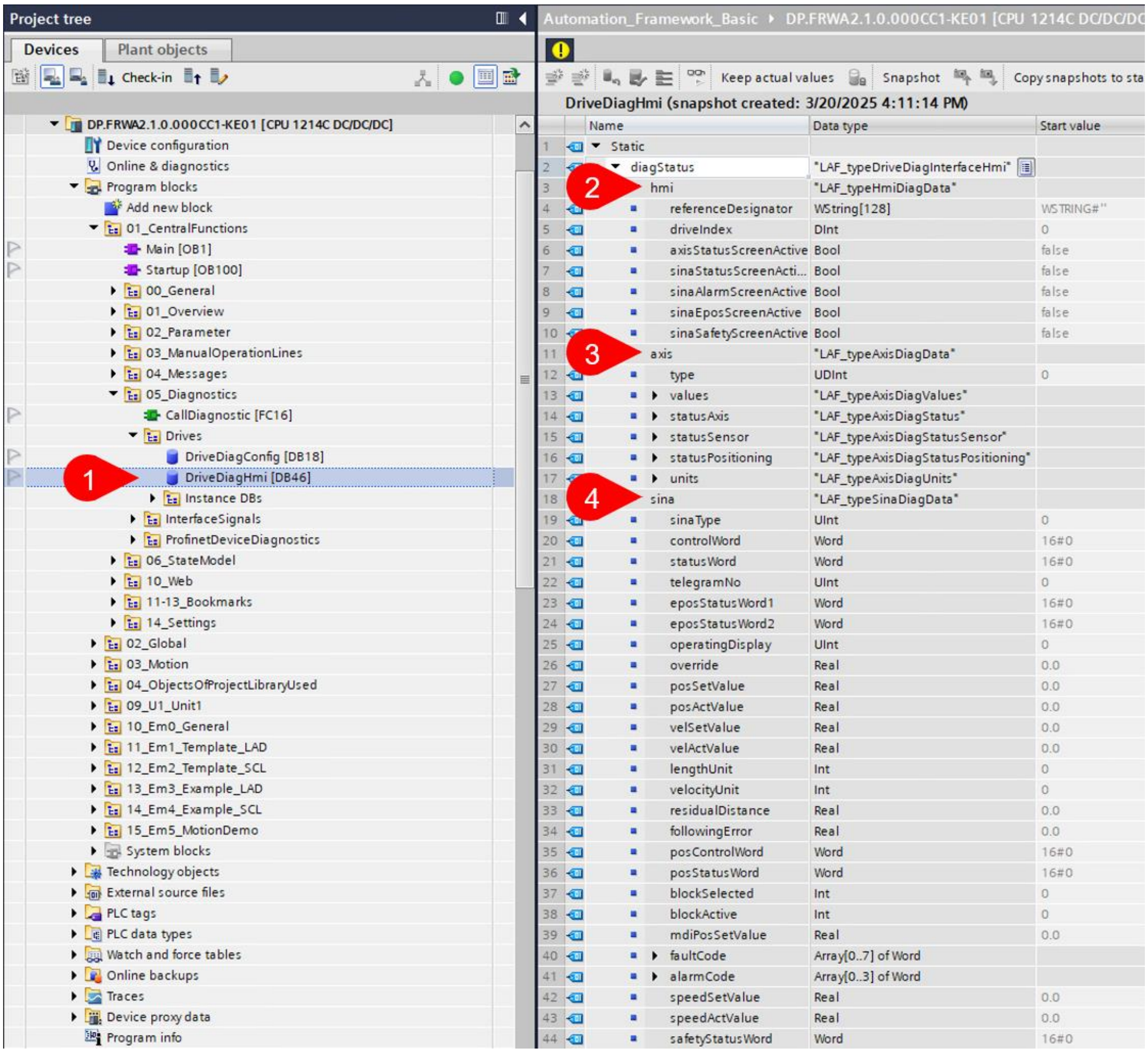
Name	Item	Link	Telegram	Length		Extension			Type	Partner	Partner data area	Hardwa...
050CV1-DU1-Telegrams	1											
Send (Actual value)			Free telegram	1	words	–		→	CD	DP.FRWA1...	I 1102...1103	359
Receive (Setpoint)			Free telegram	1	words	–		←	CD	DP.FRWA1.1...	Q 1102...1103	359
<Add telegram>												
050CV1-DU2-Telegrams	2											
Send (Actual value)			Free telegram	1	words	–		→	CD	DP.FRWA1.1...	I 1104...1105	362
Receive (Setpoint)			Free telegram	1	words	–		←	CD	DP.FRWA1.1...	Q 1104...1105	362
<Add telegram>												
050CV1-DU3-Telegrams	3											
Send (Actual value)			SIEMENS telegram 105	10	words	0	words	→	CD	DP.FRWA1.1...	I 1106...1125	365
Receive (Setpoint)			SIEMENS telegram 105	10	words	0	words	←	CD	DP.FRWA1.1...	Q 1106...1125	365
<Add telegram>												
050CV1-DU4-Telegrams	4											
Send (Actual value)			SIEMENS telegram 111	12	words	0	words	→	CD	DP.FRWA1.1...	I 1126...1149	368
Receive (Setpoint)			SIEMENS telegram 111	12	words	0	words	←	CD	DP.FRWA1.1...	Q 1126...1149	368

Figure 8-31: Access drive object number in SINAMICS S120 multi-axis system

HMI interface

The interface between PLC and HMI is the data block “DriveDiagHmi” (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 function block “LAF_GetAxisStatus”.
- Sinamics status information (4), which are filled by the function blocks “LAF_GetSinaStatus”, “LAF_GetSinaAlarms” and “LAF_GetSinaEpos”.



- DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC]
  - Device configuration
  - Online & diagnostics
  - Program blocks
    - Add new block
    - 01_CentralFunctions
      - Main [OB1]
      - Startup [OB100]
      - 00_General
      - 01_Overview
      - 02_Parameter
      - 03_ManualOperationLines
      - 04_Messages
      - 05_Diagnostics
        - CallDiagnostic [FC16]
        - Drives
          - DriveDiagConfig [DB18]
          - DriveDiagHmi [DB46]
          - Instance DBs
        - InterfaceSignals
        - ProfinetDeviceDiagnostics
      - 06_StateModel
      - 10_Web
      - 11-13_Bookmarks
      - 14_Settings
    - 02_Global
    - 03_Motion
    - 04_ObjectsOfProjectLibraryUsed
    - 09_U1_Unit1
    - 10_Em0_General
    - 11_Em1_Template_LAD
    - 12_Em2_Template_SCL
    - 13_Em3_Example_LAD
    - 14_Em4_Example_SCL
    - 15_Em5_MotionDemo
    - System blocks
  - Technology objects
  - External source files
  - PLC tags
  - PLC data types
  - Watch and force tables
  - Online backups
  - Traces
  - Device proxy data
  - Program info

DriveDiagHmi (snapshot created: 3/20/2025 4:11:14 PM)

	Name	Data type	Start value
1	Static		
2	diagStatus	"LAF_typeDriveDiagInterfaceHmi"	
3	hmi	"LAF_typeHmiDiagData"	
4	referenceDesignator	WString[128]	WSTRING#''
5	driveIndex	DInt	0
6	axisStatusScreenActive	Bool	false
7	sinaStatusScreenActi...	Bool	false
8	sinaAlarmScreenActive	Bool	false
9	sinaEposScreenActive	Bool	false
10	sinaSafetyScreenActive	Bool	false
11	axis	"LAF_typeAxisDiagData"	
12	type	UDInt	0
13	values	"LAF_typeAxisDiagValues"	
14	statusAxis	"LAF_typeAxisDiagStatus"	
15	statusSensor	"LAF_typeAxisDiagStatusSensor"	
16	statusPositioning	"LAF_typeAxisDiagStatusPositioning"	
17	units	"LAF_typeAxisDiagUnits"	
18	sina	"LAF_typeSinaDiagData"	
19	sinaType	UInt	0
20	controlWord	Word	16#0
21	statusWord	Word	16#0
22	telegramNo	UInt	0
23	eposStatusWord1	Word	16#0
24	eposStatusWord2	Word	16#0
25	operatingDisplay	UInt	0
26	override	Real	0.0
27	posSetValue	Real	0.0
28	posActValue	Real	0.0
29	velSetValue	Real	0.0
30	velActValue	Real	0.0
31	lengthUnit	Int	0
32	velocityUnit	Int	0
33	residualDistance	Real	0.0
34	followingError	Real	0.0
35	posControlWord	Word	16#0
36	posStatusWord	Word	16#0
37	blockSelected	Int	0
38	blockActive	Int	0
39	mdiPosSetValue	Real	0.0
40	faultCode	Array[0..7] of Word	
41	alarmCode	Array[0..3] of Word	
42	speedSetValue	Real	0.0
43	speedActValue	Real	0.0
44	safetyStatusWord	Word	16#0

Figure 8-32: Structure of HMI interface data block “DriveDiagHmi”

NOTE

The “DriveDiagHmi” 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 function “CallDiagnostic” (1) and share the two data blocks “DriveDiagConfig” and “DriveDiagHmi” (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.

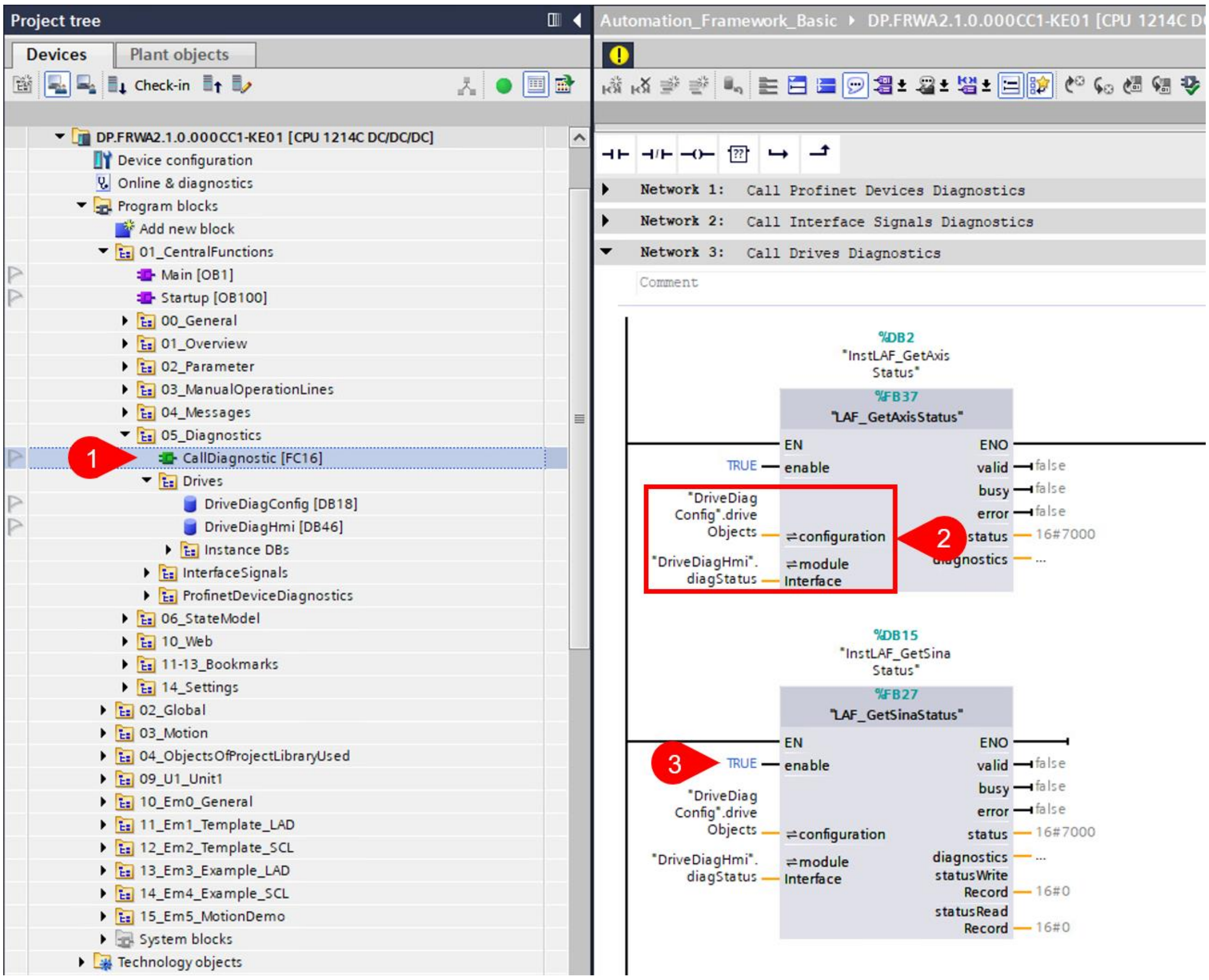


Figure 8-33: Function block calls for drive diagnostics

NOTE

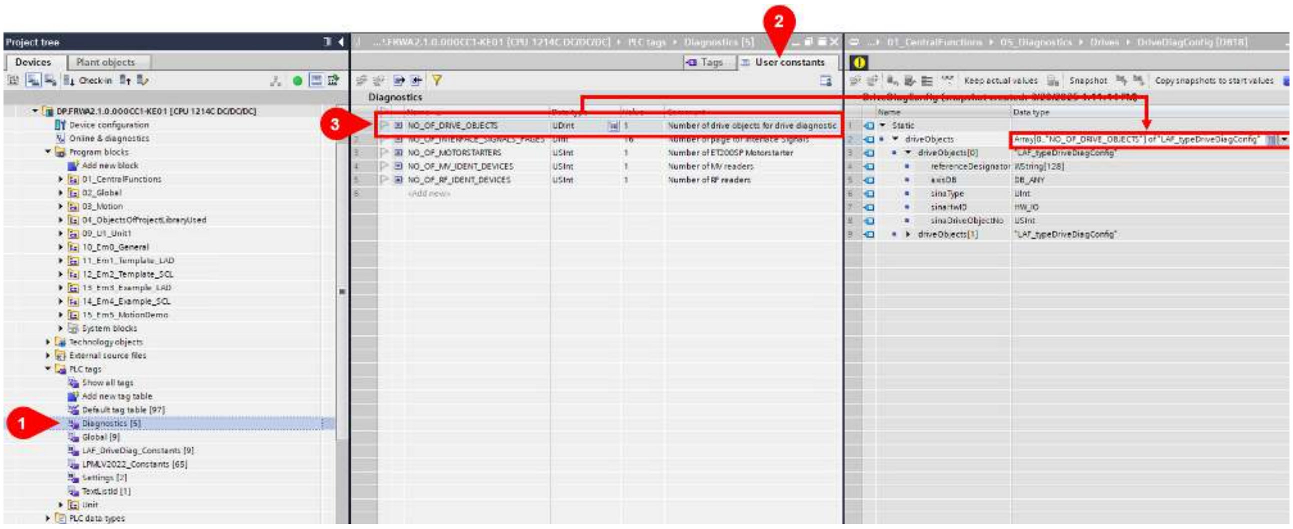
The drive diagnostics function blocks are called in the function “CallDiagnostic” and connected to the “DriveDiagConfig” and “DriveDiagHmi” data blocks in the preconfigured TIA Portal project of the AF.

8.6.5.3. 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 data block “DriveDiagConfig” in the PLC program and to extend the third navigation in the HMI.

PLC engineering

First navigate to “Diagnostics” tag table (1) in the PLC tags. Select the “User constants” (2) and set the value of the “NO_OF_DRIVE_OBJECTS” 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.



Diagnostics

	Name	Data type	Value	Comment
1	NO_OF_DRIVE_OBJECTS	UDInt	1	Number of drive objects for drive diagnostic
2	NO_OF_INTERFACE_SIGNALS_PAGES	UInt	16	Number of page for interface Signals
3	NO_OF_MOTORSTARTERS	USInt	1	Number of ET200SP Motorstarter
4	NO_OF_MV_IDENT_DEVICES	USInt	1	Number of MV readers
5	NO_OF_RF_IDENT_DEVICES	USInt	1	Number of RF readers
6	<Add new>			

01_CentralFunctions ▸ 05_Diagnostics ▸ Drives ▸ DriveDiagConfig [DB18]

	Name	Data type
1	Static	
2	driveObjects	Array[0.."NO_OF_DRIVE_OBJECTS"] of "LAF_typeDriveDiagConfig"
3	driveObjects[0]	"LAF_typeDriveDiagConfig"
4	referenceDesignator	WString[128]
5	axisDB	DB_ANY
6	sinaType	UInt
7	sinaHwID	HW_IO
8	sinaDriveObjectNo	USInt
9	driveObjects[1]	"LAF_typeDriveDiagConfig"

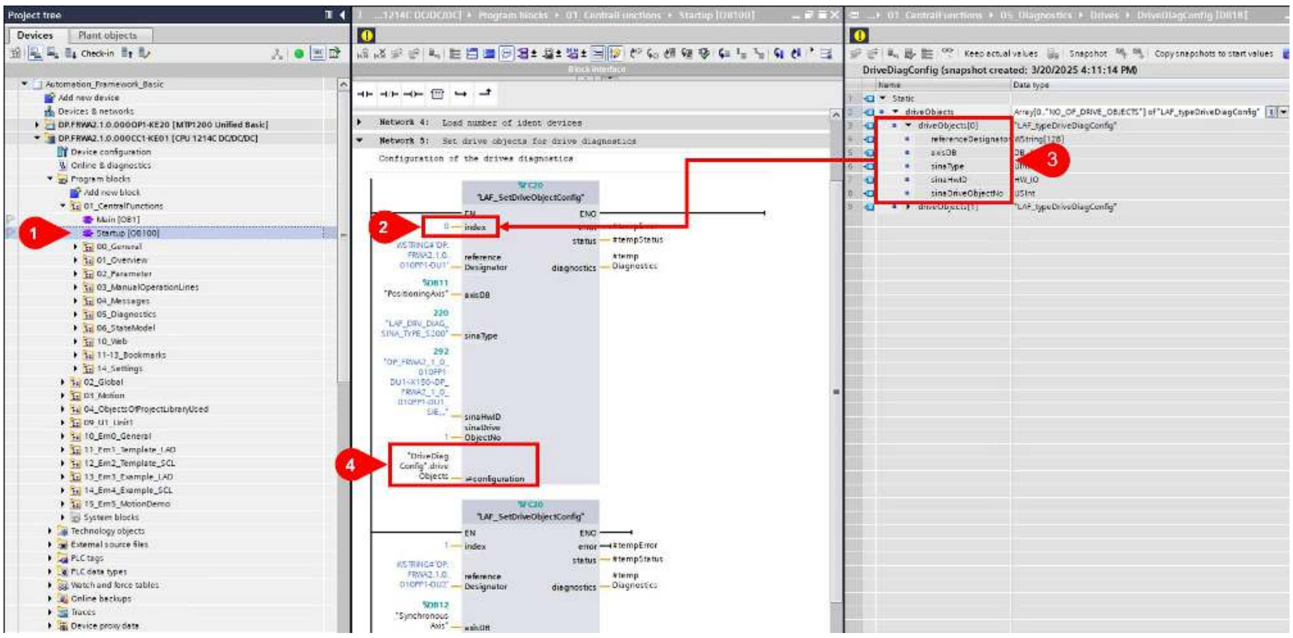
Figure 8-34: Set number of drive objects for drive diagnostics

Example:

- 1x SINAMICS G120 with SinaSpeed
- 2x SINAMICS S210 with TO_PositioningAxis and TO_SynchronousAxis
- 1x SINAMICS S210 with EPOS
- “NO_OF_DRIVE_OBJECTS” = 3

In this example “NO_OF_DRIVE_OBJECTS” means the array contains 4 elements “[0..3]”.

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

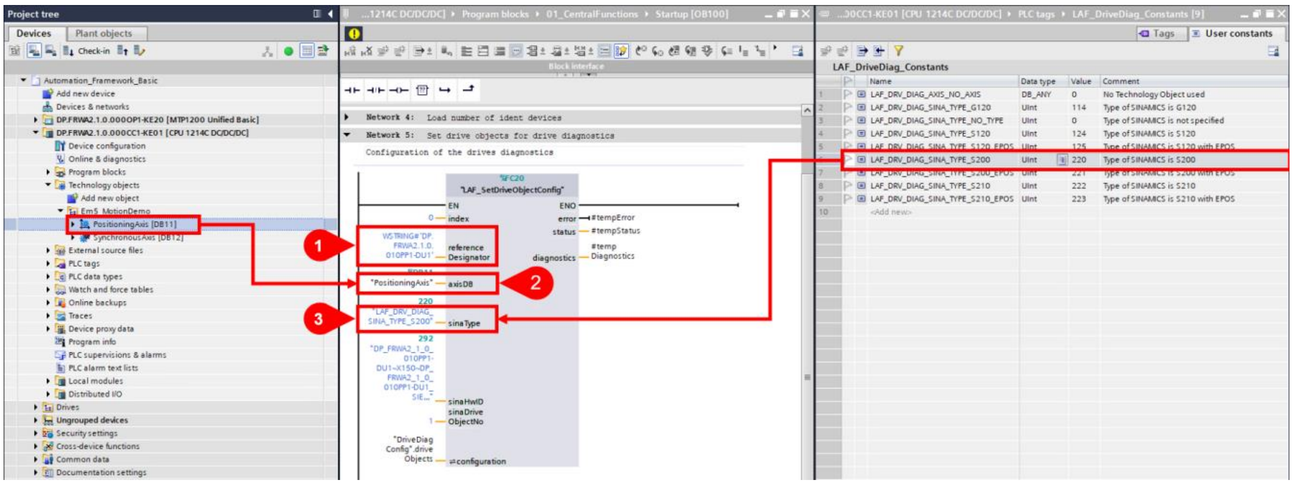


DriveDiagConfig (snapshot created: 3/20/2025 4:11:14 PM)

	Name	Data type
1	Static	
2	driveObjects	Array[0.."NO_OF_DRIVE_OBJECTS"] of "LAF_typeDriveDiagConfig"
3	driveObjects[0]	"LAF_typeDriveDiagConfig"
4	referenceDesignator	WString[128]
5	axisDB	DB
6	sinaType	UInt
7	sinaHwID	HW_IO
8	sinaDriveObjectNo	USInt
9	driveObjects[1]	"LAF_typeDriveDiagConfig"

Figure 8-35: Call of function “LAF_SetDriveObjectConfig” 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 “LAF_DriveDiag_Constants” tag table are available.

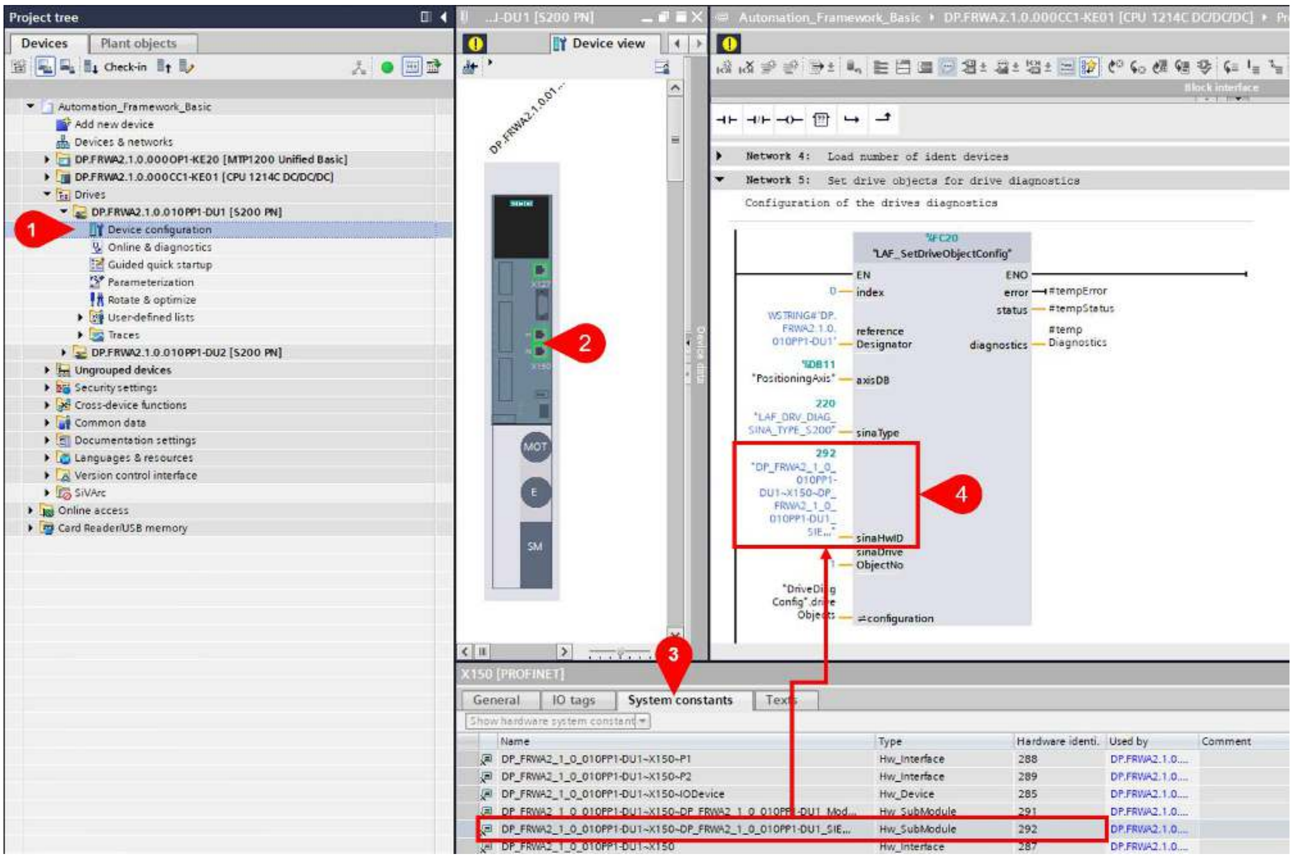


LAF_DriveDiag_Constants

	Name	Data type	Value	Comment
1	LAF_DRV_DIAG_AXIS_NO_AXIS	DB_ANY	0	No Technology Object used
2	LAF_DRV_DIAG_SINA_TYPE_G120	UInt	114	Type of SINAMICS is G120
3	LAF_DRV_DIAG_SINA_TYPE_NO_TYPE	UInt	0	Type of SINAMICS is not specified
4	LAF_DRV_DIAG_SINA_TYPE_S120	UInt	124	Type of SINAMICS is S120
5	LAF_DRV_DIAG_SINA_TYPE_S120_EPOS	UInt	125	Type of SINAMICS is S120 with EPOS
6	LAF_DRV_DIAG_SINA_TYPE_S200	UInt	220	Type of SINAMICS is S200
7	LAF_DRV_DIAG_SINA_TYPE_S200_EPOS	UInt	221	Type of SINAMICS is S200 with EPOS
8	LAF_DRV_DIAG_SINA_TYPE_S210	UInt	222	Type of SINAMICS is S210
9	LAF_DRV_DIAG_SINA_TYPE_S210_EPOS	UInt	223	Type of SINAMICS is S210 with EPOS
10	<Add new>			

Figure 8-36: Assign reference designator, TO and type of SINAMICS to function “LAF_SetDriveObjectConfig”

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 function “LAF_SetDriveObjectConfig” via drag and drop.

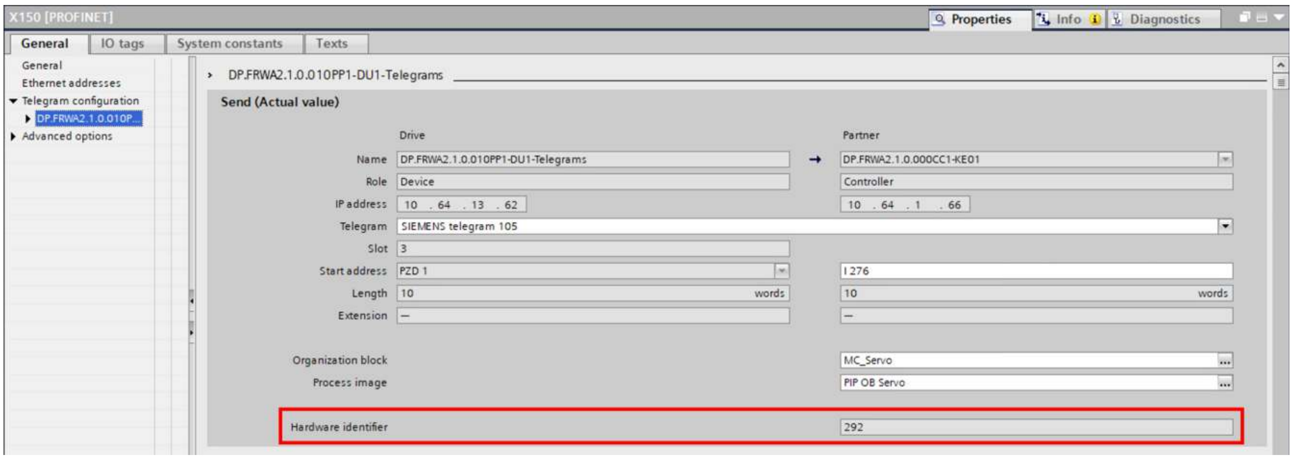

Project tree
Devices
Plant objects
Automation_Framework_Basic
Add new device
Devices & networks
DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic]
DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC]
Drives
DP.FRWA2.1.0.010PP1-DU1 [S200 PN]
Device configuration
Online & diagnostics
Guided quick startup
Parameterization
Rotate & optimize
User-defined lists
Traces
DP.FRWA2.1.0.010PP1-DU2 [S200 PN]
Ungrouped devices
Security settings
Cross-device functions
Common data
Documentation settings
Languages & resources
Version control interface
SiVArc
Online access
Card Reader/USB memory
1
2
3
4
Device view
MOT
E
SM
Automation_Framework_Basic ▸ DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC]
Network 4: Load number of ident devices
Network 5: Set drive objects for drive diagnostics
Configuration of the drives diagnostics
%FC20
"LAF_SetDriveObjectConfig"
EN
ENO
0 — index
error — #tempError
status — #tempStatus
WSTRING#'DP.FRWA2.1.0.010PP1-DU1' — referenceDesignator
diagnostics — #tempDiagnostics
%DB11 "PositioningAxis" — axisDB
220 "LAF_DRV_DIAG_SINA_TYPE_S200" — sinaType
292 "DP_FRWA2_1_0_010PP1-DU1~X150~DP_FRWA2_1_0_010PP1-DU1_SIE..." — sinaHwID
1 — sinaDriveObjectNo
"DriveDiagConfig".driveObjects — configuration
X150 [PROFINET]
General
IO tags
System constants
Texts
Show hardware system constants

Name	Type	Hardware identi.	Used by	Comment
DP_FRWA2_1_0_010PP1-DU1~X150~P1	Hw_Interface	288	DP.FRWA2.1.0....	
DP_FRWA2_1_0_010PP1-DU1~X150~P2	Hw_Interface	289	DP.FRWA2.1.0....	
DP_FRWA2_1_0_010PP1-DU1~X150~IODevice	Hw_Device	285	DP.FRWA2.1.0....	
DP_FRWA2_1_0_010PP1-DU1~X150~DP_FRWA2_1_0_010PP1-DU1_Mod...	Hw_SubModule	291	DP.FRWA2.1.0....	
DP_FRWA2_1_0_010PP1-DU1~X150~DP_FRWA2_1_0_010PP1-DU1_SIE...	Hw_SubModule	292	DP.FRWA2.1.0....	
DP_FRWA2_1_0_010PP1-DU1~X150	Hw_Interface	287	DP.FRWA2.1.0....	

Figure 8-37: Assign system constant of the PROFIdrive telegram to function “LAF_SetDriveObjectConfig”

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.



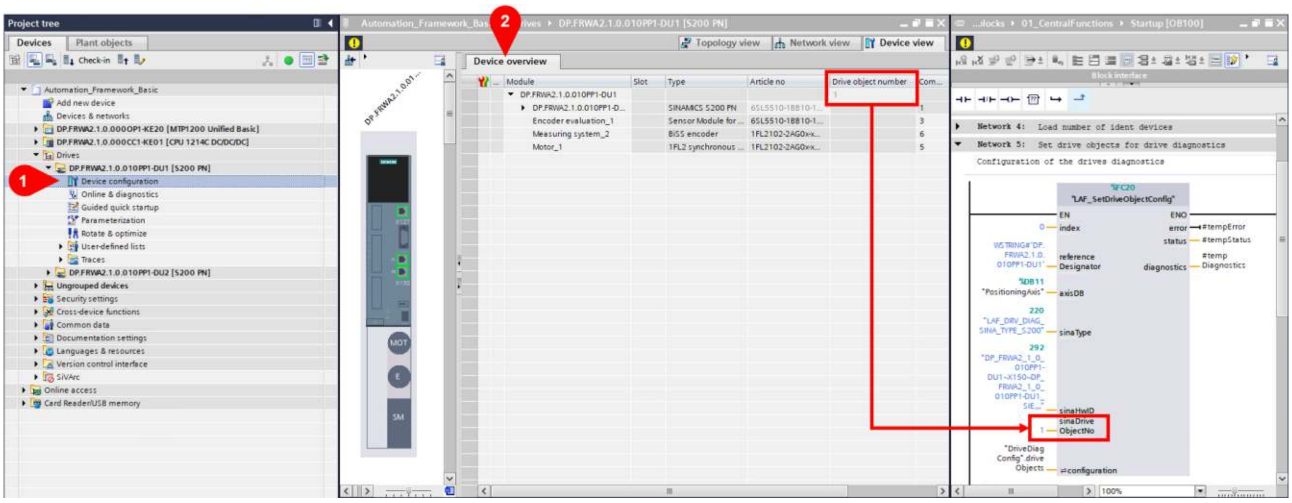
DP.FRWA2.1.0.010PP1-DU1-Telegrams

Send (Actual value)

	Drive	Partner
Name	DP.FRWA2.1.0.010PP1-DU1-Telegrams	DP.FRWA2.1.0.000CC1-KE01
Role	Device	Controller
IP address	10 . 64 . 13 . 62	10 . 64 . 1 . 66
Telegram	SIEMENS telegram 105	
Slot	3	
Start address	PZD 1	I 276
Length	10 words	10 words
Extension	–	–
Organization block		MC_Servo
Process image		PIP OB Servo
Hardware identifier		292

Figure 8-38: Access to the hardware identifier of the PROFIdrive telegram

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



DP.FRWA2.1.0.010PP1-DU1 [S200 PN]

Module	Slot	Type	Article no	Drive object number	Com...
DP.FRWA2.1.0.010PP1-DU1				1	
DP.FRWA2.1.0.010PP1-D...		SINAMICS S200 PN	6SL5510-1BB10-1...		1
Encoder evaluation_1		Sensor Module for ...	6SL5510-1BB10-1...		3
Measuring system_2		BiSS encoder	1FL2102-2AG0x-x...		6
Motor_1		1FL2 synchronous ...	1FL2102-2AG0x-x...		5

```
Network 4:  Load number of ident devices
Network 5:  Set drive objects for drive diagnostics
    Configuration of the drives diagnostics

%FC20
"LAF_SetDriveObjectConfig"
EN                                              ENO
0 — index                                 error — #tempError
                                          status — #tempStatus
WSTRING#'DP.FRWA2.1.0.010PP1-DU1' — referenceDesignator
                                          diagnostics — #tempDiagnostics
%DB11 "PositioningAxis" — axisDB
220 "LAF_DRV_DIAG_SINA_TYPE_S200" — sinaType
292 "DP_FRWA2_1_0_010PP1-DU1~X150~DP_FRWA2_1_0_010PP1-DU1_SIE..." — sinaHwID
1 — sinaDriveObjectNo
"DriveDiagConfig".driveObjects — configuration
```

Figure 8-39: Assign drive object number to function “LAF_SetDriveObjectConfig”

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 extending the third navigation. Via the third navigation, the operator selects the relevant drive.

NOTE The 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.

First navigate to “Text and graphic lists” (1) and extend entries for the text list “Drives” (2). The number of text list entries must match the number of array elements “DriveDiagConfig” data block (3). Also, each value of a text list entry must match the same index value of the configuration array.

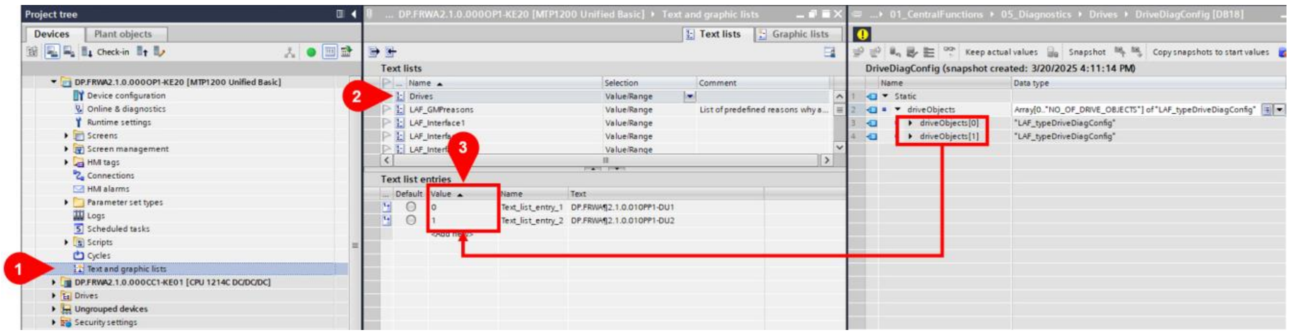


Figure 8-40: Extend entries of the text list “Drives” in the HMI

Next add a drive name for each text list entry in the “Text” collum (1). These texts are shown on the navigation buttons in the HMI during runtime (2).Recommendation: Use the reference designator for the text entry.

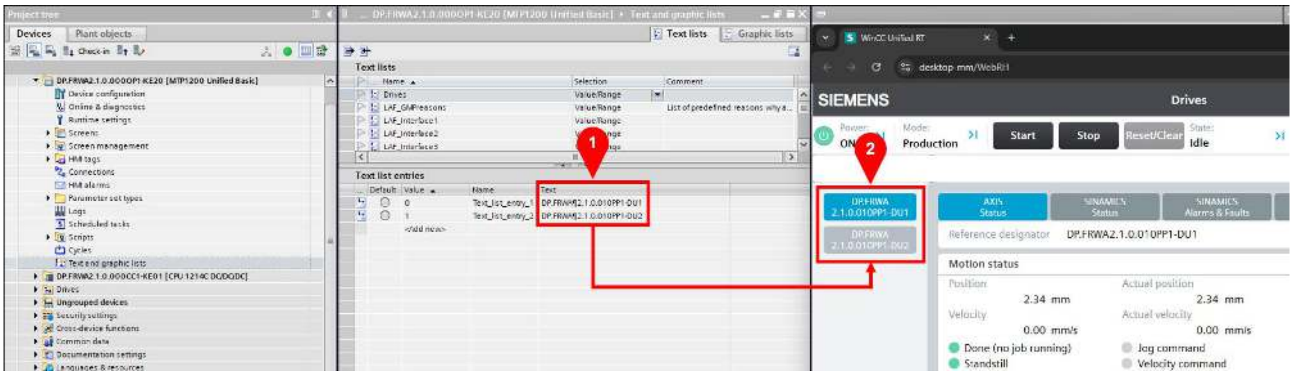


Figure 8-41: Add texts for drive diagnostics navigation buttons

The AFBasic provides 8 preconfigured buttons for the drive diagnostics navigation in the screen “ThirdDrives” (1). Only the relevant buttons are shown in the HMI during runtime.

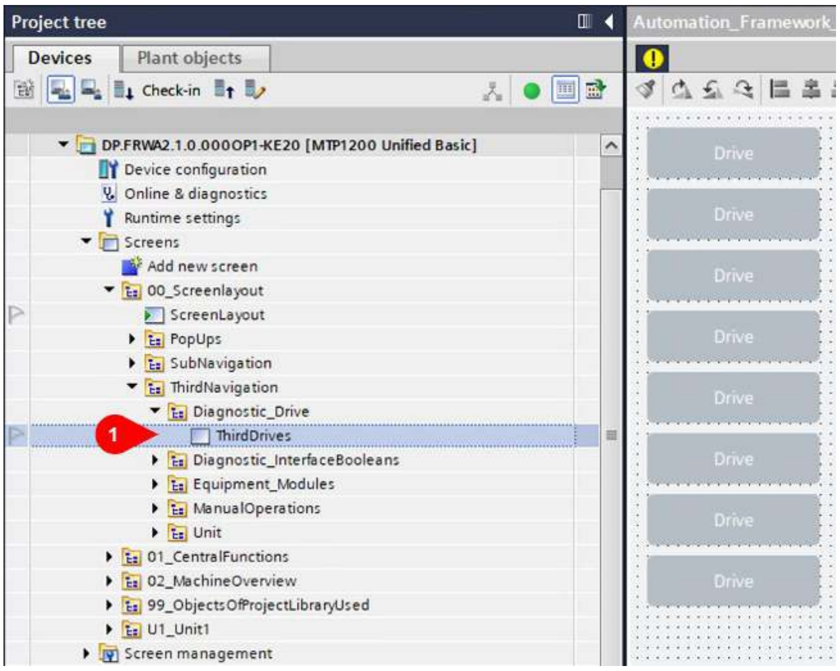


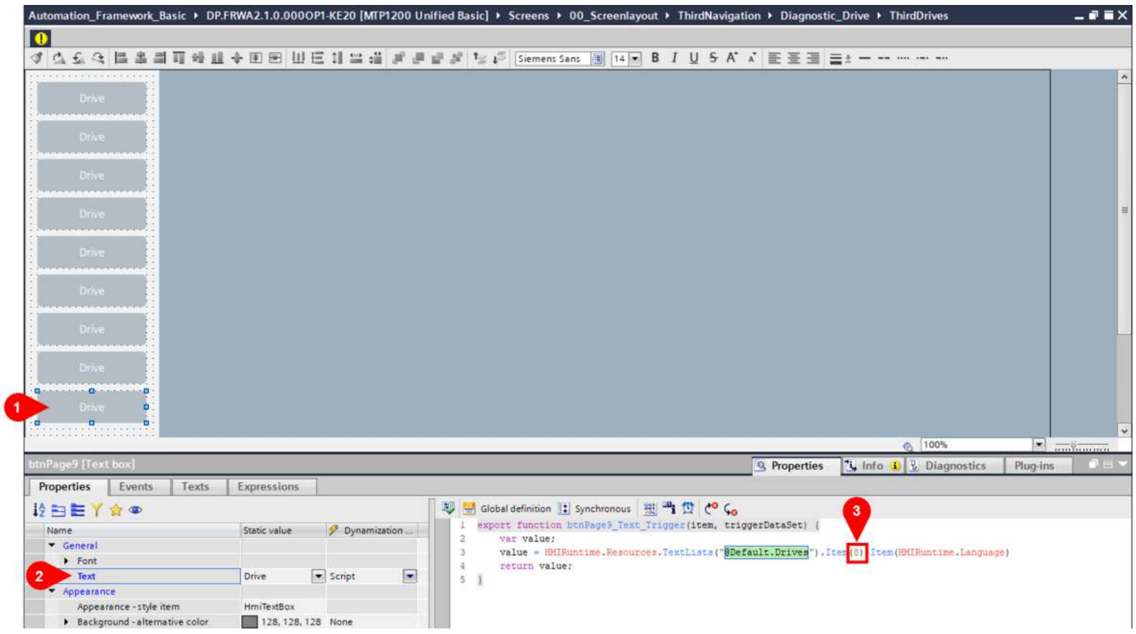
Figure 8-42: Preconfigured drive diagnostics navigation screen “ThirdDrives”

NOTE

If 8 or less drives are used, no additional HMI engineering is required.

If more than 8 drives are used follow the next steps in this chapter to add up to 4 more buttons to the drive diagnostics navigation. The steps are only shown for one button. If more buttons are needed, repeat the steps for each additional button.

To add an extra button, copy the last of the preconfigured buttons and paste it below in the screen “ThirdDrives” (1). Select the pasted button and navigate to “Text” in the properties (2). Then increment the value of the “DriveObjectIndex” in the Java Script (3).

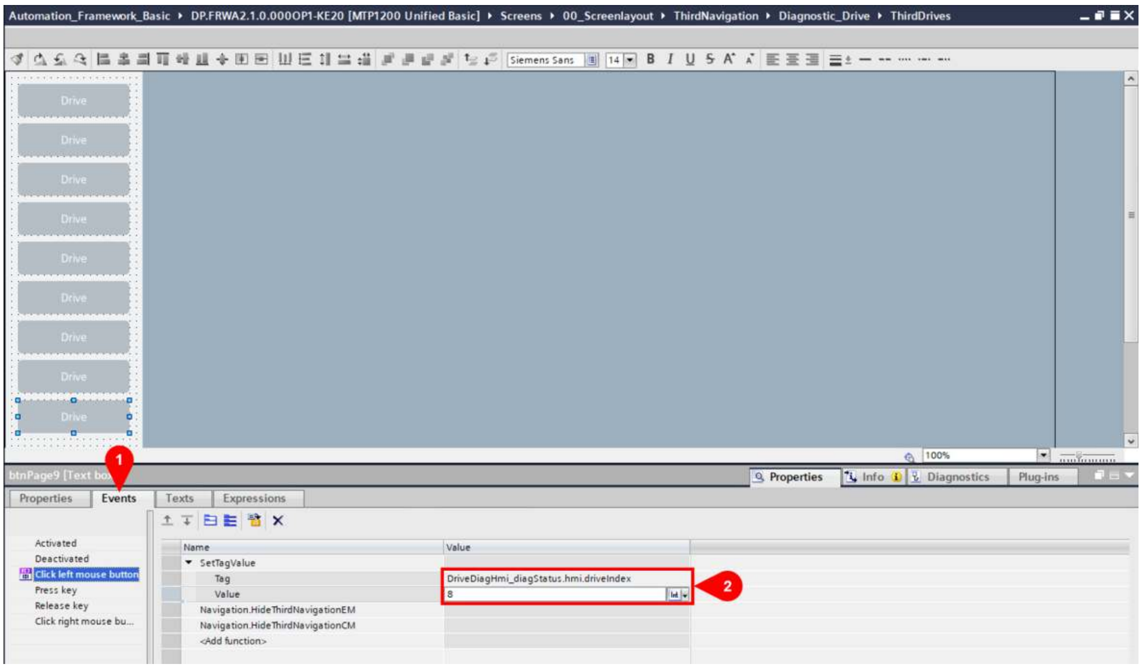

Automation_Framework_Basic ▸ DP.FRWA2.1.0.0000P1-KE20 [MTP1200 Unified Basic] ▸ Screens ▸ 00_Screenlayout ▸ ThirdNavigation ▸ Diagnostic_Drive ▸ ThirdDrives
Drive
1
btnPage9 [Text box]
Properties
Info
Diagnostics
Plug-ins
Properties
Events
Texts
Expressions
Global definition
Synchronous
2
3

Name	Static value	Dynamization ...
General		
Font		
Text	Drive	Script
Appearance		
Appearance - style item	HmiTextBox	
Background - alternative color	128, 128, 128	None

```
export function btnPage9_Text_Trigger(item, triggerDataSet) {
    var value;
    value = HMIRuntime.Resources.TextLists("@Default.Drives").Item(8).Item(HMIRuntime.Language)
    return value;
}
```

Figure 8-43: Button text dynamization for drive diagnostics navigation

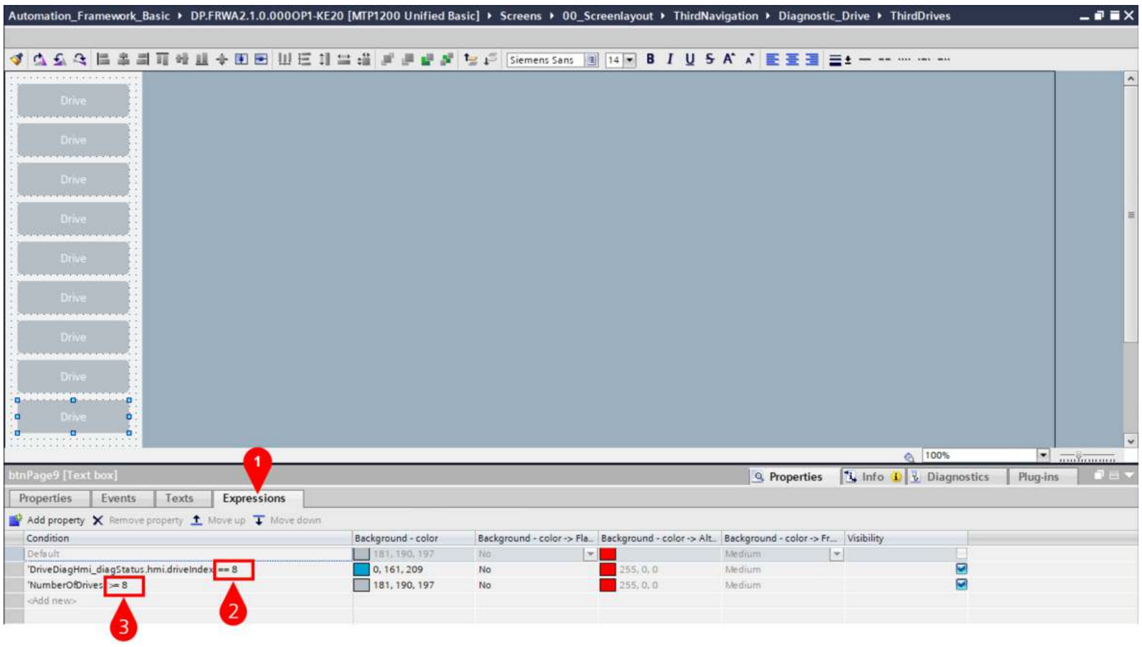
Next navigate to “Events” (1) and increment the value of the “DriveObjectIndex” in event (2).



Name	Value
SetTagValue	
Tag	DriveDiagHmi_diagStatus.hmi.driveIndex
Value	8
Navigation.HideThirdNavigationEM	
Navigation.HideThirdNavigationCM	
<Add function>	

Figure 8-44: Button event for drive diagnostics navigation

Last navigate to “Expressions” (1) and increment the value of the “DriveObjectIndex” in the expressions for the background color dynamization (2) and for the visibility (3).


Automation_Framework_Basic ▸ DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic] ▸ Screens ▸ 00_Screenlayout ▸ ThirdNavigation ▸ Diagnostic_Drive ▸ ThirdDrives
Drive
btnPage9 [Text box]
Properties
Info
Diagnostics
Plug-ins
Properties
Events
Texts
Expressions
Add property
Remove property
Move up
Move down
1
2
3

Condition	Background - color	Background - color -> Fla..	Background - color -> Alt..	Background - color -> Fr...	Visibility
Default	181, 190, 197	No	🟥	Medium	☐
'DriveDiagHmi_diagStatus.hmi.driveIndex' == 8	0, 161, 209	No	🟥 255, 0, 0	Medium	✓
'NumberOfDrives' >= 8	181, 190, 197	No	🟥 255, 0, 0	Medium	✓
<Add new>					

Figure 8-45: Button background color dynamization for drive diagnostics navigation

NOTE

The value of “DriveObjectIndex” must match the index of the array element in the data block “DriveDiagConfig” for the three steps above. Double check the value for the Java Script, the event and the expressions.

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").

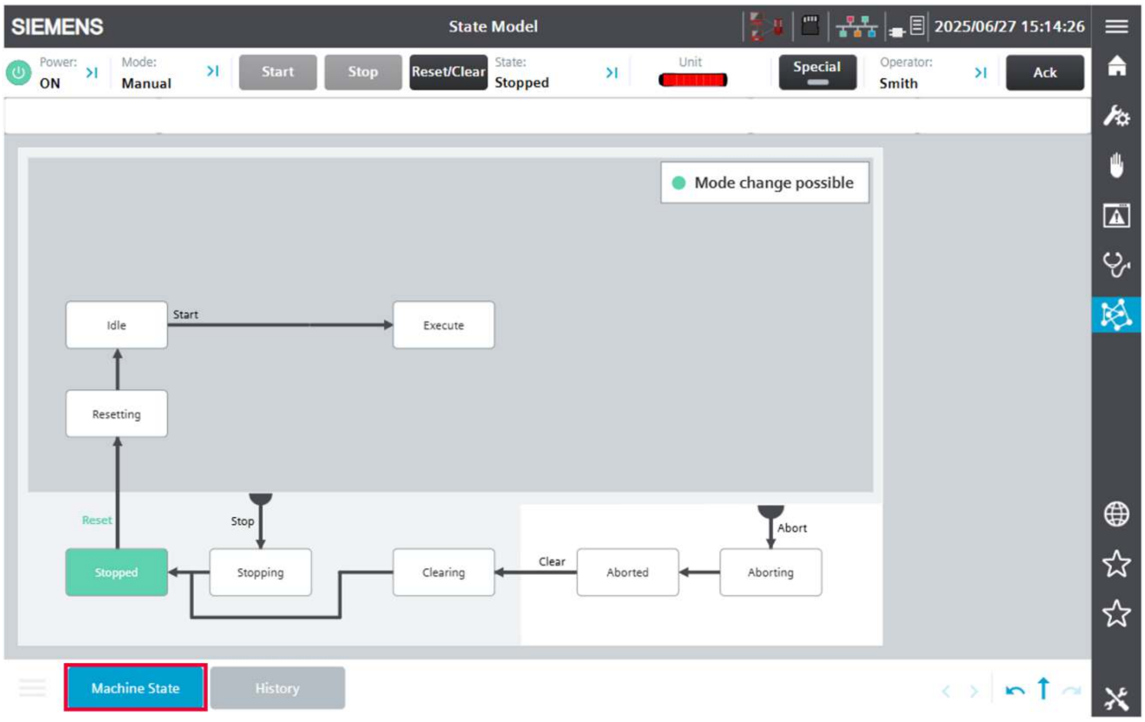
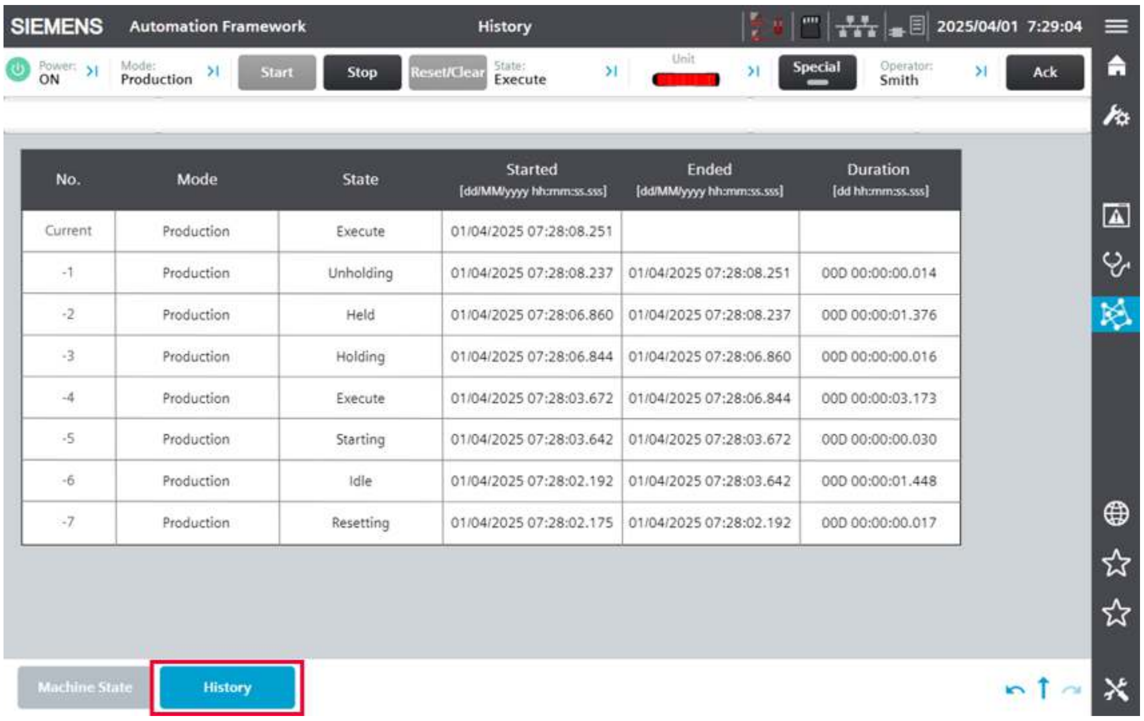


Figure 8-46: State model screen



No.	Mode	State	Started [dd/MM/yyyy hh:mm:ss.sss]	Ended [dd/MM/yyyy hh:mm:ss.sss]	Duration [dd hh:mm:ss.sss]
Current	Production	Execute	01/04/2025 07:28:08.251		
-1	Production	Unholding	01/04/2025 07:28:08.237	01/04/2025 07:28:08.251	00D 00:00:00.014
-2	Production	Held	01/04/2025 07:28:06.860	01/04/2025 07:28:08.237	00D 00:00:01.376
-3	Production	Holding	01/04/2025 07:28:06.844	01/04/2025 07:28:06.860	00D 00:00:00.016
-4	Production	Execute	01/04/2025 07:28:03.672	01/04/2025 07:28:06.844	00D 00:00:03.173
-5	Production	Starting	01/04/2025 07:28:03.642	01/04/2025 07:28:03.672	00D 00:00:00.030
-6	Production	Idle	01/04/2025 07:28:02.192	01/04/2025 07:28:03.642	00D 00:00:01.448
-7	Production	Resetting	01/04/2025 07:28:02.175	01/04/2025 07:28:02.192	00D 00:00:00.017

Figure 8-47: State and mode history

8.8. 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 bookmark was created (2).

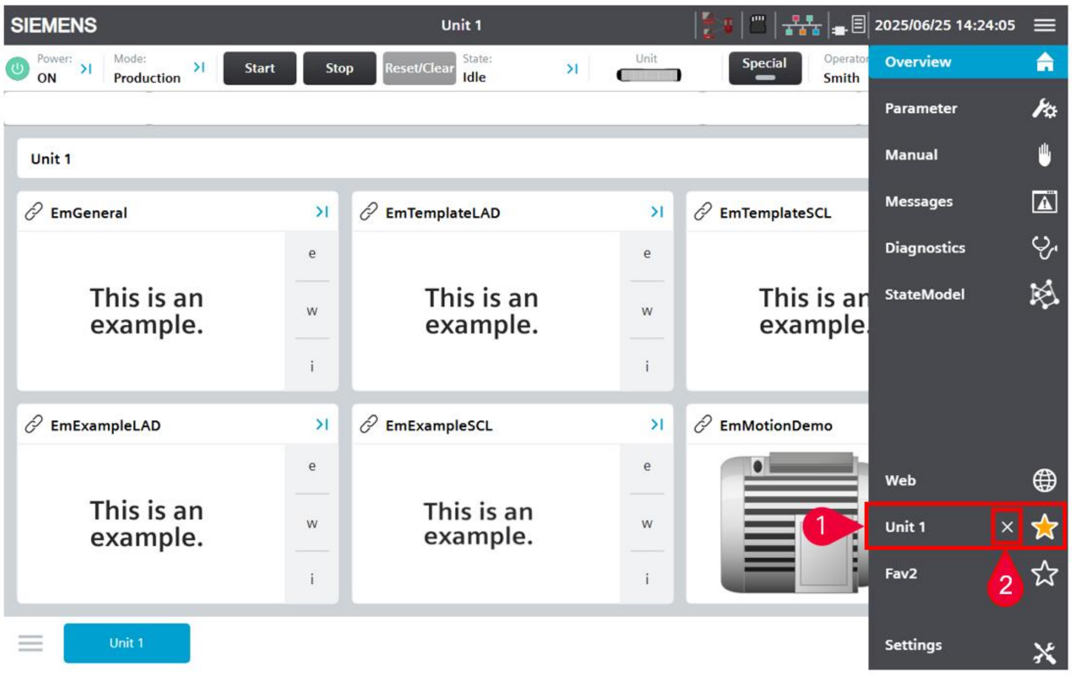


Figure 8-48: Bookmarks

8.9. 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 cycle times (1). 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.

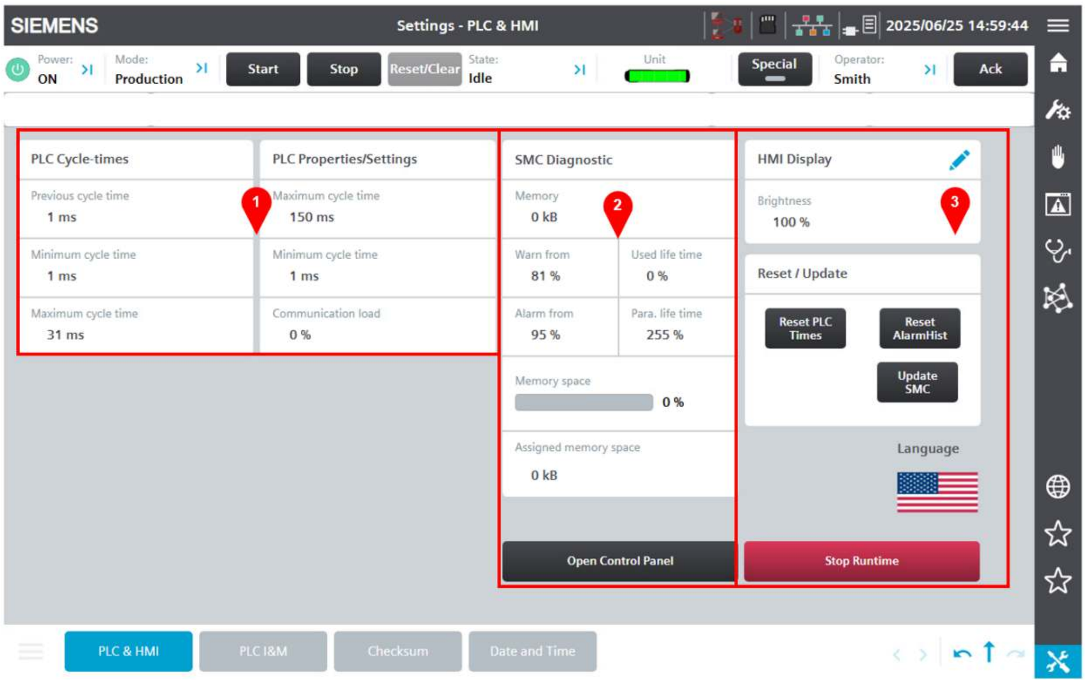
The predefined parameters in the PLC as well as the set communication load can also be displayed.

An evaluation of the SIMATIC Memory Card state is also presented, as well as Open the control panel button (2).

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 (3):

- Stop WinCC Unified Runtime
- Update trigger
- Switching Language


SIEMENS Settings - PLC & HMI 2025/06/25 14:59:44

Power: ON | Mode: Production | Start | Stop | Reset/Clear | State: Idle | Unit | Special | Operator: Smith | Ack

PLC Cycle-times
Previous cycle time 1 ms
Minimum cycle time 1 ms
Maximum cycle time 31 ms

PLC Properties/Settings
Maximum cycle time 150 ms
Minimum cycle time 1 ms
Communication load 0 %

SMC Diagnostic
Memory 0 kB
Warn from 81 %
Used life time 0 %
Alarm from 95 %
Para. life time 255 %
Memory space 0 %
Assigned memory space 0 kB
Open Control Panel

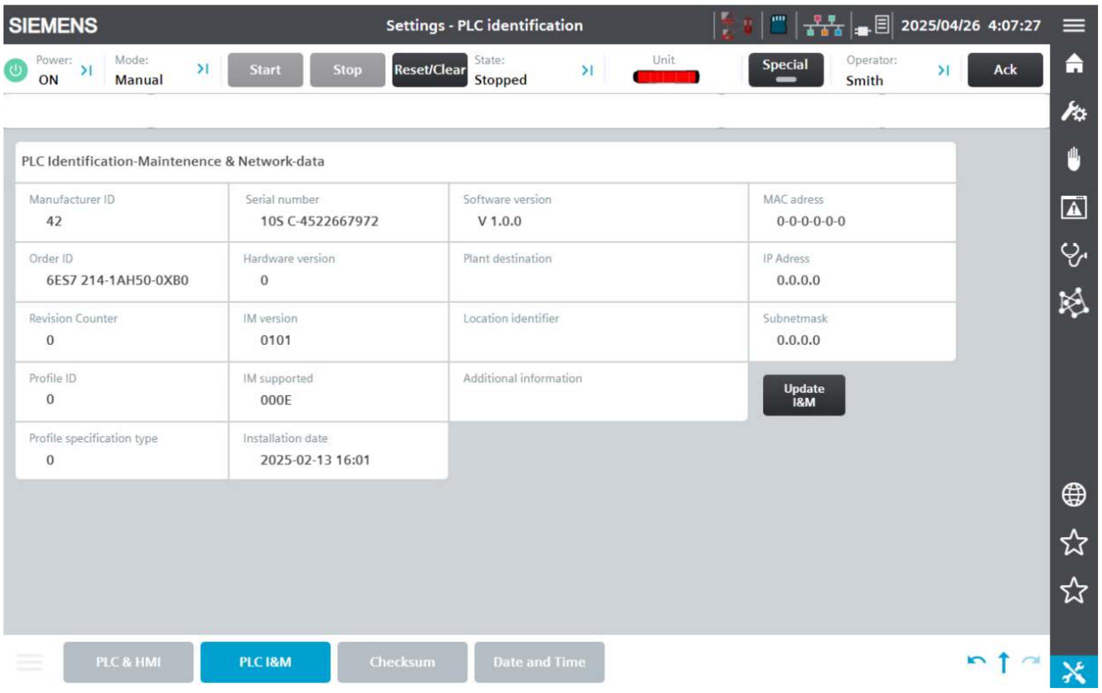
HMI Display
Brightness 100 %
Reset / Update
Reset PLC Times
Reset AlarmHist
Update SMC
Language
Stop Runtime

PLC & HMI | PLC I&M | Checksum | Date and Time

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.



Power: ON | Mode: Manual | State: Stopped | Operator: Smith

PLC Identification-Maintenence & Network-data

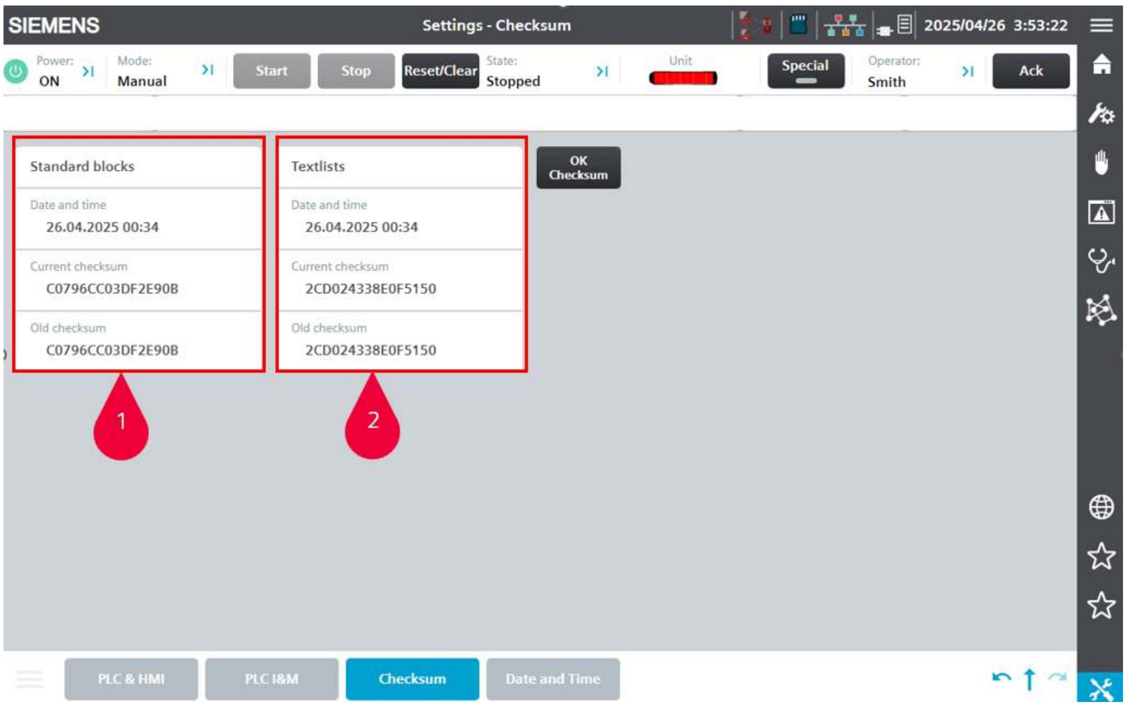
Manufacturer ID 42	Serial number 10S C-4522667972	Software version V 1.0.0	MAC adress 0-0-0-0-0-0
Order ID 6ES7 214-1AH50-0XB0	Hardware version 0	Plant destination	IP Adress 0.0.0.0
Revision Counter 0	IM version 0101	Location identifier	Subnetmask 0.0.0.0
Profile ID 0	IM supported 000E	Additional information	
Profile specification type 0	Installation date 2025-02-13 16:01		

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) and for PLC text lists (2). 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 buttons "Checksum"(SubMenu), "Ok Checksum" and "Settings" (Main Menu) are animated.



Standard blocks

Date and time
26.04.2025 00:34

Current checksum
C0796CC03DF2E90B

Old checksum
C0796CC03DF2E90B

Textlists

Date and time
26.04.2025 00:34

Current checksum
2CD024338E0F5150

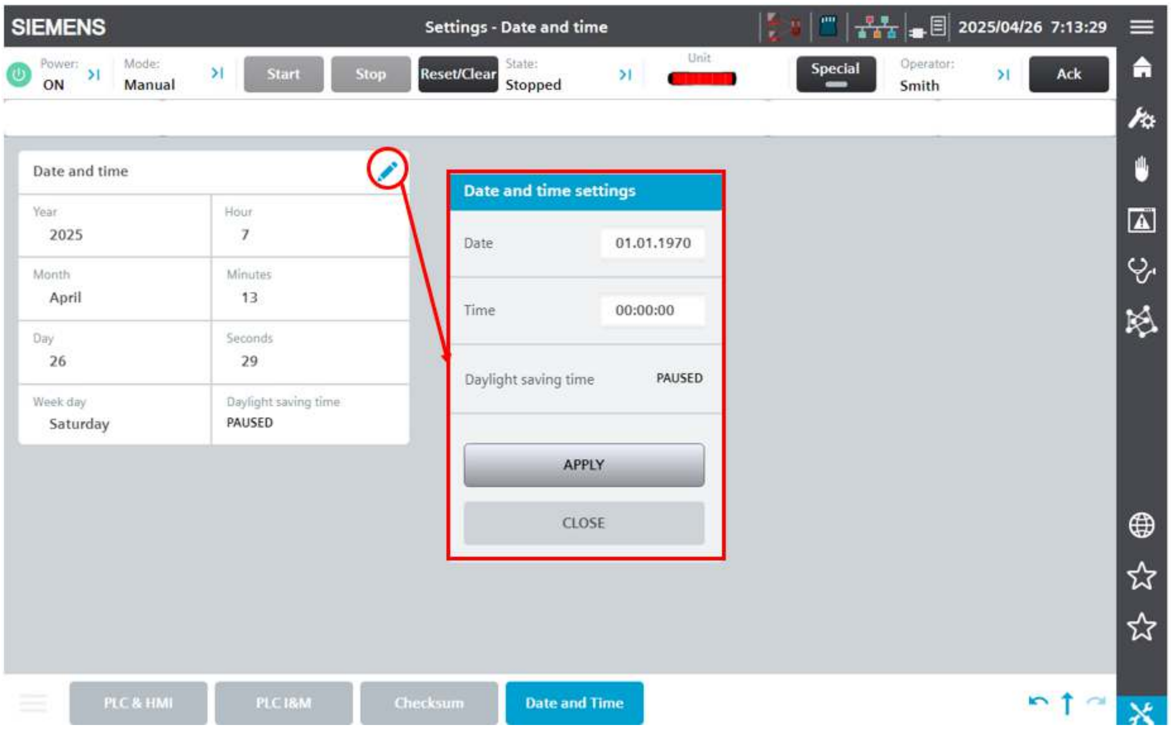
Old checksum
2CD024338E0F5150

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 Software Unit "CentralFunctions" → Program blocks → 14_Settings → CallSettings.



Date and time

Year 2025	Hour 7
Month April	Minutes 13
Day 26	Seconds 29
Week day Saturday	Daylight saving time PAUSED

Date and time settings

Date	01.01.1970
Time	00:00:00
Daylight saving time	PAUSED

APPLY

CLOSE

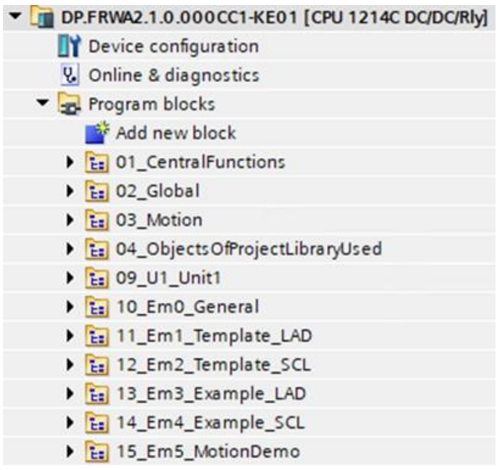
Figure 8-52: Settings Date and Time

9. Units

This chapter describes the general structure, interfaces, commands, and data flow of units.

One unit is implemented in the AFBasic project:

- Unit 1: Example implementation programmed with LAD.



DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/Rly]
Device configuration
Online & diagnostics
Program blocks
Add new block
01_CentralFunctions
02_Global
03_Motion
04_ObjectsOfProjectLibraryUsed
09_U1_Unit1
10_Em0_General
11_Em1_Template_LAD
12_Em2_Template_SCL
13_Em3_Example_LAD
14_Em4_Example_SCL
15_Em5_MotionDemo

Figure 9-1: Unit and EMs implementation

9.1. General Structure

Each Unit consists of several folders. The first folder represents the unit itself. Additional folders represent each equipment modules that are assigned to the unit.

Main program structure

- Handling of general releases and safety status.
- Mode and state management. This done by the evaluating the feedback of the EMs, reading the control interfaces, pre-processing of the commands and the mode-state management itself.
- Internal Alarms are processed and managed.

Signal flow

The basic cyclic execution of unit commands starts with the assessment of the feedback obtained from the EMs and the status of the unit itself. This evaluation is performed by the FB "LUC_StatusCollector". The outcome is the creation of a current process interface. This interface is then evaluated together with the external control interfaces, such as commands from an 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 summary based on this information.
The FB “LPMLV2022_UnitModeStateManager” then evaluates the commands and sets the current mode and state. Additional information of the state machine is written to the DB “Units” via the FC "LUC_SetModeCurrent", FC “LUC_SetStateCurrent”, FC “LPMLV2022_UnitModeChangeAllowed” and FB “LUC_UnitStateCommandsAllowed”.
From DB “Units”, the EMs read the unit mode and state and execute their logic accordingly. The EMs then send their updated feedback signals to the unit, which are then evaluated again in the new cycle.

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

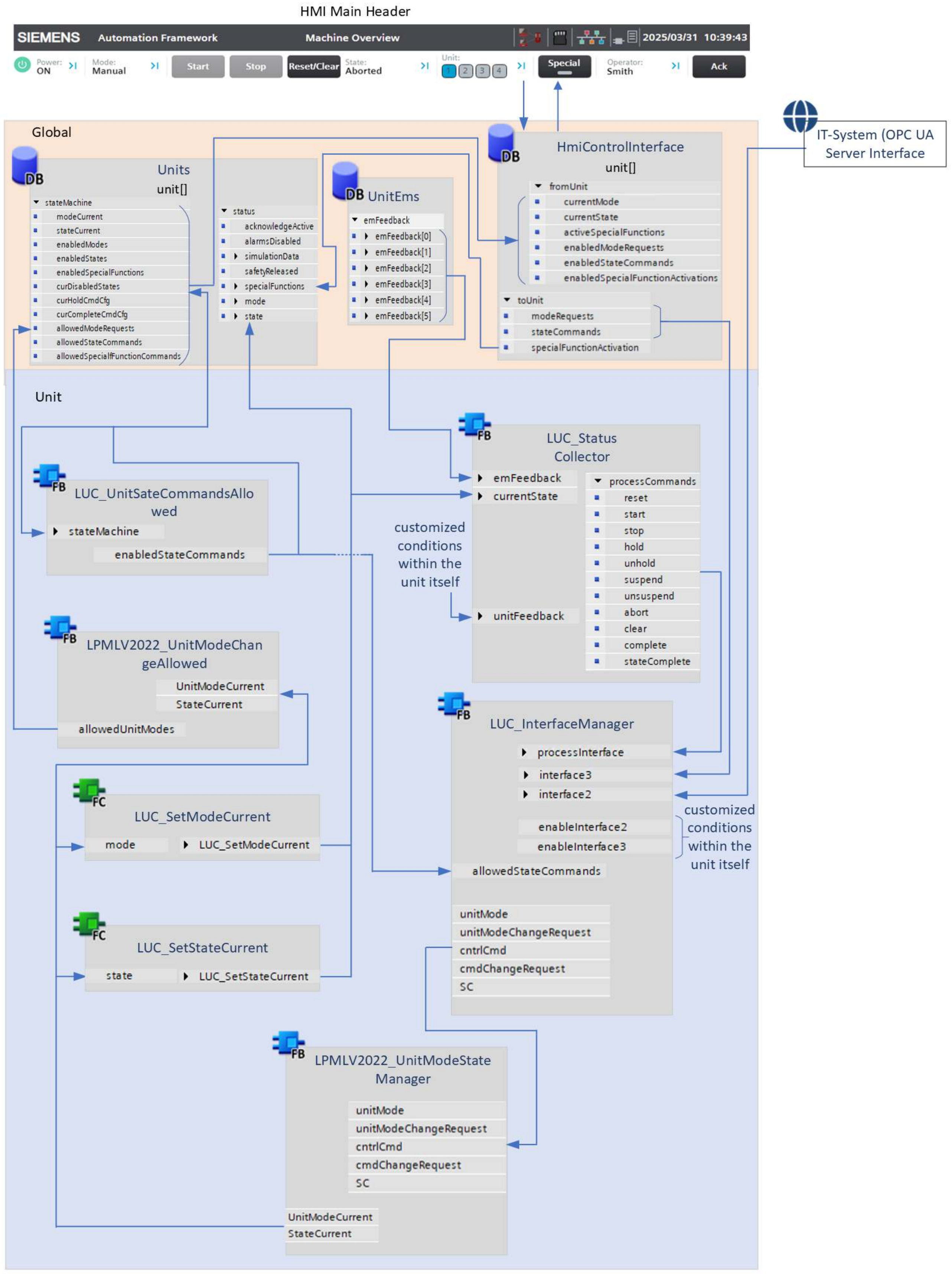
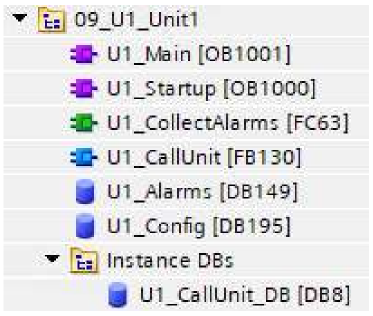


Figure 9-2: Signal flow of a unit

9.2. Interfaces and blocks

The general program architecture of a unit is based on the description in the chapter Program Structure. 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 unit implemented within the AF Basic.

A unit contains the following blocks:



```
09_U1_Unit1
    U1_Main [OB1001]
    U1_Startup [OB1000]
    U1_CollectAlarms [FC63]
    U1_CallUnit [FB130]
    U1_Alarms [DB149]
    U1_Config [DB195]
    Instance DBs
        U1_CallUnit_DB [DB8]
```

Figure 9-3: 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”, FC “LUC_StateModelConfig” transfers the user setting settings into the stateMode manager configuration which are used by the state mode manager during runtime. In addition, information of the enabled modes and states are provided to the unit interface through the FC “LUC_SummarizeEnabledModesAndStates”. The enabled special functions configured in the “Config DB” are also provided to the unit interface in this OB “Startup”.

OB Main

In OB “Main”, FB “CallUnit” is called, which contains the call of all further blocks and code. It can be seen, that data from outside the own folder, e.g. from “Global” is handed over via the interface of FB “CallUnit”. Data within the own folder is not handed over via the block interface, but handeled inside the block directly by using direct data block access.

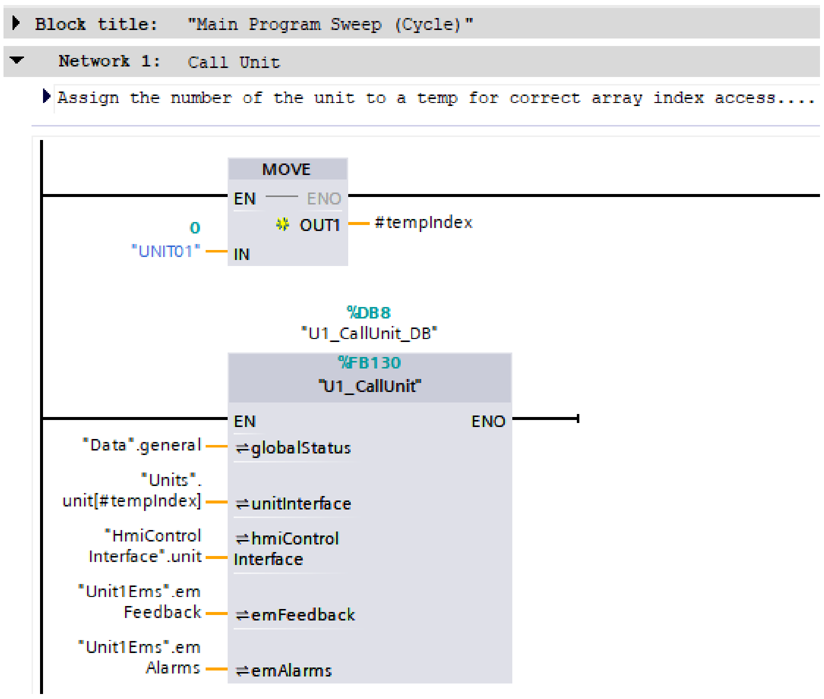

```
Block title:   "Main Program Sweep (Cycle)"
Network 1:   Call Unit
Assign the number of the unit to a temp for correct array index access....
```

Figure 9-4: Implementation of OB Main of the unit

FC CollectAlarms

This block is used for the alarm concept implemented within the Automation framework Basic. Further explanation of how the block is used can be found within the chapter Diagnostics Concept under the section Alarm Implementation in the PLC

DB Alarms

Contains alarms that can be configured by the used. They are evaluated by FC “CollectAlarms”.

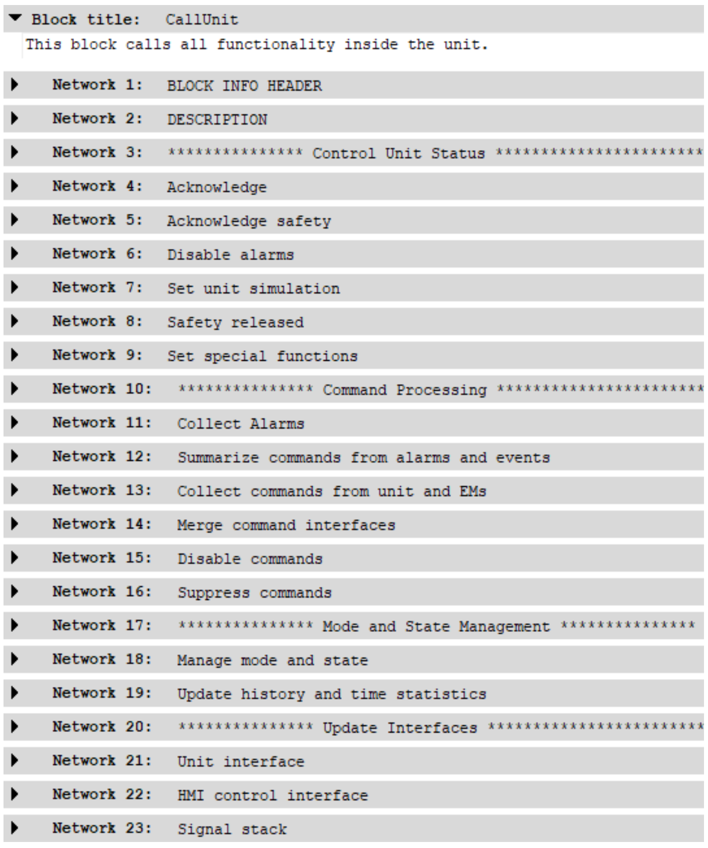
DB Config (Unit)

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
- Special functions enabled for the unit

FB CallUnit

The function block FB “CallUnit” performs the followings actions:



```
▼ Block title:   CallUnit
  This block calls all functionality inside the unit.

▶   Network 1:   BLOCK INFO HEADER
▶   Network 2:   DESCRIPTION
▶   Network 3:   *************** Control Unit Status ***********************
▶   Network 4:   Acknowledge
▶   Network 5:   Acknowledge safety
▶   Network 6:   Disable alarms
▶   Network 7:   Set unit simulation
▶   Network 8:   Safety released
▶   Network 9:   Set special functions
▶   Network 10:   *************** Command Processing ***********************
▶   Network 11:   Collect Alarms
▶   Network 12:   Summarize commands from alarms and events
▶   Network 13:   Collect commands from unit and EMs
▶   Network 14:   Merge command interfaces
▶   Network 15:   Disable commands
▶   Network 16:   Suppress commands
▶   Network 17:   *************** Mode and State Management ***************
▶   Network 18:   Manage mode and state
▶   Network 19:   Update history and time statistics
▶   Network 20:   *************** Update Interfaces ************************
▶   Network 21:   Unit interface
▶   Network 22:   HMI control interface
▶   Network 23:   Signal stack
```

Figure 9-5 Content of FB “CallUnit”

Control Unit Status

- Set unit simulation / Acknowledge / Disable alarms

If simulation, acknowledge or disable alarms is active on a global level, inherit this status and set active in this unit also. Information is written to DB “Units”, 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 occurs, an abort command is sent to the mode state manager.

- Set special functions

Special functions commands to the unit are copied from the DB “HmiControlInterface” to the DB “Units”, so that all EMs can inherit them an react accordingly.

Command Processing

- Disable state commands

The user can define preconditions to disbale state commands. This preconditions can be process variables, mode, states, etc.

- Collect Alarms

Any event can be assigned to an alarm within the collect alarms function. This specific alarm is mapped to a reaction for the unit (e.g. abort, stop, hold). More information on the process diagnostic within the AF$_{\text{Basic}}$ is given in chapter Diagnostic Concept.

- Summarize commands from alarms and events

This block summarizes commands from events, and alarms from DB “Alarms”. The alarms from DB” Alarms” can contain reaction categories (e.g. abort, stop, hold) which are summarized here and sent to the unit as commands.

- Collect commands from the unit and EMs

Here the Collection of commands coming internally from the unit (unitFeedback) and commands and stateComplete from all underlaying linked equipment modules (emFeedback) takes place. They then get Merged and forwarded to the FB “InterfaceManager” as process commands.

- Merge commands interfaces

The different commands from all interfaces are evaluated and merged before they are sent as set of commands to the mode state manager. These could be commands from the operator via HMI or IT, or state change request due to the process status evaluation.

- Suppress commands

The summarized command will not be forwarded to the ModeState Manager if the command was previously disabled.

Mode and State Management

- Manage mode and state

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

- Update history and 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.

Update Interfaces

- Unit interface

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

- HMI control interface
- Update of unit status information to the Hmi control interface.
- Stack Light

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

9.3. HMI Screen

To access the overview screen of a specific unit, the operator has two options:

1. Select the unit directly from the main machine overview screen.

2. Use the navigation bar, which can be opened by clicking the icon in the bottom-left corner of the screen.

For simulation purposes, a Simulation Activation button has been added to the main overview screen.

Additionally, the collective status of the safety zones is visualized via a virtual E-Stop button. If all safety zones are released, the button appears deactivated. If any safety zone remains active, the button is shown as activated—clearly indicating that the system is not yet fully safe to operate.

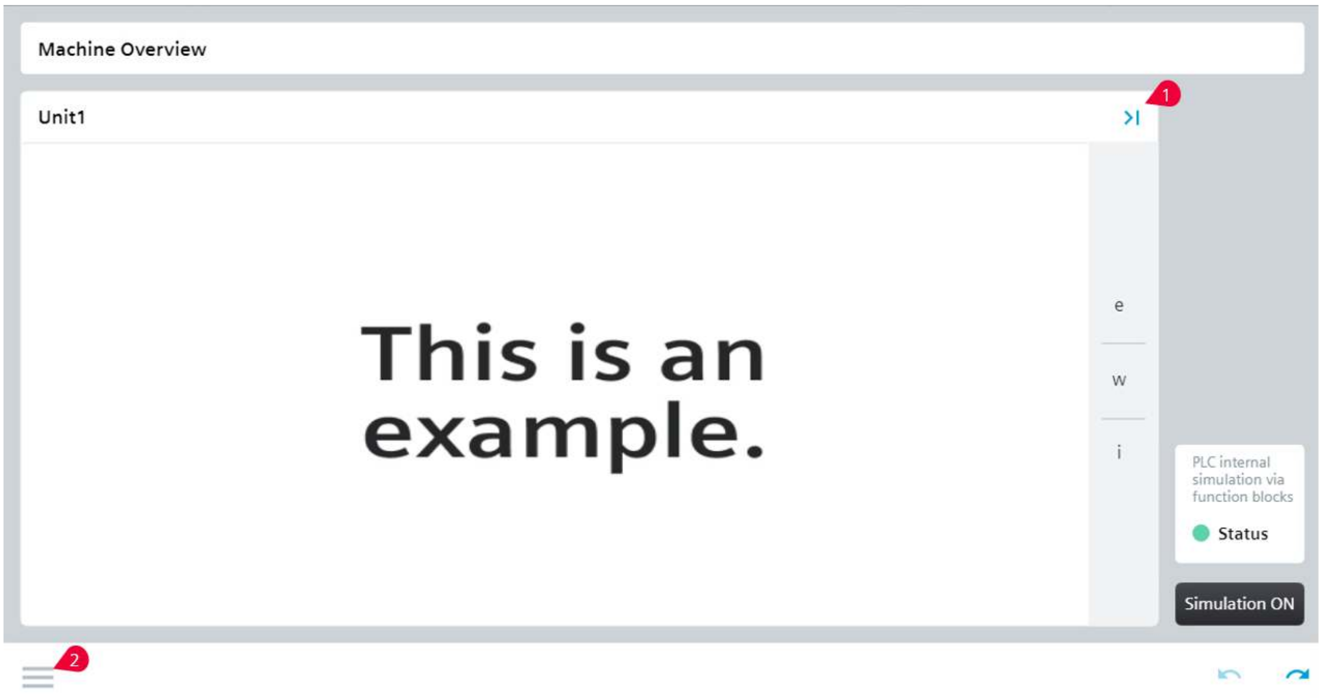


Figure 9-6: Overall machine view

When a unit is selected, its corresponding overview screen opens, displaying the associated Equipment Modules (EMs).

An example implementation for Unit 3 is shown in Figure 9-8: Demo Unit 3 Overview Screen. In customer-specific projects, this screen can be adapted to include tailored machine schematics or dedicated EM views.

Operators can access individual equipment screens either by clicking directly on the equipment within the overview screen or by using the navigation bar, which also provides access to the EM level of the machine.

The detailed structure of the Equipment Module screens is described in Chapter 10.3: HMI Screen.

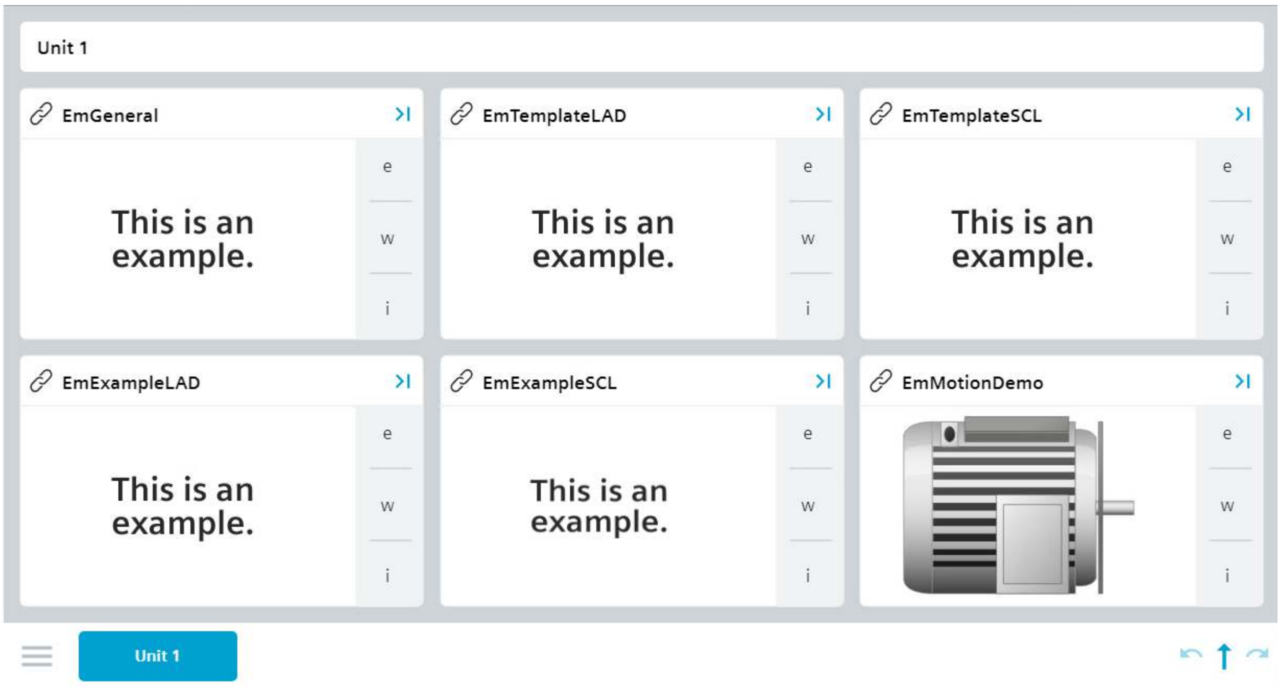


Figure 9-7: Demo Unit 1 Overview Screen

9.4. Mode and State Manager

The AFBasic 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 share the current Mode and State with 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 user’s 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.

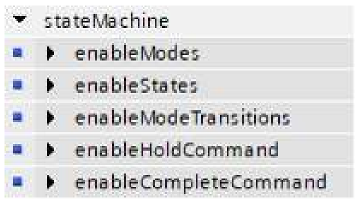


Figure 9-8: Configuration of the state machine in the DB Config of the unit

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 “HmiControlInterface” of Global folder. Each unit that was projected includes 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.

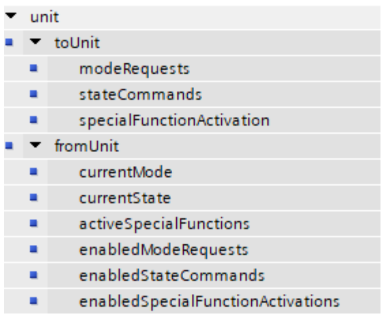


Figure 9-9: ModeStateManager Interface in DB “HmiControlInterface” in Global folder

History and statistics

A history of the previous states and mode is also stored for each unit within the DB “Units” in the UDT “history” of the specific unit. The last 8 steps of the state machine are provided. The FB “LUC_UpdateModeStateHistory” (called in FB “CallUnit”) updates that history upon each mode or state change.

A 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 (See the pop-up windows below) located on the main header. The **Mode Requests** section allows for the selection of the overarching operational mode of the equipment, dictating the available procedures and the level of automation. The available modes are:

- Production: The standard automated operating mode.
- Maintenance: A mode that allows the user to access specific functions for service, calibration, or repairs.
- Manual: A mode that permits direct operator intervention and control over individual CMs.

The **State Commands** provide the means to govern the procedural execution within the selected mode, adhering to the state model. The commands include:

- Core-Commands: Start, Stop, Reset, and Clear for fundamental control over the operational cycle.
- Execution Management: Hold, Suspend, Complete, and Abort to manage the progression of a batch or process. Hold initiates a controlled pause, Suspend brings the machine to a shutdown state that can be resumed, Complete finalizes the current operation, and Abort terminates all activity immediately and requires a reset.
- State Resumption: Unhold and Unsuspend are used to resume operation from the corresponding paused states, bringing the unit back into an active condition.

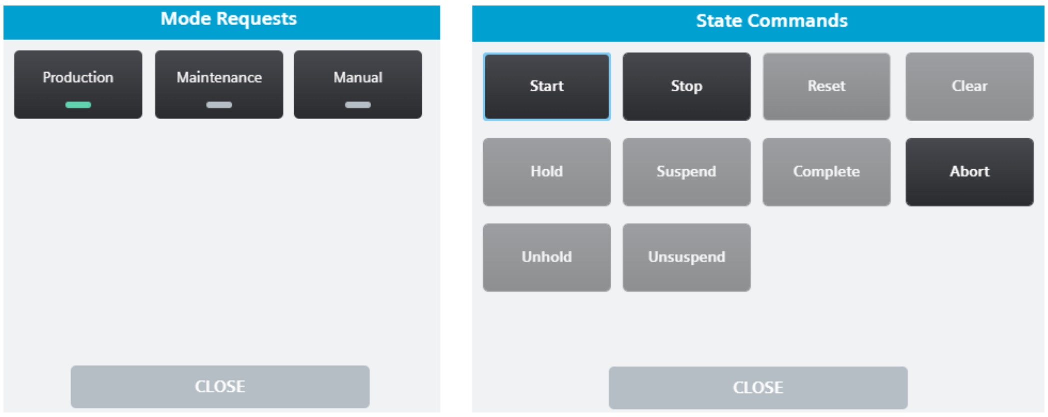


Figure 9-10: Mode Request and State Commands pop-ups

From the main navigation bar located on the right side of the HMI, as indicated on the figure below, the user can access a dedicated screen to monitor the machine's state model. Activating the designated icon (1), as highlighted in the figure below, opens the **Machine State** view (2). This screen presents a graphical representation of the configured state machine, illustrating all possible states (e.g., Idle, Starting, Execute, Held, Stopped) and the transitions between them for the currently selected operational mode. The active state is visually emphasized to provide an immediate understanding of the machine's status.

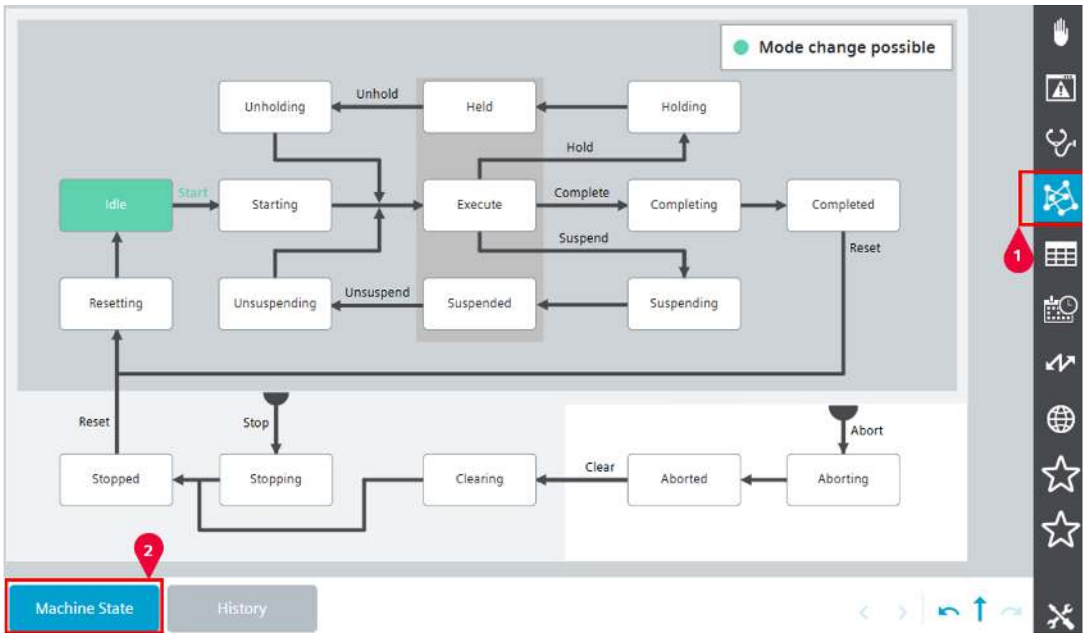
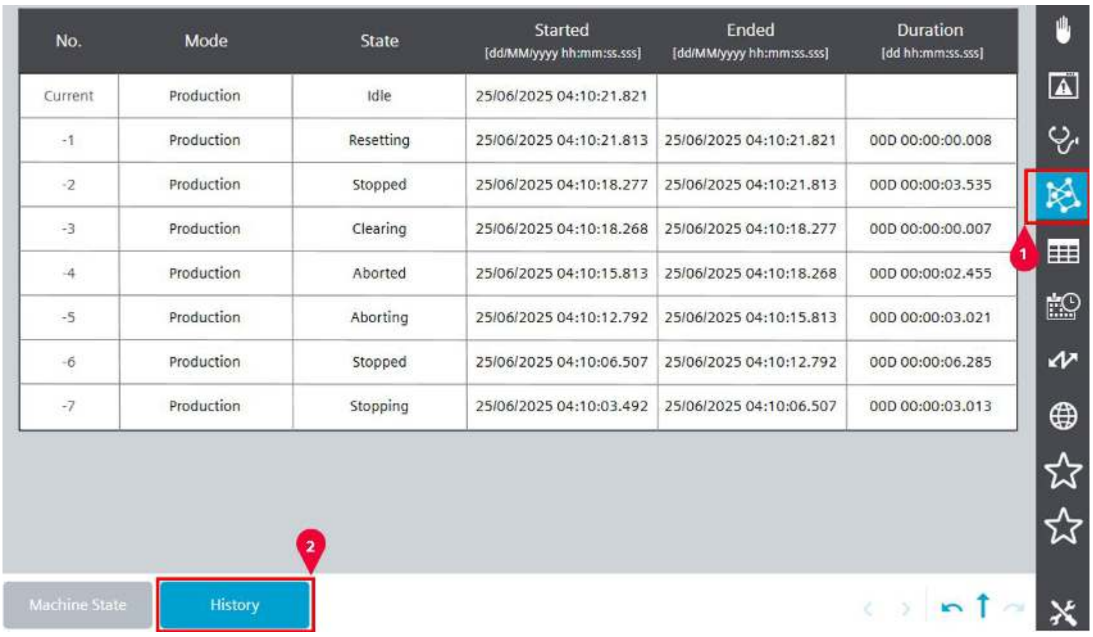


Figure 9-11: State model view

Furthermore, this interface includes a History tab. Selecting this tab navigates to a detailed, time-stamped log of the state machine's recent activity. As shown in the figure below (2), this historical view provides a chronological list of states, recording the precise start time, end time, and total duration for each state entered. This functionality allows for in-depth diagnostics and performance analysis of the machine's operational sequence.



No.	Mode	State	Started [dd/MM/yyyy hh:mm:ss.sss]	Ended [dd/MM/yyyy hh:mm:ss.sss]	Duration [dd hh:mm:ss.sss]
Current	Production	Idle	25/06/2025 04:10:21.821		
-1	Production	Resetting	25/06/2025 04:10:21.813	25/06/2025 04:10:21.821	00D 00:00:00.008
-2	Production	Stopped	25/06/2025 04:10:18.277	25/06/2025 04:10:21.813	00D 00:00:03.535
-3	Production	Clearing	25/06/2025 04:10:18.268	25/06/2025 04:10:18.277	00D 00:00:00.007
-4	Production	Aborted	25/06/2025 04:10:15.813	25/06/2025 04:10:18.268	00D 00:00:02.455
-5	Production	Aborting	25/06/2025 04:10:12.792	25/06/2025 04:10:15.813	00D 00:00:03.021
-6	Production	Stopped	25/06/2025 04:10:06.507	25/06/2025 04:10:12.792	00D 00:00:06.285
-7	Production	Stopping	25/06/2025 04:10:03.492	25/06/2025 04:10:06.507	00D 00:00:03.013

Figure 9-12: History view

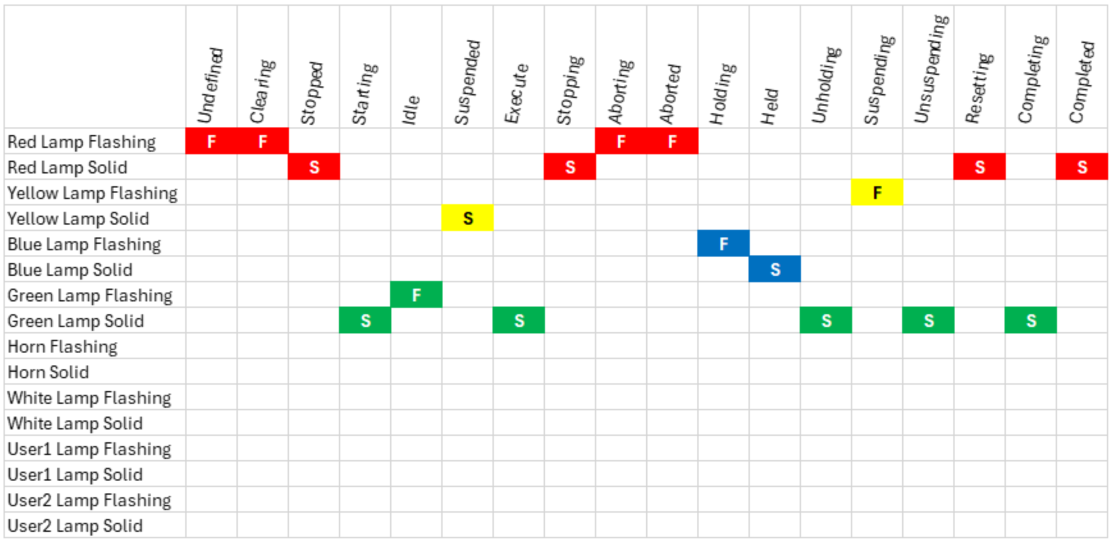
NOTE

The visualization of OMAC state model of the SIMATIC OMAC PackML V2022 library (LPML) \LPML\ 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 AFBasic:



	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 stack light

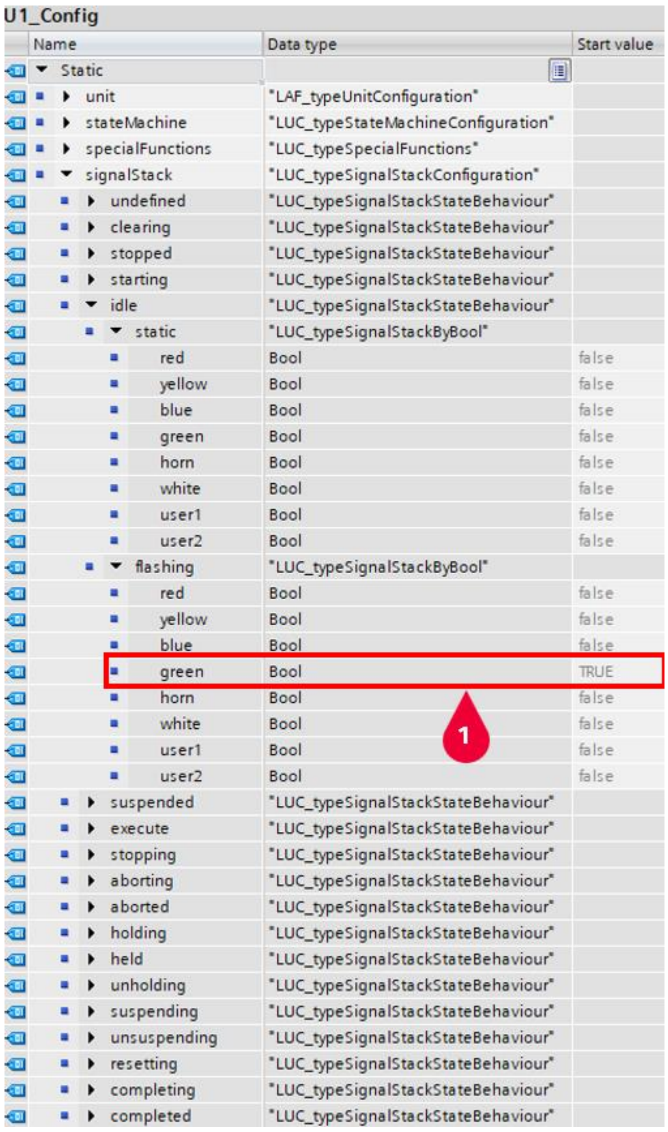
The AFBasic offers a preconfigured implementation of such a unit stack light. This implementation is not arbitrary; it is robustly designed in accordance with IEC 60073, which governs the principles for coding information using visual (lights) and auditory signals and aligns with the OMAC guideline Part 6: PackML User interface - HMI.

The configuration of these stack lights is centralized within DB "Config", this DB allows for defining different signal behaviors, primarily static (a solid light) and flashing. For each of these behaviors, the user can precisely determine the status of each individual signal using a simple Bool true/false value. The available signals include a standard color palette (red, yellow, blue, green, white), an audible horn, and additional user-definable signals (user1, user2).

For example, to configure the machine to display a flashing green light when the unit is in idle state, the user would navigate to signalStack -> idle -> flashing and set the "green" variable to true as shown on the figure below (1).

NOTICE

Please have in mind that the stack light functionality has not been fully implemented in the AF project. It was inserted as a suggestion. The image was implemented without stack light symbols in the visualization.



U1_Config

Name	Data type	Start value
▼ Static		
▸ unit	"LAF_typeUnitConfiguration"	
▸ stateMachine	"LUC_typeStateMachineConfiguration"	
▸ specialFunctions	"LUC_typeSpecialFunctions"	
▼ signalStack	"LUC_typeSignalStackConfiguration"	
▸ undefined	"LUC_typeSignalStackStateBehaviour"	
▸ clearing	"LUC_typeSignalStackStateBehaviour"	
▸ stopped	"LUC_typeSignalStackStateBehaviour"	
▸ starting	"LUC_typeSignalStackStateBehaviour"	
▼ idle	"LUC_typeSignalStackStateBehaviour"	
▼ static	"LUC_typeSignalStackByBool"	
red	Bool	false
yellow	Bool	false
blue	Bool	false
green	Bool	false
horn	Bool	false
white	Bool	false
user1	Bool	false
user2	Bool	false
▼ flashing	"LUC_typeSignalStackByBool"	
red	Bool	false
yellow	Bool	false
blue	Bool	false
green	Bool	TRUE
horn	Bool	false
white	Bool	false
user1	Bool	false
user2	Bool	false
▸ suspended	"LUC_typeSignalStackStateBehaviour"	
▸ execute	"LUC_typeSignalStackStateBehaviour"	
▸ stopping	"LUC_typeSignalStackStateBehaviour"	
▸ aborting	"LUC_typeSignalStackStateBehaviour"	
▸ aborted	"LUC_typeSignalStackStateBehaviour"	
▸ holding	"LUC_typeSignalStackStateBehaviour"	
▸ held	"LUC_typeSignalStackStateBehaviour"	
▸ unholding	"LUC_typeSignalStackStateBehaviour"	
▸ suspending	"LUC_typeSignalStackStateBehaviour"	
▸ unsuspending	"LUC_typeSignalStackStateBehaviour"	
▸ resetting	"LUC_typeSignalStackStateBehaviour"	
▸ completing	"LUC_typeSignalStackStateBehaviour"	
▸ completed	"LUC_typeSignalStackStateBehaviour"	

Figure 9-14: Stack light configuration in DB “Config”

9.6. How to add a new Unit in the PLC

To add a new Unit, an existing Unit Folder can be used to copy and paste into the Program blocks folders, following the numbering conversion as per **Figure: 9-15**, where Folders numbered 10 and above contain units and their associated equipment modules, following a specific numbering format for clarity and ease of access.

The format is as follows:

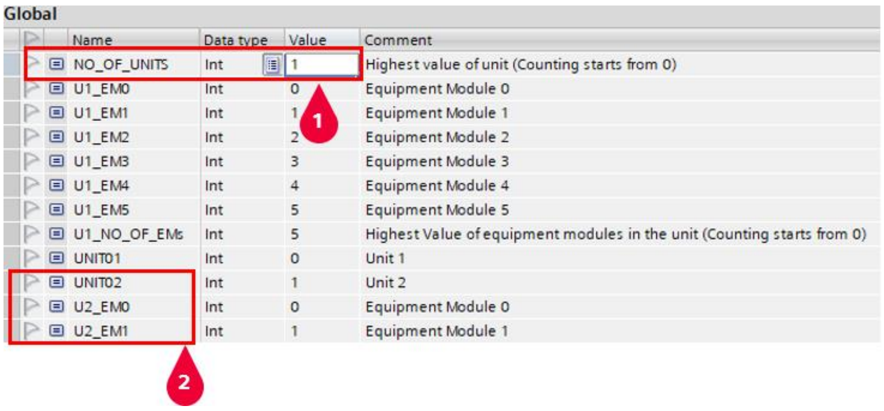
First Digit: Represents the unit number.

Second Digit: Represents the equipment module number associated with that unit.

Example Breakdown can be the "12" in a folder named "12_Em2_Template_SCL" indicates:

Unit 1 (first digit = 1); Equipment Module 2 (second digit = 2). As an example, another unit called U2_NewUnit will be added and two Ems to demo the numbering conversion.

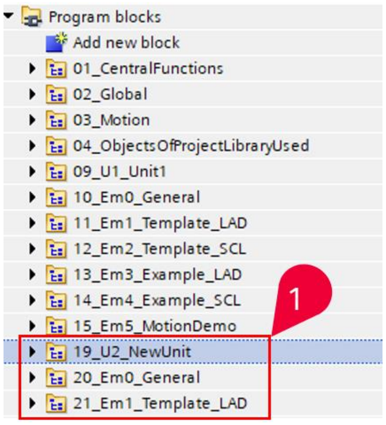
- In the program blocks, go to PLC tags and select the Global PLC tag table. Within user constants tab, the following steps have to be taken



Name	Data type	Value	Comment
NO_OF_UNITS	Int	1	Highest value of unit (Counting starts from 0)
U1_EM0	Int	0	Equipment Module 0
U1_EM1	Int	1	Equipment Module 1
U1_EM2	Int	2	Equipment Module 2
U1_EM3	Int	3	Equipment Module 3
U1_EM4	Int	4	Equipment Module 4
U1_EM5	Int	5	Equipment Module 5
U1_NO_OF_EMs	Int	5	Highest Value of equipment modules in the unit (Counting starts from 0)
UNIT01	Int	0	Unit 1
UNIT02	Int	1	Unit 2
U2_EM0	Int	0	Equipment Module 0
U2_EM1	Int	1	Equipment Module 1

Figure 9-15: Global PLC Tag list

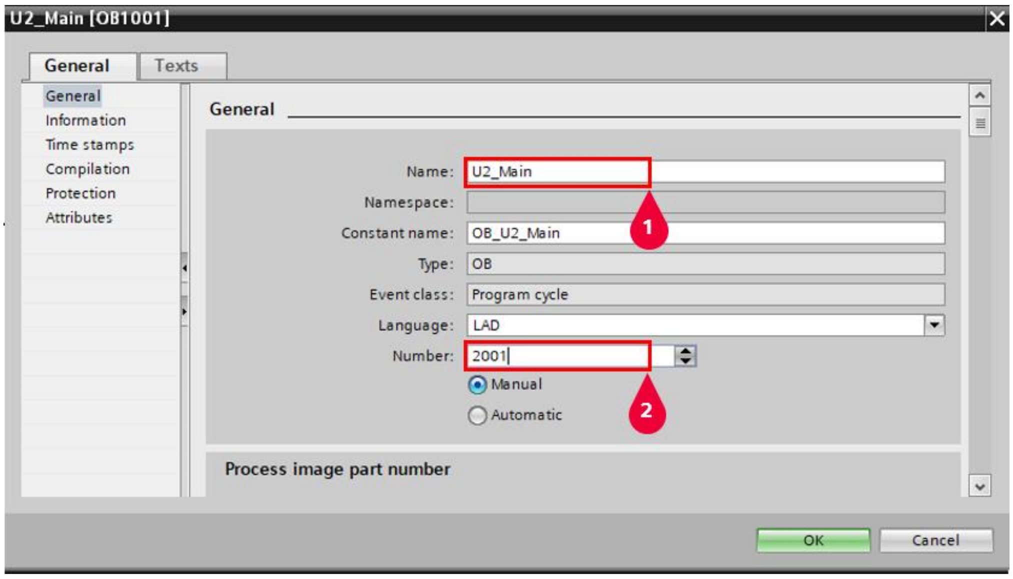
- Increase NO_OF_UNITS by 1(1) so that is shows the highest value of the last unit within the whole machine.
- Add a new constant with the name of the additional unit and assign the next possible higher number(2).
- Copy an existing unit Folder of the AFB and paste it to the project by right clicking on the program blocks in the PLC project tree and select paste.
- Rename the folder with a proper name and number following the conversion descussed on the General structure (see Figure:9-15 below).



Program blocks
- Add new block
- 01_CentralFunctions
- 02_Global
- 03_Motion
- 04_ObjectsOfProjectLibraryUsed
- 09_U1_Unit1
- 10_Em0_General
- 11_Em1_Template_LAD
- 12_Em2_Template_SCL
- 13_Em3_Example_LAD
- 14_Em4_Example_SCL
- 15_Em5_MotionDemo
- 19_U2_NewUnit
- 20_Em0_General
- 21_Em1_Template_LAD

Figure 9-16: Adding new unit to program block's structure

- Open the OB "U#_Main" and OB "U#_Startup" Properties of the new unit (see Figure:9-16 below) and rename to the applicable name of the Unit (1). Change the OB numbers of the OB "U#_Main" and "U#_Startup" (2) according to the OB numbering convention presented in chapter: Startup and cyclic program execution


U2_Main [OB1001]

General

Name: U2_Main
Namespace:
Constant name: OB_U2_Main
Type: OB
Event class: Program cycle
Language: LAD
Number: 2001
[✓ Manual] Automatic

Figure 9-17: Properties of the new Unit OB

- Rename the rest of the blocks to reflect the newly added unit.

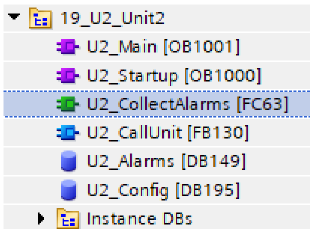
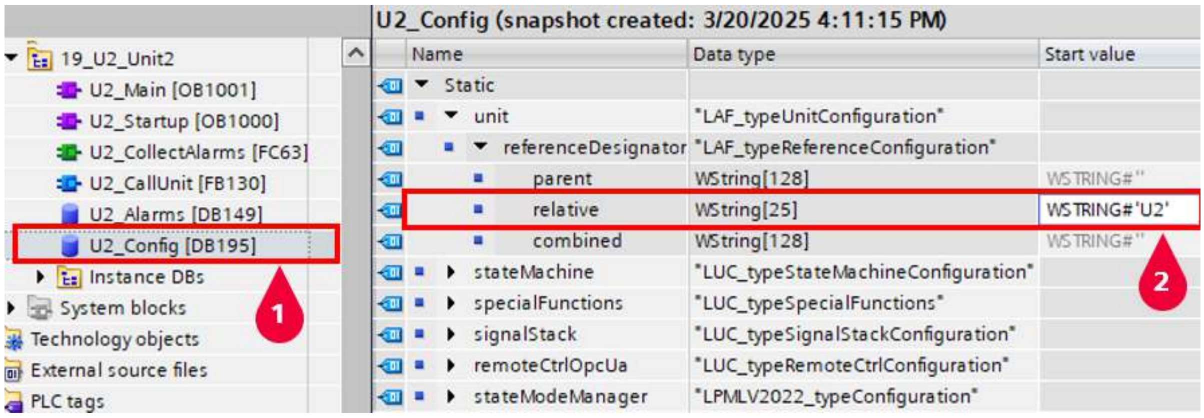

19_U2_Unit2
U2_Main [OB1001]
U2_Startup [OB1000]
U2_CollectAlarms [FC63]
U2_CallUnit [FB130]
U2_Alarms [DB149]
U2_Config [DB195]
Instance DBs

Figure 9-18: Newly added Unit block

- Open the "Config" (1) DB of the new unit and adjust the "referenceDesignator.relative" (2) to reflect the new unit.



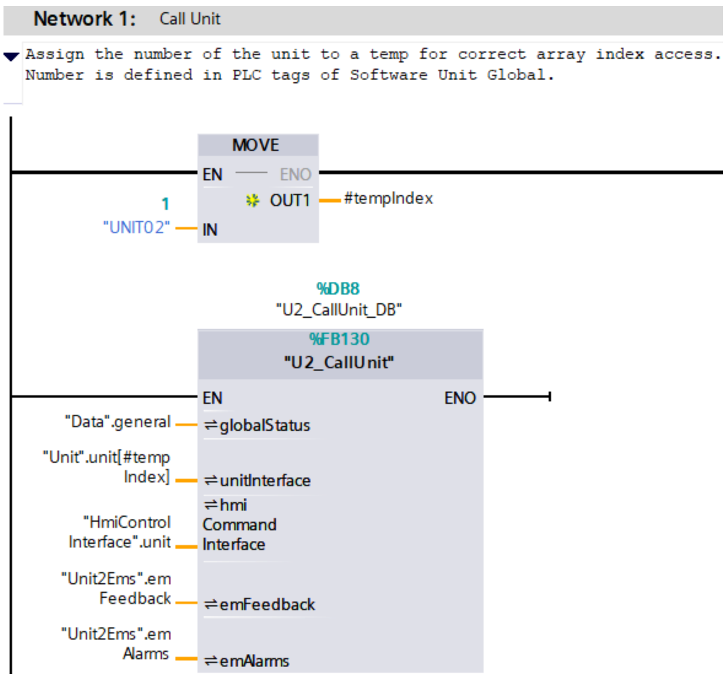
U2_Config (snapshot created: 3/20/2025 4:11:15 PM)

Name	Data type	Start value
Static		
unit	"LAF_typeUnitConfiguration"	
referenceDesignator	"LAF_typeReferenceConfiguration"	
parent	WString[128]	WSTRING#''
relative	WString[25]	WSTRING#'U2'
combined	WString[128]	WSTRING#''
stateMachine	"LUC_typeStateMachineConfiguration"	
specialFunctions	"LUC_typeSpecialFunctions"	
signalStack	"LUC_typeSignalStackConfiguration"	
remoteCtrlOpcUa	"LUC_typeRemoteCtrlConfiguration"	
stateModeManager	"LPMLV2022_typeConfiguration"	

Figure 9-19: Reference Designator adjustment

- Open the OB "Main" of the new unit and assign the user constant that was created in the Global Folder at the Move-block that is called before the FB "CallUnit" is called

- Assing the right Global interface for the unit, by connecting the corresponding interfaces to the “Data”, “Unit”, “HmiControl”, “UnitEms”, and “UnitAlarms”.

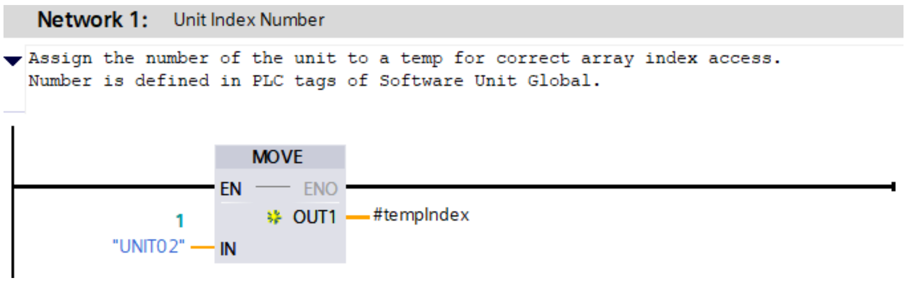


Network 1: Call Unit

```
Assign the number of the unit to a temp for correct array index access.
Number is defined in PLC tags of Software Unit Global.
```

Figure 9-20: OB “Main” of the new Unit

- Open the OB "Startup" of the new unit and assign the user constant that was created in the Global Folder at the Move-block in Network 1: "Unit Index Number".



Network 1: Unit Index Number

```
Assign the number of the unit to a temp for correct array index access.
Number is defined in PLC tags of Software Unit Global.
```

Figure 9-21: OB “Startup” of the new Unit

- The unit can then be configured as required.

9.7. 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.

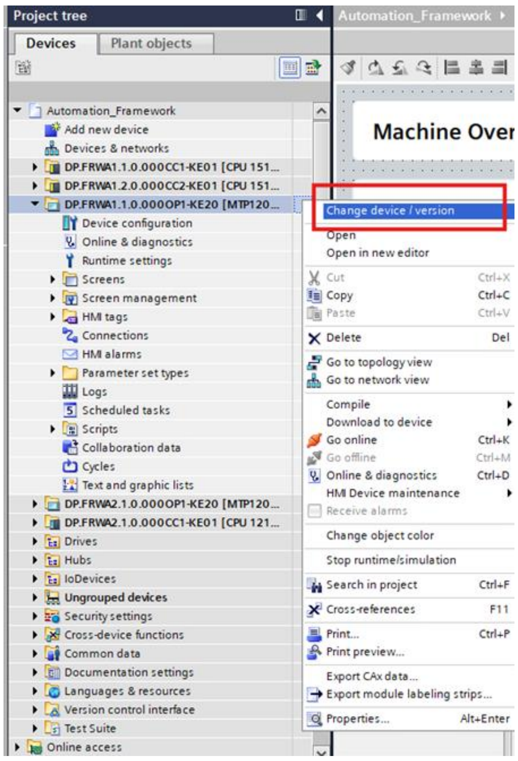
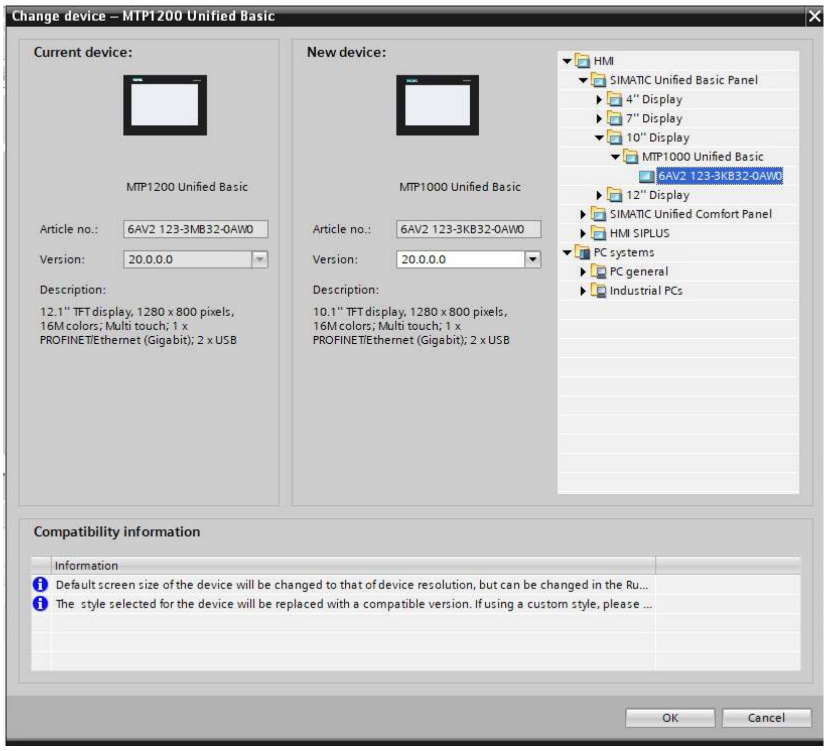


Figure 9-40: Selection of the “Change device/version” property of the HMI


Change device – MTP1200 Unified Basic

Current device:

MTP1200 Unified Basic

Article no.: 6AV2 123-3MB32-0AW0

Version: 20.0.0.0

Description:

12.1" TFT display, 1280 x 800 pixels, 16M colors; Multi touch; 1 x PROFINET/Ethernet (Gigabit); 2 x USB

New device:

MTP1000 Unified Basic

Article no.: 6AV2 123-3KB32-0AW0

Version: 20.0.0.0

Description:

10.1" TFT display, 1280 x 800 pixels, 16M colors; Multi touch; 1 x PROFINET/Ethernet (Gigabit); 2 x USB

Compatibility information

Information
Default screen size of the device will be changed to that of device resolution, but can be changed in the Ru...
The style selected for the device will be replaced with a compatible version. If using a custom style, please ...

Figure 9-41: 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:

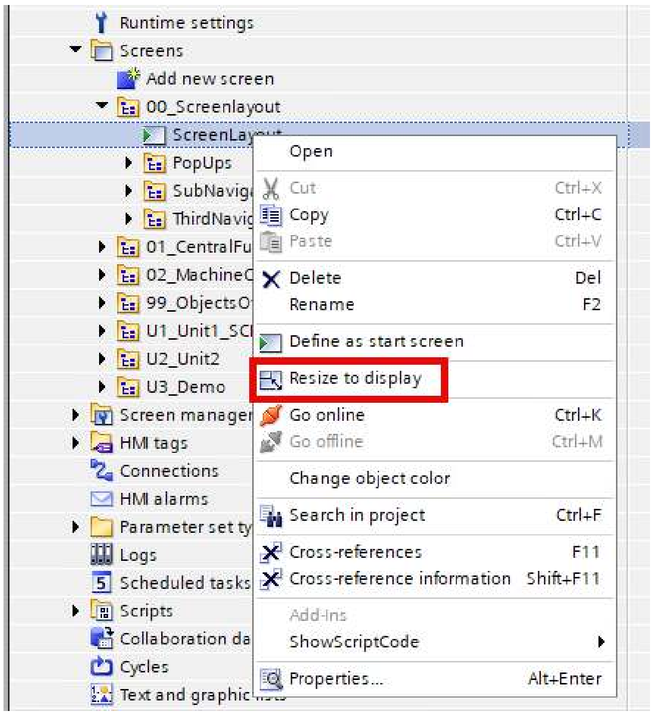


Figure 9-42: “Resize to display” property

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.

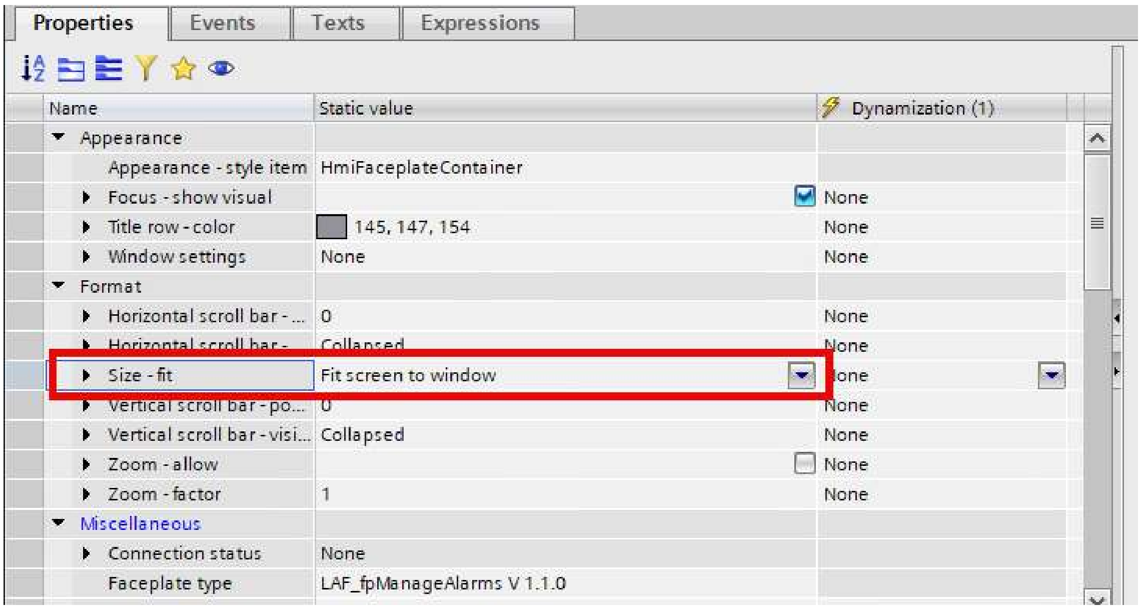
3. Adjusting Faceplate Containers

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.



Name	Static value	Dynamization (1)
Appearance		
Appearance - style item	HmiFaceplateContainer	
Focus - show visual	✓	None
Title row - color	145, 147, 154	None
Window settings	None	None
Format		
Horizontal scroll bar - ...	0	None
Horizontal scroll bar -	Collapsed	None
Size - fit	Fit screen to window	None
Vertical scroll bar - po...	0	None
Vertical scroll bar - visi...	Collapsed	None
Zoom - allow		None
Zoom - factor	1	None
Miscellaneous		
Connection status	None	
Faceplate type	LAF_fpManageAlarms V 1.1.0	

Figure 9-43: "Fit to screen window" property in the faceplate's properties

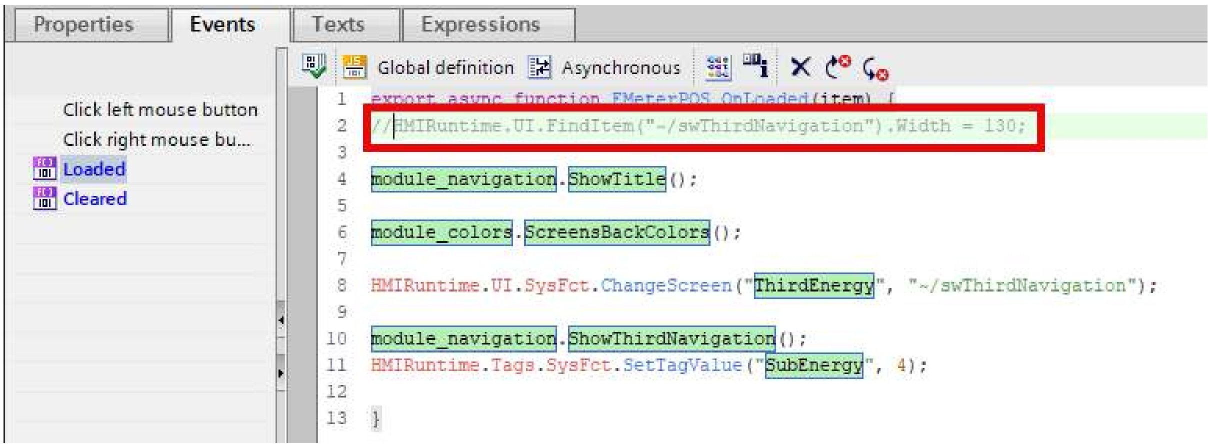
4. Managing Static Size Values

Some objects or elements may have static size values defined by the user, 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.



```
export async function FMeterPOS_OnLoaded(item) {
//HMIRuntime.UI.FindItem("~/swThirdNavigation").Width = 130;

module_navigation.ShowTitle();

module_colors.ScreensBackColors();

HMIRuntime.UI.SysFct.ChangeScreen("ThirdEnergy", "~/swThirdNavigation");

module_navigation.ShowThirdNavigation();
HMIRuntime.Tags.SysFct.SetTagValue("SubEnergy", 4);

}
```

Figure 9-44: 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

This chapter describes the general structure, interfaces, commands, and data flow of equipment modules.

10.1. General structure

An EM always consists of exactly one folder 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 .
- The general evaluation of whether and when the CMs, that represent actuators, are reset and activated takes place before the CMs are called.
- The module's alarms and events are processedSignal Flow.
- Finally, The feedback of the completion of the current state and the commands to the Unit will then be built.

General mode and state are coming from the unit level. This determines if the equipment level runs in automatic mode, maintenance or manual 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 AFBasic, 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 and SCL.

The sequence communicates with the control modules using 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”.

Signal Flow

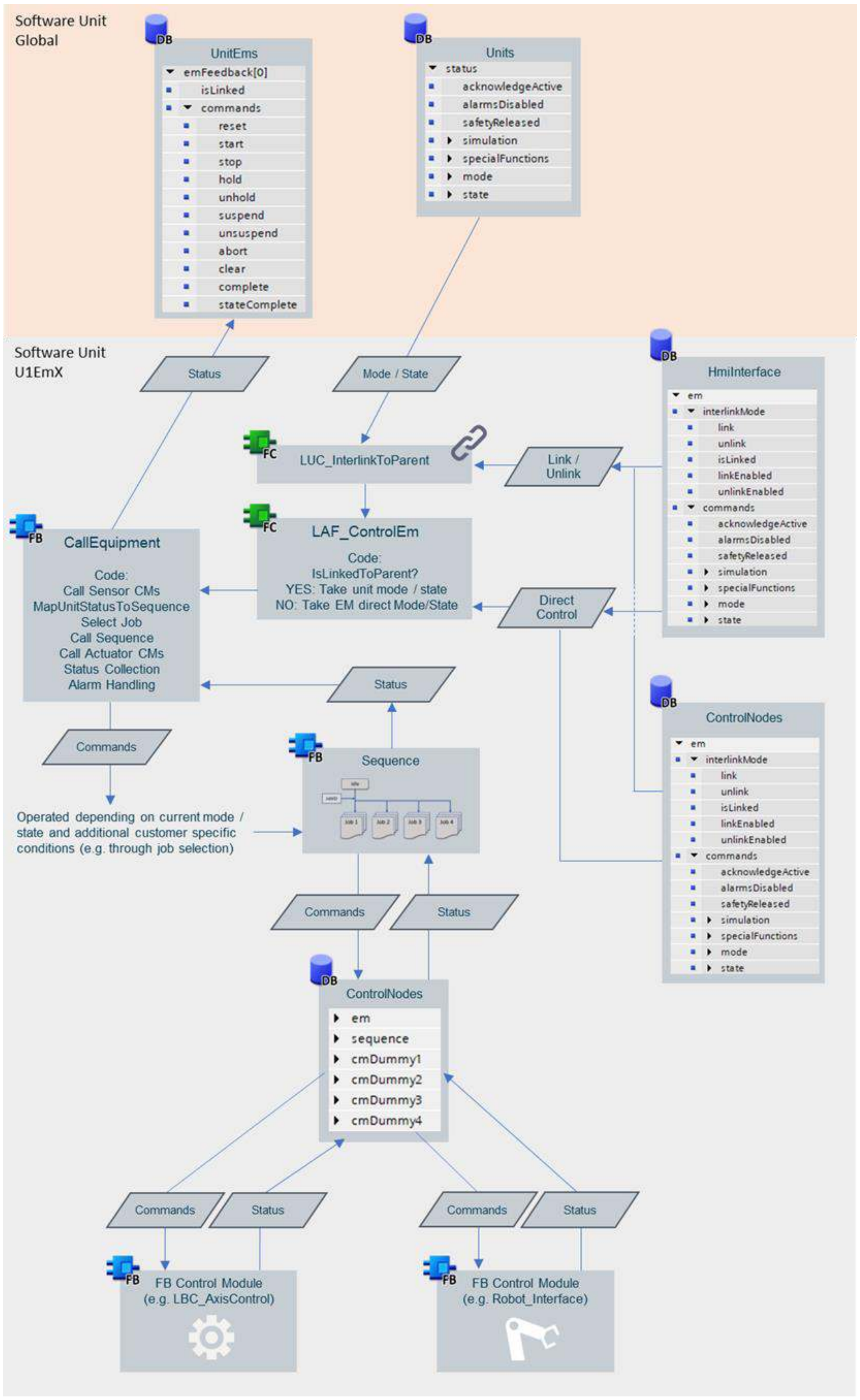


Figure 10-1: Signal flow in an equipment module

10.2. Interfaces and blocks

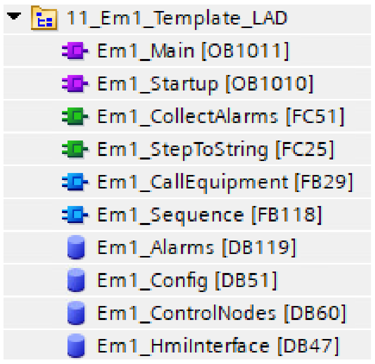
The main responsibilities of the equipment modules include:

- 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 folders that represents 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.



11_Em1_Template_LAD
- Em1_Main [OB1011]
- Em1_Startup [OB1010]
- Em1_CollectAlarms [FC51]
- Em1_StepToString [FC25]
- Em1_CallEquipment [FB29]
- Em1_Sequence [FB118]
- Em1_Alarms [DB119]
- Em1_Config [DB51]
- Em1_ControlNodes [DB60]
- Em1_HmiInterface [DB47]

Figure 10-2: 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.

OB Main

In OB "Main", FB "CallEquipment" is called, which contains the call of all further blocks and code. All data from outside the own folder, e.g. from "Global" is handed over via the interface of FB "CallEquipment". Data within the own folder is not handed over via the block interface, but handeled inside the block directly by using direct data block access.

DB Alarms

Contains alarms that can be configured by the used. They are evaluated by FC "CollectAlarms".

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.

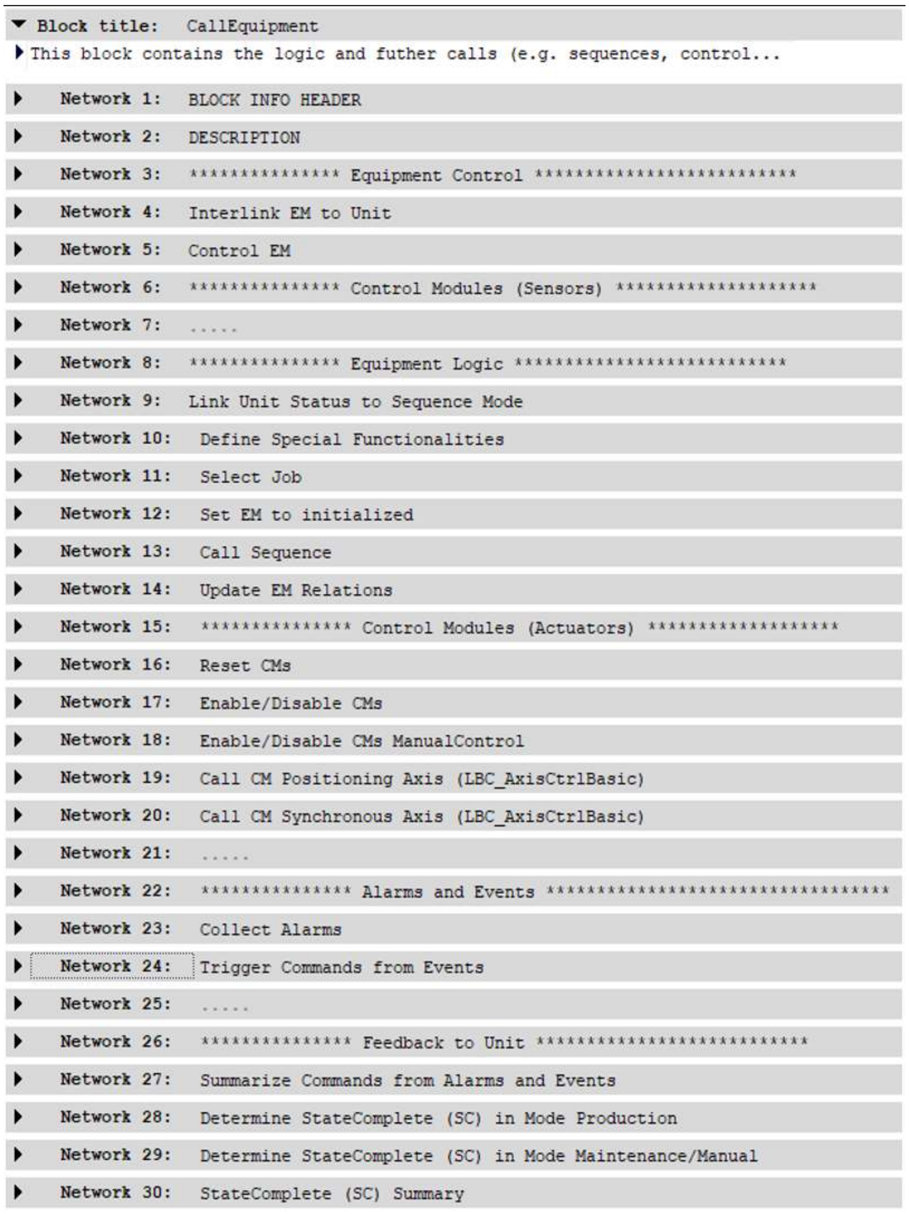
FC StepToString

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.

FB CallEquipment performs the followings actions



```
Block title:   CallEquipment
This block contains the logic and futher calls (e.g. sequences, control...

Network 1:   BLOCK INFO HEADER
Network 2:   DESCRIPTION
Network 3:   *************** Equipment Control *************************
Network 4:   Interlink EM to Unit
Network 5:   Control EM
Network 6:   *************** Control Modules (Sensors) ********************
Network 7:   .....
Network 8:   *************** Equipment Logic ***************************
Network 9:   Link Unit Status to Sequence Mode
Network 10:   Define Special Functionalities
Network 11:   Select Job
Network 12:   Set EM to initialized
Network 13:   Call Sequence
Network 14:   Update EM Relations
Network 15:   *************** Control Modules (Actuators) *******************
Network 16:   Reset CMs
Network 17:   Enable/Disable CMs
Network 18:   Enable/Disable CMs ManualControl
Network 19:   Call CM Positioning Axis (LBC_AxisCtrlBasic)
Network 20:   Call CM Synchronous Axis (LBC_AxisCtrlBasic)
Network 21:   .....
Network 22:   *************** Alarms and Events *********************************
Network 23:   Collect Alarms
Network 24:   Trigger Commands from Events
Network 25:   .....
Network 26:   *************** Feedback to Unit ***************************
Network 27:   Summarize Commands from Alarms and Events
Network 28:   Determine StateComplete (SC) in Mode Production
Network 29:   Determine StateComplete (SC) in Mode Maintenance/Manual
Network 30:   StateComplete (SC) Summary
```

Figure 10-3: Content of FB “CallEquipment” based on the example of Em5 Motion Demo

Block Info Header

This header is part of code2Docu, it serves as a quick reference for users to identify the block’s basic properties and version tracking.

Description

This network provides a more detailed textual explanation of the block's functionality, purpose, and implementation details. It serves as comprehensive documentation that would typically include detailed explanation of the block's purpose, summary of its functionality, description of inputs/outputs and their purposes, special considerations or operational notes references to related documentation or standards, dependencies on other blocks or systems, usage instructions for maintenance personnel. This network does not contain executable code but is essential documentation element that help users understand the purpose and functionality of the equipment module before proceeding to the actual implementation code.

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 or from the DB “ControlNodes”. 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 .

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 is 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.

Trigger Commands from Events

Not all events need to trigger an alarm but trigger a command. Therefore, the user can define their events that trigger a command to the unit without being assigned to an alarm.

Summarize Commands from Alarms and Events

This block derives commands from alarms and events. These bit alarms are processed and generate the corresponding reaction (e.g. abort, stop, hold). These are handed over as commands to the status collector.

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.

10.3. Hmi Screen

The equipment screens are fully customizable and should be tailored to the specific requirements of each machine. In this demonstration, simplified example implementations are provided for the following Equipment Module: EM Motion Demo.

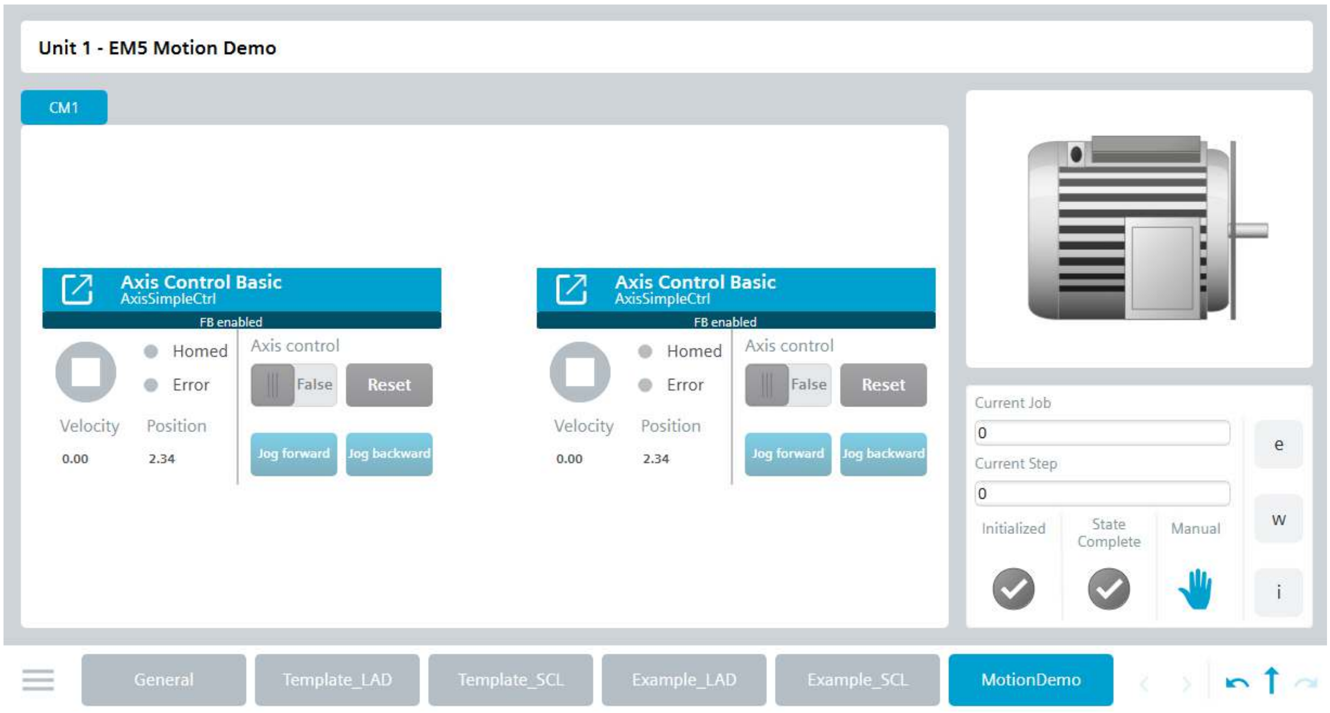


Figure 10-4: Equipment screen of EmMotionDemo

The equipment screen includes several faceplates that allow manual control and monitoring of key functions:

- Jobs
- Sequence
- Linking state, and—if the EM is unlinked—simulated unit mode and state.
- Control Modules

The faceplates for controlling sequence, jobs, and linking state can be accessed by pressing the hand icon located at the bottom-right corner of the screen. This action opens a popup window, as shown in the following figure:

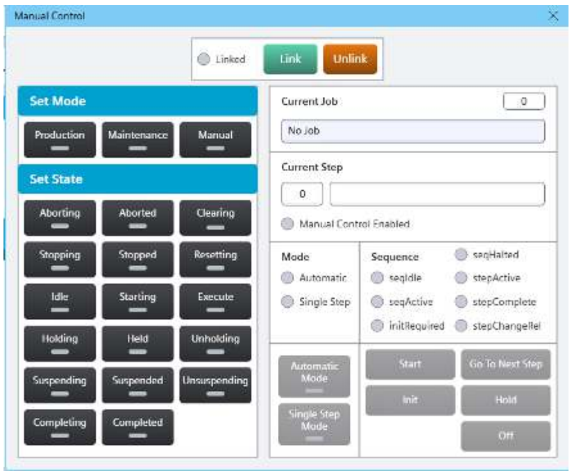


Figure 10-5: Manual Control Faceplates of the EM

10.4. Interlink to unit

The AF$_{Basic}$ 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.

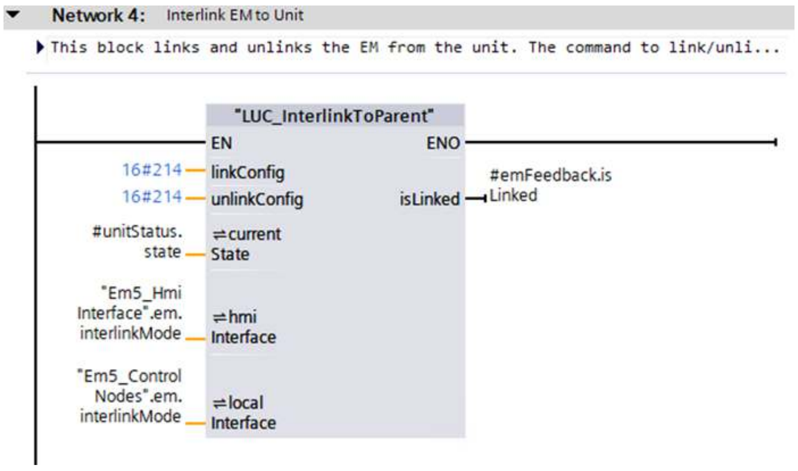

Network 4: Interlink EM to Unit
This block links and unlinks the EM from the unit. The command to link/unli...
"LUC_InterlinkToParent"
EN
ENO
16#214
linkConfig
16#214
unlinkConfig
isLinked
#emFeedback.isLinked
#unitStatus.state
⇌currentState
"Em5_HmiInterface".em.interlinkMode
⇌hmiInterface
"Em5_ControlNodes".em.interlinkMode
⇌localInterface

Figure 10-6: 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 by The DB “HmiInterface” or from the DB “ControlNodes”.

The process of linking or unlinking is only possible when the parent, the Unit that contains the respective equipment module, is in a certain active state. The "linkConfig" and "unlinkConfig" inputs allow the user to decide which states can be enabled for linking or unlinking of an EM, respectively. Each state is represented by a bit inside the configuration, and the OMAC enumeration is used for this representation (bit 0: Undefined; bit 1: Clearing; bit 2: Stopped; etc.). The default value of these configuration parameters is 16#214, which enables the states of Idle, Stopped and Aborted for both linking and unlinking of an EM.

Link an Equipment Module to a Unit:

Connect a logic to the input link or send a link command from the HMI or from DB “ControlNodes”.

Unlink an Equipment Module from a Unit:

Connect a logic to the input unlink or send a unlink command from the HMI or from DB “ControlNodes”.

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 from the unitStatus is applied to the EM. The mode, the state and special functions can be controlled directly by the “localInterface” if “enModeStateCtrlByLocal” control interface is enabled or by the operator through the “hmiInterface” if “enModeStateCtrlByHmi” control interface is enabled. That means that any mode, state and special function can be given to the EM. In case both control interfaces are enabled, local interface has priority than HMI. There is no state machine in the EM, so no automatic mode, state or special function 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.

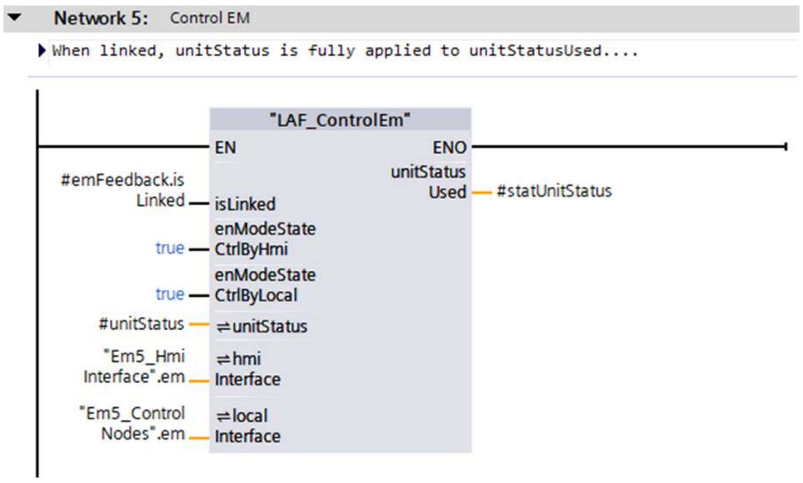
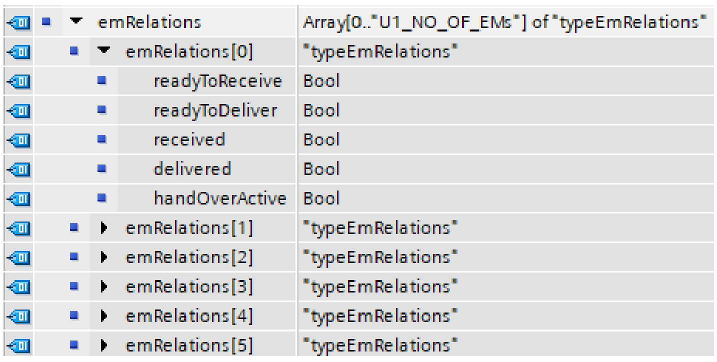

Network 5: Control EM
When linked, unitStatus is fully applied to unitStatusUsed....
"LAF_ControlEm"
EN
ENO
#emFeedback.isLinked — isLinked
unitStatusUsed — #statUnitStatus
true — enModeStateCtrlByHmi
true — enModeStateCtrlByLocal
#unitStatus — unitStatus
"Em5_HmiInterface".em — hmiInterface
"Em5_ControlNodes".em — localInterface

Figure 10-7: Direct control function of an EM

10.6. EM – EM link

The EM within a unit exhibits interdependency in their processing workflow. Consequently, an internal signal exchange mechanism can be implemented among the EMs. The DB 'Unit#Ems' serves as the central repository for this signal exchange process. The interfaces between individual EMs are encapsulated within the "emRelations" data structure. The figure below illustrates the DB structure used for the implementation of the "emRelations"



emRelations	Array[0.."U1_NO_OF_EMs"] of "typeEmRelations"
emRelations[0]	"typeEmRelations"
readyToReceive	Bool
readyToDeliver	Bool
received	Bool
delivered	Bool
handOverActive	Bool
emRelations[1]	"typeEmRelations"
emRelations[2]	"typeEmRelations"
emRelations[3]	"typeEmRelations"
emRelations[4]	"typeEmRelations"
emRelations[5]	"typeEmRelations"

Figure 10-8: Structure of EM Relation signals

This structure is used to transfer signals between an EM and its corresponding EMs in a sequential production process. It receives components from the source EM only after the source EM has completed its part of the production. Upon receipt, it processes these components before delivering them to the destination EM. Depending on its operational state, it activates signals such as "readyToDeliver" to the destination EM and "readyToReceive" to the source EM. Counter EMs can respond to these signals by executing Delivery jobs when the destination EM is ready to receive and executing Receiving jobs when the source EM is ready for delivery. This approach enables EMs to synchronize their workflows, ensuring that each module processes its designated portion of the production sequence before forwarding the partially completed component to the next downstream EM.

Additional signals play critical roles in this signaling framework:

- received: This signal indicates whether the EM has successfully received parts from its predecessor. It ensures that processes can adapt based on whether the requisite materials are available.
- delivered: This signal informs the system that parts have been successfully delivered to the next EM. It allows downstream processes to initiate once confirmation is received, ensuring a smooth workflow.
- handOverActive: This signal indicates whether the handover process is currently active. It helps coordinate the timing and control flow during the transfer of parts between EMs, preventing conflicts and ensuring that the process adheres to operational protocols.

The figure bellow demonstrates the operation of "emRelations" between two Ems.

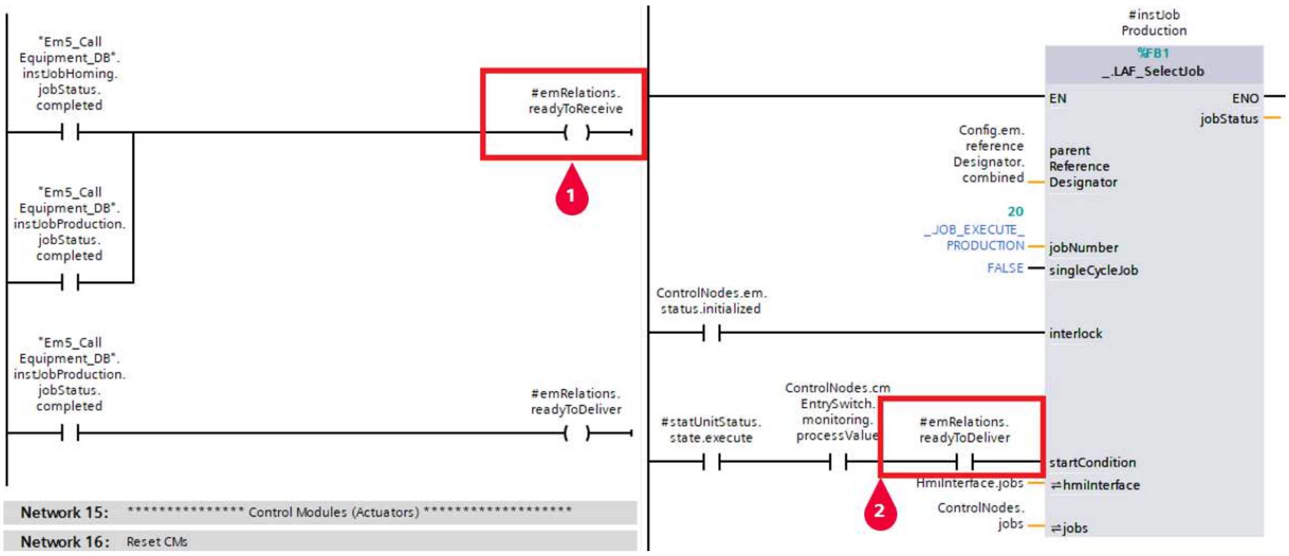

"Em5_Call Equipment_DB". instJobHoming. jobStatus. completed
"Em5_Call Equipment_DB". instJobProduction. jobStatus. completed
"Em5_Call Equipment_DB". instJobProduction. jobStatus. completed
#emRelations. readyToReceive
1
#emRelations. readyToDeliver
Network 15: *************** Control Modules (Actuators) *******************
Network 16: Reset CMs
#instJob Production
%FB1
_LAF_SelectJob
EN
ENO
jobStatus
Config.em. reference Designator. combined
parent Reference Designator
20 _JOB_EXECUTE_ PRODUCTION
jobNumber
FALSE
singleCycleJob
ControlNodes.em. status.initialized
interlock
#statUnitStatus. state.execute
ControlNodes.cm EntrySwitch. monitoring. processValue
#emRelations. readyToDeliver
startCondition
HmiInterface.jobs
hmiInterface
2
ControlNodes. jobs
jobs

Figure 10-9: Example of EM – EM link application

Demo Em above updates the counter Em regarding the Em relation status. Depending on the application, the Em will update the counter Em to indicate that it is ready to receive or deliver material (1). Additionally, the Em will utilize this status from the counter Ems as conditions for selecting jobs (2). This applies in cases where an Em is executing a specific task that ultimately requires handing over to a counter Em for further processing.

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 these jobs is made by FB "LAF_SelectJob". It provides the control of one job that the EM could execute. Therefore, every job that was defined for the EM must have its own "LAF_SelectJob" instance and a specific number (Job Number) in the user constants of the EM.

When the **startCondition** input of the FB "LAF_SelectJob" block is activated, the assigned **jobNumber** is transferred to the EM sequence, and the corresponding job branch is initiated. Each job comprises a comprehensive set of steps designed to execute a specific function. During these job steps, the interaction with the Control Module (CM) occurs, facilitating the required operational sequence.

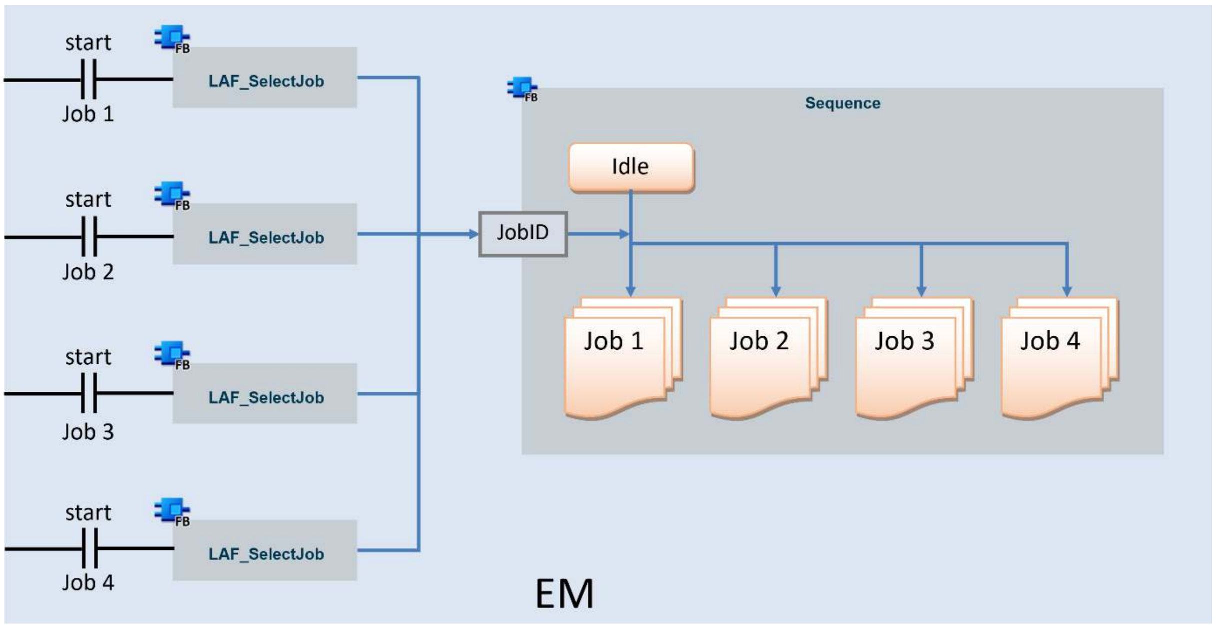


Figure 10-10: Job Management of an EM

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 AFBasic.

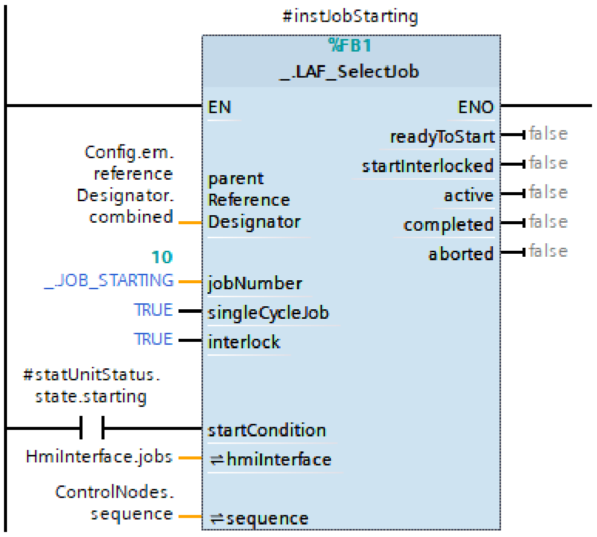


Figure 10-11: Call of the FB “LAF_SelectJob”

The **interlock** input prevents the execution of the specific job. An interlock could be e.g. a mechanical reason why the job should not be carried out. E.g. the job of moving a robot to a transfer station could be interlocked when another robot is at the same transfer station. This prevents a collision between the two robots.

The **startcondition** is the process condition, that is used by the program to start a job.

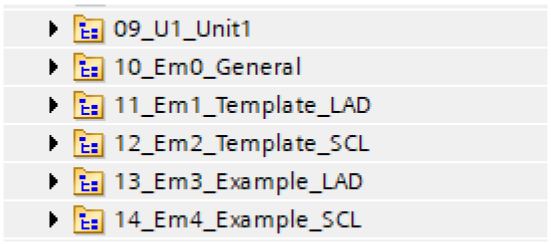
The **hmiInterface** In Out allows the operator to interact with the Job control of the EM, it consists of two primary members namely: commands which allows the user to manipulate the jobs by setting a Job number to be executed when the EM is unlinked from the unit, and status enables the job status of the EM to be visualized on the HMI through a sub-member **activeJobNumber** the current active job number can be visualized, and manual control status on the jobs through the sub-member **manualControlEnabled.** The **sequence** InOut will then transfer the data to the specific points within the PLC to share the sequence information through DB "ControlNodes".

In addition, the status of the job is available for use on the output interface of the block through:

- **readyToStart** - When not in manual mode: Ready to start when startCondition and interlock are true and When in manual mode: Ready to start when job is selected on the hmiInterface. startCondition and interlock irrelevant in that case.
- **startInterlocked** - Job could be started but interlock is FALSE.
- **active** - Job is currently active in the sequence.
- **completed** - Job has been completed and no other job has started. As soon as another job starts, this variable is reset.
- **aborted** - The sequence has been initialized while this job was active. Thus, the job was interrupted and was not completed. As soon as another job starts, this variable is reset.

10.8. Sequencers

There are Two different types of sequencers available in the Automation Framework$_{\text{Basic}}$. Depending on the requirements, a LAD or SCL sequence can be chosen. Both 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 Group folders that are shown below.



09_U1_Unit1
10_Em0_General
11_Em1_Template_LAD
12_Em2_Template_SCL
13_Em3_Example_LAD
14_Em4_Example_SCL

Figure 10-12: Group Folders 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 chapter, where the individual functions of the EM are divided into jobs. To provide the general function of the sequences and to implement the architecture without further programming effort at the EM level, the Automation Framework Basic offers the possibility to use a control and a monitor FB for each of the two programming languages mentioned above.

LAD

Within the function block of the sequenecer, two extra blocks, FB "LAF_ControlLadSequence" to the permanent pre-instruction network and sets the sequence mode, initializes, holds and changes the sequencer to single step mode.

The following figure shows the call of the block with the connections that must be made to map the signals of the DB "HmiInterface" and DB "ControlNodes" to the function block.

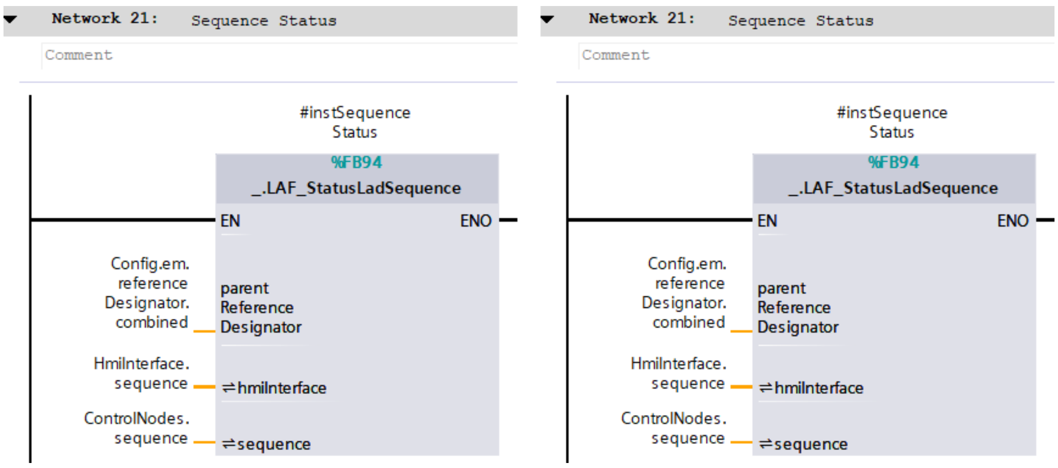
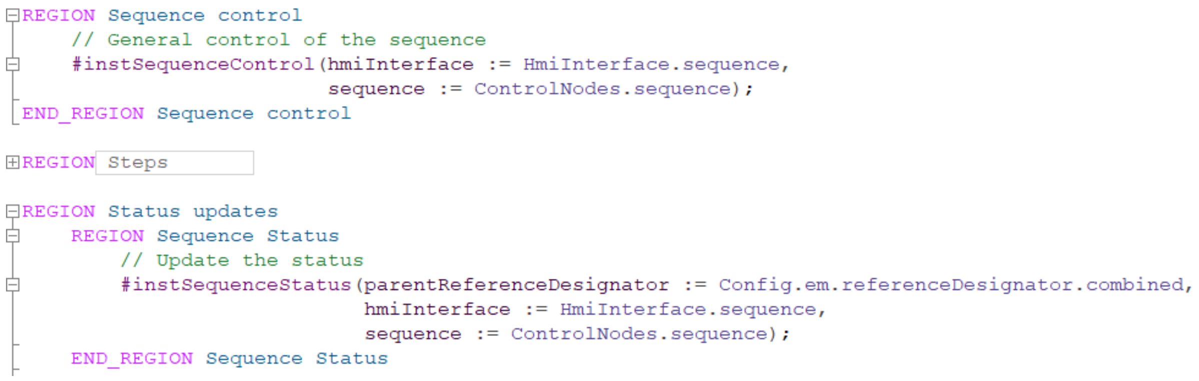


Figure 10-13: Call of the FBs "LAF_ControlLadSequence" and "LAF_StatusLadSequence"

SCL

The concept of sequence and job handling also covers the SCL programming language. The implementations of LAD and SCL is similar, as can be seen on the figure below. The interface connection for SCL is therefore the same as the one shown for the LAD blocks.



```
REGION Sequence control
    // General control of the sequence
    #instSequenceControl(hmiInterface := HmiInterface.sequence,
                         sequence := ControlNodes.sequence);
END_REGION Sequence control

REGION Steps

REGION Status updates
    REGION Sequence Status
        // Update the status
        #instSequenceStatus(parentReferenceDesignator := Config.em.referenceDesignator.combined,
                            hmiInterface := HmiInterface.sequence,
                            sequence := ControlNodes.sequence);
    END_REGION Sequence Status
```

Figure 10-14: Call of the FBs "LAF_ControlSclSequence" and "LAF_StatusSclSequence"

Post Processing

The data structure of the Job control and the provided sequencers allow us 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 additional information. The function "StepToString" just converts the step number into a string. Therefore, the block is called for the current, previous and next step.

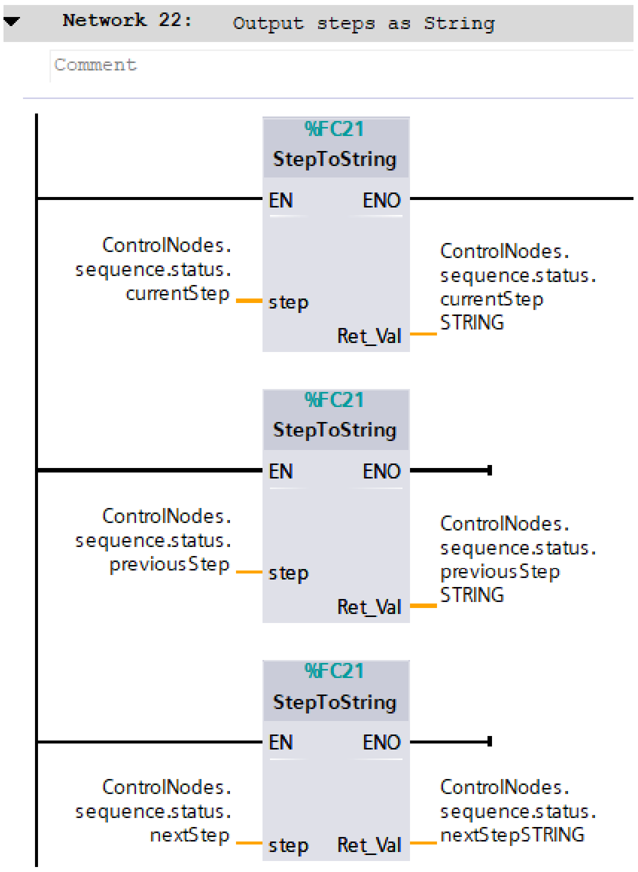


Figure 10-15: Call of the functions for additional information of the jobs and sequence

10.9. EM-Template / Example

Technical description

The template and 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$_{Basic}$ 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.

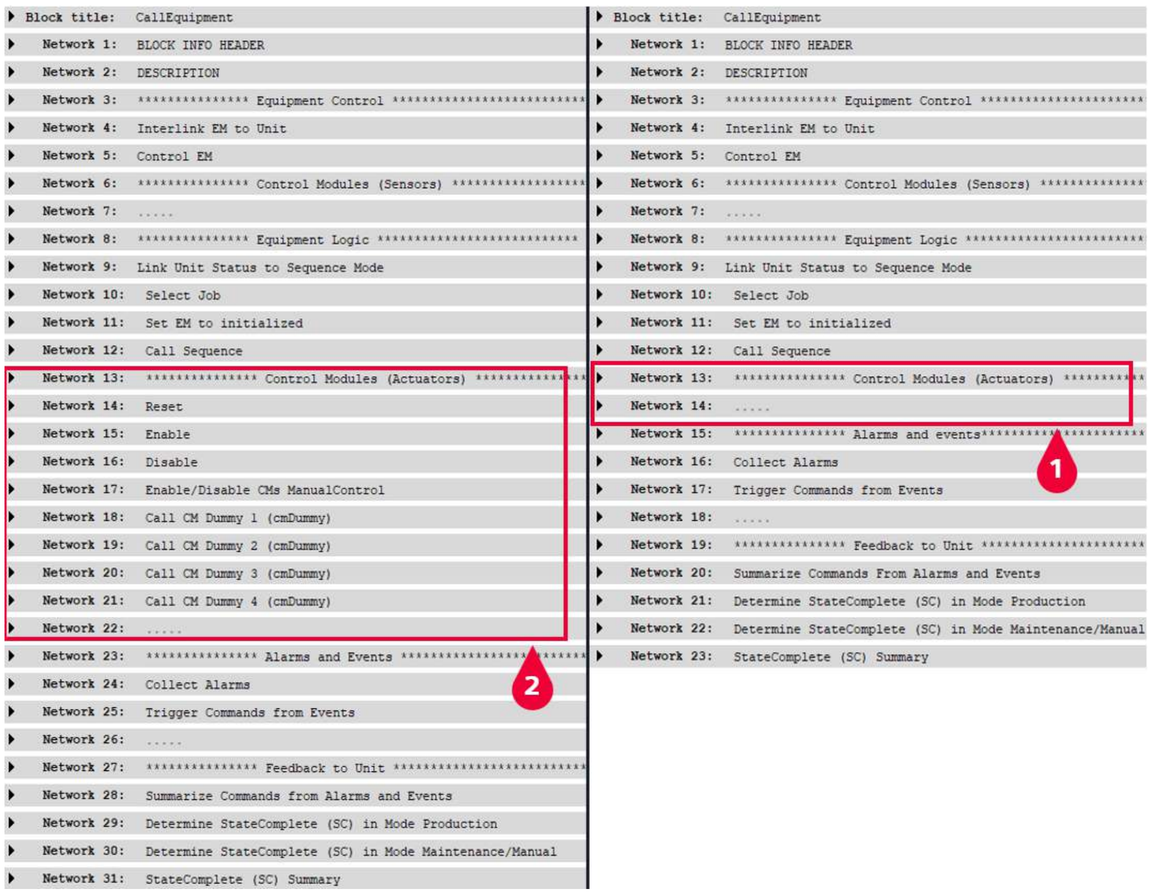
Architecture

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 of the differences between the individual EMs the following table shows their content:

Folder	Description
U1_Unit1	Programmed in LAD
Em0_General	Programmed in LAD
Em1_Template_LAD	General logic programmed in LAD; sequencer programmed in LAD
Em1_Template_SCL	General logic programmed in LAD; sequencer programmed in SCL
Em3_Example_LAD	General logic programmed in LAD; sequencer programmed in LAD
Em4_Example_SCL	General logic programmed in LAD; sequencer programmed in SCL
Em5_MotionDemo	General logic programmed in LAD; sequencer programmed in LAD

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 (2) and a sequence. The enhanced program of the function block “CallEquipment” is shown in the following figure. Notice that the template EMs have no representative CMs (1) since it is just a template.

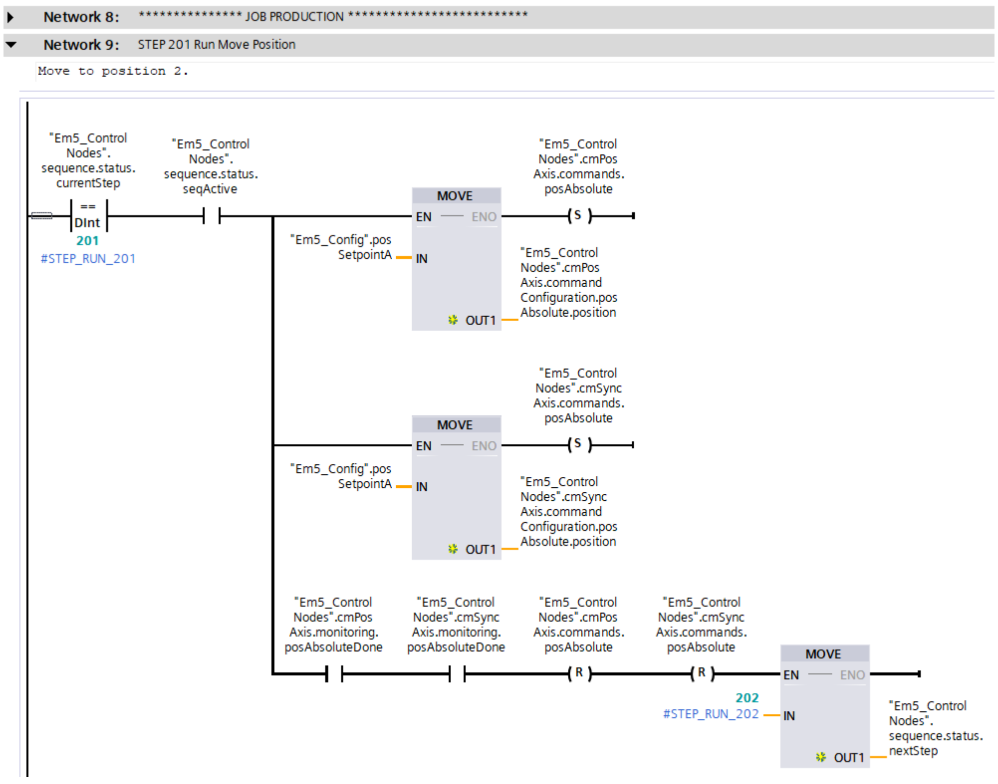


```
Block title:   CallEquipment
  Network 1:   BLOCK INFO HEADER
  Network 2:   DESCRIPTION
  Network 3:   *************** Equipment Control *************************
  Network 4:   Interlink EM to Unit
  Network 5:   Control EM
  Network 6:   *************** Control Modules (Sensors) *****************
  Network 7:   .....
  Network 8:   *************** Equipment Logic ***************************
  Network 9:   Link Unit Status to Sequence Mode
  Network 10:   Select Job
  Network 11:   Set EM to initialized
  Network 12:   Call Sequence
  Network 13:   *************** Control Modules (Actuators) **************
  Network 14:   Reset
  Network 15:   Enable
  Network 16:   Disable
  Network 17:   Enable/Disable CMs ManualControl
  Network 18:   Call CM Dummy 1 (cmDummy)
  Network 19:   Call CM Dummy 2 (cmDummy)
  Network 20:   Call CM Dummy 3 (cmDummy)
  Network 21:   Call CM Dummy 4 (cmDummy)
  Network 22:   .....
  Network 23:   *************** Alarms and Events ************************
  Network 24:   Collect Alarms
  Network 25:   Trigger Commands from Events
  Network 26:   .....
  Network 27:   *************** Feedback to Unit *************************
  Network 28:   Summarize Commands from Alarms and Events
  Network 29:   Determine StateComplete (SC) in Mode Production
  Network 30:   Determine StateComplete (SC) in Mode Maintenance/Manual
  Network 31:   StateComplete (SC) Summary
```

```
Block title:   CallEquipment
  Network 1:   BLOCK INFO HEADER
  Network 2:   DESCRIPTION
  Network 3:   *************** Equipment Control *********************
  Network 4:   Interlink EM to Unit
  Network 5:   Control EM
  Network 6:   *************** Control Modules (Sensors) **************
  Network 7:   .....
  Network 8:   *************** Equipment Logic ************************
  Network 9:   Link Unit Status to Sequence Mode
  Network 10:   Select Job
  Network 11:   Set EM to initialized
  Network 12:   Call Sequence
  Network 13:   *************** Control Modules (Actuators) ************
  Network 14:   .....
  Network 15:   *************** Alarms and events***********************
  Network 16:   Collect Alarms
  Network 17:   Trigger Commands from Events
  Network 18:   .....
  Network 19:   *************** Feedback to Unit **********************
  Network 20:   Summarize Commands From Alarms and Events
  Network 21:   Determine StateComplete (SC) in Mode Production
  Network 22:   Determine StateComplete (SC) in Mode Maintenance/Manual
  Network 23:   StateComplete (SC) Summary
```

Figure 10-16: Comparison of a Template and Example EM

The dummy CMs (2) 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 moving a 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 Demo EM of the Unit (chapter EM Demo Implementation – Motion Demo). The figure below shows the generic structure of an individual step.



Network 8: *************** JOB PRODUCTION **************************
Network 9: STEP 201 Run Move Position
Move to position 2.
"Em5_Control Nodes". sequence.status. currentStep
==
DInt
201
#STEP_RUN_201
"Em5_Control Nodes". sequence.status. seqActive
MOVE
EN ENO
"Em5_Config".pos SetpointA
IN
OUT1
"Em5_Control Nodes".cmPos Axis.commands. posAbsolute
S
"Em5_Control Nodes".cmPos Axis.command Configuration.pos Absolute.position
MOVE
EN ENO
"Em5_Config".pos SetpointA
IN
OUT1
"Em5_Control Nodes".cmSync Axis.commands. posAbsolute
S
"Em5_Control Nodes".cmSync Axis.command Configuration.pos Absolute.position
"Em5_Control Nodes".cmPos Axis.monitoring. posAbsoluteDone
"Em5_Control Nodes".cmSync Axis.monitoring. posAbsoluteDone
"Em5_Control Nodes".cmPos Axis.commands. posAbsolute
R
"Em5_Control Nodes".cmSync Axis.commands. posAbsolute
R
MOVE
EN ENO
202
#STEP_RUN_202
IN
OUT1
"Em5_Control Nodes". sequence.status. nextStep

Figure 10-17: Example for generic programming with step structure (Em5)

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 between DB "ControlNodes" and DB "HmiInterface." The "ControlNodes" database provides signals regarding the status of the EM, including interlink mode, commands to the EM, and status information. Additionally, it includes variables representing the status of the control modules (CMs) and linked hardware, such as fuses or the state of the pressurized air, which are processed within the general releases.

In contrast, the DB "HmiInterface" database contains only EM information but is supplemented by the alarm status of the EM and the User-Defined Type (UDT), enabling direct EM control through a simulated unit status.

10.11. EM Demo Implementation – Motion Demo

The equipment module “Motion Demo” is an example of how to implement motion tasks.

In motion control systems, the motion program coordinates the movement of mechanical axes through a structured execution cycle. The motion commands define the desired movement, including positioning, speed, and trajectory. Each motion cycle processes the current command, updating the target position and managing the axial movements. Different types of motion logic are integrated into the control sequence: motion-critical logic handles immediate positioning and sensor interactions, while supporting logic manages fundamental operations like enabling the system, resetting parameters, or performing homing procedures. This layered approach ensures smooth and precise motion control, allowing the system to respond to commands and environmental inputs effectively.

The example implementation contains one type of control module with the following two functionalities:

- ”LBC_AxisCtrlBasic“ Positioning Axis, to control a Axis for the Positioning Axis.
- ”LBC_AxisCrtlBasic“ Synchronous Axis, to control a synchronized Axis for the Positioning Axis.

Technical description

The Implementation of the Motion equipment module follows the same technical implementation as the other demo EMs by separating the OMAC state model from the EM level. To create modular behavior for the EM, various functionalities that the equipment can offer are divided into separate jobs. A job could encompass starting (Homing) the EM drives or executing a subprocess of production. The job and sequence management blocks are implementing this separation.

Architecture

The Motion Demo equipment module provides three 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 both the Positioning and Synchronous Axis handled by the ”LBC_AxisCtrlBasic“ block
- Aborting
  - Stop the posAxis and syncAxis and then wait until they are stopped.

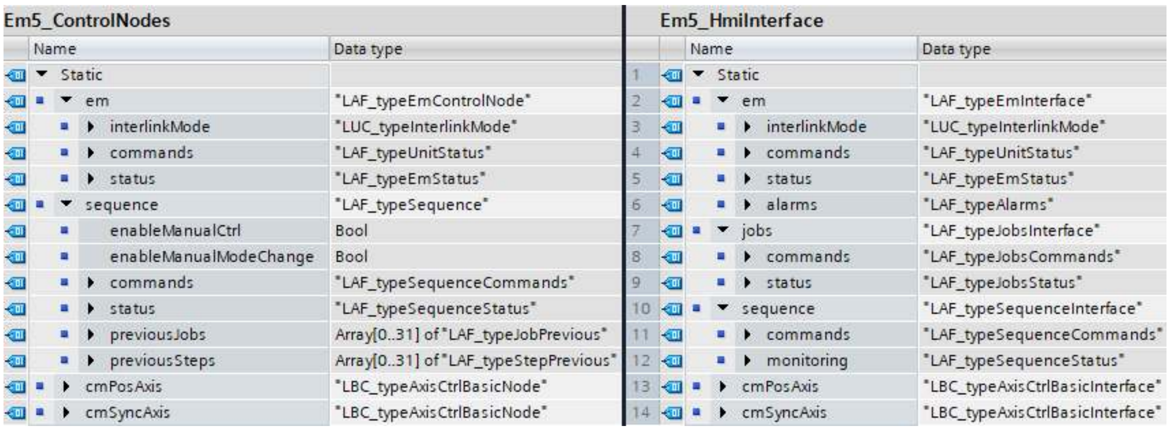
The Motion Demo equipment module is as already mentioned including two instances of the FB ”LBC_AxisCtrlBasic”. These function blocks are called as usual in the FB ”CallEquipment” within the OB ”Main” cycle of the equipment module. Thus, the existing diagnostic and HMI functionalities, which are provided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

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.

Interface function description

The primary interface of the equipment module (EM), its jobs, and the step sequencer is distributed across the DB "ControlNodes" and DB "HmiInterface". The HMI component contains commands and monitoring signals that can be connected to various operating points. The "HmiInterface" database includes a specific User-Defined Type (UDT) for EM control, allowing modification of the interlink mode and monitoring of the current alarm status.

Within the "ControlNodes" database, the interface comprises similar elements to the HMI interface, with the addition of histories of previously executed jobs or steps, along with various enable signals for jobs and sequences. Furthermore, the control modules (CMs) also store their interface data within these databases.



Em5_ControlNodes

Name	Data type
▼ Static	
▼ em	"LAF_typeEmControlNode"
▶ interlinkMode	"LUC_typeInterlinkMode"
▶ commands	"LAF_typeUnitStatus"
▶ status	"LAF_typeEmStatus"
▼ sequence	"LAF_typeSequence"
enableManualCtrl	Bool
enableManualModeChange	Bool
▶ commands	"LAF_typeSequenceCommands"
▶ status	"LAF_typeSequenceStatus"
▶ previousJobs	Array[0..31] of "LAF_typeJobPrevious"
▶ previousSteps	Array[0..31] of "LAF_typeStepPrevious"
▶ cmPosAxis	"LBC_typeAxisCtrlBasicNode"
▶ cmSyncAxis	"LBC_typeAxisCtrlBasicNode"

Em5_HmiInterface

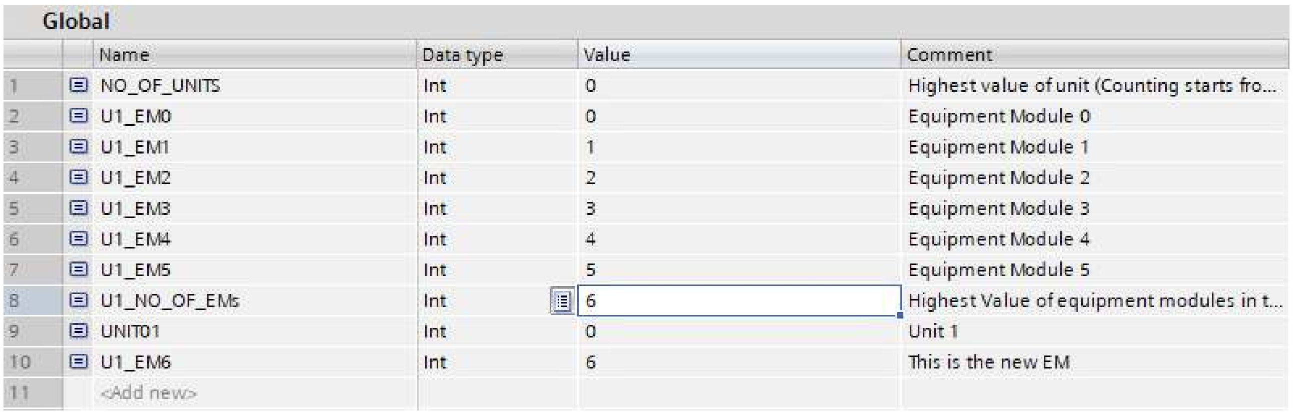
	Name	Data type
1	▼ Static	
2	▼ em	"LAF_typeEmInterface"
3	▶ interlinkMode	"LUC_typeInterlinkMode"
4	▶ commands	"LAF_typeUnitStatus"
5	▶ status	"LAF_typeEmStatus"
6	▶ alarms	"LAF_typeAlarms"
7	▼ jobs	"LAF_typeJobsInterface"
8	▶ commands	"LAF_typeJobsCommands"
9	▶ status	"LAF_typeJobsStatus"
10	▼ sequence	"LAF_typeSequenceInterface"
11	▶ commands	"LAF_typeSequenceCommands"
12	▶ monitoring	"LAF_typeSequenceStatus"
13	▶ cmPosAxis	"LBC_typeAxisCtrlBasicInterface"
14	▶ cmSyncAxis	"LBC_typeAxisCtrlBasicInterface"

Figure 10-18: Interface of the Conveyor equipment module.

10.12. How to add a new EM in the PLC

The following steps need to be performed when adding an additional equipment module. As an example, another equipment module is added to the U1_Unit1.

- Go to PLC tags and select the Global PLC tag table. Within the User constants Tab the following steps have to be taken.

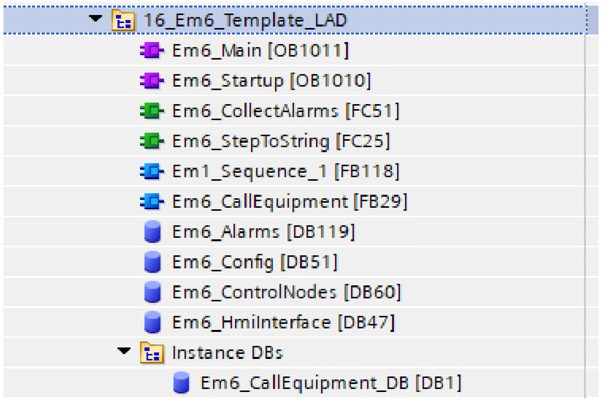


Global

	Name	Data type	Value	Comment
1	NO_OF_UNITS	Int	0	Highest value of unit (Counting starts fro...
2	U1_EM0	Int	0	Equipment Module 0
3	U1_EM1	Int	1	Equipment Module 1
4	U1_EM2	Int	2	Equipment Module 2
5	U1_EM3	Int	3	Equipment Module 3
6	U1_EM4	Int	4	Equipment Module 4
7	U1_EM5	Int	5	Equipment Module 5
8	U1_NO_OF_EMs	Int	6	Highest Value of equipment modules in t...
9	UNIT01	Int	0	Unit 1
10	U1_EM6	Int	6	This is the new EM
11	<Add new>			

Figure 10-19: Unit1 PLC Tag list

- Increase U1_NO_OF_EMs by 1 so that it 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 AFBasic that fits the best to the needs. In this case the folder "11_Em1_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 Program blocks folder in the PLC project tree and select paste.
- In the new Em Folder, adjust the name as needed.

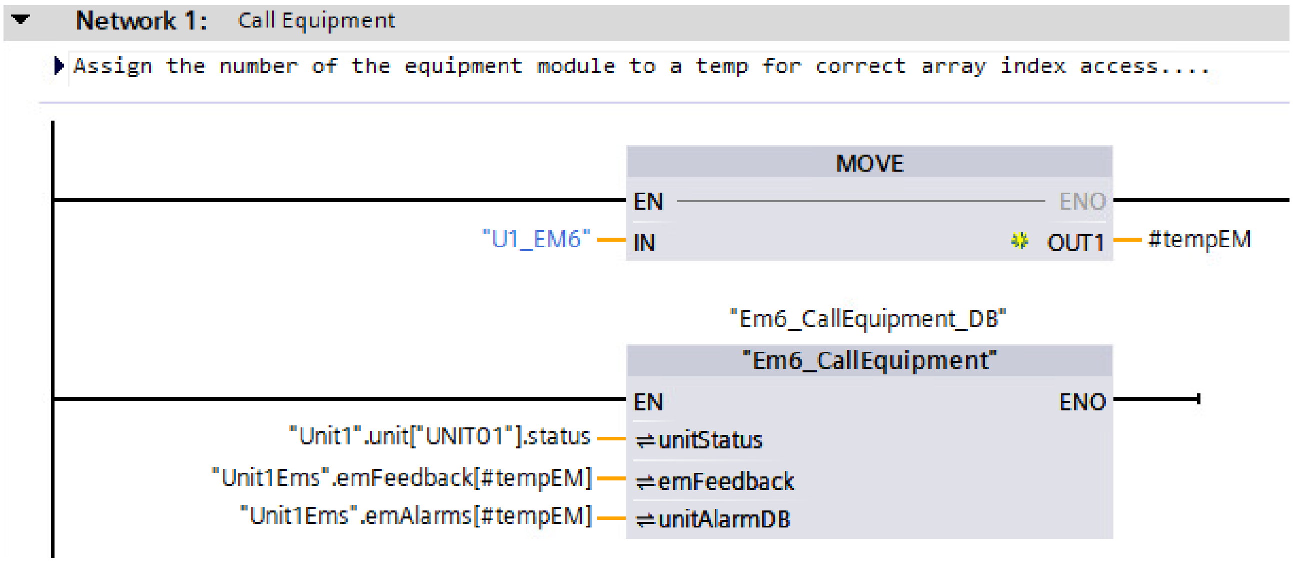


16_Em6_Template_LAD
Em6_Main [OB1011]
Em6_Startup [OB1010]
Em6_CollectAlarms [FC51]
Em6_StepToString [FC25]
Em1_Sequence_1 [FB118]
Em6_CallEquipment [FB29]
Em6_Alarms [DB119]
Em6_Config [DB51]
Em6_ControlNodes [DB60]
Em6_HmiInterface [DB47]
Instance DBs
Em6_CallEquipment_DB [DB1]

Figure 10-20: Properties of the new EM

- 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.

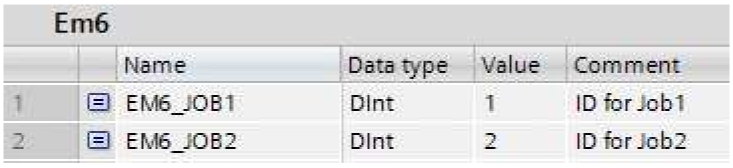
- Change the interface of the CallEquipment block to the corresponding unit signals.



```
Assign the number of the equipment module to a temp for correct array index access....
"U1_EM6" — IN   OUT1 — #tempEM
"Unit1".unit["UNIT01"].status — unitStatus
"Unit1Ems".emFeedback[#tempEM] — emFeedback
"Unit1Ems".emAlarms[#tempEM] — unitAlarmDB
```

Figure 10-21: OB Main adjustments

- Copy the PLC tag list from the EM you copied, go to the PLC tags folder and then to the Unit folder. In this case we will copy the Em1 PLC tag list and paste it in the Unit folder. Adjust the name as needed.
- 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 EM.

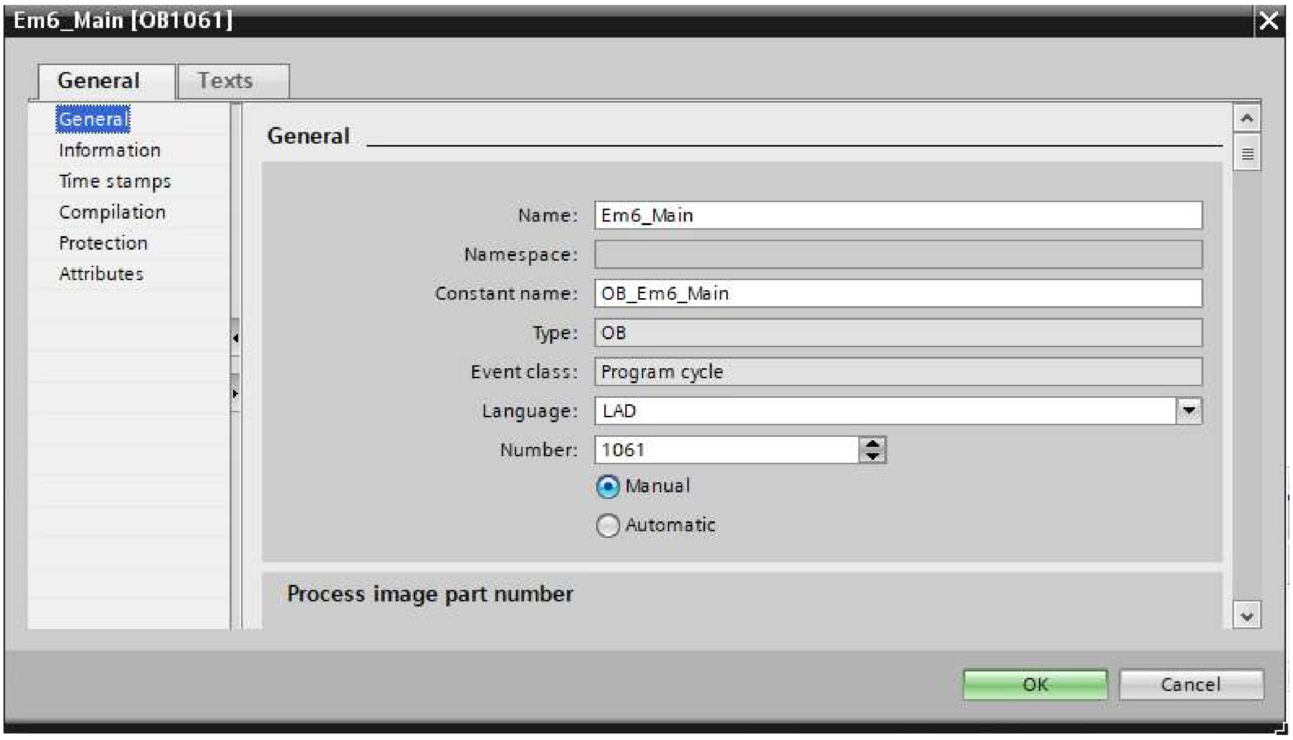


Em6

	Name	Data type	Value	Comment
1	EM6_JOB1	DInt	1	ID for Job1
2	EM6_JOB2	DInt	2	ID for Job2

Figure 10-22: 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”. Adjust the names as needed.



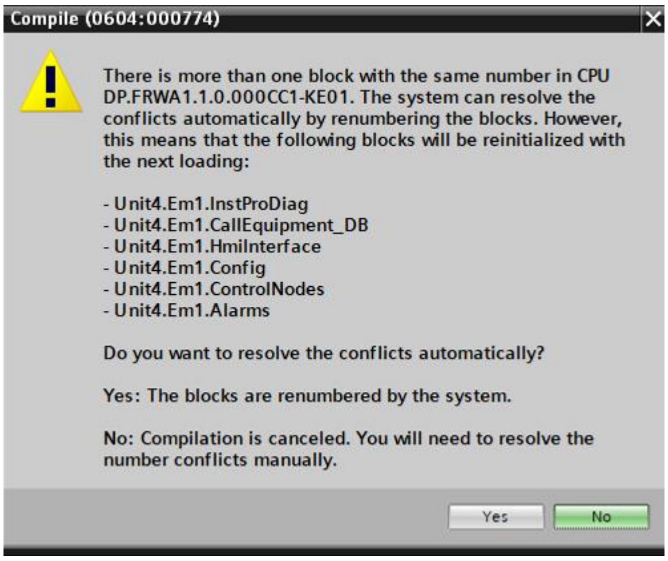
General

Name: Em6_Main
Namespace:
Constant name: OB_Em6_Main
Type: OB
Event class: Program cycle
Language: LAD
Number: 1061
[✓ Manual] Automatic

Process image part number

Figure 10-23: Properties of the new OBs

- Change the OB numbers of the OB “Main” and “Startup” according to the OB numbering convention presented in chapter Startup and cyclic
- Renumber the rest of the blocks by compiling the PLC and select to solve the conflict automatically, as shown in figure Figure 10-18: Prompt to solve block numbering
- The EM is then ready and can be adjusted to the process that should be fulfilled. 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.



There is more than one block with the same number in CPU DP.FRWA1.1.0.000CC1-KE01. The system can resolve the conflicts automatically by renumbering the blocks. However, this means that the following blocks will be reinitialized with the next loading:

- Unit4.Em1.InstProDiag
- Unit4.Em1.CallEquipment_DB
- Unit4.Em1.HmiInterface
- Unit4.Em1.Config
- Unit4.Em1.ControlNodes
- Unit4.Em1.Alarms

Do you want to resolve the conflicts automatically?

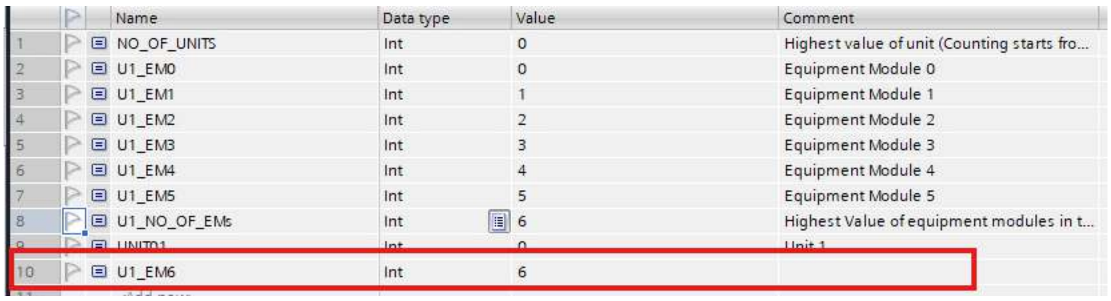
Yes: The blocks are renumbered by the system.

No: Compilation is canceled. You will need to resolve the number conflicts manually.

1. Figure 10-18: Prompt to solve block numbering

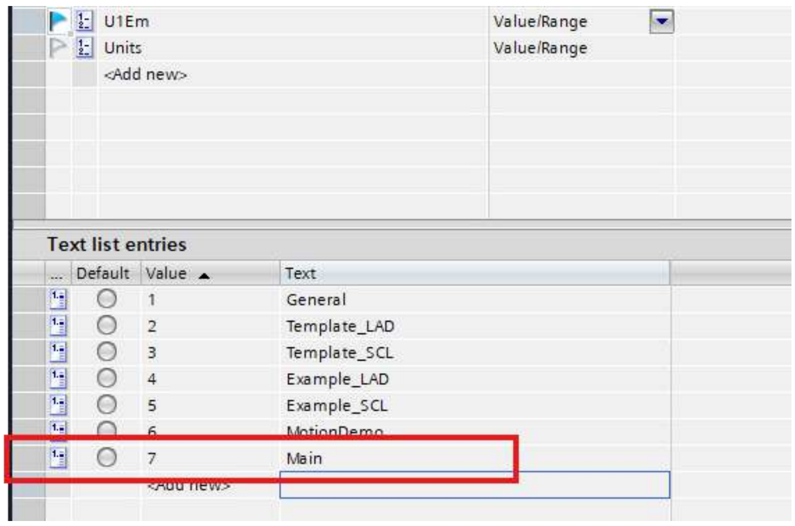
10.13. How to add a new EM in the HMI

1. Go to the Unit in the PLC where the new EM is added and check the array of the new EM. Add it to the textlists in Unified:



	Name	Data type	Value	Comment
1	NO_OF_UNITS	Int	0	Highest value of unit (Counting starts fro...
2	U1_EM0	Int	0	Equipment Module 0
3	U1_EM1	Int	1	Equipment Module 1
4	U1_EM2	Int	2	Equipment Module 2
5	U1_EM3	Int	3	Equipment Module 3
6	U1_EM4	Int	4	Equipment Module 4
7	U1_EM5	Int	5	Equipment Module 5
8	U1_NO_OF_EMs	Int	6	Highest Value of equipment modules in t...
9	UNIT01	Int	0	Unit 1
10	U1_EM6	Int	6	

Figure 10-19: EM check in the PLC



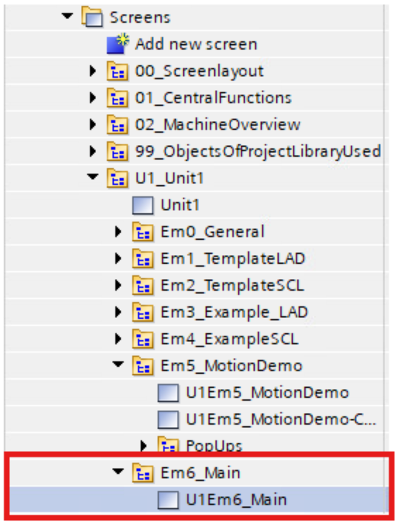
U1Em	Value/Range
Units	Value/Range
<Add new>	

Text list entries

Default	Value	Text
○	1	General
○	2	Template_LAD
○	3	Template_SCL
○	4	Example_LAD
○	5	Example_SCL
○	6	MotionDemo
○	7	Main
	<Add new>	

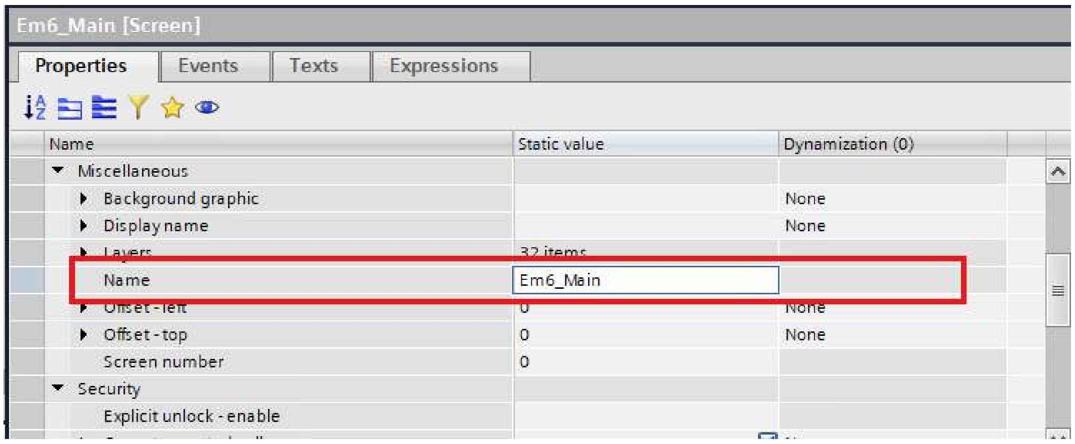
Figure 10-20: Adjustment of the textlist in the HMI

2. Add the new screen of the EM to the folder and adjust its properties:



```
Screens
  Add new screen
  00_Screenlayout
  01_CentralFunctions
  02_MachineOverview
  99_ObjectsOfProjectLibraryUsed
  U1_Unit1
    Unit1
    Em0_General
    Em1_TemplateLAD
    Em2_TemplateSCL
    Em3_Example_LAD
    Em4_ExampleSCL
    Em5_MotionDemo
      U1Em5_MotionDemo
      U1Em5_MotionDemo-C...
      PopUps
    Em6_Main
      U1Em6_Main
```

Figure 10-21: New screen for the EM

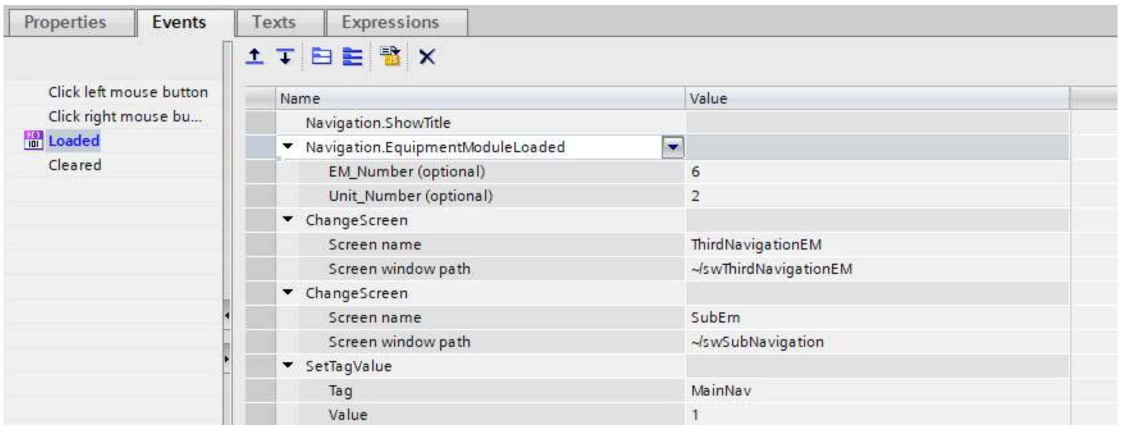

Em6_Main [Screen]

Properties | Events | Texts | Expressions

Name	Static value	Dynamization (0)
Miscellaneous		
Background graphic		None
Display name		None
Layers	32 items	
Name	Em6_Main	
Offset - left	0	None
Offset - top	0	None
Screen number	0	
Security		
Explicit unlock - enable		

Figure 10-22: Adjustment of the properties of the new screen

3. Adapt the loaded event of the Screen:


Properties | Events | Texts | Expressions

Click left mouse button
Click right mouse bu...
Loaded
Cleared

Name	Value
Navigation.ShowTitle	
Navigation.EquipmentModuleLoaded	
EM_Number (optional)	6
Unit_Number (optional)	2
ChangeScreen	
Screen name	ThirdNavigationEM
Screen window path	~/swThirdNavigationEM
ChangeScreen	
Screen name	SubEm
Screen window path	~/swSubNavigation
SetTagValue	
Tag	MainNav
Value	1

Figure 10-23: Loaded events for the new screen

The different system functions inside the loaded event refer to:

- ShowTitle : A generic function to show the title in the header.
- EquipmentModuleLoaded: The scripts that tracks the information for the correct visualization of the EM. We have to write the Number of the EM and the corresponding Unit.
- Two ChangeScreens: to visualize and set the corresponding navigations.
- SetTagValue for the MainNav: to highlight correctly the icon in the Main Navigation.

4. The SubEM and ThirdEM navigations are automatically adapted in the project. There is no need to adapt anything in the different sub and third navigation as the project provides an "automatic" adaptation depending on the tags connected to the PLC. Automation Framework provides up to 25 buttons for the units pre-configured.

NOTE

If 25 or less EMs are used, no additional HMI engineering is required.

If more than 25 EMs are used, the user only needs to "copy-paste" a button and adapt the third navigation it with the number of the equipment module.

5. Save and compile and the new EM should appear in the different navigations.

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.

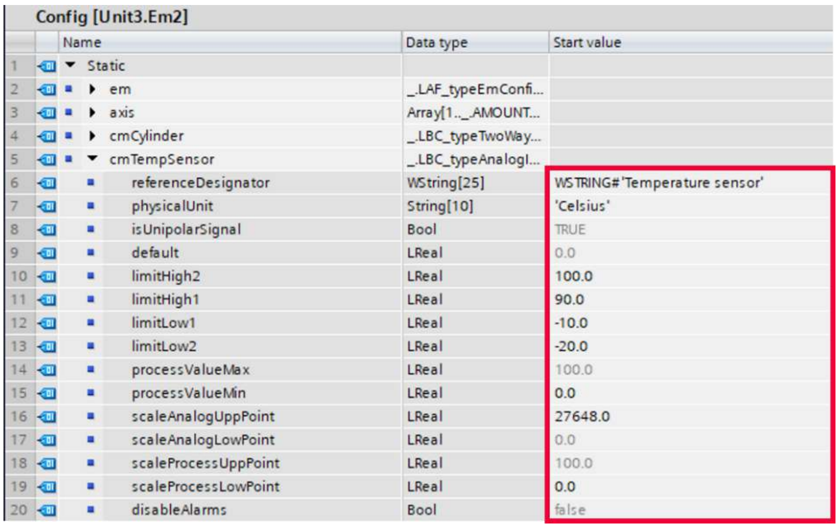
For the ease of use, a special control node version of the LBC is used in the AF$_{\text{Basic}}$. The control node FBs do simplify the FBs interface by moving the commands inputs to a commands UDT and the outputs to a monitoring UDT. Both, commands and monitoring UDTs, are within the control node UDT. 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:

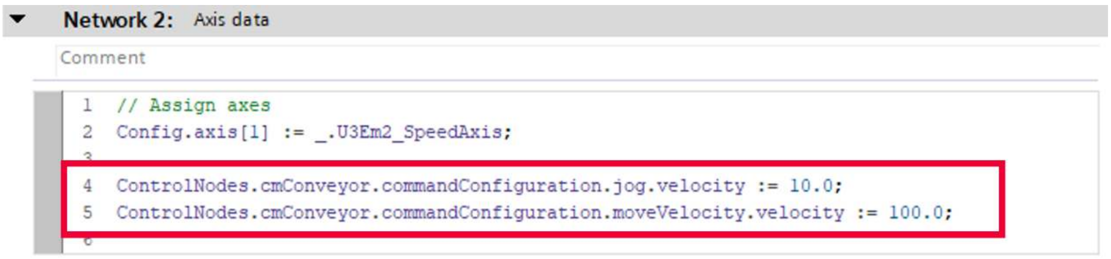


Config [Unit3.Em2]

	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:



Network 2: Axis data

Comment

```
// Assign axes
Config.axis[1] := _.U3Em2_SpeedAxis;

ControlNodes.cmConveyor.commandConfiguration.jog.velocity := 10.0;
ControlNodes.cmConveyor.commandConfiguration.moveVelocity.velocity := 100.0;
```

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. Using the LBC as an example, the input "enableManualCmds" that allows the manual control of the CM is only enabled with the variable "#tempManualControlAllowed", as seen in the next figure:

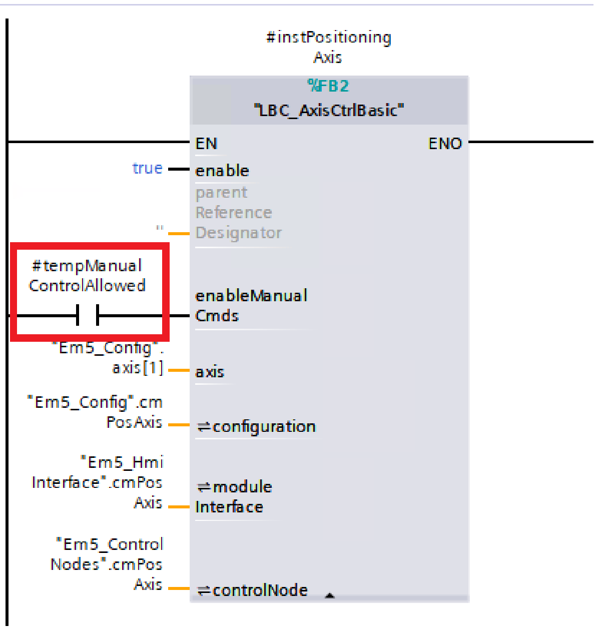


Figure 11-3: Enable Manual Commands of the LBC block

The variable “#tempManualControlAllowed” is being configured in a network from the CallEquipment FB of any equipment module, where it will be set/reset depending on whether the unit is in manual mode and whether the unit's state is a waiting state or not (according to the OMAC Model, manual operation of the CMs will not be allowed in waiting states, so it will only be permitted in the execute state). This configuration network can be seen in the next figure:

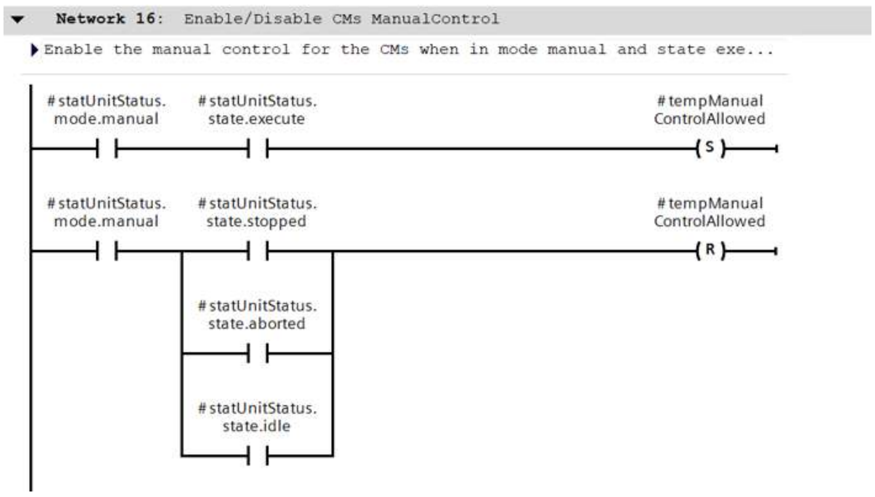

Network 16: Enable/Disable CMs ManualControl
Enable the manual control for the CMs when in mode manual and state exe...
#statUnitStatus. mode.manual
#statUnitStatus. state.execute
#tempManual ControlAllowed
S
#statUnitStatus. mode.manual
#statUnitStatus. state.stopped
#tempManual ControlAllowed
R
#statUnitStatus. state.aborted
#statUnitStatus. state.idle

Figure 11-4: Configuration network for enabling/disabling CMs Manual control

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. Motion Control

12.1. General

The S7-1200 G2 motion control (MC) is executed centrally by the Motion-OBs which are located in the folder “Motion”. The motion control instructions of a technology object (TO) are programmed in the equipment modules that belongs to the TO from a technology perspective. The following chapters explain the principle of the motion control application cycle and the use cases of the different Motion-OBs.

System MC OBs: MC_Servo/ MC_Interpolator

When the first TO is added to the S7-1200 G2 PLC, the OBs “MC_Servo” and “MC_Interpolator” for processing the functionalities of 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 by the engineer.

- "MC_Servo" performs calculations required to control the speed and position of axes.
- "MC_Interpolator" evaluates motion control instrucations, generates setpoints and monitors inputs.

User MC OBs: MC_PreInterpolator / MC_PreServo / MC_PostServo

The Motion-OBs “MC_PreInterpolator”, “MC_PreServo” and “MC_PostServo” are not know-how protected and part of the motion control application cycle (see Figure 12-1). These Motion-OBs consider special requirements regarding time-critical events and should only be added and programmed by the engineer if need:

- “MC_PreServo” should be used to modify actual values before “MC_Servo”. For example, when using an analog encoder for an axis.
- “MC_PostServo” should be used to modifiy setpoint values after “MC_Servo”. For example, to link setpoints to a characteristic line of a hydralic axis.
- "MC_PreInterpolator" should be used to adjust inputs for the "MC_Interpolator". For example, to start motion instructions immediately when a fast reaction or an instant execution for positioning is required.

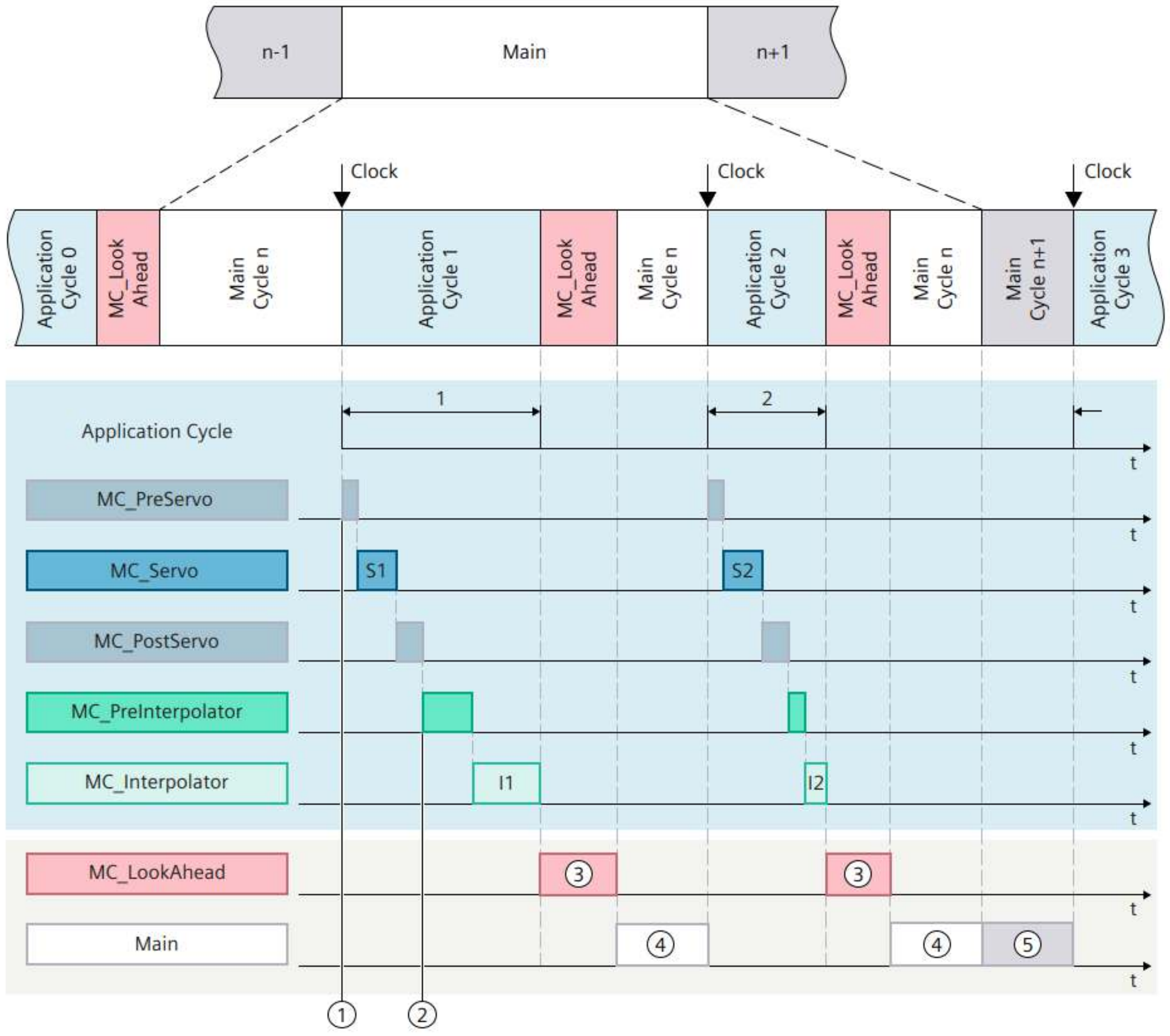
NOTE

Standard motion instructions such as “MC_Power” and “MC_Reset” and none-time-critical movement instructions should be called in the standard program cycle OB “Main” of the equipment module.

Otherwise, the PLC is loaded with too much unnecessary program in the motion control application cycle which decreases the overall PLC performance dramatically.

Additionally, only add the Motion OBs in the program which are needed.

Figure 12-1 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 12-1: Motion control application cycle and call hierarchy of Motion-OBs

12.2. Motion Folder

The folder "Motion" is very lean. It only contains the necessary Motion-OBs (1). The technology objects are configured globally in the S7-1200 G2 PLC (2). The motion instructions are programmed and called in the respective equipment module itself.

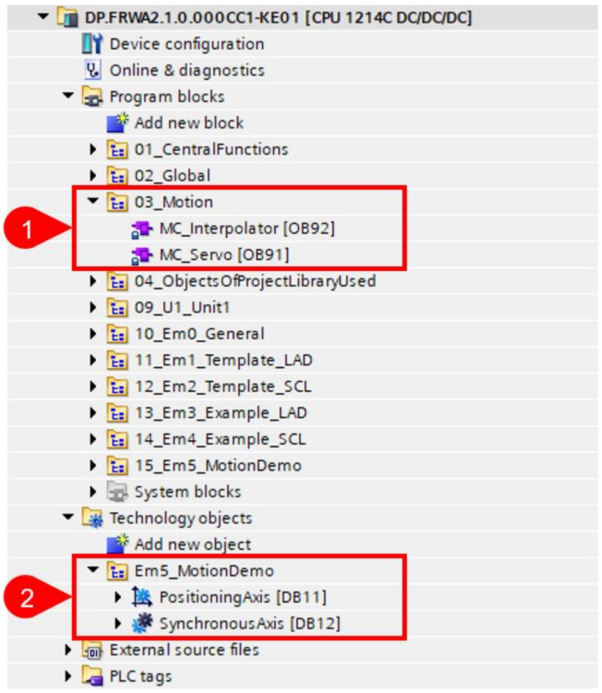
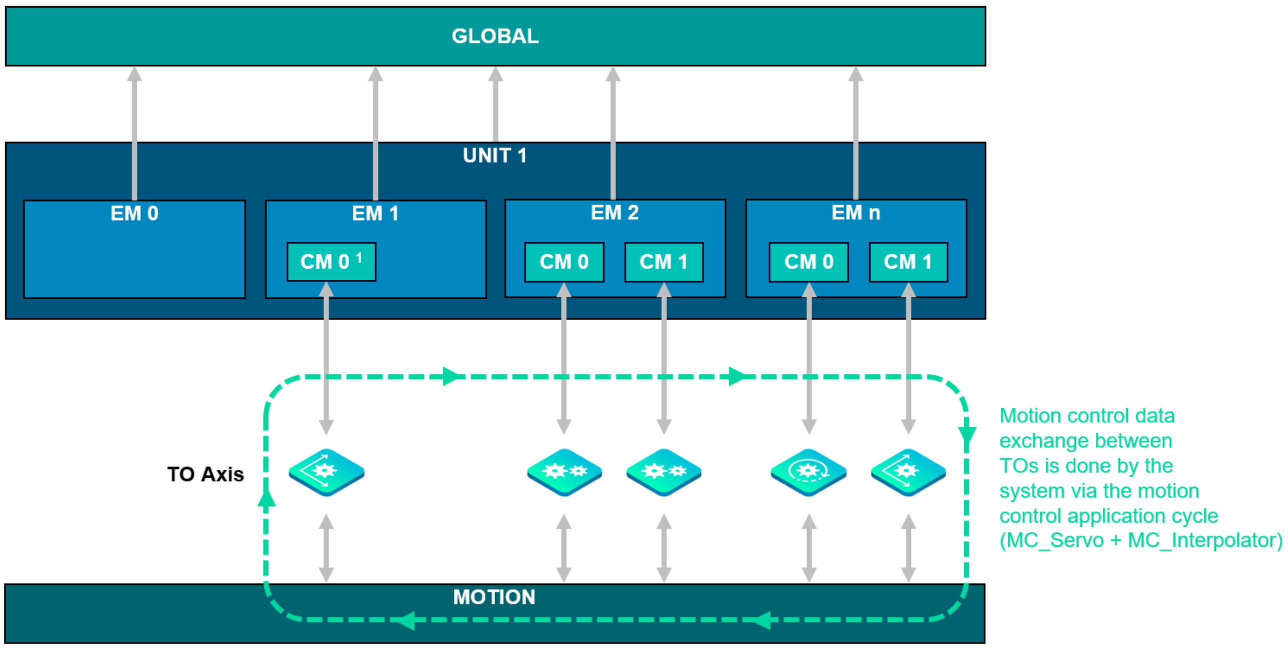


Figure 12-2: Contant of folder "Motion" and technology objects

NOTE

In the preconfigured TIA project of the AF$_{Basic}$, a motion control demo equipment module EM Demo Implementation – Motion Demo is included. It shows how motion control with a technology object is programmed in a clear, performance optimized and efficient way.

12.3. Data exchange between master axis and slave axis in different equipment modules


GLOBAL
UNIT 1
EM 0
EM 1
CM 0 [1]
EM 2
CM 0
CM 1
EM n
CM 0
CM 1
TO Axis
MOTION
Motion control data exchange between TOs is done by the system via the motion control application cycle (MC_Servo + MC_Interpolator)

[1] The Motion Control (MC) instructions are called in a Control Module (CM). All CMs are called in “CallEquipment”.

Figure 12-3: Program structure and motion control data exchange via motion control application cycle

Figure 12-3 shows the program structure with the use of technology objects in different equipment modules. The standard motion instructions such as “MC_Power” and “MC_Reset” and none-time-critical movements such as “MC_MoveAbsolute” or “MC_MoveVelocity” are called inside a block for the control module (e.g. LBC_AxisCtrlBasic as CM 0 in Figure 12-3). Each control module block is assigned to a technology object from type “Axis” (e.g. TO_SpeedAxis, TO_PositioningAxis, TO_SynchronousAxis or TO_ExternalEncoder).

In gearing or camming applications, these axes need to move in a coordinated way to fulfill the entire movement. Means, a data exchange for the coupling between these technology objects is required to perform a coordinated movement or to start the synchronization/desynchronization of master axis and slave axis when certain preconditions are fulfilled. This data exchange between the technology objects is done by the system of the SIMATIC controller via the “motion control application cycle” which is provided by the Motion-OBs. The Motion-OBs “MC_Servo” and “MC_Interpolator” are located in the folder “Motion”.

But what about the handshake signals to start the synchronization (e.g. “MC_GearIn”) or the desynchronization (e.g. “MC_GearOut”) of master axis and slave axis? Let’s say the master axis is a TO_PositioningAxis and belongs to equipment module 1 and the slave axis is a TO_SynchronousAxis and belongs to equipment module 2 (see Figure 12-4). Because the “LBC_AxisCtrlBasic” block is limited to basic motion instructions, the synchronous motion instructions for the TO_SynchronousAxis must be called in addition for CM 0 in equipment module 2.

Both axes belong to unit 1. The handshake signals are exchanged via the DB “Unit1Ems” in the folder “Global”. It is recommended to create UDTs for the handshake signals and declare tags based on these UDTs in the DB “Unit1Ems”.

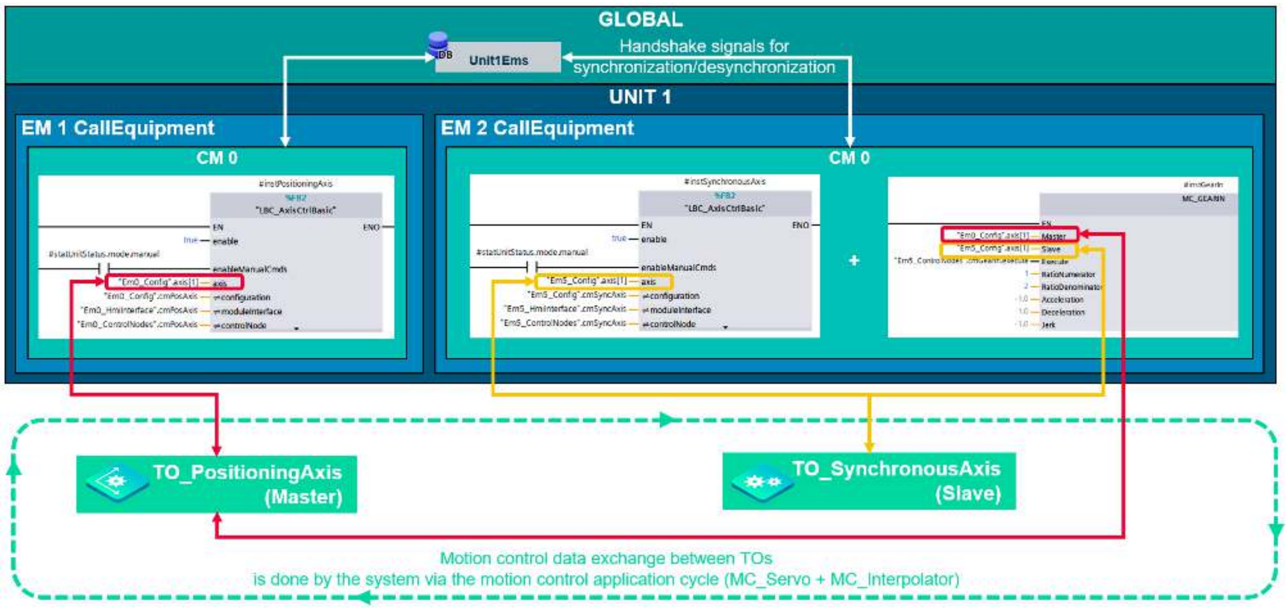

GLOBAL
DB
Unit1Ems
Handshake signals for synchronization/desynchronization
UNIT 1
EM 1 CallEquipment
CM 0
EM 2 CallEquipment
CM 0
"LBC_AxisCtrlBasic"
"Em0_Config".axis[1]
"Em5_Config".axis[1]
MC_GEARIN
TO_PositioningAxis (Master)
TO_SynchronousAxis (Slave)
Motion control data exchange between TOs is done by the system via the motion control application cycle (MC_Servo + MC_Interpolator)

Figure 12-4: Data exchange for handshake signals between master axis and slave axis in different equipment modules

NOTE

As the handshake signals are highly dependent on the application. The AFBasic does not provide fixed UDTs for the handshake signals for synchronization/desynchronization between master axis and slave axis in different equipment modules by default. This chapter explains a general approach on how these handshake signals can be implemented in a clean and modular way.

12.4. Technology objects vs. Sina-blocks

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 are in the PLC. The PLC 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 such as "SinaPos", the position controller and interpolator are 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 highlighted in green.

	Technology object	**Sina-block**
Functionality	• Speed control • Positioning • Gearing • Camming • Kinematics	• Speed control • Positioning (with EPOS)
Axis quantity	Depends on the CPU. • Up to 10 positioning axes are supported with CPU1212 C/FC (typical 4 positioning axes at 4 ms motion control application cycle) • Up to 10 positioning axes are supported with CPU1214 C/FC (typical 4 positioning axes at 4 ms motion control application cycle)	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.	• Telegram 1 (speed control) • Telegram 111, 112 and 113 (positioning via EPOS)
PROFINET IRT	Required for servo drives	Not required
Axis configuration	• In TIA Portal • User friendly configuration • Wide range of options	• In Startdrive • User friendly configuration • Limited options
PLC programming	• Variety of PLCopen blocks for MC • High functionality • High flexibility	• Limited to "Sina-blocks" and "TO BasicPosControl" (positioning only) • Low functionality • Low flexibility
Usability	High	Medium

	Technology object	**Sina-block**
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)
Simulation	• TO integrated simulation • No additional software and licenses required	• Additional software and licenses (SIMIT or DriveSim) required
Openness / Add-Ins	Supported	Supported

Table 12-1: Comparison of TOs and Sina-blocks

NOTE

In the preconfigured TIA project of the AF$_{\text{Basic}}$, the EM Demo Implementation – Motion Demo shows motion control with technology objects only. No Sina-blocks are included in this demo.

13. Process diagnostics

13.1. General

The Automation Framework_{Basic} offers a 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”.

- System diagnostics are hardware-related and described in the chapter Diagnostics.
- Process diagnostic relate to the machine actions and processes. Bit alarms are created within the PLC program, while their text and the alarm configuration are handled in the HMI.

Within the Automation Framework_{Basic}, alarms are configured in both the PLC and the HMI. Each Unit and Equipment Module will have an alarm DB containing Boolean arrays categorized into errors, warnings, and information messages. These Booleans will include their trigger logic and a corresponding alarm configuration in the HMI, where parameters such as alarm text, mode, additional information, etc., are defined.

Subsequent chapters will describe the prerequisites for the HMI and PLC, as well as the process of creating alarms using the AF_{Basic} concept

NOTE

Thanks to its modular structure, the bit alarming method can be adapted with your own implementation

13.2. Prerequisites

This chapter explains the necessary settings for the PLC and the HMI. The final settings to meet end-user requirement may differ.

13.2.1. 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, or all connected HMIs.

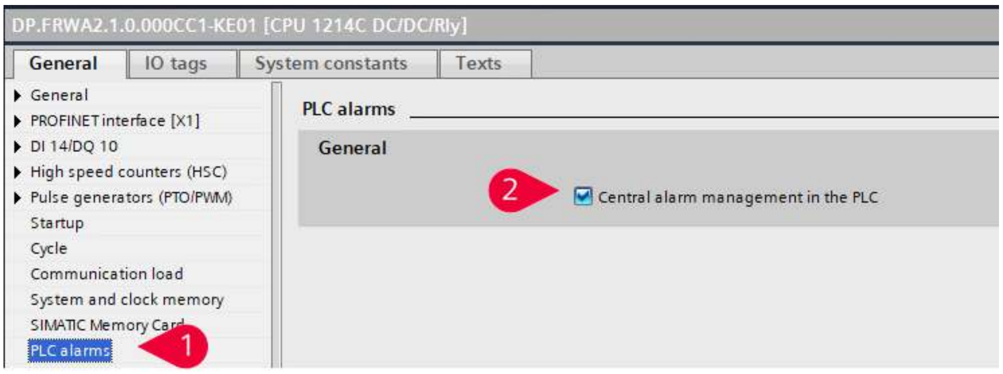
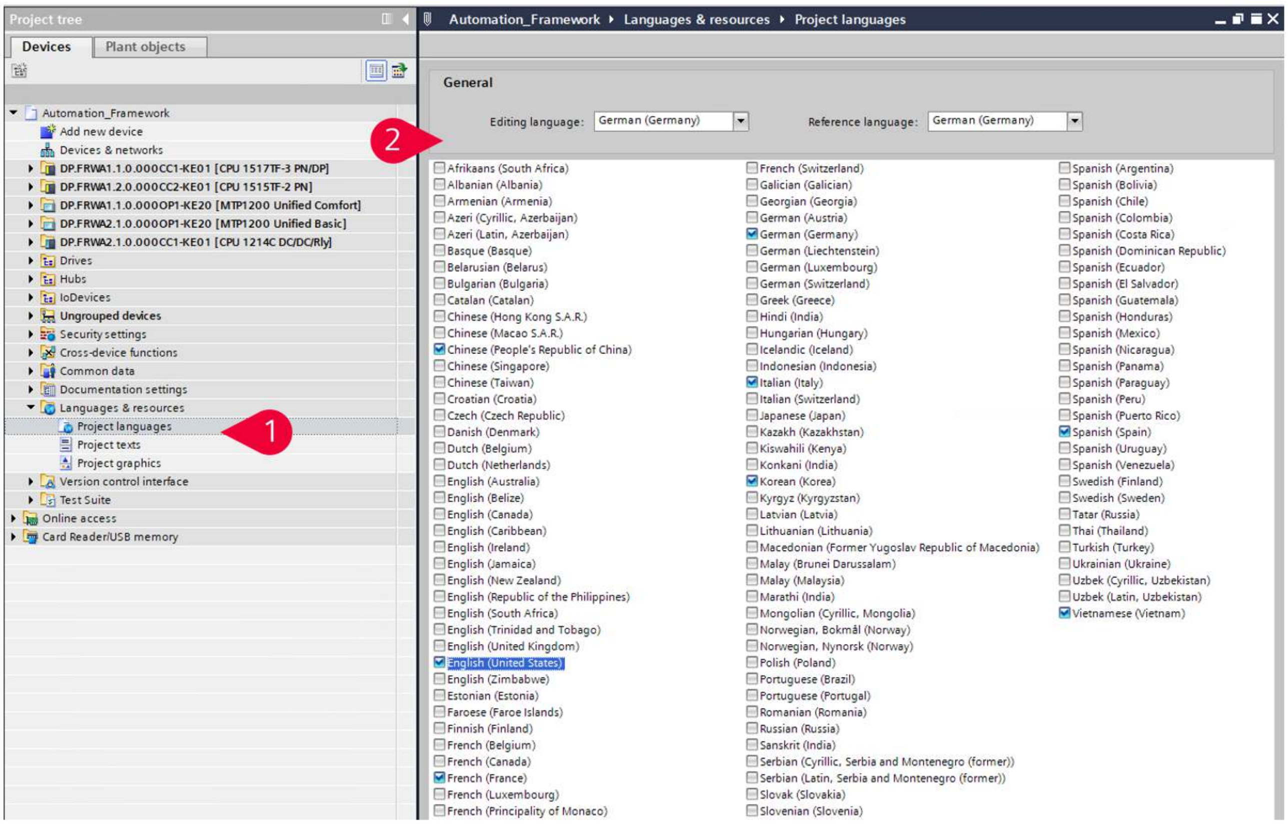


Figure 13-1: Activation of the central alarm management in the PLC

13.2.2. Multilingual support

The default language setting in the Automation Framework 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.


Project tree
Devices
Plant objects
Automation_Framework
Add new device
Devices & networks
DP.FRWA1.1.0.000CC1-KE01 [CPU 1517TF-3 PN/DP]
DP.FRWA1.2.0.000CC2-KE01 [CPU 1515TF-2 PN]
DP.FRWA1.1.0.000OP1-KE20 [MTP1200 Unified Comfort]
DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic]
DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/Rly]
Drives
Hubs
IoDevices
Ungrouped devices
Security settings
Cross-device functions
Common data
Documentation settings
Languages & resources
Project languages
Project texts
Project graphics
Version control interface
Test Suite
Online access
Card Reader/USB memory
1
2
Automation_Framework ▸ Languages & resources ▸ Project languages
General

Editing language: German (Germany)

Reference language: German (Germany)

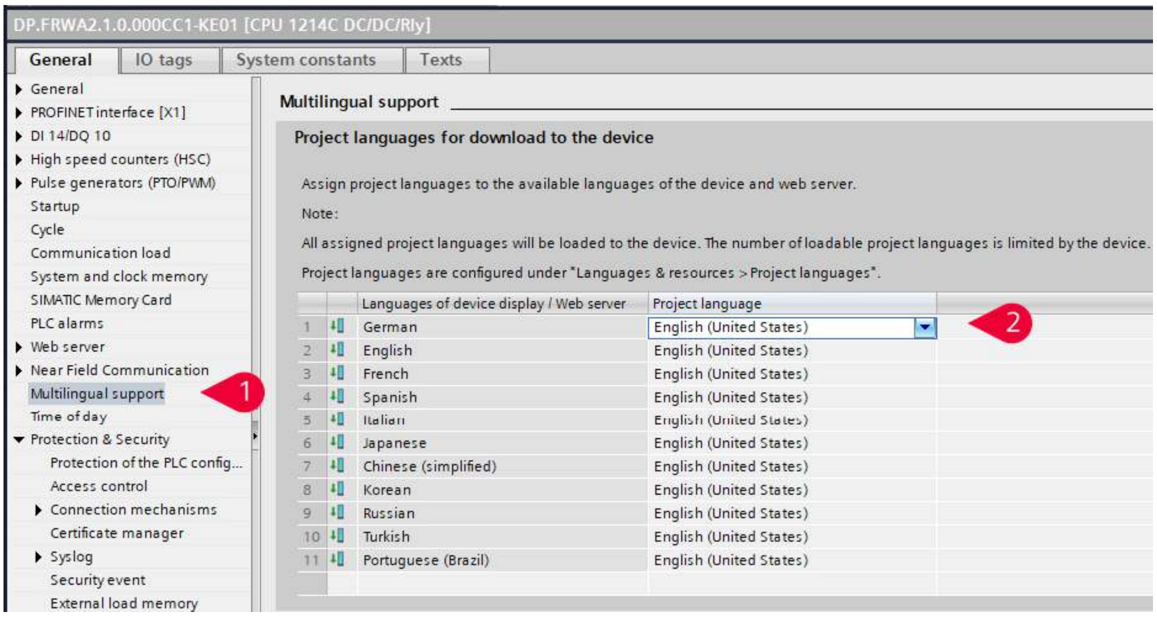
[ ] Afrikaans (South Africa)	[ ] French (Switzerland)	[ ] Spanish (Argentina)
[ ] Albanian (Albania)	[ ] Galician (Galician)	[ ] Spanish (Bolivia)
[ ] Armenian (Armenia)	[ ] Georgian (Georgia)	[ ] Spanish (Chile)
[ ] Azeri (Cyrillic, Azerbaijan)	[ ] German (Austria)	[ ] Spanish (Colombia)
[ ] Azeri (Latin, Azerbaijan)	[✓] German (Germany)	[ ] Spanish (Costa Rica)
[ ] Basque (Basque)	[ ] German (Liechtenstein)	[ ] Spanish (Dominican Republic)
[ ] Belarusian (Belarus)	[ ] German (Luxembourg)	[ ] Spanish (Ecuador)
[ ] Bulgarian (Bulgaria)	[ ] German (Switzerland)	[ ] Spanish (El Salvador)
[ ] Catalan (Catalan)	[ ] Greek (Greece)	[ ] Spanish (Guatemala)
[ ] Chinese (Hong Kong S.A.R.)	[ ] Hindi (India)	[ ] Spanish (Honduras)
[ ] Chinese (Macao S.A.R.)	[ ] Hungarian (Hungary)	[ ] Spanish (Mexico)
[✓] Chinese (People's Republic of China)	[ ] Icelandic (Iceland)	[ ] Spanish (Nicaragua)
[ ] Chinese (Singapore)	[ ] Indonesian (Indonesia)	[ ] Spanish (Panama)
[ ] Chinese (Taiwan)	[✓] Italian (Italy)	[ ] Spanish (Paraguay)
[ ] Croatian (Croatia)	[ ] Italian (Switzerland)	[ ] Spanish (Peru)
[ ] Czech (Czech Republic)	[ ] Japanese (Japan)	[ ] Spanish (Puerto Rico)
[ ] Danish (Denmark)	[ ] Kazakh (Kazakhstan)	[✓] Spanish (Spain)
[ ] Dutch (Belgium)	[ ] Kiswahili (Kenya)	[ ] Spanish (Uruguay)
[ ] Dutch (Netherlands)	[ ] Konkani (India)	[ ] Spanish (Venezuela)
[ ] English (Australia)	[✓] Korean (Korea)	[ ] Swedish (Finland)
[ ] English (Belize)	[ ] Kyrgyz (Kyrgyzstan)	[ ] Swedish (Sweden)
[ ] English (Canada)	[ ] Latvian (Latvia)	[ ] Tatar (Russia)
[ ] English (Caribbean)	[ ] Lithuanian (Lithuania)	[ ] Thai (Thailand)
[ ] English (Ireland)	[ ] Macedonian (Former Yugoslav Republic of Macedonia)	[ ] Turkish (Turkey)
[ ] English (Jamaica)	[ ] Malay (Brunei Darussalam)	[ ] Ukrainian (Ukraine)
[ ] English (New Zealand)	[ ] Malay (Malaysia)	[ ] Uzbek (Cyrillic, Uzbekistan)
[ ] English (Republic of the Philippines)	[ ] Marathi (India)	[ ] Uzbek (Latin, Uzbekistan)
[ ] English (South Africa)	[ ] Mongolian (Cyrillic, Mongolia)	[✓] Vietnamese (Vietnam)
[ ] English (Trinidad and Tobago)	[ ] Norwegian, Bokmål (Norway)	
[ ] English (United Kingdom)	[ ] Norwegian, Nynorsk (Norway)	
[✓] English (United States)	[ ] Polish (Poland)	
[ ] English (Zimbabwe)	[ ] Portuguese (Brazil)	
[ ] Estonian (Estonia)	[ ] Portuguese (Portugal)	
[ ] Faroese (Faroe Islands)	[ ] Romanian (Romania)	
[ ] Finnish (Finland)	[ ] Russian (Russia)	
[ ] French (Belgium)	[ ] Sanskrit (India)	
[ ] French (Canada)	[ ] Serbian (Cyrillic, Serbia and Montenegro (former))	
[✓] French (France)	[ ] Serbian (Latin, Serbia and Montenegro (former))	
[ ] French (Luxembourg)	[ ] Slovak (Slovakia)	
[ ] French (Principality of Monaco)	[ ] Slovenian (Slovenia)	

Figure 13-2: Overall project language settings

All languages that are activated in this configuration can now be adapted within the project texts. An additional step is to assign the project languages to the respective system languages of the PLC and its web server. To achieve this , follow these steps:

- Select the 'Multilingual Support' setting within the properties of the PLC.

- Assign the project languages that should be displayed to the system languages.



Multilingual support

Project languages for download to the device

Assign project languages to the available languages of the device and web server.

Note:

All assigned project languages will be loaded to the device. The number of loadable project languages is limited by the device.

Project languages are configured under "Languages & resources > Project languages".

	Languages of device display / Web server	Project language
1	German	English (United States)
2	English	English (United States)
3	French	English (United States)
4	Spanish	English (United States)
5	Italian	English (United States)
6	Japanese	English (United States)
7	Chinese (simplified)	English (United States)
8	Korean	English (United States)
9	Russian	English (United States)
10	Turkish	English (United States)
11	Portuguese (Brazil)	English (United States)

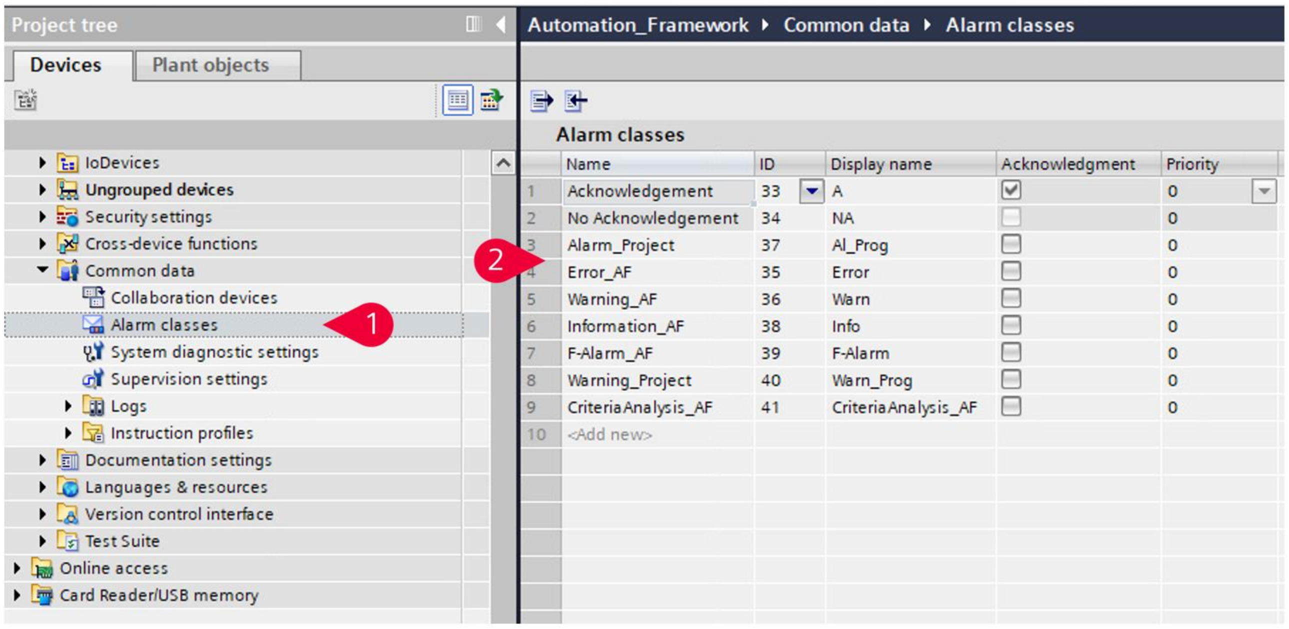
Figure 13-3: Project language assignment

All assigned project languages will be loaded onto the PLC. The PLC S7-1200 G2 can support a maximum of three different project languages at the same time. When the system language is switched, the corresponding project texts are automatically updated to the selected language.

NOTE Additionally, pay attention to the HMI language settings.

13.2.3. Common alarm class settings

In each TIA Portal project, it is possible to create custom alarm classes in addition to the standard alarm classes "Acknowledgment" and "No Acknowledgment". For newly defined classes, you can specify names, display names, priorities and whether acknowledgement is required.



Alarm classes

	Name	ID	Display name	Acknowledgment	Priority
1	Acknowledgement	33	A	✓	0
2	No Acknowledgement	34	NA		0
3	Alarm_Project	37	Al_Prog		0
4	Error_AF	35	Error		0
5	Warning_AF	36	Warn		0
6	Information_AF	38	Info		0
7	F-Alarm_AF	39	F-Alarm		0
8	Warning_Project	40	Warn_Prog		0
9	CriteriaAnalysis_AF	41	CriteriaAnalysis_AF		0
10	<Add new>				

Figure 13-4: AF Alarm Classes configuration

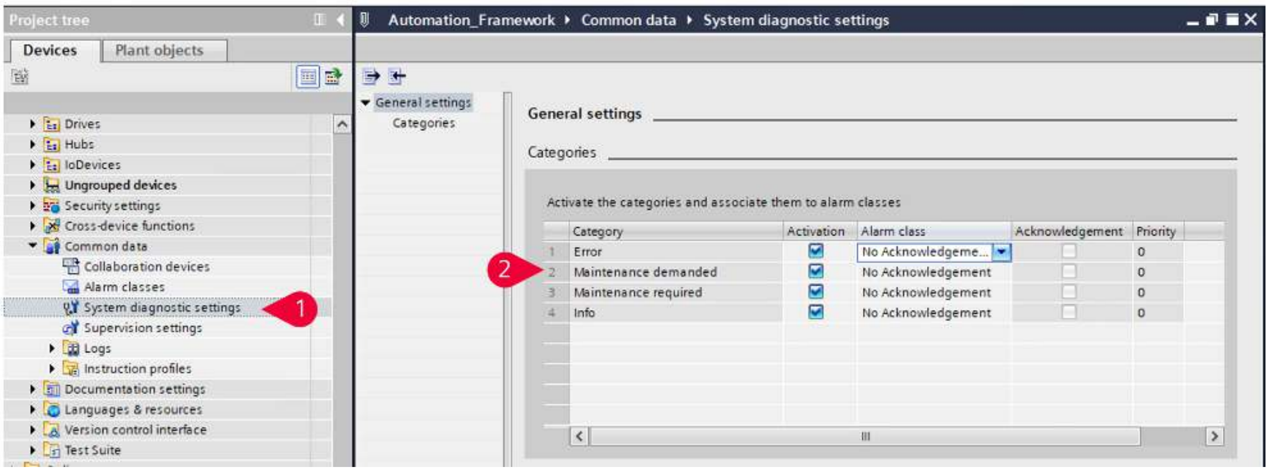
The AFBasic provides the following user defined Alarm classes:

- Alarm_Project
- Alarm_AF
- Warning_AF
- Information_AF
- F-Alarm_AF
- Warning_Project
- CriteriaAnalysis_AF

The Alarm classes can then be assigned to the diagnostic settings described in the following chapters.

13.2.4. 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.



Activate the categories and associate them to alarm classes

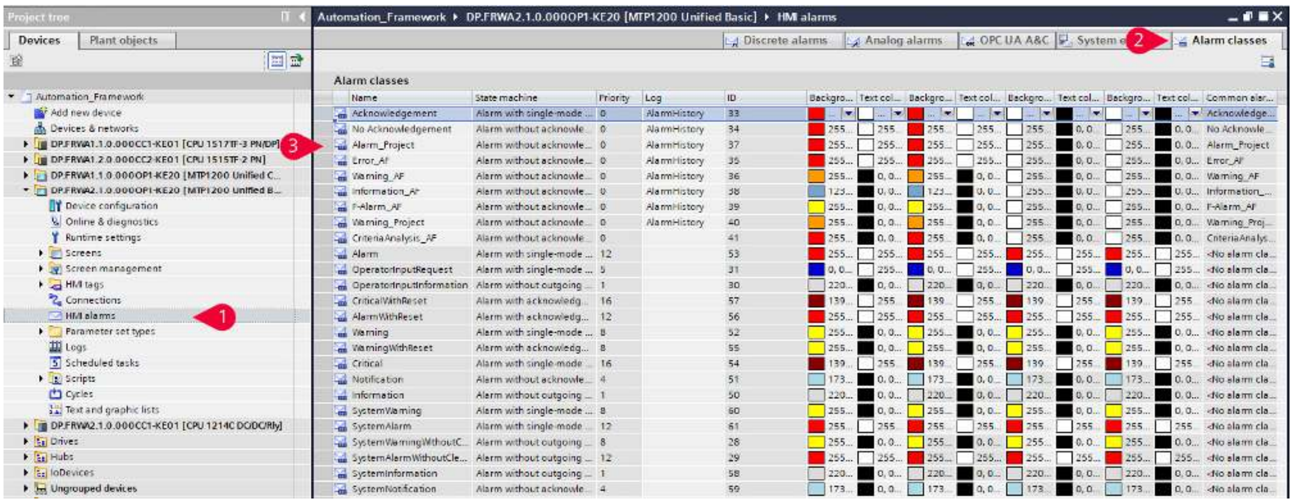
	Category	Activation	Alarm class	Acknowledgement	Priority
1	Error	✓	No Acknowledgeme...	☐	0
2	Maintenance demanded	✓	No Acknowledgement	☐	0
3	Maintenance required	✓	No Acknowledgement	☐	0
4	Info	✓	No Acknowledgement	☐	0

Figure 13-5: System diagnostic settings in the AF

13.2.5. HMI alarm class settings

HMI alarms can be freely configured, and custom alarm classes can be assigned. In the Automation framework, the alarm classes defined in the Common Alarm Class Settings are used for configuring alarms in the HMI.

Different colors can be set based on the alarm status. In the configured visualization device (1), you can access on the 'Alarm Classes' tab (2) the configuration overview (3).



Alarm classes

Name	State machine	Priority	Log	ID	Backgro...	Text col...	Backgro...	Text col...	Backgro...	Text col...	Backgro...	Text col...	Common alar...
Acknowledgement	Alarm with single-mode ...	0	AlarmHistory	33	🟥 ...	⬜ ...	🟥 ...	⬜ ...	⬜ ...	⬛ ...	⬜ ...	⬛ ...	Acknowledge...
No Acknowledgement	Alarm without acknowle...	0	AlarmHistory	34	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	No Acknowle...
Alarm_Project	Alarm without acknowle...	0	AlarmHistory	37	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	Alarm_Project
Error_AF	Alarm without acknowle...	0	AlarmHistory	35	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	Error_AF
Warning_AF	Alarm without acknowle...	0	AlarmHistory	36	🟧 255...	⬛ 0, 0...	🟧 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	Warning_AF
Information_AF	Alarm without acknowle...	0	AlarmHistory	38	🟦 123...	⬛ 0, 0...	🟦 123...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	Information_...
F-Alarm_AF	Alarm without acknowle...	0	AlarmHistory	39	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	F-Alarm_AF
Warning_Project	Alarm without acknowle...	0	AlarmHistory	40	🟧 255...	⬛ 0, 0...	🟧 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	Warning_Proj...
CriteriaAnalysis_AF	Alarm without acknowle...	0		41	🟥 255...	⬛ 0, 0...	🟥 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	⬜ 255...	⬛ 0, 0...	CriteriaAnalys...
Alarm	Alarm with single-mode ...	12		53	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	<No alarm cla...
OperatorInputRequest	Alarm with single-mode ...	5		31	🟦 0, 0...	⬜ 255...	🟦 0, 0...	⬜ 255...	🟦 0, 0...	⬜ 255...	🟦 0, 0...	⬜ 255...	<No alarm cla...
OperatorInputInformation	Alarm without outgoing ...	1		30	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	<No alarm cla...
CriticalWithReset	Alarm with acknowledg...	16		57	🟥 139...	⬜ 255...	🟥 139...	⬜ 255...	🟥 139...	⬜ 255...	🟥 139...	⬜ 255...	<No alarm cla...
AlarmWithReset	Alarm with acknowledg...	12		56	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	<No alarm cla...
Warning	Alarm with single-mode ...	8		52	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	<No alarm cla...
WarningWithReset	Alarm with acknowledg...	8		55	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	<No alarm cla...
Critical	Alarm with single-mode ...	16		54	🟥 139...	⬜ 255...	🟥 139...	⬜ 255...	🟥 139...	⬜ 255...	🟥 139...	⬜ 255...	<No alarm cla...
Notification	Alarm without acknowle...	4		51	🟦 173...	⬛ 0, 0...	🟦 173...	⬛ 0, 0...	🟦 173...	⬛ 0, 0...	🟦 173...	⬛ 0, 0...	<No alarm cla...
Information	Alarm without outgoing ...	1		50	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	<No alarm cla...
SystemWarning	Alarm with single-mode ...	8		60	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	<No alarm cla...
SystemAlarm	Alarm with single-mode ...	12		61	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	<No alarm cla...
SystemWarningWithoutC...	Alarm without outgoing ...	8		28	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	🟨 255...	⬛ 0, 0...	<No alarm cla...
SystemAlarmWithoutCle...	Alarm without outgoing ...	12		29	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	🟥 255...	⬜ 255...	<No alarm cla...
SystemInformation	Alarm without outgoing ...	1		58	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	⬜ 220...	⬛ 0, 0...	<No alarm cla...
SystemNotification	Alarm without acknowle...	4		59	🟦 173...	⬛ 0, 0...	🟦 173...	⬛ 0, 0...	🟦 173...	⬛ 0, 0...	🟦 173...	⬛ 0, 0...	<No alarm cla...

Figure 13-6: 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

13.3. Diagnostic Concept

The holistic alarm concept within the Automation Framework$_{Basic}$ provides a standardized and modular approach to create alarms in a machine. Its primary purpose is to enable operators to identify the cause and location of failures within the system, ultimately increasing production availability and reducing downtime.

Within the Automation Framework$_{Basic}$, this requires a well-defined structure that integrates the alarm concept into the framework's functional architecture. This includes alarming across different layers of the ISA-88 model, from the Control Module (CM) to the Unit level.

The implemented alarm concept consists of a centralized component for machine alarming at both the Equipment Module (EM) and Unit levels. This component is based on the "CollectAlarms" block, which centralizes alarm management and simplifies the definition of unique bit alarms and messages for the process. Within this block, users categorize and map all alarms originating from control modules and the processes executed within the EM or Unit. A predefined structure in the Alarms DB is available to facilitate mapping and categorization. The Alarm DB consists of different categories such as Error, Warning, Info and Criteria Analysis where each category structure is composed by the trigger event status and the subcategory which defines the reaction or command to the system.

The following table shows the different alarm and message categories included in the DB structure and their effects on the process when triggered.

Alarm categories	**Description**
Error	An error message is displayed and triggers an automatic response or command to the system according to the subcategory defined by the user.
Warning	A warning message that indicates a potential issue and triggers an automatic response or command to the system according to the subcategory defined by the user.
Information	A notification providing status updates or general system information and triggers an automatic response or command to the system according to the subcategory defined by the user.
Criteria Analysis	A category used for evaluating specific conditions in the system based on predefined criteria and triggers an automatic response or command to the system according to the subcategory defined by the user.

Table 13-1 Alarm Categories and their description

These different categories are used to map the different alarms to the machine's process, allowing them to directly influence the behavior of the machine through the subcategories. The state changes are handled by the "LAF_SummarizeCommandsToUnit" 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.

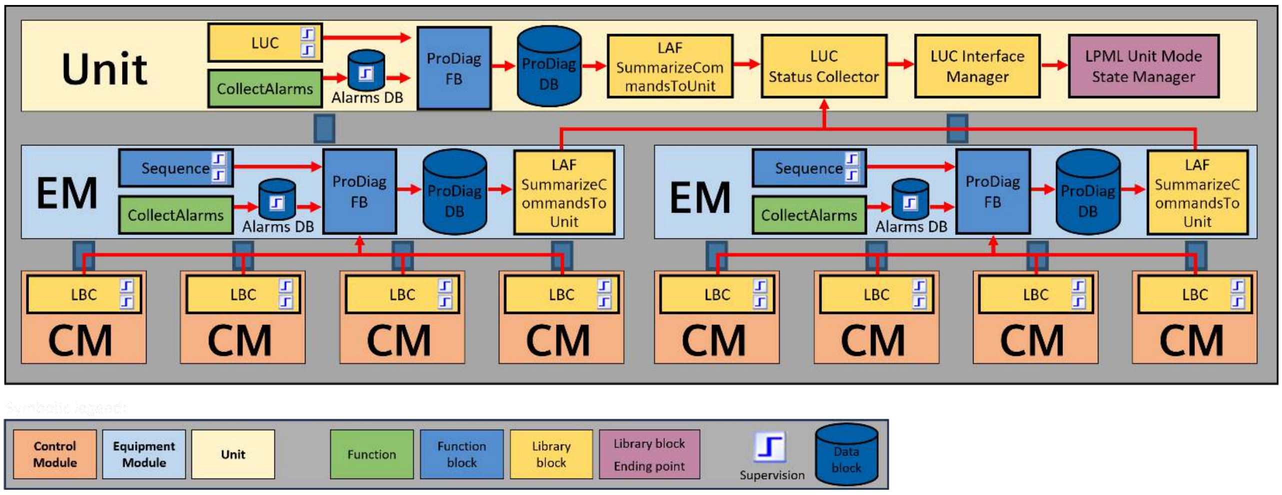
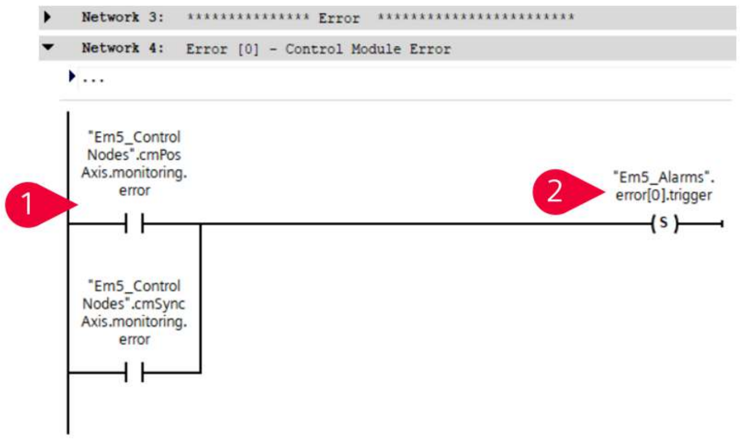


Figure 13-7: Architecture of the Diagnostic Concept

13.3.1. Alarm Implementation in the PLC

As previously mentioned, the user defines all EM or Unit alarms within the “CollectAlarm” block. This block contains predefined sections based on alarm categories. To add an alarm, follow these steps:

- Define a condition that should trigger the alarm and place it in the section of the "CollectAlarm" block according to the alarm category (1).
- Insert the assignment and connect a free element of the array that is related to the trigger of the alarm category from the DB “Alarms” (2).

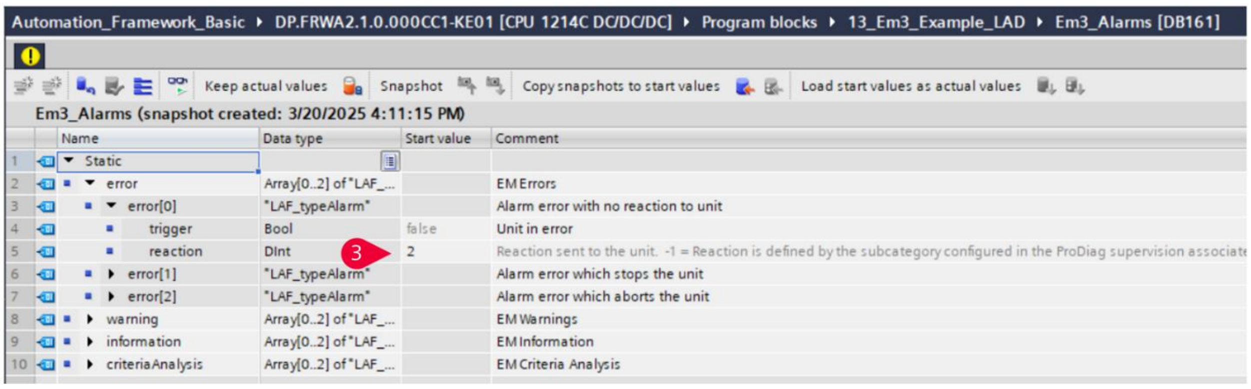


```
Network 3:   *************** Error   ************************
Network 4:   Error [0] - Control Module Error
...
```

"Em5_Control Nodes".cmPos Axis.monitoring. error
1
2
"Em5_Alarms". error[0].trigger
S
"Em5_Control Nodes".cmSync Axis.monitoring. error

Figure 13-8 Alamr Mapping in FC CollectAlarms

- Open the DB "Alarms" of the correspondig EM or Unit, for our example is DB “Em3_Alarms”, select the assigned element and add the reaction to the system (3). The reaction is defined by an integer value with the following reactions to the system. Reactions: -1 = Reaction is defined by the subcategory configured in the ProDiag (non applicable for 1200-G2, 1 = Stop, 2 = No Reaction, 3 = Abort, 4 = Hold, 5 = Suspend, 6 = Reset, 7 = Start, 8 = Unhold, 9 = Unsuspend, 10 = Clear, 11 = Complete.


Keep actual values
Snapshot
Copy snapshots to start values
Load start values as actual values
3

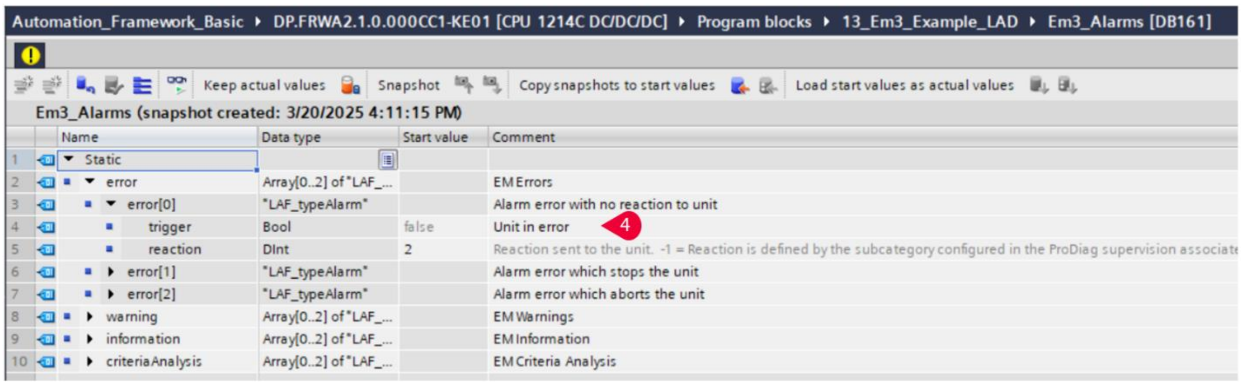
Automation_Framework_Basic ▸ DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ 13_Em3_Example_LAD ▸ Em3_Alarms [DB161]

Em3_Alarms (snapshot created: 3/20/2025 4:11:15 PM)

	Name	Data type	Start value	Comment
1	▼ Static			
2	▼ error	Array[0..2] of "LAF_...		EM Errors
3	▼ error[0]	"LAF_typeAlarm"		Alarm error with no reaction to unit
4	trigger	Bool	false	Unit in error
5	reaction	DInt	2	Reaction sent to the unit. -1 = Reaction is defined by the subcategory configured in the ProDiag supervision associate
6	▸ error[1]	"LAF_typeAlarm"		Alarm error which stops the unit
7	▸ error[2]	"LAF_typeAlarm"		Alarm error which aborts the unit
8	▸ warning	Array[0..2] of "LAF_...		EM Warnings
9	▸ information	Array[0..2] of "LAF_...		EM Information
10	▸ criteriaAnalysis	Array[0..2] of "LAF_...		EM Criteria Analysis

Figure 13-9: Subcategories assignment to different alarm events

- Open the DB "Alarms" of the correspondig EM or Unit, for our example is DB “Em3_Alarms”, select the assigned element and add a comment that is describing the alarm. This step is optional but strongly recommended to identify which array element is already in use (3).


Keep actual values
Snapshot
Copy snapshots to start values
Load start values as actual values
4

Automation_Framework_Basic ▸ DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC] ▸ Program blocks ▸ 13_Em3_Example_LAD ▸ Em3_Alarms [DB161]

Em3_Alarms (snapshot created: 3/20/2025 4:11:15 PM)

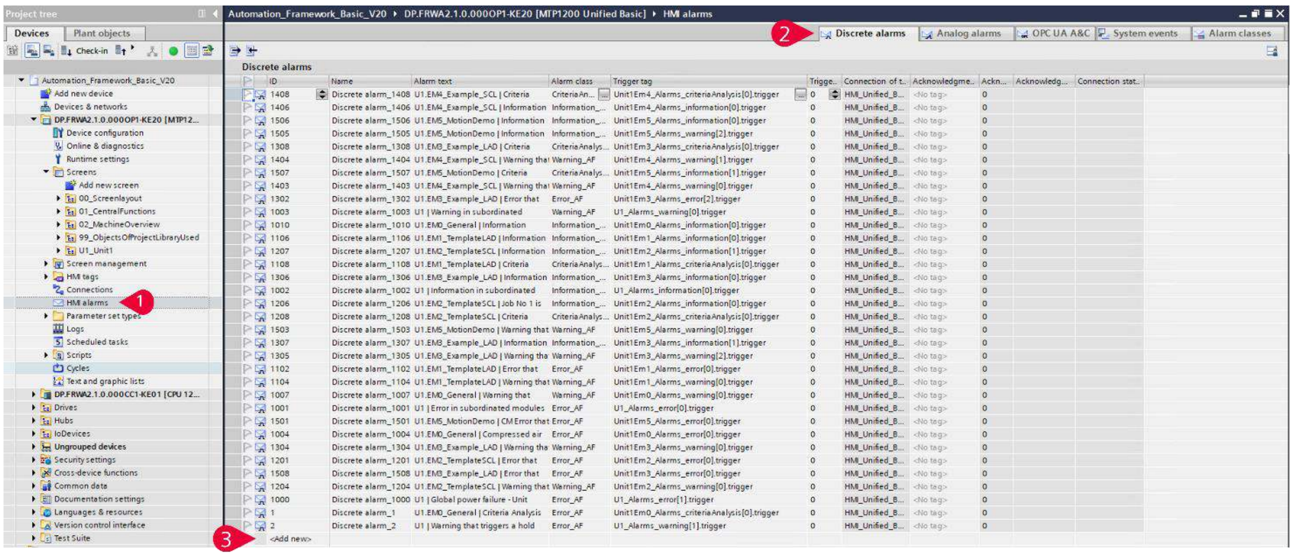
	Name	Data type	Start value	Comment
1	▼ Static			
2	▼ error	Array[0..2] of "LAF_...		EM Errors
3	▼ error[0]	"LAF_typeAlarm"		Alarm error with no reaction to unit
4	trigger	Bool	false	Unit in error
5	reaction	DInt	2	Reaction sent to the unit. -1 = Reaction is defined by the subcategory configured in the ProDiag supervision associate
6	▸ error[1]	"LAF_typeAlarm"		Alarm error which stops the unit
7	▸ error[2]	"LAF_typeAlarm"		Alarm error which aborts the unit
8	▸ warning	Array[0..2] of "LAF_...		EM Warnings
9	▸ information	Array[0..2] of "LAF_...		EM Information
10	▸ criteriaAnalysis	Array[0..2] of "LAF_...		EM Criteria Analysis

Figure 13-10 Add comment to the configured element in the Alarm DB

13.3.2. Alarm Implementation in the HMI

Once the alarm has been called in the “CollectAlarms” function and the trigger conditions are set, a new discrete alarm must be created in the HMI. The HMI discrete alarm consists of an alarm text, alarm class, trigger tag, mode, and other parameters. To add a discrete alarm, follow these steps:

- Open the HMI Alarms menu inside the HMI folder(1).
- Select the discrete alarms tab (2).
- Click “Add new” to add a new alarm(3).



Project tree

Devices | Plant objects

Check-in

- Automation_Framework_Basic_V20
  - Add new device
  - Devices & networks
  - DP.FRWA2.1.0.000OP1-KE20 [MTP12...
    - Device configuration
    - Online & diagnostics
    - Runtime settings
    - Screens
      - Add new screen
      - 00_Screenlayout
      - 01_CentralFunctions
      - 02_MachineOverview
      - 99_ObjectsOfProjectLibraryUsed
      - U1_Unit1
    - Screen management
    - HMI tags
    - Connections
    - HMI alarms
    - Parameter set types
    - Logs
    - Scheduled tasks
    - Scripts
    - Cycles
    - Text and graphic lists
  - DP.FRWA2.1.0.000CC1-KE01 [CPU 12...
  - Drives
  - Hubs
  - IoDevices
  - Ungrouped devices
  - Security settings
  - Cross-device functions
  - Common data
  - Documentation settings
  - Languages & resources
  - Version control interface
  - Test Suite

Automation_Framework_Basic_V20 ▸ DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic] ▸ HMI alarms

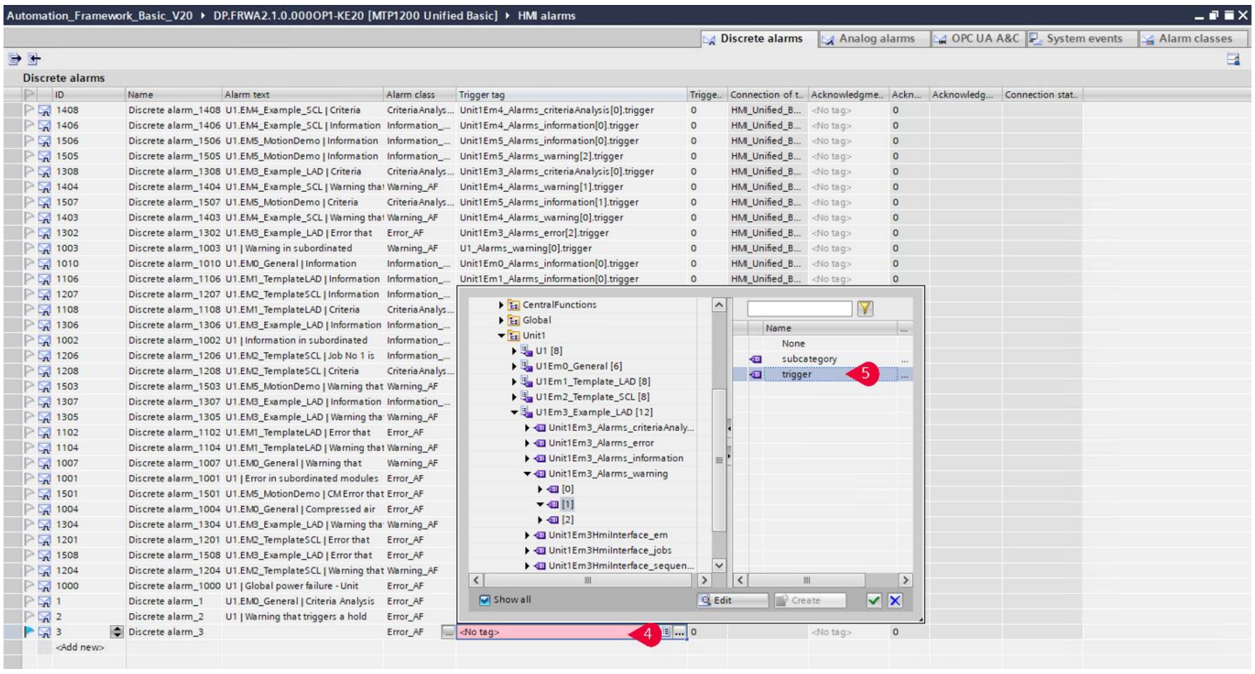
Discrete alarms | Analog alarms | OPC UA A&C | System events | Alarm classes

Discrete alarms

ID	Name	Alarm text	Alarm class	Trigger tag	Trigge...	Connection of t...	Acknowledgme...	Ackn...	Acknowledg...	Connection stat...
1408	Discrete alarm_1408	U1.EM4_Example_SCL | Criteria	CriteriaAn...	Unit1Em4_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1406	Discrete alarm_1406	U1.EM4_Example_SCL | Information	Information_...	Unit1Em4_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1506	Discrete alarm_1506	U1.EM5_MotionDemo | Information	Information_...	Unit1Em5_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1505	Discrete alarm_1505	U1.EM5_MotionDemo | Information	Information_...	Unit1Em5_Alarms_warning[2].trigger	0	HMI_Unified_B...	<No tag>	0		
1308	Discrete alarm_1308	U1.EM3_Example_LAD | Criteria	CriteriaAnalys...	Unit1Em3_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1404	Discrete alarm_1404	U1.EM4_Example_SCL | Warning that	Warning_AF	Unit1Em4_Alarms_warning[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1507	Discrete alarm_1507	U1.EM5_MotionDemo | Criteria	CriteriaAnalys...	Unit1Em5_Alarms_information[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1403	Discrete alarm_1403	U1.EM4_Example_SCL | Warning that	Warning_AF	Unit1Em4_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1302	Discrete alarm_1302	U1.EM3_Example_LAD | Error that	Error_AF	Unit1Em3_Alarms_error[2].trigger	0	HMI_Unified_B...	<No tag>	0		
1003	Discrete alarm_1003	U1 | Warning in subordinated	Warning_AF	U1_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1010	Discrete alarm_1010	U1.EM0_General | Information	Information_...	Unit1Em0_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1106	Discrete alarm_1106	U1.EM1_TemplateLAD | Information	Information_...	Unit1Em1_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1207	Discrete alarm_1207	U1.EM2_TemplateSCL | Information	Information_...	Unit1Em2_Alarms_information[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1108	Discrete alarm_1108	U1.EM1_TemplateLAD | Criteria	CriteriaAnalys...	Unit1Em1_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1306	Discrete alarm_1306	U1.EM3_Example_LAD | Information	Information_...	Unit1Em3_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1002	Discrete alarm_1002	U1 | Information in subordinated	Information_...	U1_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1206	Discrete alarm_1206	U1.EM2_TemplateSCL | Job No 1 is	Information_...	Unit1Em2_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1208	Discrete alarm_1208	U1.EM2_TemplateSCL | Criteria	CriteriaAnalys...	Unit1Em2_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1503	Discrete alarm_1503	U1.EM5_MotionDemo | Warning that	Warning_AF	Unit1Em5_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1307	Discrete alarm_1307	U1.EM3_Example_LAD | Information	Information_...	Unit1Em3_Alarms_information[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1305	Discrete alarm_1305	U1.EM3_Example_LAD | Warning tha	Warning_AF	Unit1Em3_Alarms_warning[2].trigger	0	HMI_Unified_B...	<No tag>	0		
1102	Discrete alarm_1102	U1.EM1_TemplateLAD | Error that	Error_AF	Unit1Em1_Alarms_error[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1104	Discrete alarm_1104	U1.EM1_TemplateLAD | Warning that	Warning_AF	Unit1Em1_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1007	Discrete alarm_1007	U1.EM0_General | Warning that	Warning_AF	Unit1Em0_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1001	Discrete alarm_1001	U1 | Error in subordinated modules	Error_AF	U1_Alarms_error[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1501	Discrete alarm_1501	U1.EM5_MotionDemo | CM Error that	Error_AF	Unit1Em5_Alarms_error[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1004	Discrete alarm_1004	U1.EM0_General | Compressed air	Error_AF	Unit1Em0_Alarms_error[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1304	Discrete alarm_1304	U1.EM3_Example_LAD | Warning tha	Warning_AF	Unit1Em3_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1201	Discrete alarm_1201	U1.EM2_TemplateSCL | Error that	Error_AF	Unit1Em2_Alarms_error[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1508	Discrete alarm_1508	U1.EM3_Example_LAD | Error that	Error_AF	Unit1Em3_Alarms_error[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1204	Discrete alarm_1204	U1.EM2_TemplateSCL | Warning that	Warning_AF	Unit1Em2_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1000	Discrete alarm_1000	U1 | Global power failure - Unit	Error_AF	U1_Alarms_error[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1	Discrete alarm_1	U1.EM0_General | Criteria Analysis	Error_AF	Unit1Em0_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
2	Discrete alarm_2	U1 | Warning that triggers a hold	Error_AF	U1_Alarms_warning[1].trigger	0	HMI_Unified_B...	<No tag>	0		
<Add new>										

Figure 13-11 Open discrete alarms configuration

- Go to the trigger tag section and select the tag configured to trigger the alarm (4). In this example, the complete errorStop array, contained in the Alarms DB, was previously created as a variable in the HMI. Choose the corresponding tag (5).



Automation_Framework_Basic_V20 ▸ DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic] ▸ HMI alarms

Discrete alarms | Analog alarms | OPC UA A&C | System events | Alarm classes

Discrete alarms

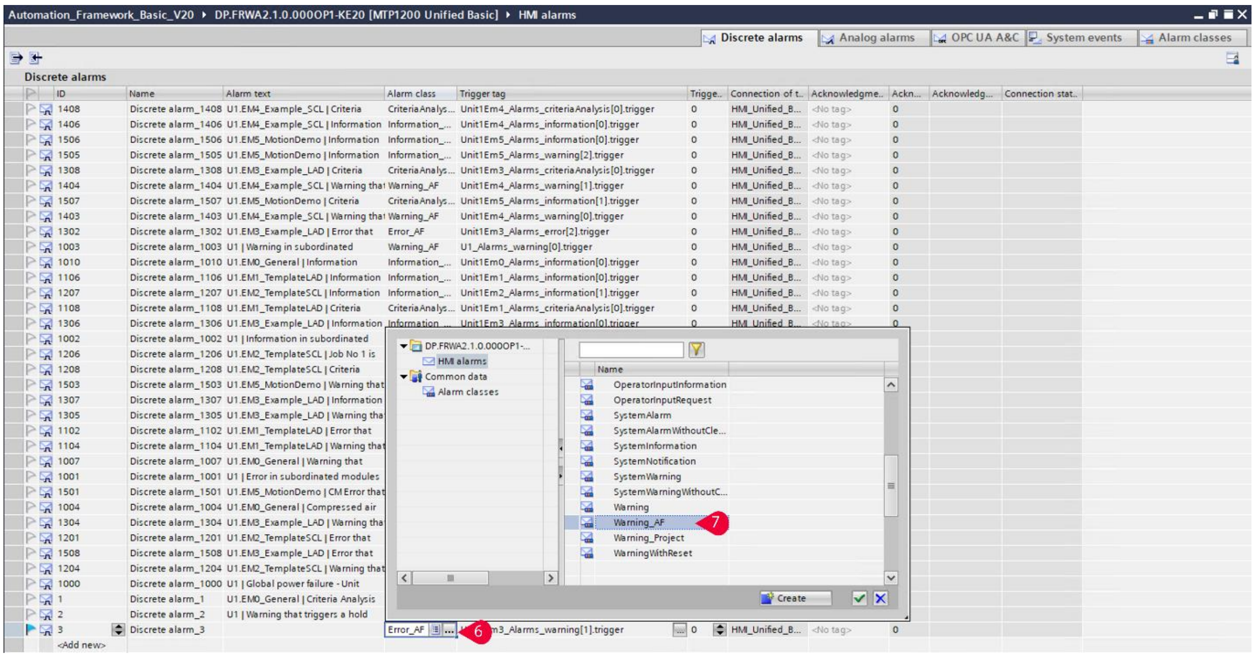
ID	Name	Alarm text	Alarm class	Trigger tag	Trigge...	Connection of t...	Acknowledgme...	Ackn...	Acknowledg...	Connection stat...
1408	Discrete alarm_1408	U1.EM4_Example_SCL | Criteria	CriteriaAnalys...	Unit1Em4_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1406	Discrete alarm_1406	U1.EM4_Example_SCL | Information	Information_...	Unit1Em4_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1506	Discrete alarm_1506	U1.EM5_MotionDemo | Information	Information_...	Unit1Em5_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1505	Discrete alarm_1505	U1.EM5_MotionDemo | Information	Information_...	Unit1Em5_Alarms_warning[2].trigger	0	HMI_Unified_B...	<No tag>	0		
1308	Discrete alarm_1308	U1.EM3_Example_LAD | Criteria	CriteriaAnalys...	Unit1Em3_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1404	Discrete alarm_1404	U1.EM4_Example_SCL | Warning that	Warning_AF	Unit1Em4_Alarms_warning[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1507	Discrete alarm_1507	U1.EM5_MotionDemo | Criteria	CriteriaAnalys...	Unit1Em5_Alarms_information[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1403	Discrete alarm_1403	U1.EM4_Example_SCL | Warning that	Warning_AF	Unit1Em4_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1302	Discrete alarm_1302	U1.EM3_Example_LAD | Error that	Error_AF	Unit1Em3_Alarms_error[2].trigger	0	HMI_Unified_B...	<No tag>	0		
1003	Discrete alarm_1003	U1 | Warning in subordinated	Warning_AF	U1_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1010	Discrete alarm_1010	U1.EM0_General | Information	Information_...	Unit1Em0_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1106	Discrete alarm_1106	U1.EM1_TemplateLAD | Information	Information_...	Unit1Em1_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1207	Discrete alarm_1207	U1.EM2_TemplateSCL | Information	Information_...							
1108	Discrete alarm_1108	U1.EM1_TemplateLAD | Criteria	CriteriaAnalys...							
1306	Discrete alarm_1306	U1.EM3_Example_LAD | Information	Information_...							
1002	Discrete alarm_1002	U1 | Information in subordinated	Information_...							
1206	Discrete alarm_1206	U1.EM2_TemplateSCL | Job No 1 is	Information_...							
1208	Discrete alarm_1208	U1.EM2_TemplateSCL | Criteria	CriteriaAnalys...							
1503	Discrete alarm_1503	U1.EM5_MotionDemo | Warning that	Warning_AF							
1307	Discrete alarm_1307	U1.EM3_Example_LAD | Information	Information_...							
1305	Discrete alarm_1305	U1.EM3_Example_LAD | Warning tha	Warning_AF							
1102	Discrete alarm_1102	U1.EM1_TemplateLAD | Error that	Error_AF							
1104	Discrete alarm_1104	U1.EM1_TemplateLAD | Warning that	Warning_AF							
1007	Discrete alarm_1007	U1.EM0_General | Warning that	Warning_AF							
1001	Discrete alarm_1001	U1 | Error in subordinated modules	Error_AF							
1501	Discrete alarm_1501	U1.EM5_MotionDemo | CM Error that	Error_AF							
1004	Discrete alarm_1004	U1.EM0_General | Compressed air	Error_AF							
1304	Discrete alarm_1304	U1.EM3_Example_LAD | Warning tha	Warning_AF							
1201	Discrete alarm_1201	U1.EM2_TemplateSCL | Error that	Error_AF							
1508	Discrete alarm_1508	U1.EM3_Example_LAD | Error that	Error_AF							
1204	Discrete alarm_1204	U1.EM2_TemplateSCL | Warning that	Warning_AF							
1000	Discrete alarm_1000	U1 | Global power failure - Unit	Error_AF							
1	Discrete alarm_1	U1.EM0_General | Criteria Analysis	Error_AF							
2	Discrete alarm_2	U1 | Warning that triggers a hold	Error_AF							
3	Discrete alarm_3		Error_AF	<No tag>	0		<No tag>	0		
<Add new>										

- CentralFunctions
- Global
- Unit1
  - U1 [8]
  - U1Em0_General [6]
  - U1Em1_Template_LAD [8]
  - U1Em2_Template_SCL [8]
  - U1Em3_Example_LAD [12]
    - Unit1Em3_Alarms_criteriaAnaly...
    - Unit1Em3_Alarms_error
    - Unit1Em3_Alarms_information
    - Unit1Em3_Alarms_warning
      - [0]
      - [1]
      - [2]
    - Unit1Em3HmiInterface_em
    - Unit1Em3HmiInterface_jobs
    - Unit1Em3HmiInterface_sequen...

Name
None
subcategory
trigger

Figure 13-12 Trigger tag configuration

- Assign the corresponding alarm class for the configured alarm (6). Use the classes previously configured for the AF (7).



Automation_Framework_Basic_V20 ▸ DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic] ▸ HMI alarms

Discrete alarms

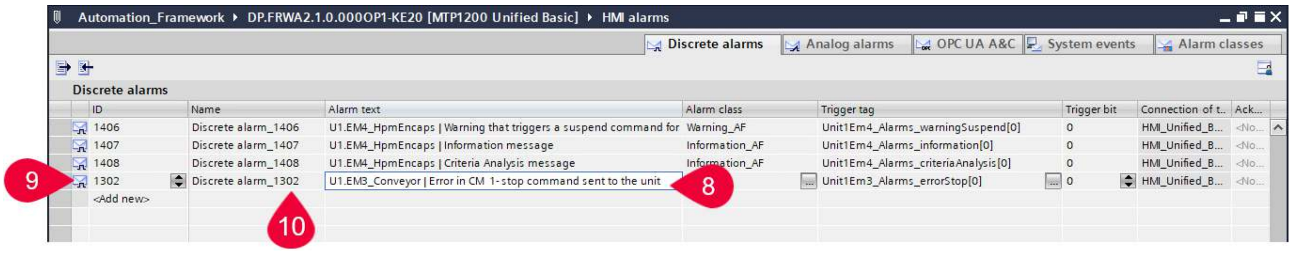
ID	Name	Alarm text	Alarm class	Trigger tag	Trigge..	Connection of t..	Acknowledgme..	Ackn...	Acknowledg...	Connection stat..
1408	Discrete alarm_1408	U1.EM4_Example_SCL | Criteria	CriteriaAnalys...	Unit1Em4_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1406	Discrete alarm_1406	U1.EM4_Example_SCL | Information	Information_...	Unit1Em4_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1506	Discrete alarm_1506	U1.EM5_MotionDemo | Information	Information_...	Unit1Em5_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1505	Discrete alarm_1505	U1.EM5_MotionDemo | Information	Information_...	Unit1Em5_Alarms_warning[2].trigger	0	HMI_Unified_B...	<No tag>	0		
1308	Discrete alarm_1308	U1.EM3_Example_LAD | Criteria	CriteriaAnalys...	Unit1Em3_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1404	Discrete alarm_1404	U1.EM4_Example_SCL | Warning that	Warning_AF	Unit1Em4_Alarms_warning[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1507	Discrete alarm_1507	U1.EM5_MotionDemo | Criteria	CriteriaAnalys...	Unit1Em5_Alarms_information[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1403	Discrete alarm_1403	U1.EM4_Example_SCL | Warning that	Warning_AF	Unit1Em4_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1302	Discrete alarm_1302	U1.EM3_Example_LAD | Error that	Error_AF	Unit1Em3_Alarms_error[2].trigger	0	HMI_Unified_B...	<No tag>	0		
1003	Discrete alarm_1003	U1 | Warning in subordinated	Warning_AF	U1_Alarms_warning[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1010	Discrete alarm_1010	U1.EM0_General | Information	Information_...	Unit1Em0_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1106	Discrete alarm_1106	U1.EM1_TemplateLAD | Information	Information_...	Unit1Em1_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1207	Discrete alarm_1207	U1.EM2_TemplateSCL | Information	Information_...	Unit1Em2_Alarms_information[1].trigger	0	HMI_Unified_B...	<No tag>	0		
1108	Discrete alarm_1108	U1.EM1_TemplateLAD | Criteria	CriteriaAnalys...	Unit1Em1_Alarms_criteriaAnalysis[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1306	Discrete alarm_1306	U1.EM3_Example_LAD | Information	Information_...	Unit1Em3_Alarms_information[0].trigger	0	HMI_Unified_B...	<No tag>	0		
1002	Discrete alarm_1002	U1 | Information in subordinated								
1206	Discrete alarm_1206	U1.EM2_TemplateSCL | Job No 1 is								
1208	Discrete alarm_1208	U1.EM2_TemplateSCL | Criteria								
1503	Discrete alarm_1503	U1.EM5_MotionDemo | Warning that								
1307	Discrete alarm_1307	U1.EM3_Example_LAD | Information								
1305	Discrete alarm_1305	U1.EM3_Example_LAD | Warning tha								
1102	Discrete alarm_1102	U1.EM1_TemplateLAD | Error that								
1104	Discrete alarm_1104	U1.EM1_TemplateLAD | Warning that								
1007	Discrete alarm_1007	U1.EM0_General | Warning that								
1001	Discrete alarm_1001	U1 | Error in subordinated modules								
1501	Discrete alarm_1501	U1.EM5_MotionDemo | CM Error that								
1004	Discrete alarm_1004	U1.EM0_General | Compressed air								
1304	Discrete alarm_1304	U1.EM3_Example_LAD | Warning tha								
1201	Discrete alarm_1201	U1.EM2_TemplateSCL | Error that								
1508	Discrete alarm_1508	U1.EM3_Example_LAD | Error that								
1204	Discrete alarm_1204	U1.EM2_TemplateSCL | Warning that								
1000	Discrete alarm_1000	U1 | Global power failure - Unit								
1	Discrete alarm_1	U1.EM0_General | Criteria Analysis								
2	Discrete alarm_2	U1 | Warning that triggers a hold								
3	Discrete alarm_3		Error_AF	m3_Alarms_warning[1].trigger	0	HMI_Unified_B...	<No tag>	0		
<Add new>										

DP.FRWA2.1.0.000OP1-...
HMI alarms
Common data
Alarm classes

Name
OperatorInputInformation
OperatorInputRequest
SystemAlarm
SystemAlarmWithoutCle...
SystemInformation
SystemNotification
SystemWarning
SystemWarningWithoutC...
Warning
Warning_AF
Warning_Project
WarningWithReset

Figure 13-13 Alarm class selection

- Write the alarm text. This is the information that will be displayed when the alarm is triggered (8). Use meaningful text that helps identify and resolve the issue, and include the information configured in the PLC tag referenceDesignatorCombined. This tag can be found within the Config DB of the Unit or EM.
- Assign the Alarm ID, which is a unique identifier for the alarms identification(9)
- Add the alarm ID to the Alarm name (10)



Automation_Framework ▸ DP.FRWA2.1.0.000OP1-KE20 [MTP1200 Unified Basic] ▸ HMI alarms

Discrete alarms

ID	Name	Alarm text	Alarm class	Trigger tag	Trigger bit	Connection of t..	Ack...
1406	Discrete alarm_1406	U1.EM4_HpmEncaps | Warning that triggers a suspend command for	Warning_AF	Unit1Em4_Alarms_warningSuspend[0]	0	HMI_Unified_B...	<No...
1407	Discrete alarm_1407	U1.EM4_HpmEncaps | Information message	Information_AF	Unit1Em4_Alarms_information[0]	0	HMI_Unified_B...	<No...
1408	Discrete alarm_1408	U1.EM4_HpmEncaps | Criteria Analysis message	Information_AF	Unit1Em4_Alarms_criteriaAnalysis[0]	0	HMI_Unified_B...	<No...
1302	Discrete alarm_1302	U1.EM3_Conveyor | Error in CM 1-stop command sent to the unit		Unit1Em3_Alarms_errorStop[0]	0	HMI_Unified_B...	<No...
<Add new>							

Figure 13-14 Alarm text , ID and name definition.

NOTE

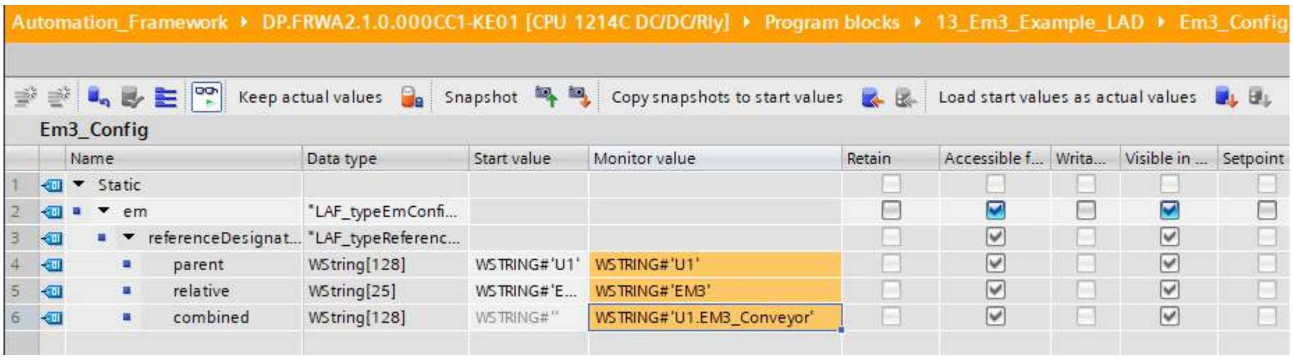
The alarm ID structure in the Automation Framework Basic was defined to ensure a consistent and unique ID for the alarms. However, the user can create their own format.

It is composed as follows:

Unit No. (1st digit) + Em No.(2nd digit) + Alarm No. (3rd and 4th digits) = Unit ‘1’ + EM ‘3’ + Alarm ‘02’

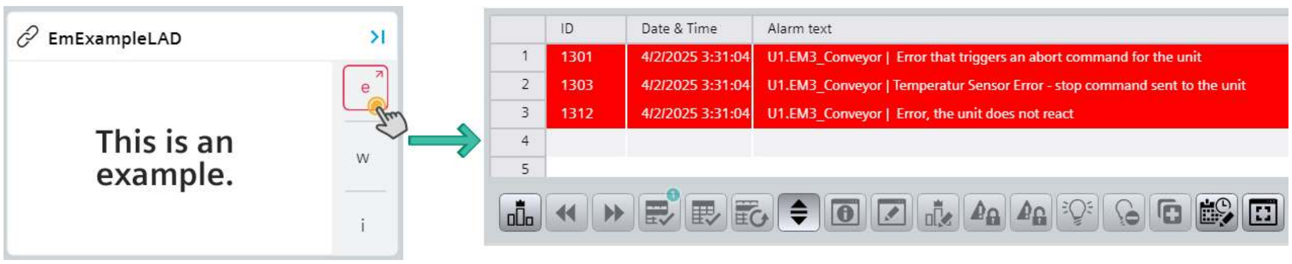
NOTE

The faceplate *"LAF_fpManageAlarms"* performs a filtering function to display active errors, warnings, or information messages at the EM or Unit level. For this to work correctly, the alarm text must include the combined reference designation


Automation_Framework ▸ DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/Rly] ▸ Program blocks ▸ 13_Em3_Example_LAD ▸ Em3_Config
Keep actual values
Snapshot
Copy snapshots to start values
Load start values as actual values
Em3_Config

	Name	Data type	Start value	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static								
2	em	"LAF_typeEmConfi...				✓		✓	
3	referenceDesignat...	"LAF_typeReferenc...				✓		✓	
4	parent	WString[128]	WSTRING#'U1'	WSTRING#'U1'		✓		✓	
5	relative	WString[25]	WSTRING#'E...	WSTRING#'EM3'		✓		✓	
6	combined	WString[128]	WSTRING#''	WSTRING#'U1.EM3_Conveyor'		✓		✓	

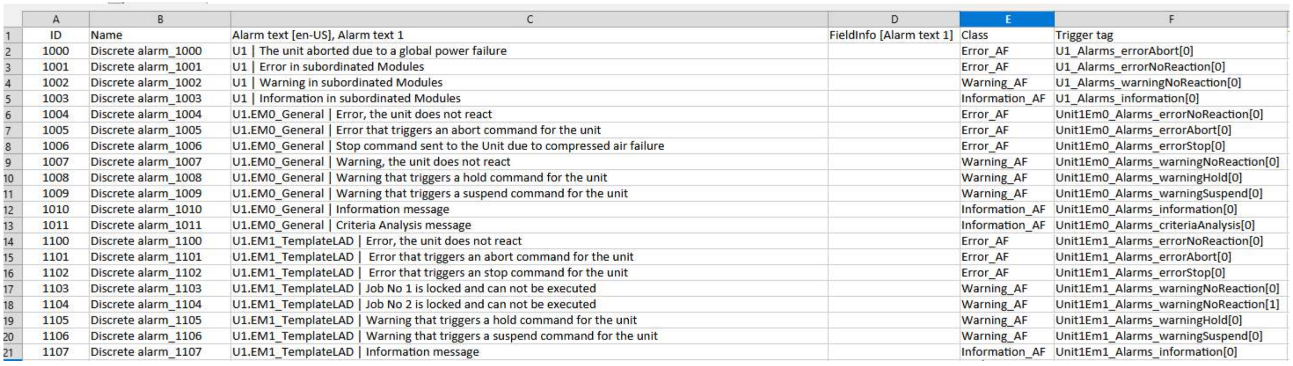
Figure 13-15 Unit or EM Reference Designation



	ID	Date & Time	Alarm text
1	1301	4/2/2025 3:31:04	U1.EM3_Conveyor | Error that triggers an abort command for the unit
2	1303	4/2/2025 3:31:04	U1.EM3_Conveyor | Temperatur Sensor Error - stop command sent to the unit
3	1312	4/2/2025 3:31:04	U1.EM3_Conveyor | Error, the unit does not react
4			
5			

Figure 13-16 LAF_fpManageAlarms filters and displays the active alarms in the EM or Unit

Alarms configuration can also be managed via export and import file functions. Using a spreadsheet for editing allows users to significantly accelerate the creation or modification of alarms in the HMI.



	A	B	C	D	E	F
1	ID	Name	Alarm text [en-US], Alarm text 1	FieldInfo [Alarm text 1]	Class	Trigger tag
2	1000	Discrete alarm_1000	U1 | The unit aborted due to a global power failure		Error_AF	U1_Alarms_errorAbort[0]
3	1001	Discrete alarm_1001	U1 | Error in subordinated Modules		Error_AF	U1_Alarms_errorNoReaction[0]
4	1002	Discrete alarm_1002	U1 | Warning in subordinated Modules		Warning_AF	U1_Alarms_warningNoReaction[0]
5	1003	Discrete alarm_1003	U1 | Information in subordinated Modules		Information_AF	U1_Alarms_information[0]
6	1004	Discrete alarm_1004	U1.EM0_General | Error, the unit does not react		Error_AF	Unit1Em0_Alarms_errorNoReaction[0]
7	1005	Discrete alarm_1005	U1.EM0_General | Error that triggers an abort command for the unit		Error_AF	Unit1Em0_Alarms_errorAbort[0]
8	1006	Discrete alarm_1006	U1.EM0_General | Stop command sent to the Unit due to compressed air failure		Error_AF	Unit1Em0_Alarms_errorStop[0]
9	1007	Discrete alarm_1007	U1.EM0_General | Warning, the unit does not react		Warning_AF	Unit1Em0_Alarms_warningNoReaction[0]
10	1008	Discrete alarm_1008	U1.EM0_General | Warning that triggers a hold command for the unit		Warning_AF	Unit1Em0_Alarms_warningHold[0]
11	1009	Discrete alarm_1009	U1.EM0_General | Warning that triggers a suspend command for the unit		Warning_AF	Unit1Em0_Alarms_warningSuspend[0]
12	1010	Discrete alarm_1010	U1.EM0_General | Information message		Information_AF	Unit1Em0_Alarms_information[0]
13	1011	Discrete alarm_1011	U1.EM0_General | Criteria Analysis message		Information_AF	Unit1Em0_Alarms_criteriaAnalysis[0]
14	1100	Discrete alarm_1100	U1.EM1_TemplateLAD | Error, the unit does not react		Error_AF	Unit1Em1_Alarms_errorNoReaction[0]
15	1101	Discrete alarm_1101	U1.EM1_TemplateLAD | Error that triggers an abort command for the unit		Error_AF	Unit1Em1_Alarms_errorAbort[0]
16	1102	Discrete alarm_1102	U1.EM1_TemplateLAD | Error that triggers an stop command for the unit		Error_AF	Unit1Em1_Alarms_errorStop[0]
17	1103	Discrete alarm_1103	U1.EM1_TemplateLAD | Job No 1 is locked and can not be executed		Warning_AF	Unit1Em1_Alarms_warningNoReaction[0]
18	1104	Discrete alarm_1104	U1.EM1_TemplateLAD | Job No 2 is locked and can not be executed		Warning_AF	Unit1Em1_Alarms_warningNoReaction[1]
19	1105	Discrete alarm_1105	U1.EM1_TemplateLAD | Warning that triggers a hold command for the unit		Warning_AF	Unit1Em1_Alarms_warningHold[0]
20	1106	Discrete alarm_1106	U1.EM1_TemplateLAD | Warning that triggers a suspend command for the unit		Warning_AF	Unit1Em1_Alarms_warningSuspend[0]
21	1107	Discrete alarm_1107	U1.EM1_TemplateLAD | Information message		Information_AF	Unit1Em1_Alarms_information[0]

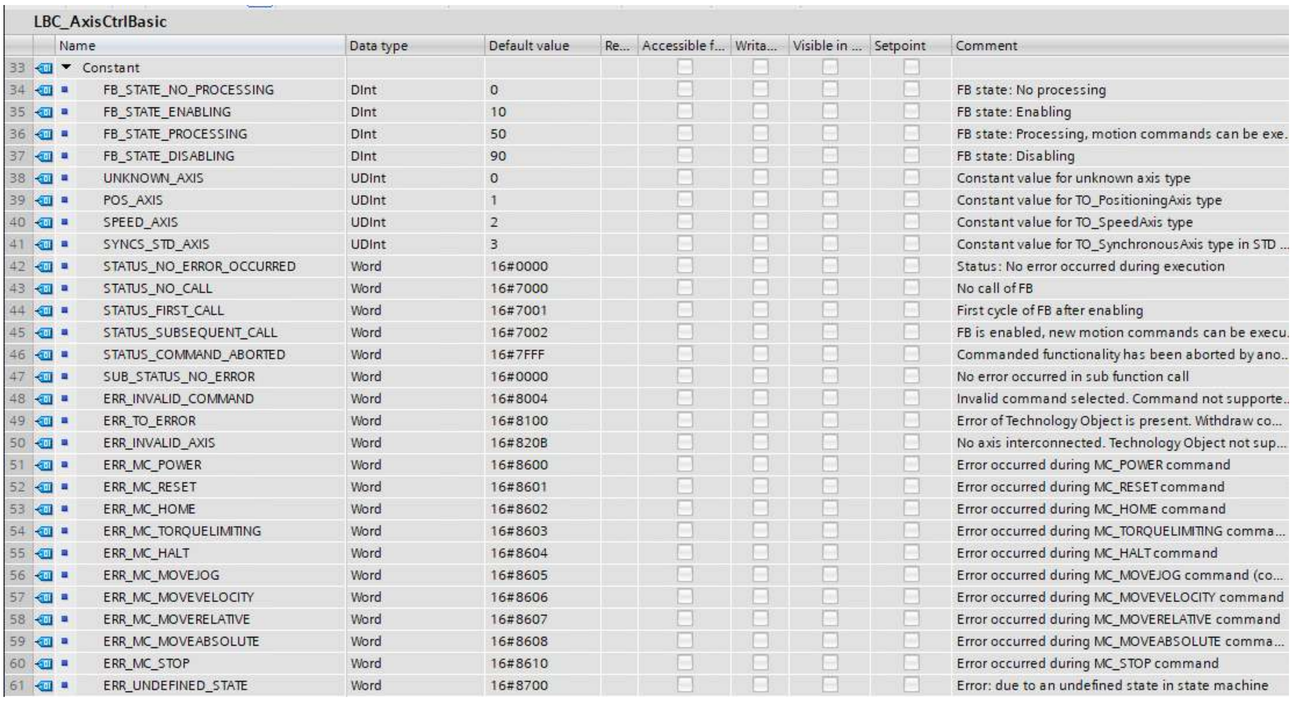
Figure 13-17 Editing Alarms using a spreadsheet software

13.3.3. Error handling within the LBC Library Blocks

Error Detection

The LBC blocks 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).


LBC_AxisCtrlBasic

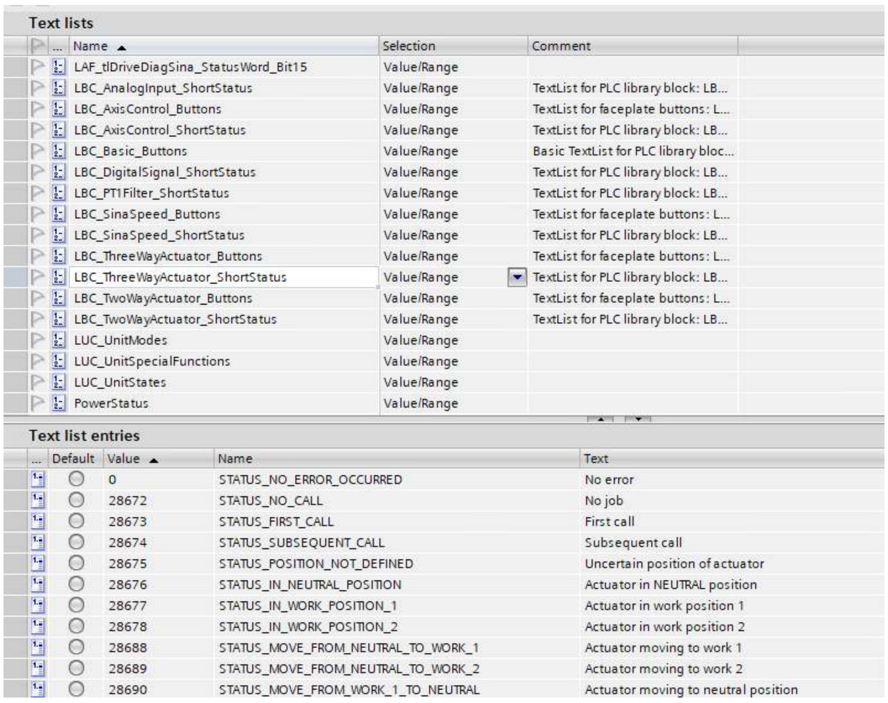
	Name	Data type	Default value	Re...	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
33	▼ Constant				☐	☐	☐	☐	
34	FB_STATE_NO_PROCESSING	DInt	0		☐	☐	☐	☐	FB state: No processing
35	FB_STATE_ENABLING	DInt	10		☐	☐	☐	☐	FB state: Enabling
36	FB_STATE_PROCESSING	DInt	50		☐	☐	☐	☐	FB state: Processing, motion commands can be exe.
37	FB_STATE_DISABLING	DInt	90		☐	☐	☐	☐	FB state: Disabling
38	UNKNOWN_AXIS	UDInt	0		☐	☐	☐	☐	Constant value for unknown axis type
39	POS_AXIS	UDInt	1		☐	☐	☐	☐	Constant value for TO_PositioningAxis type
40	SPEED_AXIS	UDInt	2		☐	☐	☐	☐	Constant value for TO_SpeedAxis type
41	SYNCS_STD_AXIS	UDInt	3		☐	☐	☐	☐	Constant value for TO_SynchronousAxis type in STD ...
42	STATUS_NO_ERROR_OCCURRED	Word	16#0000		☐	☐	☐	☐	Status: No error occurred during execution
43	STATUS_NO_CALL	Word	16#7000		☐	☐	☐	☐	No call of FB
44	STATUS_FIRST_CALL	Word	16#7001		☐	☐	☐	☐	First cycle of FB after enabling
45	STATUS_SUBSEQUENT_CALL	Word	16#7002		☐	☐	☐	☐	FB is enabled, new motion commands can be execu.
46	STATUS_COMMAND_ABORTED	Word	16#7FFF		☐	☐	☐	☐	Commanded functionality has been aborted by ano..
47	SUB_STATUS_NO_ERROR	Word	16#0000		☐	☐	☐	☐	No error occurred in sub function call
48	ERR_INVALID_COMMAND	Word	16#8004		☐	☐	☐	☐	Invalid command selected. Command not supporte..
49	ERR_TO_ERROR	Word	16#8100		☐	☐	☐	☐	Error of Technology Object is present. Withdraw co...
50	ERR_INVALID_AXIS	Word	16#820B		☐	☐	☐	☐	No axis interconnected. Technology Object not sup...
51	ERR_MC_POWER	Word	16#8600		☐	☐	☐	☐	Error occurred during MC_POWER command
52	ERR_MC_RESET	Word	16#8601		☐	☐	☐	☐	Error occurred during MC_RESET command
53	ERR_MC_HOME	Word	16#8602		☐	☐	☐	☐	Error occurred during MC_HOME command
54	ERR_MC_TORQUELIMITING	Word	16#8603		☐	☐	☐	☐	Error occurred during MC_TORQUELIMITING comma...
55	ERR_MC_HALT	Word	16#8604		☐	☐	☐	☐	Error occurred during MC_HALT command
56	ERR_MC_MOVEJOG	Word	16#8605		☐	☐	☐	☐	Error occurred during MC_MOVEJOG command (co...
57	ERR_MC_MOVEVELOCITY	Word	16#8606		☐	☐	☐	☐	Error occurred during MC_MOVEVELOCITY command
58	ERR_MC_MOVERELATIVE	Word	16#8607		☐	☐	☐	☐	Error occurred during MC_MOVERELATIVE command
59	ERR_MC_MOVEABSOLUTE	Word	16#8608		☐	☐	☐	☐	Error occurred during MC_MOVEABSOLUTE comma...
60	ERR_MC_STOP	Word	16#8610		☐	☐	☐	☐	Error occurred during MC_STOP command
61	ERR_UNDEFINED_STATE	Word	16#8700		☐	☐	☐	☐	Error: due to an undefined state in state machine

Figure 13-18: Status code and constant for error handling in the LBC_AxisCtrlBasic block

NOTE

The LBC normal blocks used in the Automation Framework can also be used in the Automation Framework Basic version, but they have not been optimized for the G2 or for reduced code memory consumption, as is the case with the LBC_Basic blocks.

The alarm message is generated using an HMI text list provided for each LBC block, and its specific information about the status of each control module will be displayed on the screen with the specific faceplate.



Text lists

Name	Selection	Comment
LAF_tlDriveDiagSina_StatusWord_Bit15	Value/Range	
LBC_AnalogInput_ShortStatus	Value/Range	TextList for PLC library block: LB...
LBC_AxisControl_Buttons	Value/Range	TextList for faceplate buttons: L...
LBC_AxisControl_ShortStatus	Value/Range	TextList for PLC library block: LB...
LBC_Basic_Buttons	Value/Range	Basic TextList for PLC library bloc...
LBC_DigitalSignal_ShortStatus	Value/Range	TextList for PLC library block: LB...
LBC_PT1Filter_ShortStatus	Value/Range	TextList for PLC library block: LB...
LBC_SinaSpeed_Buttons	Value/Range	TextList for faceplate buttons: L...
LBC_SinaSpeed_ShortStatus	Value/Range	TextList for PLC library block: LB...
LBC_ThreeWayActuator_Buttons	Value/Range	TextList for faceplate buttons: L...
LBC_ThreeWayActuator_ShortStatus	Value/Range	TextList for PLC library block: LB...
LBC_TwoWayActuator_Buttons	Value/Range	TextList for faceplate buttons: L...
LBC_TwoWayActuator_ShortStatus	Value/Range	TextList for PLC library block: LB...
LUC_UnitModes	Value/Range	
LUC_UnitSpecialFunctions	Value/Range	
LUC_UnitStates	Value/Range	
PowerStatus	Value/Range	

Text list entries

Default	Value	Name	Text
○	0	STATUS_NO_ERROR_OCCURRED	No error
○	28672	STATUS_NO_CALL	No job
○	28673	STATUS_FIRST_CALL	First call
○	28674	STATUS_SUBSEQUENT_CALL	Subsequent call
○	28675	STATUS_POSITION_NOT_DEFINED	Uncertain position of actuator
○	28676	STATUS_IN_NEUTRAL_POSITION	Actuator in NEUTRAL position
○	28677	STATUS_IN_WORK_POSITION_1	Actuator in work position 1
○	28678	STATUS_IN_WORK_POSITION_2	Actuator in work position 2
○	28688	STATUS_MOVE_FROM_NEUTRAL_TO_WORK_1	Actuator moving to work 1
○	28689	STATUS_MOVE_FROM_NEUTRAL_TO_WORK_2	Actuator moving to work 2
○	28690	STATUS_MOVE_FROM_WORK_1_TO_NEUTRAL	Actuator moving to neutral position

Figure 13-19: HMI Text lists for LBC blocks

14. Simulation

14.1. Overview

NOTE

Additional software and license are required for 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.

Supported simulation features for S7–1200 G2

Feature	S7-PLCSIM	S7-PLCSIM Advanced
General simulation support for S7–1200 G2	Yes	No
Debugging and monitoring of PLC program and tags	PLC logic only, safety and standard program, basic I/O simulation	Not supported
Validation of user management and access control (UMAC)	Not supported	Not supported
Testing Motion Control applications with technology objects (TOs)	Update 1 (see link below) or physical S7–1200 G2 device required (TO can be tested in simulation mode without physical drive)	Not supported

Table 14-1: Supported simulation features for S7–1200 G2

NOTE

PLCSIM V20 Update 1 download link: https://support.industry.siemens.com/cs/es/en/view/109963851

14.2. PLC and HMI integration of simulation signals

PLC integration

Simulation can be activated in the PLC by setting the respective bits in the data block "Data" in folder "Global" to "TRUE". Program code in the PLC, which uses PLC internal simulation program code or should be not active while the simulation is enabled, must react to these bits:

- Data.general.simulation.withSimit
  This bit indicates that a SIMIT simulation is connected. It is not used yet and is intended for future use cases.
- Data.general.simulation.plcInternal
  This bit indicates that internal simulation is activated. In this case, the simulation is done with additional program code in the PLC internally. The start value of this bit is "TRUE" by default.

NOTE

Additional program code, which belongs to simulation, should be programmed in separate FBs and DBs. This clearly separates the user program and the simulation program code. In addition, the simulation program code can be deleted easily.

The AF$_{Basic}$ example project does not show any simulation program code. The used technology objects are set to simulation mode by default. If real hardware is used, the simulation mode must be disabled manually in the configuration of the technology object.

HMI integration

The PLC internal simulation can by be activated or deactivated by the operator via the “Simulation” button in the “MachineOverview” screen in the HMI. An active PLC internal simulation is shown with a green status (1).

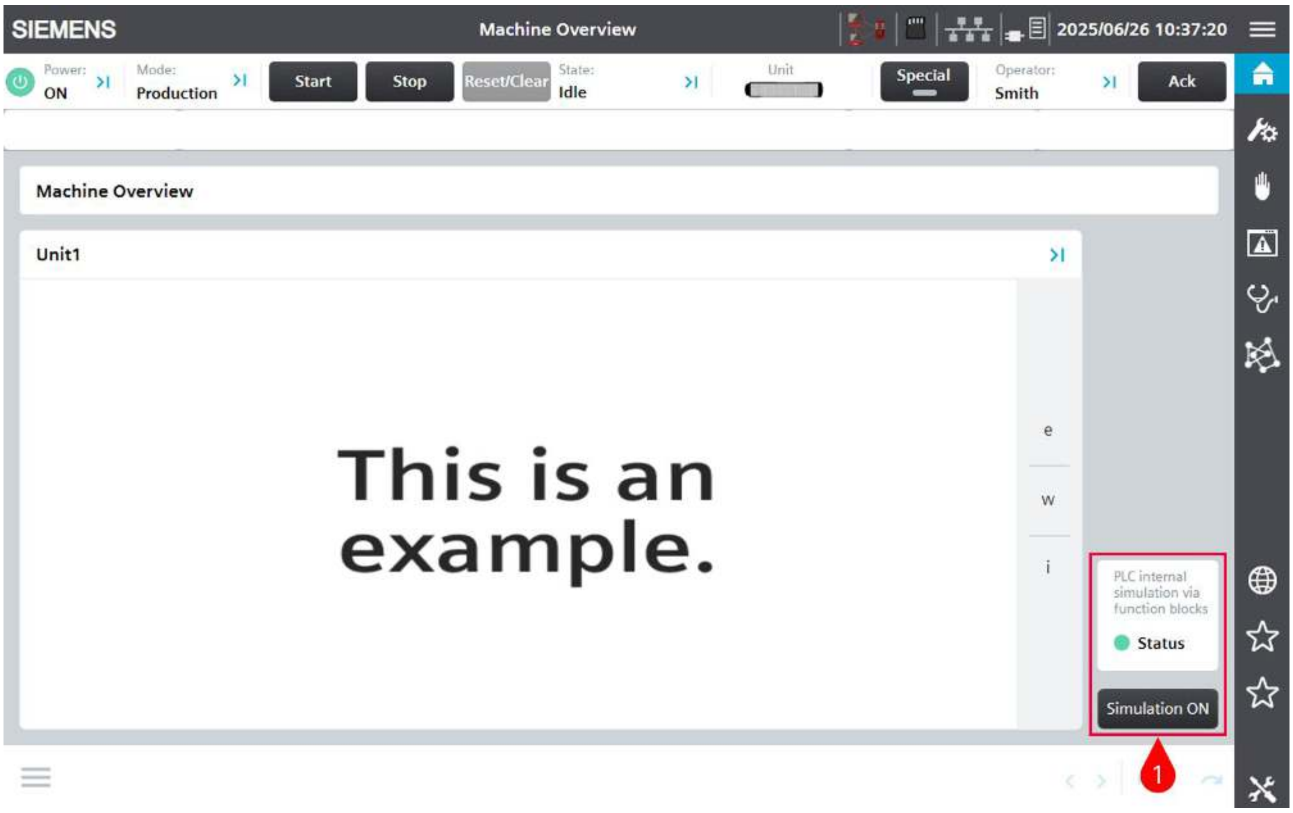


Figure 14-1: HMI objects in “MachineOverview” screen which belong to simulation

NOTE

If the simulation is not required in the HMI, then delete these HMI objects in the “MachineOverview” screen.

14.3. PLC simulation with PLCSIM

The internal simulation is activated by default in the AFBasic example project. The technology objects are set to simulation mode by default. Follow the steps below to start a PLC simulation with PLCSIM.

Start S7-PLCSIM.

Create a new PLC instance for the S7-1200 G2 by clicking the "+" icon (1).

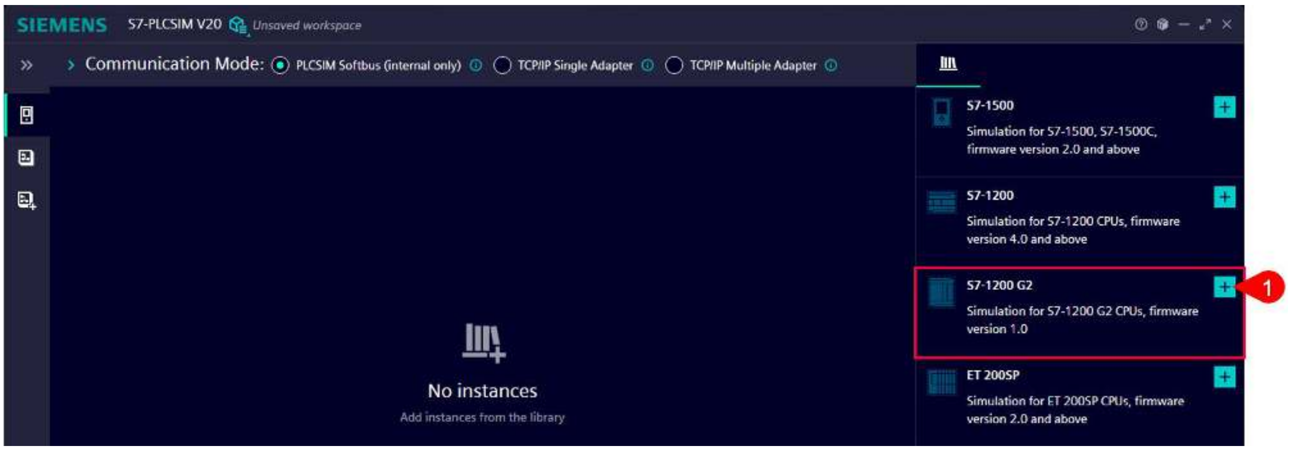

SIEMENS
S7-PLCSIM V20
Unsaved workspace
Communication Mode:
PLCSIM Softbus (internal only)
TCP/IP Single Adapter
TCP/IP Multiple Adapter
S7-1500
Simulation for S7-1500, S7-1500C, firmware version 2.0 and above
S7-1200
Simulation for S7-1200 CPUs, firmware version 4.0 and above
S7-1200 G2
Simulation for S7-1200 G2 CPUs, firmware version 1.0
ET 200SP
Simulation for ET 200SP CPUs, firmware version 2.0 and above
No instances
Add instances from the library
1

Figure 14-2: How to create a new PLC instance in PLCSIM

Start the PLC instance by clicking the "Power" icon (1).

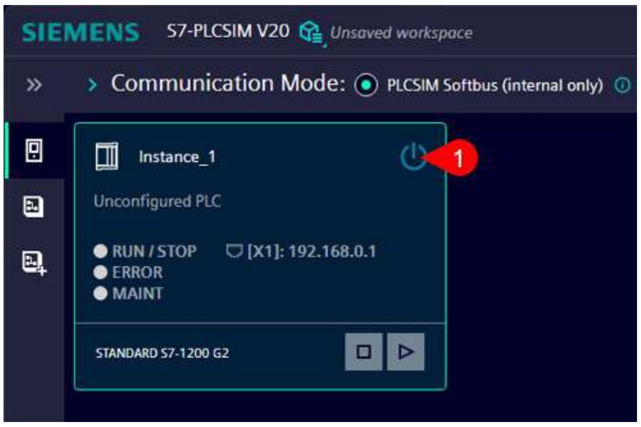
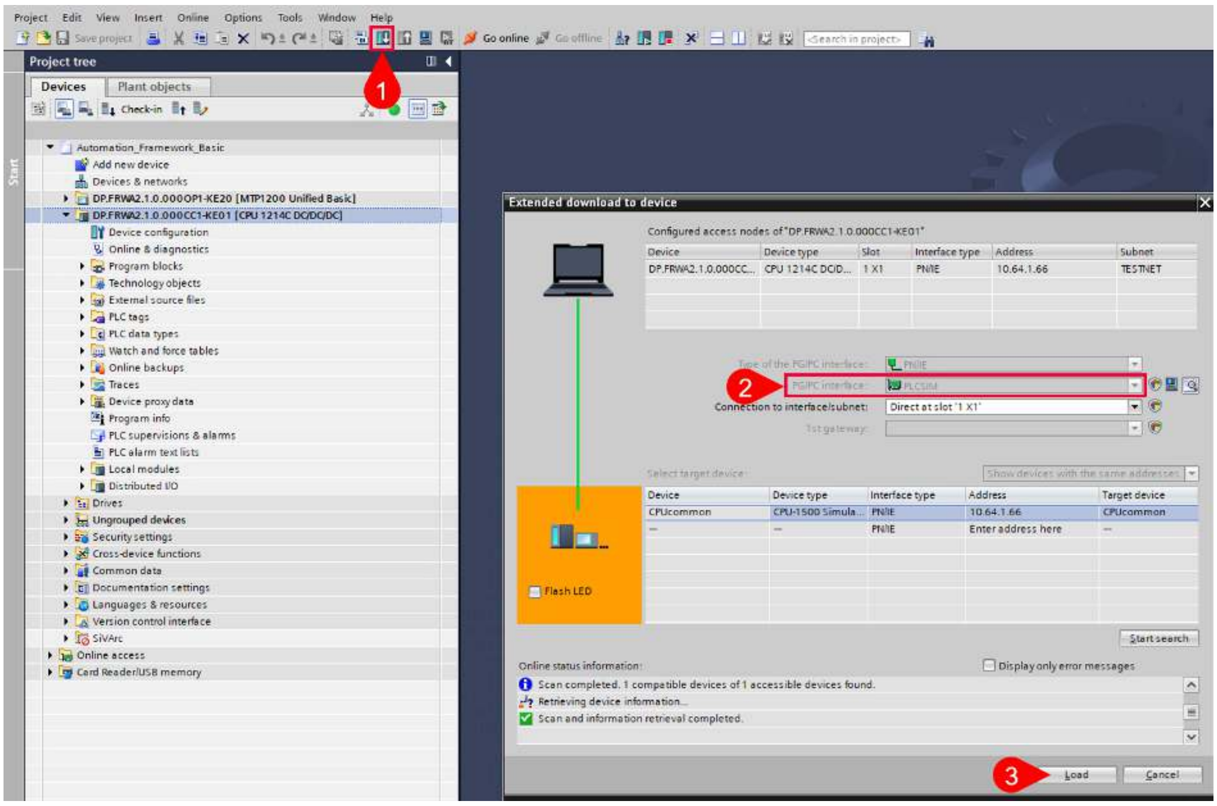


Figure 14-3: How to start a simulated PLC instance in S7-PLCSIM

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). If any following pop-ups occur, continue with the required actions.


1
2
3
Project tree
Devices
Plant objects
Automation_Framework_Basic
Add new device
Devices & networks
DP.FRWA2.1.0.0000P1-KE20 [MTP1200 Unified Basic]
DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC]
Flash LED
Start search
Load
Cancel

Extended download to device

Configured access nodes of "DP.FRWA2.1.0.000CC1-KE01"

Device	Device type	Slot	Interface type	Address	Subnet
DP.FRWA2.1.0.000CC...	CPU 1214C DC/D...	1 X1	PN/IE	10.64.1.66	TESTNET

Type of the PG/PC interface: PN/IE
PG/PC interface: PLCSIM
Connection to interface/subnet: Direct at slot '1 X1'
1st gateway:

Select target device: Show devices with the same addresses

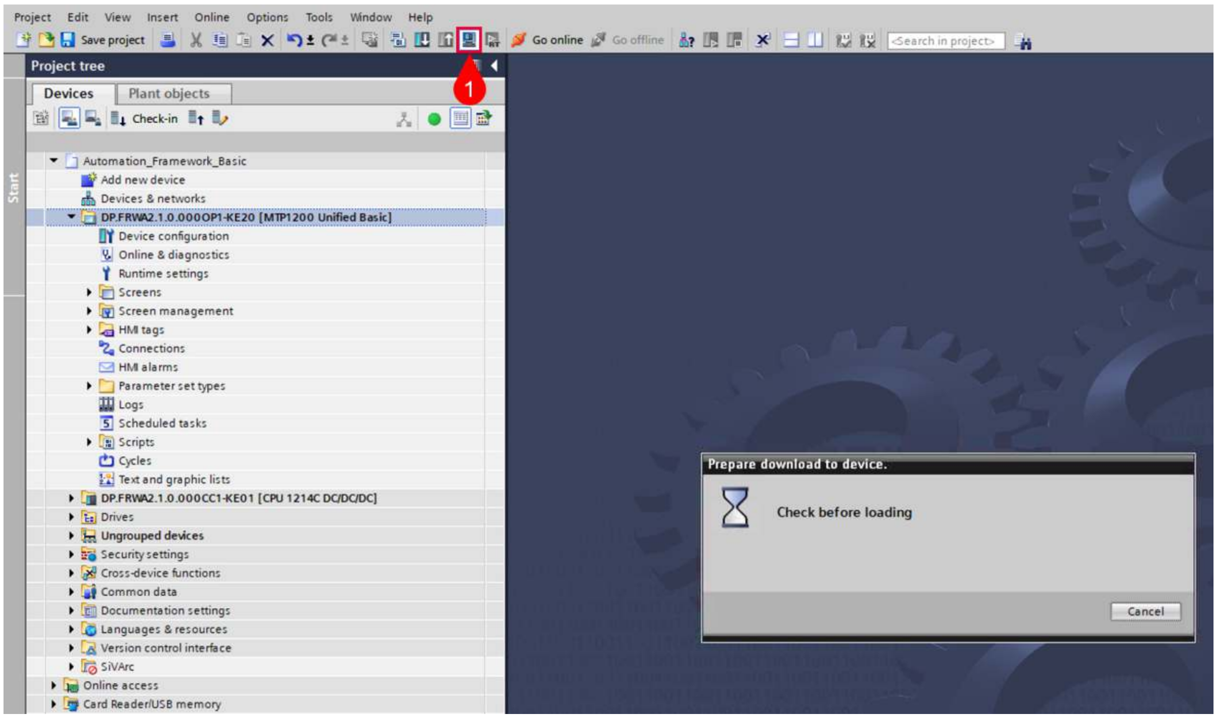
Device	Device type	Interface type	Address	Target device
CPUcommon	CPU-1500 Simula...	PN/IE	10.64.1.66	CPUcommon
--	--	PN/IE	Enter address here	--

Online status information: Display only error messages

Scan completed. 1 compatible devices of 1 accessible devices found.
Retrieving device information...
Scan and information retrieval completed.

Figure 14-4: How to download the PLC to a simulated PLC instance

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.


1
Project tree
Devices
Plant objects
Automation_Framework_Basic
DP.FRWA2.1.0.0000P1-KE20 [MTP1200 Unified Basic]
DP.FRWA2.1.0.000CC1-KE01 [CPU 1214C DC/DC/DC]
Cancel

Prepare download to device.

Check before loading

Figure 14-5: How to start the simulation of the HMI application

Go back to S7-PLCSIM and navigate to "SimView" (1). For example, add a sim table (2) from the library to simulate digital inputs or to monitor digital output.

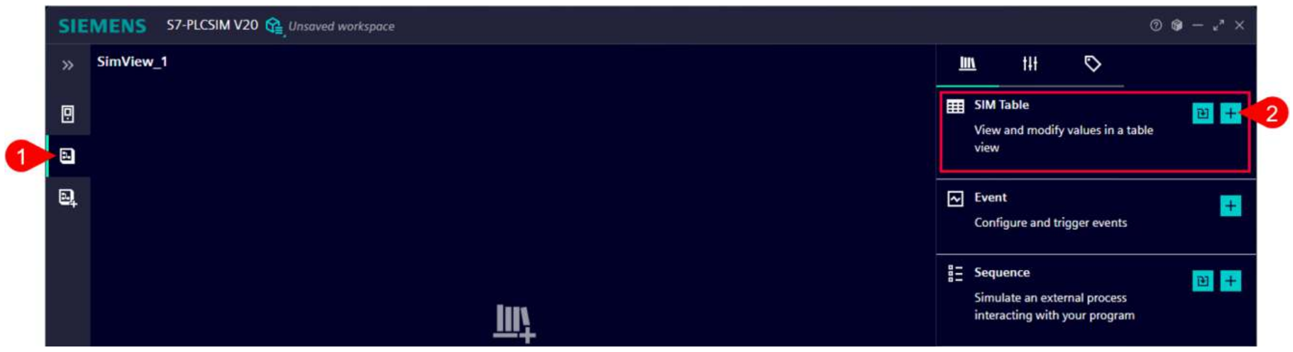

SIEMENS
S7-PLCSIM V20
Unsaved workspace
SimView_1
SIM Table
View and modify values in a table view
Event
Configure and trigger events
Sequence
Simulate an external process interacting with your program
1
2

Figure 14-6: How to add a sim table in S7-PLCSIM

NOTE

S7-PLCSIM offers additional simulation options such as events and sequences (see Figure 14-6). Check the online help by clicking the "?" icon in the top right corner of S7-PLCSIM to learn more about it.

Add PLC tags manually by entering the physical address (e.g. "Q0.0:P") into the "Address" field of the sim table or import the PLC tags from the running simulated PLC instance. For the import, navigate to "Properties" (1). If needed, set a filter and click on "Load Selected Tags" (2) to add the PLC tags to the sim table.

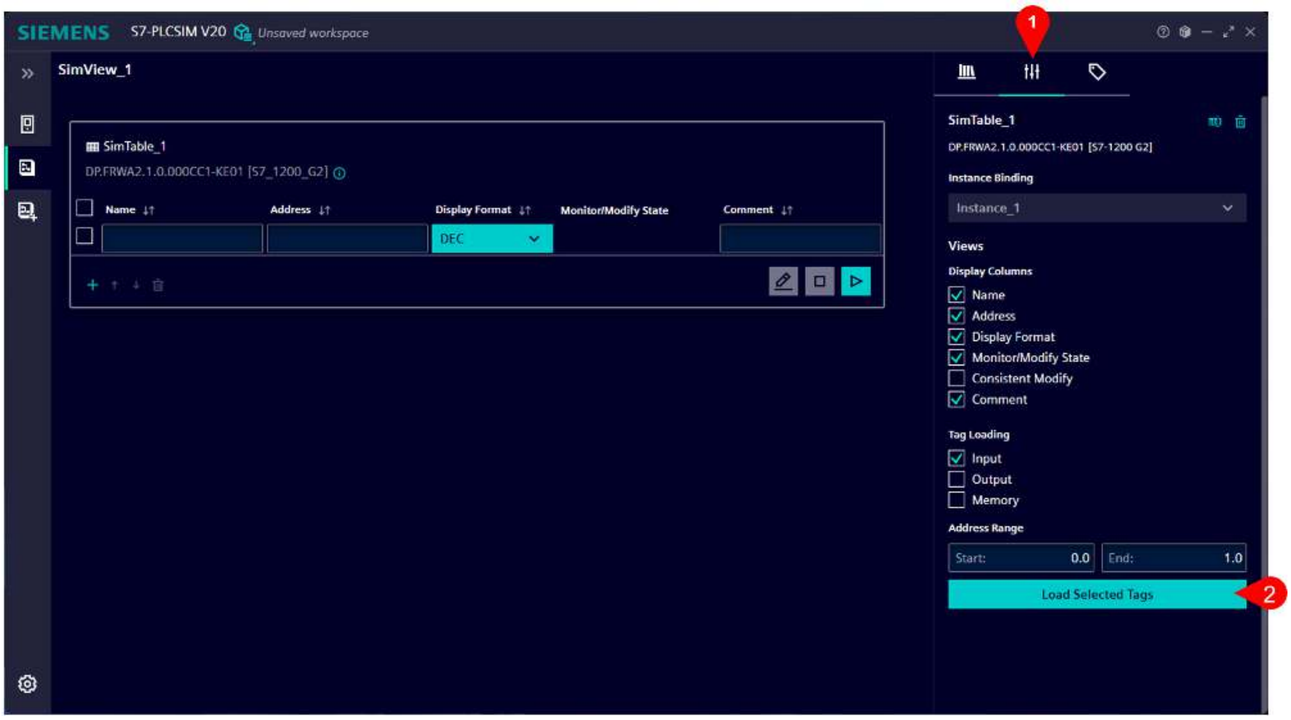
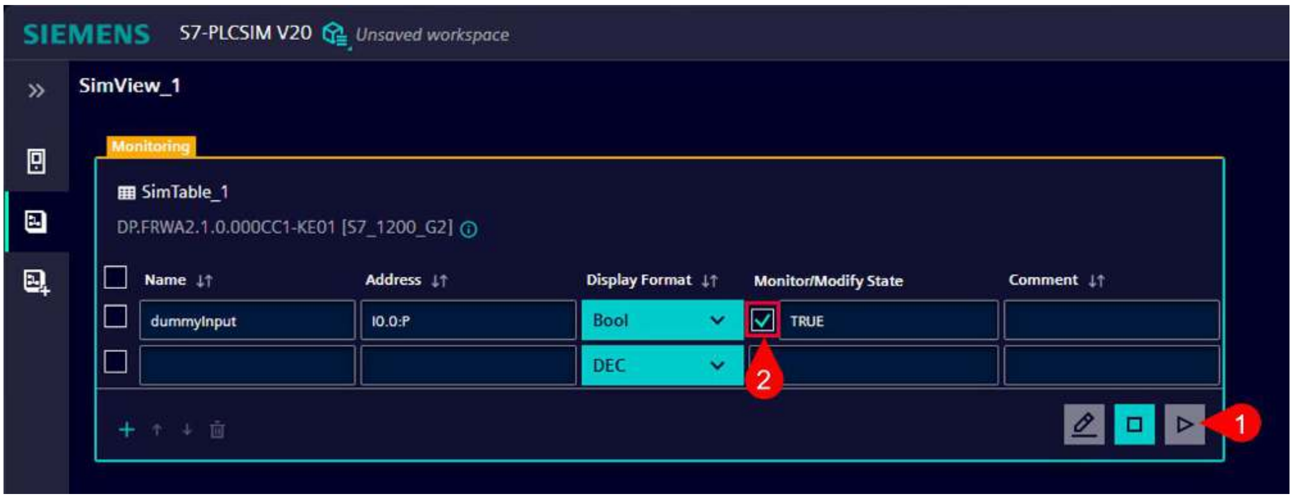


Figure 14-7: How to add PLC tags to a sim table in S7-PLCSIM

Start the sim table (1) and modify the state of the digital inputs (2) or monitor the state of the digital outputs in the online view.

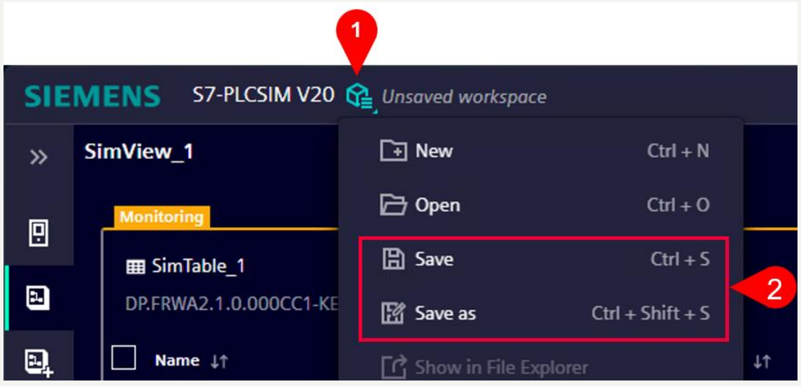


Name	Address	Display Format	Monitor/Modify State	Comment
dummyInput	I0.0:P	Bool	✓ TRUE	
		DEC		

Figure 14-8: How to start a sim table in S7-PLCSIM

NOTE

Tipp: Save the PLCSIM workspace to reuse it next time (1) – (2).



15. User Management

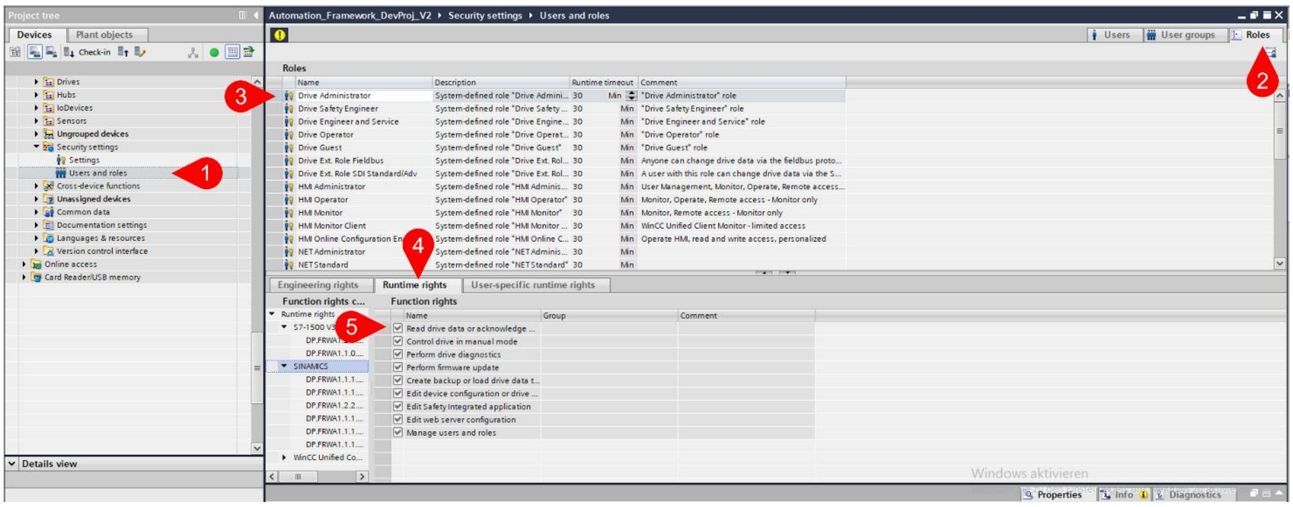
15.1. General

In the engineering system, you can specify whether you are using local, the user data is stored in the device, or central user management where the user data is loaded from UMC. By default, the use of the local user management is defined in TIA Portal.

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:



Automation_Framework_DevProj_V2 ▸ Security settings ▸ Users and roles

Roles

Name	Description	Runtime timeout		Comment
Drive Administrator	System-defined role "Drive Admini...	30	Min	"Drive Administrator" role
Drive Safety Engineer	System-defined role "Drive Safety ...	30	Min	"Drive Safety Engineer" role
Drive Engineer and Service	System-defined role "Drive Engine...	30	Min	"Drive Engineer and Service" role
Drive Operator	System-defined role "Drive Operat...	30	Min	"Drive Operator" role
Drive Guest	System-defined role "Drive Guest"	30	Min	"Drive Guest" role
Drive Ext. Role Fieldbus	System-defined role "Drive Ext. Rol...	30	Min	Anyone can change drive data via the fieldbus proto...
Drive Ext. Role SDI Standard/Adv	System-defined role "Drive Ext. Rol...	30	Min	A user with this role can change drive data via the S...
HMI Administrator	System-defined role "HMI Adminis...	30	Min	User Management, Monitor, Operate, Remote access...
HMI Operator	System-defined role "HMI Operator"	30	Min	Monitor, Operate, Remote access - Monitor only
HMI Monitor	System-defined role "HMI Monitor"	30	Min	Monitor, Remote access - Monitor only
HMI Monitor Client	System-defined role "HMI Monitor ...	30	Min	WinCC Unified Client Monitor - limited access
HMI Online Configuration En	System-defined role "HMI Online C...	30	Min	Operate HMI, read and write access, personalized
NET Administrator	System-defined role "NET Adminis...	30	Min	
NET Standard	System-defined role "NET Standard"	30	Min	

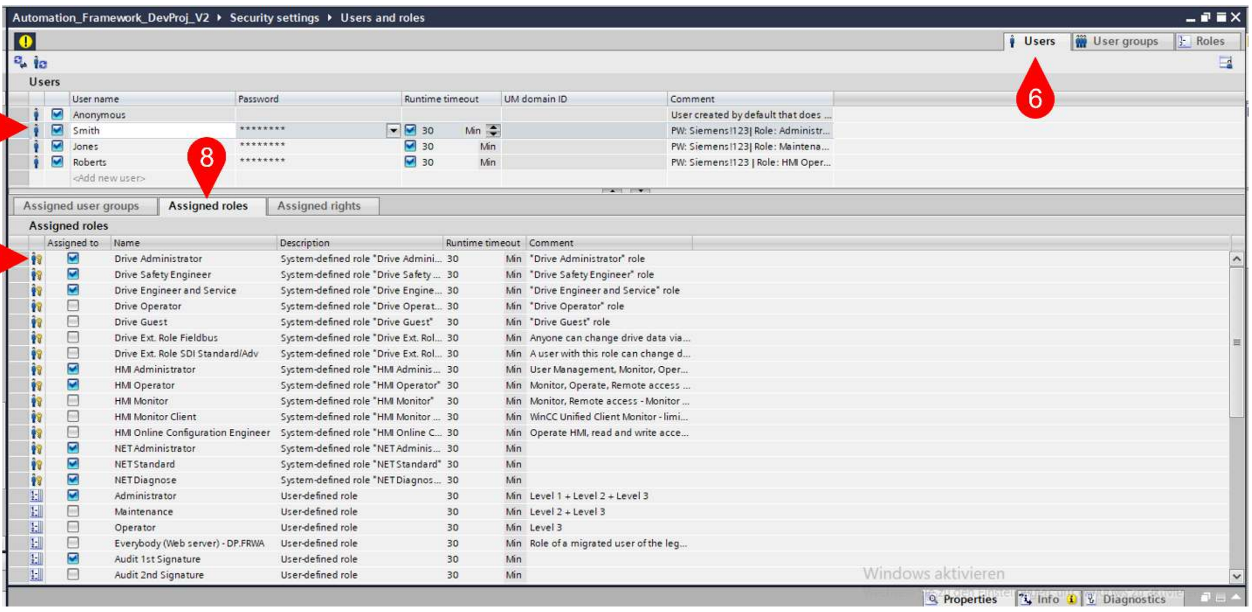
Engineering rights | Runtime rights | User-specific runtime rights

Function rights c...
- Runtime rights
  - S7-1500 V3
    - DP.FRWA1....
    - DP.FRWA1.1.0....
  - SINAMICS
    - DP.FRWA1.1.1....
    - DP.FRWA1.1.1....
    - DP.FRWA1.2.2....
    - DP.FRWA1.1.1....
    - DP.FRWA1.1.1....
    - DP.FRWA1.1.1....
  - WinCC Unified Co...

Function rights

	Name	Group	Comment
✓	Read drive data or acknowledge ...		
✓	Control drive in manual mode		
✓	Perform drive diagnostics		
✓	Perform firmware update		
✓	Create backup or load drive data t...		
✓	Edit device configuration or drive ...		
✓	Edit Safety Integrated application		
✓	Edit web server configuration		
✓	Manage users and roles		

Figure 15-1: Configurate roles and rights



Users

		User name	Password	Runtime timeout		UM domain ID	Comment
	✓	Anonymous					User created by default that does ...
	✓	Smith	********	✓ 30	Min		PW: Siemens!123| Role: Administr...
	✓	Jones	********	✓ 30	Min		PW: Siemens!123| Role: Maintena...
	✓	Roberts	********	✓ 30	Min		PW: Siemens!123 | Role: HMI Oper...
		<Add new user>					

Assigned roles

	Assigned to	Name	Description	Runtime timeout		Comment
	✓	Drive Administrator	System-defined role "Drive Admini...	30	Min	"Drive Administrator" role
	✓	Drive Safety Engineer	System-defined role "Drive Safety ...	30	Min	"Drive Safety Engineer" role
	✓	Drive Engineer and Service	System-defined role "Drive Engine...	30	Min	"Drive Engineer and Service" role
		Drive Operator	System-defined role "Drive Operat...	30	Min	"Drive Operator" role
		Drive Guest	System-defined role "Drive Guest"	30	Min	"Drive Guest" role
		Drive Ext. Role Fieldbus	System-defined role "Drive Ext. Rol...	30	Min	Anyone can change drive data via...
		Drive Ext. Role SDI Standard/Adv	System-defined role "Drive Ext. Rol...	30	Min	A user with this role can change d...
	✓	HMI Administrator	System-defined role "HMI Adminis...	30	Min	User Management, Monitor, Oper...
	✓	HMI Operator	System-defined role "HMI Operator"	30	Min	Monitor, Operate, Remote access ...
		HMI Monitor	System-defined role "HMI Monitor"	30	Min	Monitor, Remote access - Monitor ...
		HMI Monitor Client	System-defined role "HMI Monitor ...	30	Min	WinCC Unified Client Monitor - limi...
		HMI Online Configuration Engineer	System-defined role "HMI Online C...	30	Min	Operate HMI, read and write acce...
	✓	NET Administrator	System-defined role "NET Adminis...	30	Min	
	✓	NET Standard	System-defined role "NET Standard"	30	Min	
	✓	NET Diagnose	System-defined role "NET Diagnos...	30	Min	
	✓	Administrator	User-defined role	30	Min	Level 1 + Level 2 + Level 3
		Maintenance	User-defined role	30	Min	Level 2 + Level 3
		Operator	User-defined role	30	Min	Level 3
		Everybody (Web server) - DP.FRWA	User-defined role	30	Min	Role of a migrated user of the leg...
	✓	Audit 1st Signature	User-defined role	30	Min	
		Audit 2nd Signature	User-defined role	30	Min	

Figure 15-2: Configurate users and assigned roles

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

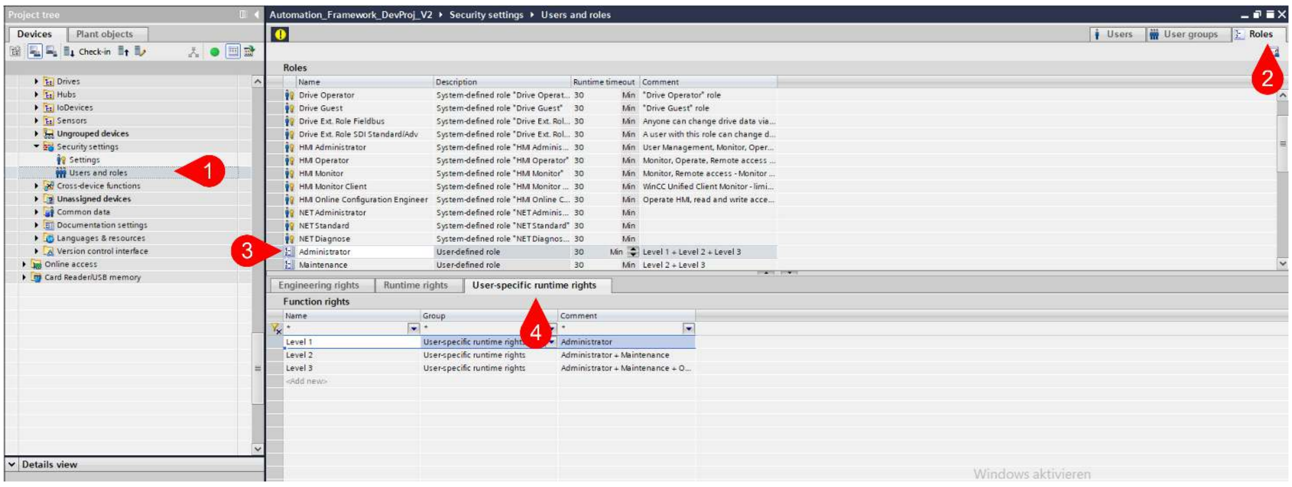
It is recommended to activate UMAC or Project protection earliest in commissioning phase if you need to make sure that functions or configurations are not changed from anybody anymore, latest after the handover to the end user.

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/de/en/view/109826862.

15.2. Demo Implementation

The AFBasic 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).



Roles

Name	Description	Runtime timeout		Comment
Drive Operator	System-defined role "Drive Operat...	30	Min	"Drive Operator" role
Drive Guest	System-defined role "Drive Guest"	30	Min	"Drive Guest" role
Drive Ext. Role Fieldbus	System-defined role "Drive Ext. Rol...	30	Min	Anyone can change drive data via...
Drive Ext. Role SDI Standard/Adv	System-defined role "Drive Ext. Rol...	30	Min	A user with this role can change d...
HMI Administrator	System-defined role "HMI Adminis...	30	Min	User Management, Monitor, Oper...
HMI Operator	System-defined role "HMI Operator"	30	Min	Monitor, Operate, Remote access ...
HMI Monitor	System-defined role "HMI Monitor"	30	Min	Monitor, Remote access - Monitor ...
HMI Monitor Client	System-defined role "HMI Monitor ...	30	Min	WinCC Unified Client Monitor - limi...
HMI Online Configuration Engineer	System-defined role "HMI Online C...	30	Min	Operate HMI, read and write acce...
NET Administrator	System-defined role "NET Adminis...	30	Min	
NET Standard	System-defined role "NET Standard"	30	Min	
NET Diagnose	System-defined role "NET Diagnos...	30	Min	
Administrator	User-defined role	30	Min	Level 1 + Level 2 + Level 3
Maintenance	User-defined role	30	Min	Level 2 + Level 3

Engineering rights | Runtime rights | User-specific runtime rights

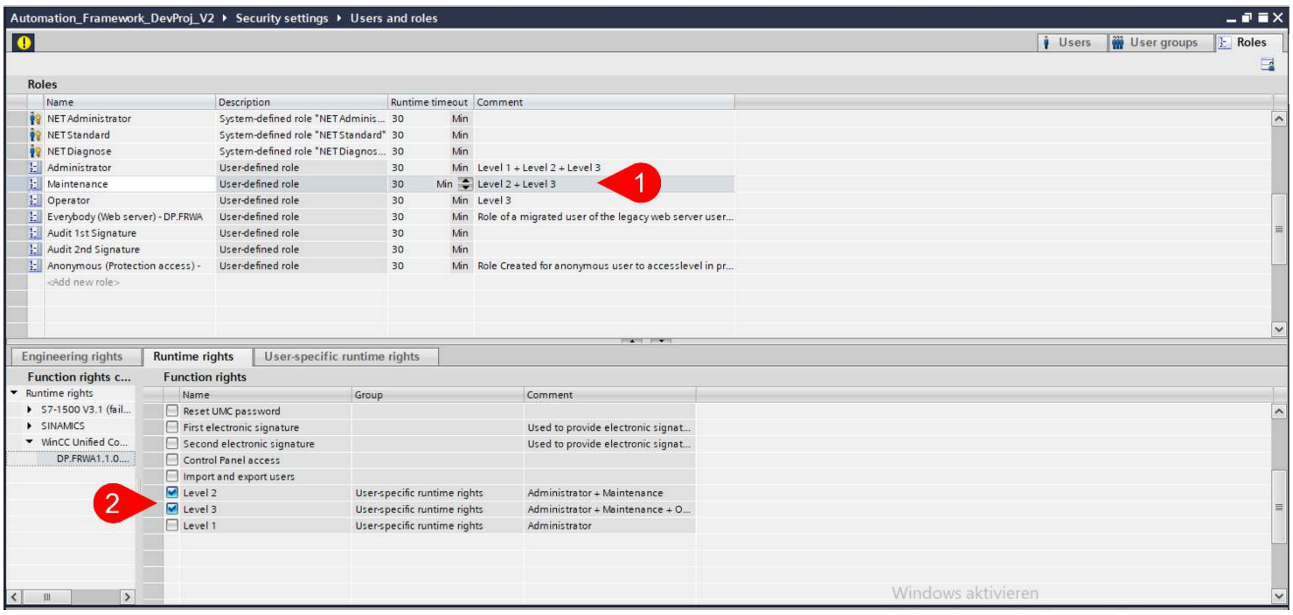
Function rights

Name	Group	Comment
*	*	*
Level 1	User-specific runtime rights	Administrator
Level 2	User-specific runtime rights	Administrator + Maintenance
Level 3	User-specific runtime rights	Administrator + Maintenance + O...
<Add new>		

Figure 15-3: 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) or create a new one. 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).



Roles

Name	Description	Runtime timeout		Comment
NET Administrator	System-defined role "NET Adminis...	30	Min	
NET Standard	System-defined role "NET Standard"	30	Min	
NET Diagnose	System-defined role "NET Diagnos...	30	Min	
Administrator	User-defined role	30	Min	Level 1 + Level 2 + Level 3
Maintenance	User-defined role	30	Min	Level 2 + Level 3
Operator	User-defined role	30	Min	Level 3
Everybody (Web server) - DP.FRWA	User-defined role	30	Min	Role of a migrated user of the legacy web server user...
Audit 1st Signature	User-defined role	30	Min	
Audit 2nd Signature	User-defined role	30	Min	
Anonymous (Protection access) -	User-defined role	30	Min	Role Created for anonymous user to accesslevel in pr...
<Add new role>				

Engineering rights | Runtime rights | User-specific runtime rights

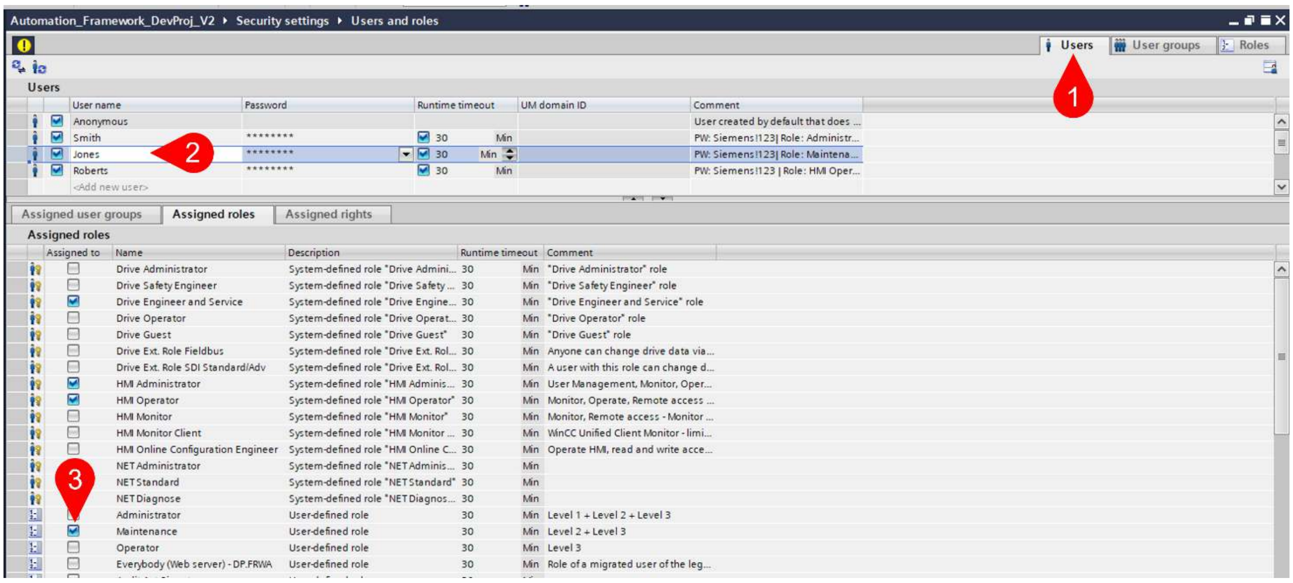
Function rights

	Name	Group	Comment
☐	Reset UMC password		
☐	First electronic signature		Used to provide electronic signat...
☐	Second electronic signature		Used to provide electronic signat...
☐	Control Panel access		
☐	Import and export users		
☑	Level 2	User-specific runtime rights	Administrator + Maintenance
☑	Level 3	User-specific runtime rights	Administrator + Maintenance + O...
☐	Level 1	User-specific runtime rights	Administrator

Figure 15-4: 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).



Users

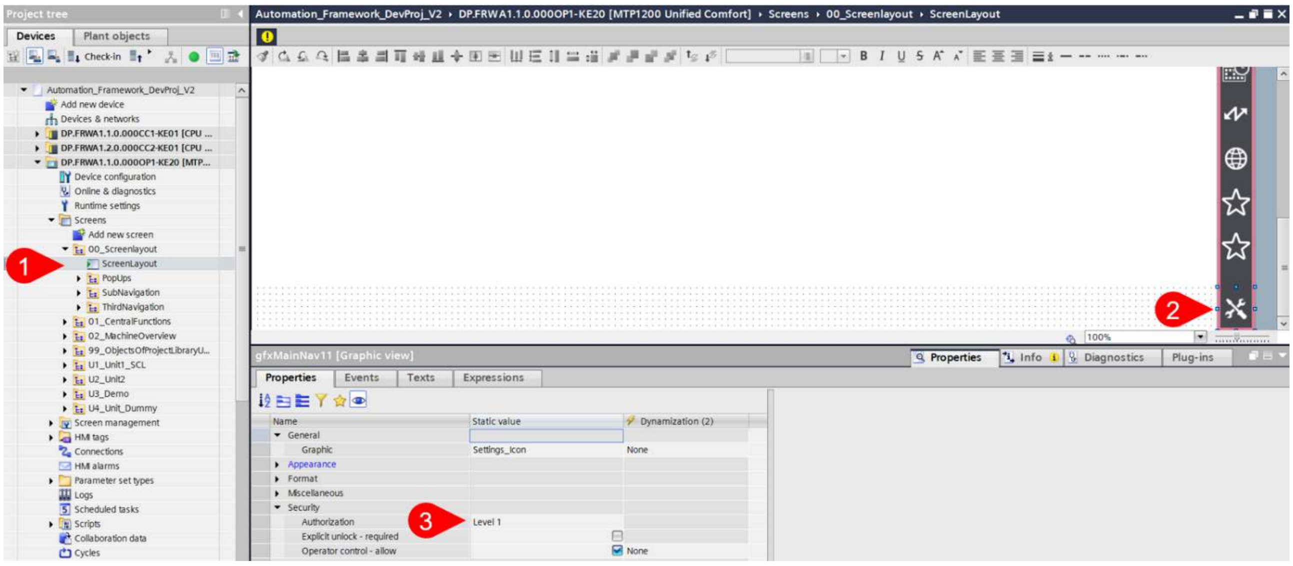
	User name	Password	Runtime timeout		UM domain ID	Comment
✓	Anonymous					User created by default that does ...
✓	Smith	********	✓ 30	Min		PW: Siemens!123| Role: Administr...
✓	Jones	********	✓ 30	Min		PW: Siemens!123| Role: Maintena...
✓	Roberts	********	✓ 30	Min		PW: Siemens!123 | Role: HMI Oper...
	<Add new user>					

Assigned roles

Assigned to	Name	Description	Runtime timeout		Comment
☐	Drive Administrator	System-defined role "Drive Admini...	30	Min	"Drive Administrator" role
☐	Drive Safety Engineer	System-defined role "Drive Safety ...	30	Min	"Drive Safety Engineer" role
✓	Drive Engineer and Service	System-defined role "Drive Engine...	30	Min	"Drive Engineer and Service" role
☐	Drive Operator	System-defined role "Drive Operat...	30	Min	"Drive Operator" role
☐	Drive Guest	System-defined role "Drive Guest"	30	Min	"Drive Guest" role
☐	Drive Ext. Role Fieldbus	System-defined role "Drive Ext. Rol...	30	Min	Anyone can change drive data via...
☐	Drive Ext. Role SDI Standard/Adv	System-defined role "Drive Ext. Rol...	30	Min	A user with this role can change d...
✓	HMI Administrator	System-defined role "HMI Adminis...	30	Min	User Management, Monitor, Oper...
✓	HMI Operator	System-defined role "HMI Operator"	30	Min	Monitor, Operate, Remote access ...
☐	HMI Monitor	System-defined role "HMI Monitor"	30	Min	Monitor, Remote access - Monitor ...
☐	HMI Monitor Client	System-defined role "HMI Monitor ...	30	Min	WinCC Unified Client Monitor - limi...
☐	HMI Online Configuration Engineer	System-defined role "HMI Online C...	30	Min	Operate HMI, read and write acce...
	NET Administrator	System-defined role "NET Adminis...	30	Min	
	NET Standard	System-defined role "NET Standard"	30	Min	
	NET Diagnose	System-defined role "NET Diagnos...	30	Min	
☐	Administrator	User-defined role	30	Min	Level 1 + Level 2 + Level 3
✓	Maintenance	User-defined role	30	Min	Level 2 + Level 3
☐	Operator	User-defined role	30	Min	Level 3
☐	Everybody (Web server) - DP.FRWA	User-defined role	30	Min	Role of a migrated user of the leg...

Figure 15-5: 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 AFBasic 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 AFBasic HMI.



Name	Static value	Dynamization (2)
▾ General		
Graphic	Settings_Icon	None
▸ Appearance		
▸ Format		
▸ Miscellaneous		
▾ Security		
Authorization	Level 1	
Explicit unlock - required	☐	
Operator control - allow	✓	None

Figure 15-6: 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																	

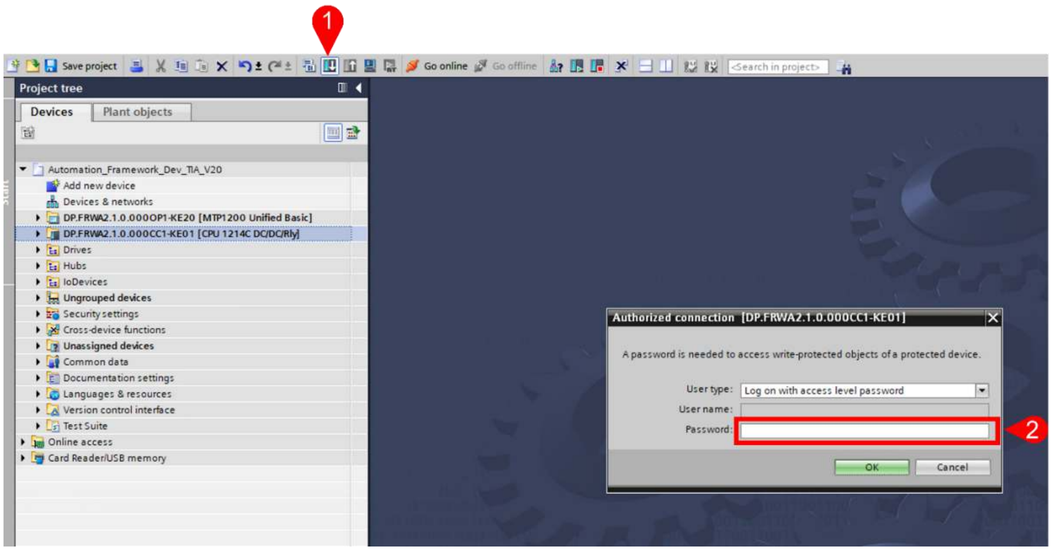
Table 15-1: Runtime and user specific rights

16. Run the demo

The AFBasic 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 AFBasic 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 AFBasic 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.



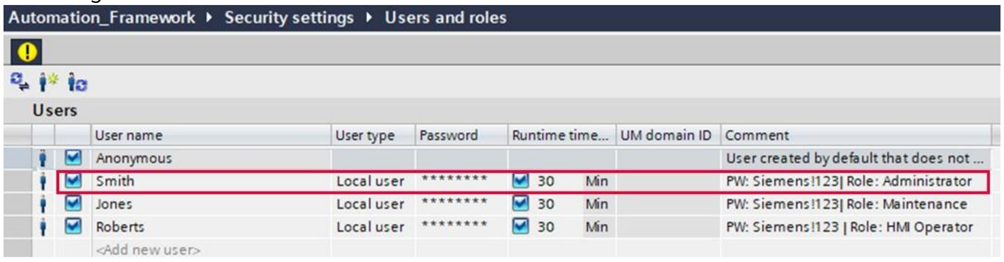
Authorized connection [DP.FRWA2.1.0.000CC1-KE01]

A password is needed to access write-protected objects of a protected device.

User type: Log on with access level password
User name:
Password:

Figure 16-1: Download of the compiled program in the PLC

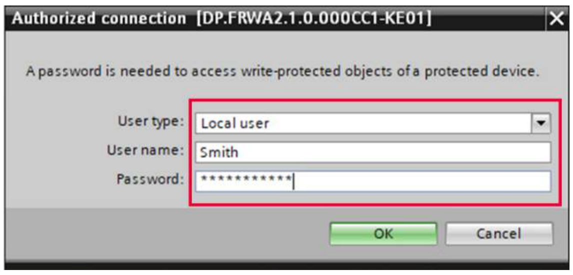
The user type to choose is "Project user" (2). Then the "User name" and "Password" field must be fulfilled. The information of this credentials can be found within the Security settings in the "User and roles" menu (See Chapter 15 User Management). For demo purposes, the user must have an administrator role. In this case the login data will be the following:


Automation_Framework ▸ Security settings ▸ Users and roles

Users

	User name	User type	Password	Runtime time...		UM domain ID	Comment
✓	Anonymous						User created by default that does not ...
✓	Smith	Local user	********	✓ 30	Min		PW: Siemens!123| Role: Administrator
✓	Jones	Local user	********	✓ 30	Min		PW: Siemens!123| Role: Maintenance
✓	Roberts	Local user	********	✓ 30	Min		PW: Siemens!123 | Role: HMI Operator
	<Add new user>						

Figure 16-2: Login data of user with administrator role


Authorized connection [DP.FRWA2.1.0.000CC1-KE01]

A password is needed to access write-protected objects of a protected device.

User type: Local user
User name: Smith
Password: ***********

Figure 16-3: 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.

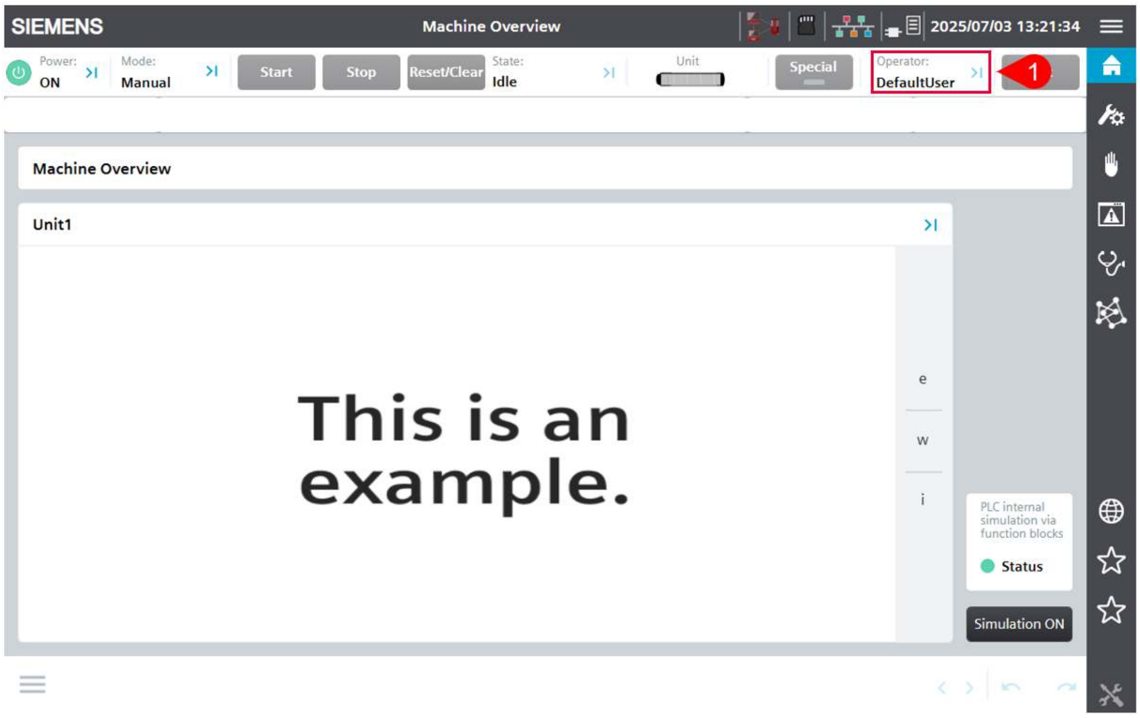


Figure 16-4: Non operable HMI. The user has to login with administrator credentials

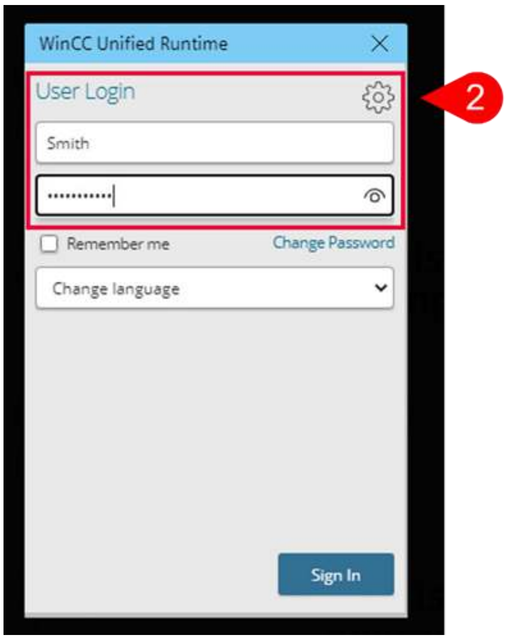


Figure 16-5: Login in the HMI with administrator credentials

17. Additional information

17.1. 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 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

17.2. 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

17.3. 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 can be used in the field for operating and maintaining the installed devices.

https://support.industry.siemens.com/cs/ww/en/view/109816217

Siemens OPC UA Modeling Editor (SiOME)

With the free "Siemens OPC UA Modellingo Editor" (SiOME) tool, we have created an editor for defining your own OPC UA Information Models or mapping existing companion specificati ns 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

SIMATIC Controller Profiling

With SIMATIC Controller Profiling, you can analyze and evaluate the runtime behavior of your user program on a SIMATIC S7-1500 controller from firmware 3.1 in depth and S7-1200 G2 controller. All relevant information can be graphically displayed and evaluated using Google Chrome.

https://support.industry.siemens.com/cs/document/109750245

18. Appendix

18.1. 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.

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: siemens.com/SupportRequest

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 "Siemens Industry Online Support" app. The app is available for iOS and Android:

18.2. Links to 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/109987391
\4\	TIA Portal Add-In Code2Docu for generating documentation https://support.industry.siemens.com/cs/ww/en/view/109809007
\5\	Siemens Information Model Kit for manufacturing use cases https://support.industry.siemens.com/cs/ww/en/view/109814835

Table 18-1: Links and literature

18.3. Change documentation

Version	Date	Modification
2.0.0	2025/04/07	First version
2.1.0	2025/05/14	Update: • Improved performance : Reduces runtime of the following blocks per PLC cycle by optimizing code and distributing load to different subsequent PLC cycles. Additionally, improved behavior, interface and memory consumption. - FC LAF_PlcDiag - FC LAF_Checksum - FB LAF_MemCardDiag • Improved HMI resource consumption: Reduced amount of used HMI ressources by optimizing tag connection for the following screens : - Settings - SettingsChecksum - SettingsDateTime - SettingsIM
2.1.0		No official version
2.1.1	2025/05/20	Update: • Fixed bug in LAF_fpDirectEmControl.

2.2.0	2025/6/30	• Added symbols and single letters texts to all configured project languages • Improved Documentation and added chapter for Manual Operation • Updated the Sub and Third Navigation structure • Added the chapter 7.3.7 for the new scripts for the navigation • Updated the chapter about how to remove a unit on the HMI • Added the chapter about how to add a new unit on the HMI • Updated the chapter about how to resize the HMI • Added the chapter 7.6 for the style setup • Renamed FB "CallGlobal" to FB "CallGeneral" • Improved read local PLC time handling - Changed type of Global.Data.general.localPlcDateAndTime from LAF_typeGlobalDateTime to DTL to make it compatible with S7-1200 G2. (G2 doesn't support data type DT). - In FB CallGeneral, substituded FB LAF_ReadPlcDateAndTime with the system function RD_LOC_T, as LAF_ReadPlcDateAndTime is not needed any more. - Changed Global.Settings.setLocalPlcTime.newDateAndTime from DT to DTL to be compatible with S7-1200 G2. - In FB CallSettings, substituded FB LAF_SetPlcDateAndTime with the system function WR_LOC_T, as LAF_SetPlcDateAndTime is not needed any more. - Added a FC "LAF_ConvertDTL" to be called before the relevant SICAR blocks to provide the needed data type "LSicar_typeGlobalDateTime". • Text list improvement : Corrected the numbering of the entries in the follwoing text lists : - PLC_Checksum_PD - PLC_ControlOnOff_PD - SMC_Diag_AlarmWarnInfo_PD • Information model kit types renamed: - From LAF_typeMachineInformation to LAF_typeInformation - From_typeMachineStatus to LAF_typePerformanceIndicators • Update LBC Library **LAF Library** **NEW:** LAF_LinkUnitStatusToSequence / V2.2.0 - First released LAF_ManCmMultiplexing / V2.2.0 - First released version LAF_ReadBoolSignals / V2.1.0 - First released version LAF_ReadWordSignals / V2.1.0 - First released version **UPDATE:** LAF_Checksum / V2.2.1 - Update header info - Reduced runtime of the block per PLC cycle by distributing the work load to multiple subsequent PLC cycles. - Included Alarms, per default ProDiag. If no ProDiag is configured, alarms could be published via Program Alarms by enabling the respective input of the block. - Update header info LAF_ConvertDTL / V2.2.0 - Interface types updated, no functional changes. LAF_ClearAlarms / V2.2.0 - Header information updated and removed commandsToUnit - Alarms array data type updated from Bool _LAF_typeEvents_ type. - Interface data type _LAF_typesEvents_ updated. LAF_ClockGenerator / V2.0.0 - Block information updated

LAF_CmDummyCn / V2.2.0
- Update block info
- Improved control module behaviour: Added command2, added error when command1 and command2 are set the same time, added

LAF_ControlEm / V2.2.0
- Creation of inputs to enable HMI and local interfaces to set mode and state, _enModeStateCtrlByHmi_ and _enModeStateCtrlByLocal_. Fixed bug in assigning the _unitStatus_ to the local interface.
- Updated interface type _LUC_typeCommands_,_LAF_typeEmControlNode_ and _LAF_typeEmInterface_. Implemented special functions evaluation from local and HMI interfaces. Updated evaluation of the enable status of the EM, now is always enabled independant if the EM is linked or unlinked. Rearranged the order of _hmiInterface_ and _localInterface_ interfaces.

LAF_ControlLadSequence / V2.2.0
- Updated block info
- Updated header info
- Improved code for the initialization logic. Removed ImmediateStop as implicit functionality and realized that functionality as a normal job. Added rising edge detection in the single mode step. Improved general code and simplified interfaces.

LAF_ControlSclSequence / V2.2.0
- Updated header info
- Improved code for the initialization logic. Removed ImmediateStop

LAF_HmiHeaderMultiUnitManager / V2.0.0
- Block information properties updated
- Updated unit data information from _LAF_typeUnitInterface_ to _LUC_typeControlInterfaceCompact_.

LAF_HmiHeaderSingleUnitManager / V2.2.0
- Implemented special functions managment between PLC and HMI. Updated unit data from _LUC_typeMonitoring_ to _LUC_typeControlInterfaceCompact_. Interface variables _modeRequestButtons_ and _stateCommandButtons_ replaced with _hmiInterface_.
- Interface types updated, no functional changes.

LAF_ManEmConfig / V2.2.0
- Corrected Predefined parameter for "screenData". Changed max areas number const from 7 to 6.
- Updated block with call to Multiplexing FC and new adapted UDTs

LAF_ManOperationCm / V2.2.0
- Updated header
- Updated UDT for ManOperation

LAF_ManualOperationFaceplate / V2.2.0
- Updated header
- Adapted UDT for manOperation

LAF_ManualOperationInternal / V2.2.0
- Updated header. Corrected Description
- Adapted new UDT for ManOperation

LAF_ManualOperationMain / V2.2.0
- Updated header; Corrected Predefined parameter for "screenData"
- Adapted new UDT version for manOperation

LAF_MemCardDiag / V2.2.1
- Update block info
- Reduced runtime of the block per PLC cycle by distributing the work load to multiple subsequent PLC cycles.
- Included Alarms, per default ProDiag. If no ProDiag is configured, alarms could be published via Program Alarms by enabling the respective input of the block.
- Update header info

LAF_PowerOnOff / V2.0.0
- Update block call

Version	Date	Modification
		LAF_ReadCycleLimits / V2.0.0 - Update block info LAF_SelectJob / V2.2.1 - Locked Output behaviour changed - Update block info - Improved the code and the behaviour. Removed the Output _jobStatus_ and InOut _jobs_ and moved all relevant variables in the InOut _sequence_. The connection between the FB LAF_SelectJob and the Sequence is realized by the InOut _sequence_. - In the code, replaced _sequence.status.manualControlAllowed_ by _sequence.enableManualCtrl_. _sequence.status.manualControlAllowed_ is set by the FB LAF_Status. Sequence and therefore the value change is seen by LAF_SelectJob one cycle after the value change of _sequence.enableManualCtrl_. That caused unwanted LAF_SetAxisSimulation / V2.2.0 - Block information updated - Simulation Trigger logic improved LAF_SetDriveObjectConfig / V2.0.0 - Block information updated LAF_StatusLadSequence / V2.2.0 - Updated block info - Updated header info - Improved general code and simplified interfaces. LAF_StatusSclSequence / V2.2.0 - Updated block info - Updated header info - Improved general code and simplified interfaces. LAF_SummarizeCommandsToUnit / V2.2.0 - Unification of prodiag and bit alarms. Extention of commands sent to the unit. Implementation of subcategories evaluation in bit alarms. - Extention of subcategories according to the commands sent to the unit. There is a subcategory defined for each posible commnd that can be sent to the unit. LAF_UpdateJobHistory / V2.2.0 - Header information updated - Changed history InOut to a [*] to enabled flexible length LAF_UpdateStepHistory / V2.2.0 - Header information updated - Changed history InOut to a [*] to enabled flexible length **DELETED**: LAF_HmiHeaderManager / V1.1.0 LAF_ManageAlarms / V1.1.0 LAF_MapUnitStatusToSequence / V1.2.0 LAF_ReadPlcDateAndTime / V1.1.0 LAF_SetPlcDateAndTime / V1.1.0
2.2.1	2025/07/11	**Bugfixes** • HMI Third Navigation’s size bug fixed. • LAF_CopyJsonFiles bug fixed and updated to V 2.1.1

Version	Date	Modification
2.2.2	2025/09/30	**Bugfixes** LAF_SelectJob / V2.2.2 - Output "#startPending" (in < V2.2.2 = Output "#readyToStart") now gets updated correctly LAF_fpDirectEmControl / V2.2.2 - The event on the buttons write into the “commands” structure now so we are not bypassing the block via the “status” structure. CallEquipment U3EM4 - In some networks GlobalDBs where accessed ddirectly in the code, instead of accessing the GlobalDBs from the interface. LUC_typeHeaderConfig / V2.2 - Fixed typo in variable names. ManualOperationLines Screen - Fixed SubNavigation bug so all manual operation lines appear when changing from one equipment module to the other through subNavigation. **Updates** LAF_typeJobStatus / V2.2.2 - Changed variable "#readyToStart" to "#startPending" for better understanding LAF_SelectJob / V2.2.2 - Changed output "#readyToStart" to "#startPending" for better understanding - It is now possible to restart a singleCyleJob with a new rising edge at input "#startCondition". So far, another job had to be called first before a singleCycleJob could be restarted again. LUC_StatusCollector / V2.2.0 - All commands are now cleared during the first PLC cycle and collected during the second. This ensures that only commands from fully executed units and equipment modules are captured, providing up-to-date data for processing. LAF_typeInterfaceSignalsHmi / V2.1.1 - Changed default values for “screenActive” as TRUE and “selectedPage” as 1. **NEW** LAF_fpUnitSelection / V2.2.0 - Added the faceplate for the Unit selection in a popup: “Select Unit” for better usability. Navigation.UnitSelected Script - Created a script to show in the header the selected units from a SelectUnit popup, that contains the new LAF_fpUnitSelection faceplate.

Table 18-2: Changelog