SIEMENS

Machine Template Documentation

SIMATIC S7 / WinCC Unified / TIA PORTAL

https://support.industry.siemens.com/cs/ww/en/view/109817223

Siemens Industry Online Support

Legal information

Use of application examples

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

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

Disclaimer of liability

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

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

Other information

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

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

Security information

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

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

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

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

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

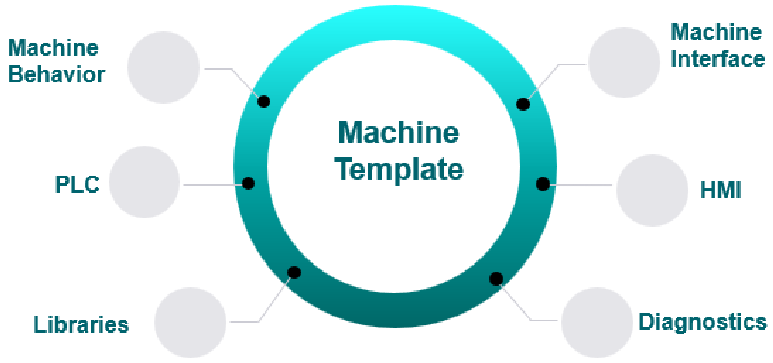
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under https://www.siemens.com/cert.

Table of contents

1 Introduction

1.1 Overview

Figure 1-1: Machine Template Overview



The Machine Template is a TIA Portal project for machine builders that helps to implement the automation program for different machines. The Machine Template provides the basic structure of a PLC and HMI program and a standardized interface to control and monitor the machine status from higher level IT systems.

This document describes the content of the Machine Template and explains how the Machine Template can be used as a basis for implementing an own automation program.

There are two main scenarios for using this Machine Template.

- The machine is already implemented, but to meet the end customer requirements, it is necessary to provide a standardized interface to connect the machine to end customer IT system.
- Start from scratch to implement a new machine based on this template and the standardization concept, including HMI and libraries.

Key features

The Machine Template contains the following features:

- PLC and HMI based on a modular structure
- Machine behavior (Example implementation of a mode state manager based on LPMLV Library)
- Machine interface
- Simple integration of existing standardized libraries
- Standardized Communication based on OPC UA Server Interface
- Diagnostics with ProDiag
- Integrated HMI template
- Template for equipment module including process sequences
- Template for control module for sensors and actors

Goals

- Reach a holistic architecture for successful combination of technologies
- Enable easy integration of Siemens offerings like libraries, templates etc.

Benefits for the machine builder

- Fast implementation of different machines using the Machine Template
- Reduced effort due to reusability of the modular concept
- Added value through the flexibility to manage further changes like individual customer requirements

NOTE

The Machine Template is a TIA Portal project that can be used as template to implement the automation program for different machines. It does not claim to be comprehensive and will be further extended with each version.

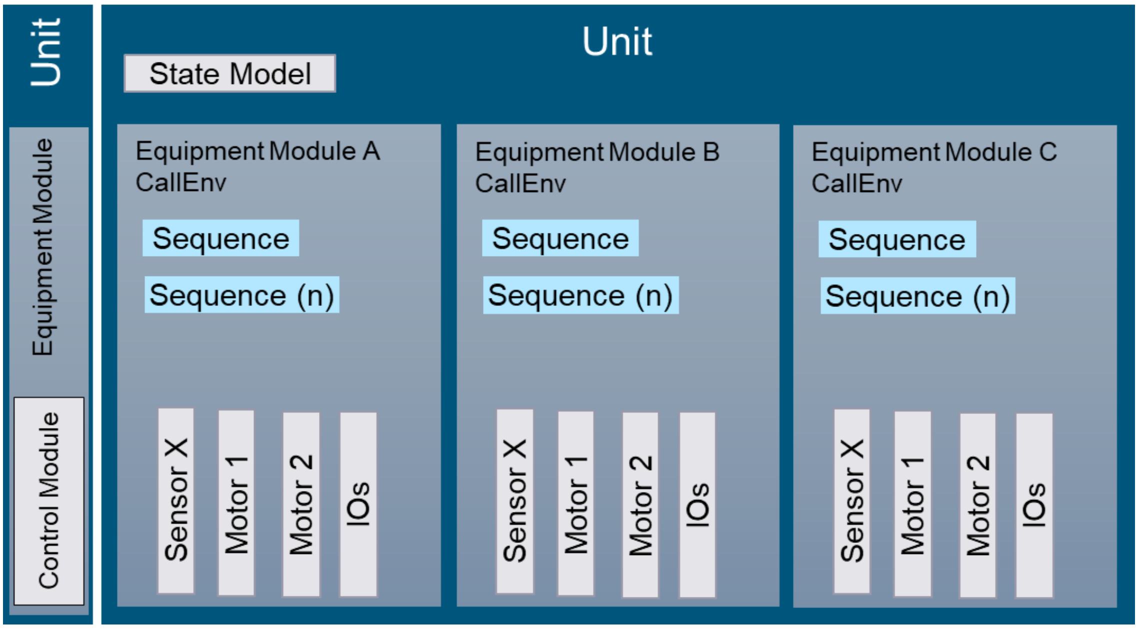
1.2 Modular Software Design based on ISA-88

Concept

The ISA-88 model is an important foundation for the modular programming structure of production machines. The Machine Template based on ISA-88 covers the physical model from the unit level to the control module level, as they are the essential components of a PLC controlled production machine.

The figure below shows the hierarchical structure of the levels as an implementation example for a better understanding of the modular project architecture.

Figure 1-2: Hierarchical software structure



Unit

- A unit is known as a machine
- A unit is a collection of equipment modules and control modules
- A unit is usually centred on one PLC and provides status and process related data for the higher-level system from a production/operational perspective
- The complete work order is often included in the unit
- A unit has a state model (in this document based on the Library of Unit Control)

Equipment Module

- An equipment module is a collection of control modules, assemblies, or other equipment modules.
- It can carry out a finite number of processing activities.
- An equipment module contains all the necessary modules to carry out these processing activities.
- For example, an equipment module might consist of, e.g., a circulation pump, a chilled water valve, a steam valve, and a temperature controller. In this case, the equipment module would represent a temperature control system.

Control Module

- A control module is often a equivalent to an actuator / sensor.
- A control module could also be collection of sensors, actuators, other control modules, or associated processing equipment.
- It acts as a single entity from a control standpoint.
- It is the direct connection to the process through its sensor and actuators.
- Examples of control modules: valve, pump, motor, pressure controller, limit switch, etc.
- A control module (CM) in the PLC program is usually represented by a FB that comes from a provided standard library. If no library is available for the dedicated control functionality, the control module must be developed by the machine builder according to the control module standards.
  Siemens offers the Library of Basic Control (LBC) to realize control modules.

NOTE

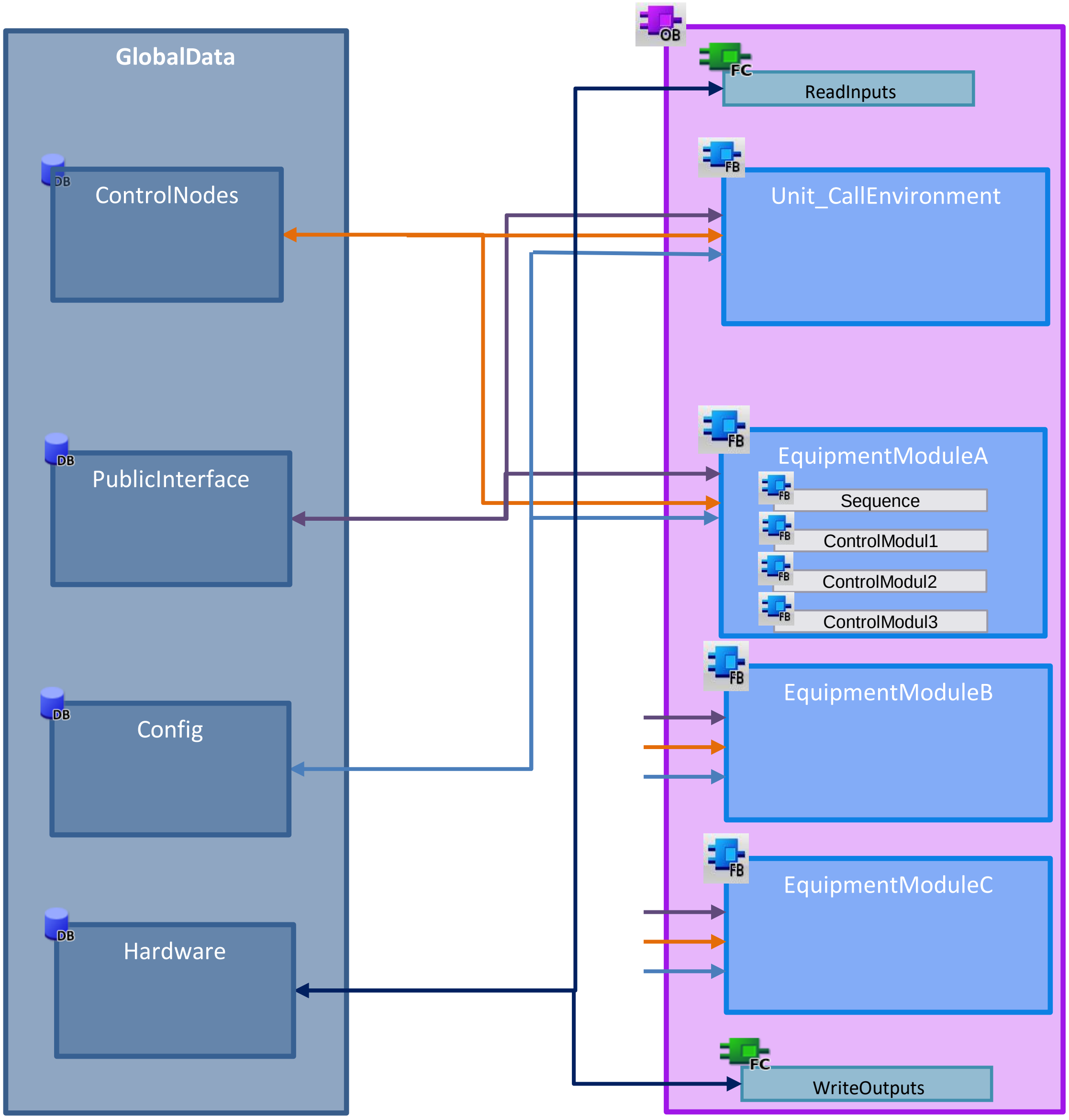
More information about Library of Basic Controls (LBC) can be found under the link:
https://support.industry.siemens.com/cs/ww/en/view/109792175

1.3 Machine Template Architecture

The data flow in the Machine Template is shown in the following figure.

One advantage of the Machine Template is that the user program of the unit is separated from the equipment modules. With that, an easy integration of additional equipment modules is ensured.

Figure 1-3: Machine Template architecture



1.4 Machine Template Approach

The Machine Template offers a standardized solution approach that can be customized and implemented in any machine.

1.4.1 Machine Behavior

The Machine Template can be used with any state model to control the machine.

Exemplarily in this version, a mode and state manager based on OMAC PackML (LPMLV2021) is used.

The functionality of the OMAC state and mode manager is extended by the Library of Unit Control (LUC).

Figure 1-4: Overall scheme

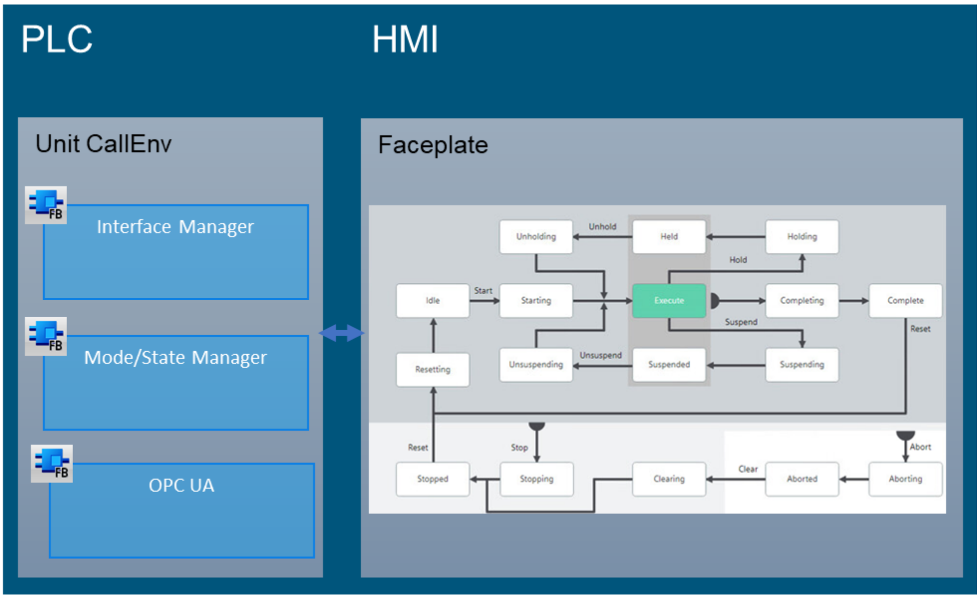
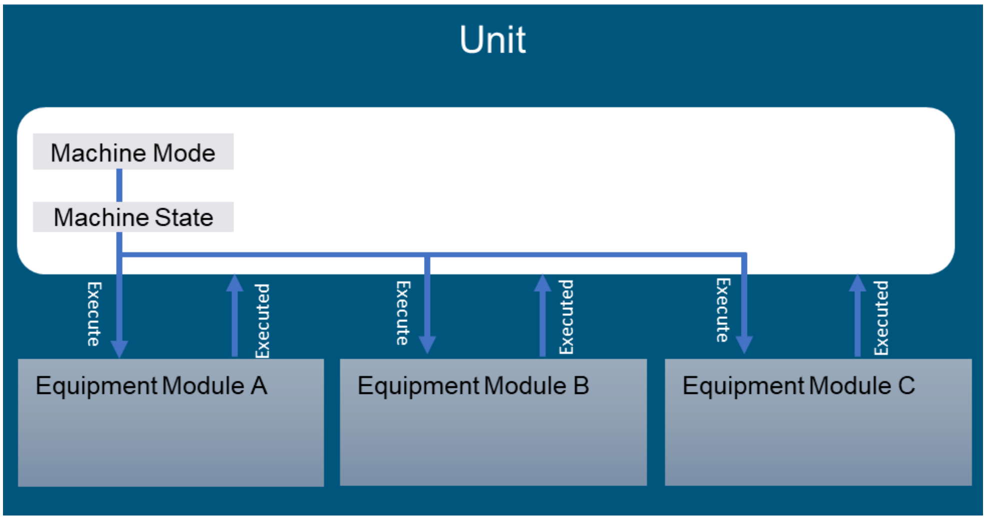


Figure 1-5: Unit structure



State Machine

A state machine is a control model that is defined by fixed operating states with defined requirements needed to transition from one state to another. The machine's operation can be summarized by providing the current state. State machines help operators to understand what the machine is doing and what is needed to move into the next state.

NOTE For more information about the state machine model which is used in this template, refer to:

https://www.omac.org/packml

https://support.industry.siemens.com/cs/ww/en/view/49970441

Machine Mode

A mode is on a higher logical level than a state and contains a set of predefined states. It defines the machine operating mode The Machine Template provides the modes Production, Maintenance, Manual and up to 5 user-defined modes.

The following are examples of machine modes specified:

- Production:
  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:
  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 (mandatory):
  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.
- Users can add additional user modes in the template. Examples may include "Cleanout", "Product Change", etc.

1.4.2 Stack Light

Each machine typically has a stack light that shows the machine status and indicates specific requests to the operator. The Machine Template contains an example implementation consisting of 5 standardized light colors that can be illuminated in a specific sequence. Additionally, two user defined colors are reserved for the extension. Moreover, a horn is also defined with simple on and off signal.

1.4.3 Machine Interface

End users usually run production lines and process plants that consist of individual components like single machines and process equipment. All those components might be manufactured and programmed by different machine builders. This can easily result in a lack of software consistency between machines, difficult and time consuming horizontal / vertical integration as well as challenging troubleshooting and maintenance.

To avoid that, the provided machine interface includes a uniform set of data structure used in the base state model. The aim is therefore to make sure that machines deliver homogeneous data to higher level information systems and enable easier line integration and serviceability.

The machine interface contains four groups:

- Machine control, which enables the change of mode and state for the state model.
- Machine information, which contains the basic description of the machine.
- Machine status, which indicates the current situation of the machine.
- Machine monitoring, which supervises the specific status or process value of modules or components in the machine.

This interface is defined and configured as an OPC UA Companion Specification with the Siemens OPC UA Editor (SiOME). With help of SiOME, the machine interface through OPC UA communication can be easily modified and extended with the specific requirements.

Further information to SiOME can be found under here:

https://support.industry.siemens.com/cs/ww/en/view/109755133

1.4.4 Communication

The OPC UA functionality of the PLC enables the user to set up additional communication between the PLC and external systems, i.e. IT systems. This could be information like diagnostics or other relevant data. To ensure data consistency, a handshake mechanism must be defined between the PLC and the external system.

1.4.5 Diagnostics

Within the Machine Template, there is a defined diagnostics mechanism, which gives the possibility to upload the diagnostics information to the information system. There are three types of diagnostics information:

- Alarm:
  This indicates an error in the machine, which mostly leads to a machine stop. An error usually must be acknowledged, so that the machine can continue operation.
- Warning:
  This indicates a warning in the machine, which doesn’t lead to a machine stop and can be solved later. This message does not need to be acknowledged.
- Information:
  This shows an information of the machine, which helps to understand what has been done by the machine.

Most information systems cannot get the alarm message from PLC system functions, like Program Alarm and ProDiag, and a separate alarming mechanism including a handshake between the PLC and the information system must be set up. Therefore, the diagnostics information needs to be uniform. Each alarm message to the information system has an ID, category and value to point out the detailed explanation in additional diagnostics table. The handshake for alarm messages will be organized by an alarm manager, that triggers the active alarm and confirms if the alarm is acknowledged by the information system.

1.4.6 HMI Template

The integrated HMI template provides basic usability for controlling the state model, checking the status and showing the diagnostics. The design of the HMI template has been inspired by the HMI Template Suite, which can also be extended with the different operating screens, dialogs and components during the configuration.

The Machine Template contains an HMI template for WinCC Unified Comfort Panel.

NOTE More information about HMI Template Suite can be found under the link: https://new.siemens.com/global/en/products/automation/simatic-hmi/hmi-template-suite.html

1.5 Hardware and software requirements

Requirements for the Machine Template

To be able to use the Machine Template described here, the following hardware and software is required.

Hardware

The following components can be used:

- SIMATIC S7-1200 (F)
- SIMATIC S7-1500 (TF) / Software Controller
- WinCC Runtime Unified or WinCC Unified Comfort Panel
- Simulation with S7-PLCSIM and PLCSIM Advanced

Software

- TIA Portal V18 Update 1
- SIMATIC STEP 7 Professional
- WinCC Runtime Unified V18 Update 1

2 Engineering

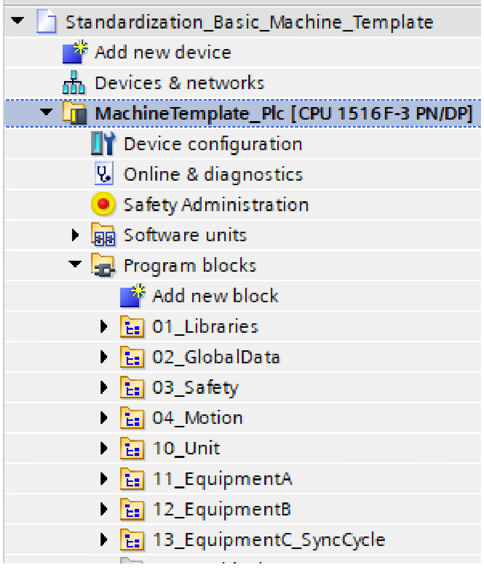
The Machine Template contains two solution approaches: Standard PLC structure and PLC structure based on software units. It provides basic functionalities that are already implemented.

This chapter contains the engineering steps for using the Machine Template.

2.1 Standard PLC Structure (without Software Units)

The PLC program has the following structure.

Figure 2-1: PLC Program structure without software units



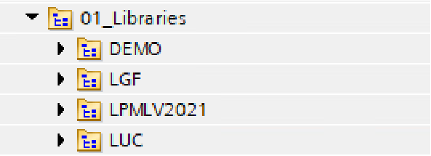
- Standardization_Basic_Machine_Template
  - Add new device
  - Devices & networks
  - MachineTemplate_Plc [CPU 1516F-3 PN/DP]
    - Device configuration
    - Online & diagnostics
    - Safety Administration
    - Software units
    - Program blocks
      - Add new block
      - 01_Libraries
      - 02_GlobalData
      - 03_Safety
      - 04_Motion
      - 10_Unit
      - 11_EquipmentA
      - 12_EquipmentB
      - 13_EquipmentC_SyncCycle

2.1.1 Libraries

This folder contains all libraries that are used in the project. Libraries offer the possibility to instantiate pre-programmed blocks in your project.

In this Machine Template, a variety of libraries are used, like LGF and LUC. More information to the used libraries can be found in the Annex.

Figure 2-2: Content of group 01_Libraries

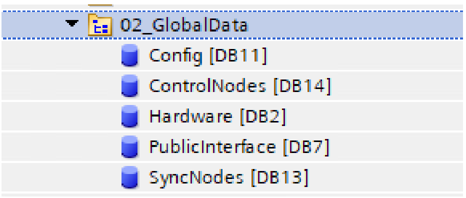


- 01_Libraries
  - DEMO
  - LGF
  - LPMLV2021
  - LUC

2.1.2 Global Data

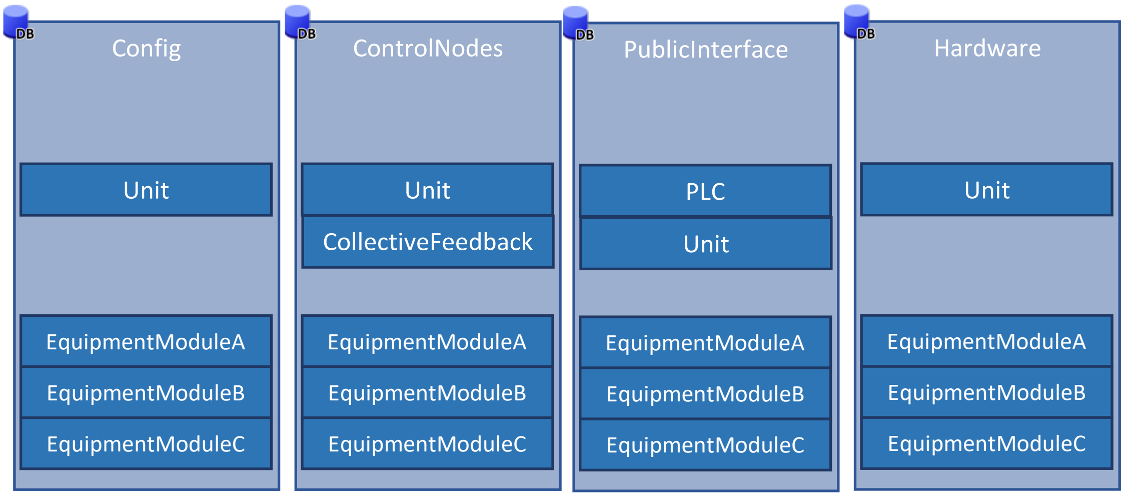
The Machine Template contains a variety of global data blocks for different use cases.

Figure 2-3: Content of group 02_GlobalData



- 02_GlobalData
  - Config [DB11]
  - ControlNodes [DB14]
  - Hardware [DB2]
  - PublicInterface [DB7]
  - SyncNodes [DB13]

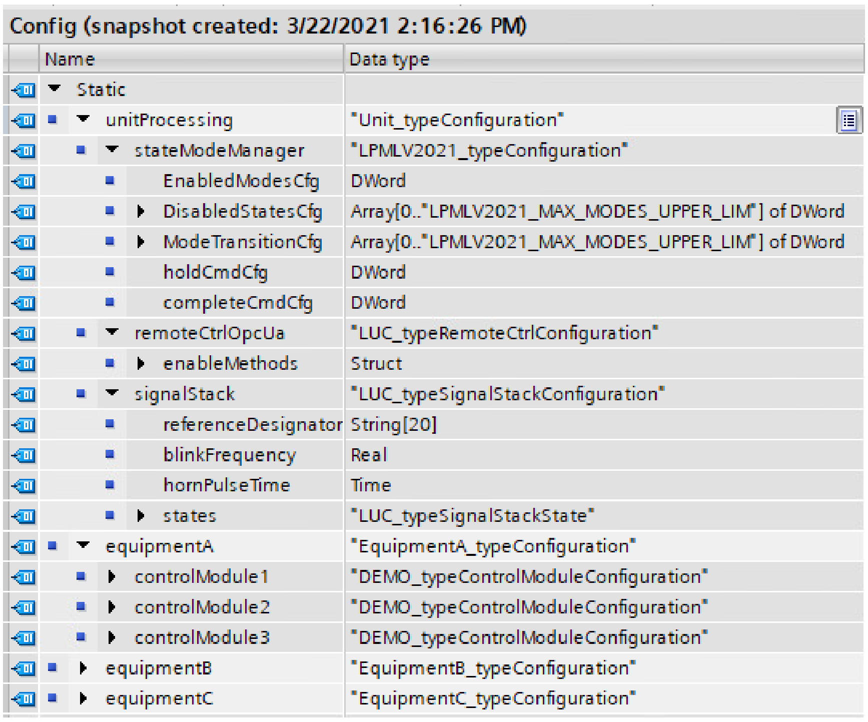
Figure 2-4: Structure of global data blocks



Config DB

The Config DB stores the configuration parameters for the entire machine and enables the user to make backups from the configuration parameters with one single snapshot, e.g., during or after commissioning and the data can be stored retentively. By using this concept, the configuration data can also be easily protected from unwanted access from the program or unauthorized users.

Figure 2-5: Content of Config DB



Config (snapshot created: 3/22/2021 2:16:26 PM)

Name	Data type
Static	
unitProcessing	"Unit_typeConfiguration"
stateModeManager	"LPMLV2021_typeConfiguration"
EnabledModesCfg	DWord
DisabledStatesCfg	Array[0.."LPMLV2021_MAX_MODES_UPPER_LIM"] of DWord
ModeTransitionCfg	Array[0.."LPMLV2021_MAX_MODES_UPPER_LIM"] of DWord
holdCmdCfg	DWord
completeCmdCfg	DWord
remoteCtrlOpcUa	"LUC_typeRemoteCtrlConfiguration"
enableMethods	Struct
signalStack	"LUC_typeSignalStackConfiguration"
referenceDesignator	String[20]
blinkFrequency	Real
hornPulseTime	Time
states	"LUC_typeSignalStackState"
equipmentA	"EquipmentA_typeConfiguration"
controlModule1	"DEMO_typeControlModuleConfiguration"
controlModule2	"DEMO_typeControlModuleConfiguration"
controlModule3	"DEMO_typeControlModuleConfiguration"
equipmentB	"EquipmentB_typeConfiguration"
equipmentC	"EquipmentC_typeConfiguration"

As exemplarily shown in figure 2-5, unitProcessing contains three groups: stateModeManager, remotrCtrlOpcUa and signalStack. These are modified in the libraries (LPMLV2021) and (LUC).

The equipment modules with their associated control modules are also part of the Config DB.

ControlNodes DB

The ControlNodes DB contains process and control information for the entire machine: For the unit, the equipment modules and the control modules.

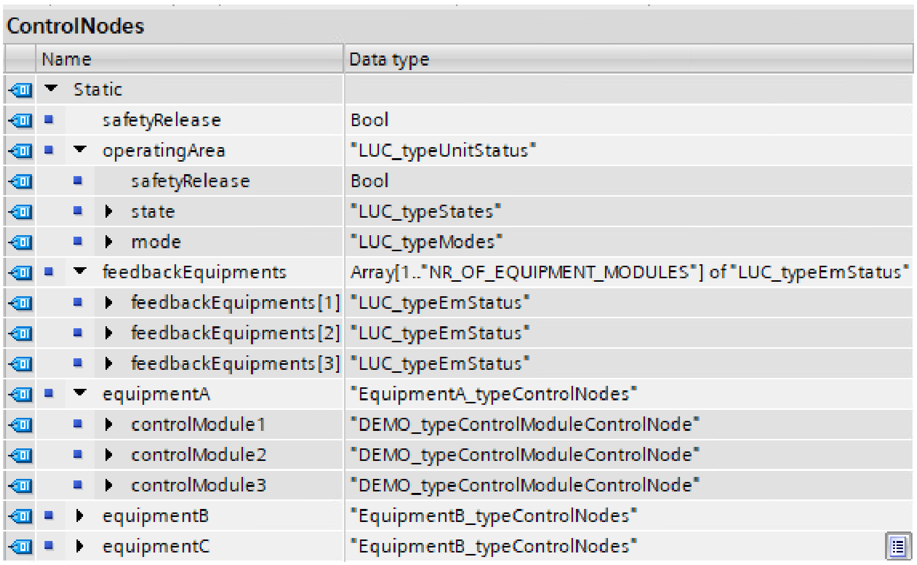
The UnitStatus contains information on the safetyRelease and the current state and mode, which are modified in the LUC.

The status of the equipment modules provides feedback signals from the current states.

The control nodes of the control modules are data structures to control and monitor the control modules. The sequences inside the equipment modules could use these control nodes to coordinate and manage the control modules.

In this template, the control nodes of the control modules are part of the global ControlNodes DB. Alternatively, the control nodes could be placed as static variables insides an FB that calls both the sequence and control modules.

Figure 2-6: Content of ControlNodes DB



ControlNodes

Name	Data type
Static	
safetyRelease	Bool
operatingArea	"LUC_typeUnitStatus"
safetyRelease	Bool
state	"LUC_typeStates"
mode	"LUC_typeModes"
feedbackEquipments	Array[1.."NR_OF_EQUIPMENT_MODULES"] of "LUC_typeEmStatus"
feedbackEquipments[1]	"LUC_typeEmStatus"
feedbackEquipments[2]	"LUC_typeEmStatus"
feedbackEquipments[3]	"LUC_typeEmStatus"
equipmentA	"EquipmentA_typeControlNodes"
controlModule1	"DEMO_typeControlModuleControlNode"
controlModule2	"DEMO_typeControlModuleControlNode"
controlModule3	"DEMO_typeControlModuleControlNode"
equipmentB	"EquipmentB_typeControlNodes"
equipmentC	"EquipmentB_typeControlNodes"

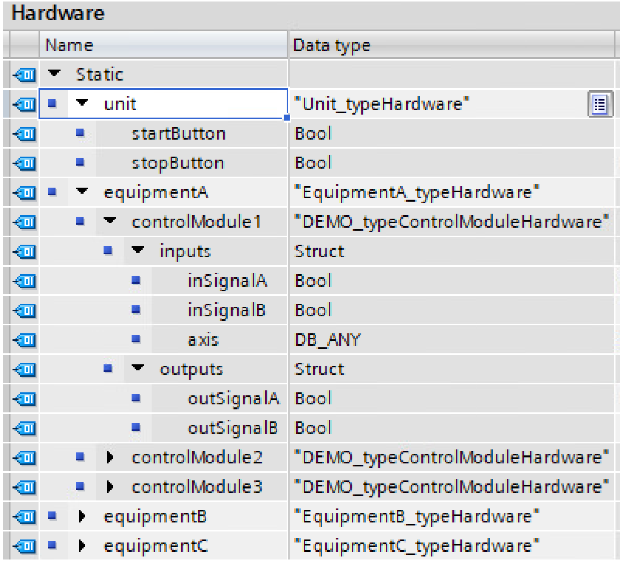
Hardware DB

The Hardware DB is the interface between the process signals from the field and the PLC program. The Hardware DB connects the control modules with the real sensors and actuators, like valves, cylinders, and axis.

To ensure the reusability of the subordinate modules, the periphery signals are connected via the FB interface following the hierarchy structure of the user program.

At the beginning of each PLC cycle, the input tags of the PLC are read and are transferred to the corresponding control module tags in the Hardware DB. At the end of each PLC cycle, the outputs tags from the Hardware DB are copied to the corresponding output tags of the PLC.

Figure 2-7: Content of Hardware DB



Hardware

Name	Data type
Static	
unit	"Unit_typeHardware"
startButton	Bool
stopButton	Bool
equipmentA	"EquipmentA_typeHardware"
controlModule1	"DEMO_typeControlModuleHardware"
inputs	Struct
inSignalA	Bool
inSignalB	Bool
axis	DB_ANY
outputs	Struct
outSignalA	Bool
outSignalB	Bool
controlModule2	"DEMO_typeControlModuleHardware"
controlModule3	"DEMO_typeControlModuleHardware"
equipmentB	"EquipmentB_typeHardware"
equipmentC	"EquipmentC_typeHardware"

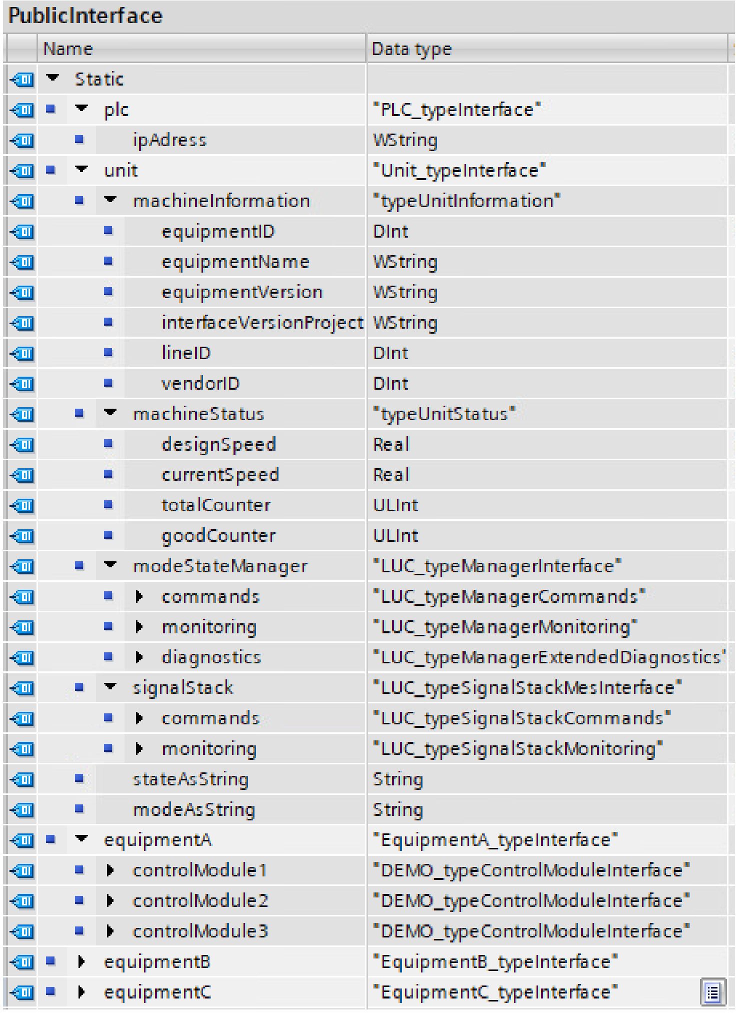
PublicInterface DB

The PublicInterface DB contains all the necessary variables that control the state machine model and indicate the status of the machine and further important components in the machine. The PublicInterface DB is not only relevant for HMI, but more importantly for the information system. The variables can be checked externally through OPC UA communication.

As shown in the following figure, there are three interface structures of the machine.

1. PLC includes the IP address of the PLC.
2. Unit contains the machine information, machine status, mode state manager and signal stack.
3. Equipments contain information of the associated control modules.

Figure 2-8: Content of PublicInterface DB



PublicInterface

Name	Data type
Static	
plc	"PLC_typeInterface"
ipAdress	WString
unit	"Unit_typeInterface"
machineInformation	"typeUnitInformation"
equipmentID	DInt
equipmentName	WString
equipmentVersion	WString
interfaceVersionProject	WString
lineID	DInt
vendorID	DInt
machineStatus	"typeUnitStatus"
designSpeed	Real
currentSpeed	Real
totalCounter	ULInt
goodCounter	ULInt
modeStateManager	"LUC_typeManagerInterface"
commands	"LUC_typeManagerCommands"
monitoring	"LUC_typeManagerMonitoring"
diagnostics	"LUC_typeManagerExtendedDiagnostics'
signalStack	"LUC_typeSignalStackMesInterface"
commands	"LUC_typeSignalStackCommands"
monitoring	"LUC_typeSignalStackMonitoring"
stateAsString	String
modeAsString	String
equipmentA	"EquipmentA_typeInterface"
controlModule1	"DEMO_typeControlModuleInterface"
controlModule2	"DEMO_typeControlModuleInterface"
controlModule3	"DEMO_typeControlModuleInterface"
equipmentB	"EquipmentB_typeInterface"
equipmentC	"EquipmentC_typeInterface"

NOTE

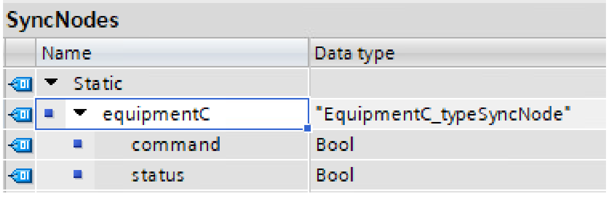
To enable the OPC UA Method for controlling state model, the remote control must be activated.

Set the variable "`modeStateManager\commands\enableRemoteControl`" to TRUE.

SyncNodes DB

The SyncNodes DB is used as an interface between the Motion OBs and provides information to the command and status of the different equipment modules. It allows easy data exchange between the standard cycle and the motion cycle.

Figure 2-9: Content of SyncNodes DB



SyncNodes

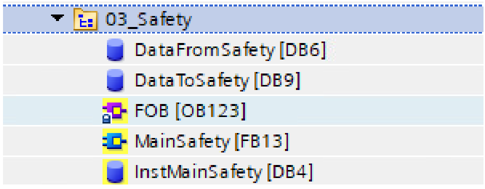
Name	Data type
Static	
equipmentC	"EquipmentC_typeSyncNode"
command	Bool
status	Bool

2.1.3 Safety

The Machine Template contains a basic concept for a modular safety program structure with minimized complexity.

Global standard data blocks are used for data exchange between the standard user program and the safety program as well as F-suitable PLC data types.

Figure 2-10: Content of the group 03_Safety



03_Safety
DataFromSafety [DB6]
DataToSafety [DB9]
FOB [OB123]
MainSafety [FB13]
InstMainSafety [DB4]

As a user of the Machine Template, you must follow the specific rules:

- Protect the F-PLC from unauthorized access
- Activate the change history
- Activate consistent upload

NOTE

For more information about the requirements of the Siemens Safety programming guideline, refer to: https://support.industry.siemens.com/cs/ww/en/view/109750255

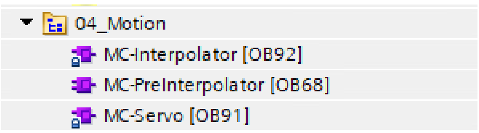
2.1.4 Motion

Task handling in different OB layers helps to manage the process and facilitate the execution of motion tasks. For example, there are processes that must be executed in a very fast clock rate, because of needed reaction times in the kinematic application.

In the Machine Template program, a demo equipment module is included which describes how to integrate motion control into the user program.

The motion folder contains OBs for motion related program logic, e.g., high-performance tasks.

Figure 2-11: Content of the group 04_Motion



- 04_Motion
  - MC-Interpolator [OB92]
  - MC-PreInterpolator [OB68]
  - MC-Servo [OB91]

MC-Interpolator/ MC-Servo

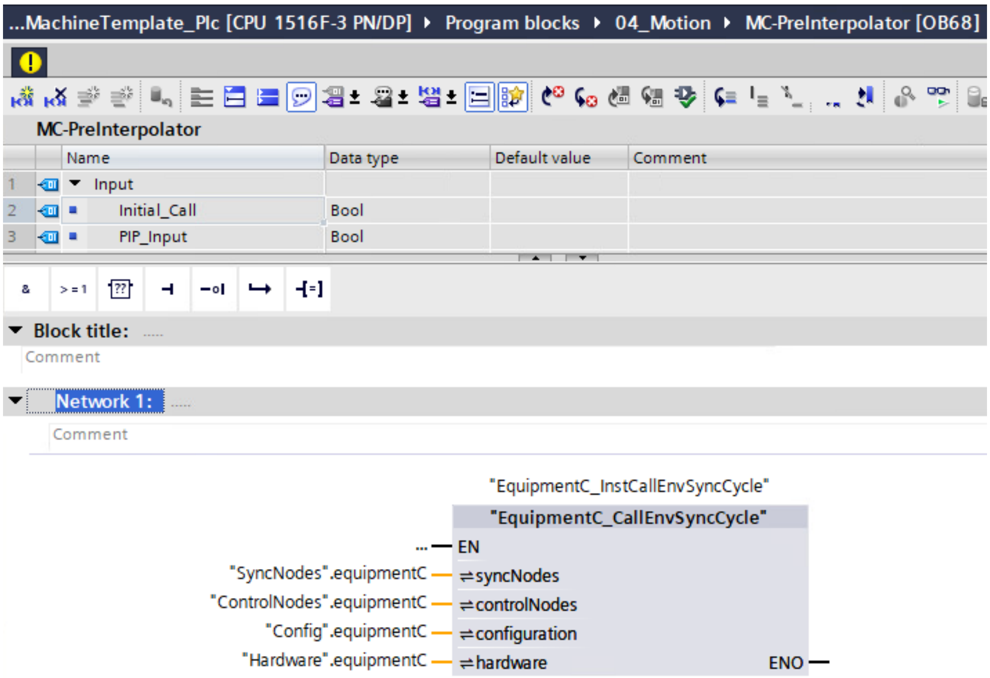
When the first technology object (TO) is added to the S7-1500 PLC, the MC-Interpolator and MC-Servo- OB for processing the TOs are created automatically. The functionality of the TOs creates its own execution level according to the application cycle. These OBs are knowhow-protected, and the content cannot be changed.

MC-PreInterpolator

To call an equipment module that requires a deterministic motion cycle, MC-PreInterpolator or MC-PostInterpolator can be used.

In the Machine Template the MC-PreInterpolator is used to call the equipment module in a synchronous cycle. This might be needed, e.g., to read out an actual value in the fast cycle or to write output signals quickly.

Figure 2-12: Call of an equipment module in MC-PreInterpolator OB



...MachineTemplate_Plc [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ 04_Motion ▸ MC-PreInterpolator [OB68]

MC-PreInterpolator

	Name	Data type	Default value	Comment
1	▼ Input			
2	Initial_Call	Bool		
3	PIP_Input	Bool		

Block title: .....

Comment

Network 1: .....

Comment

"EquipmentC_InstCallEnvSyncCycle"

"EquipmentC_CallEnvSyncCycle"

... — EN

"SyncNodes".equipmentC — syncNodes

"ControlNodes".equipmentC — controlNodes

"Config".equipmentC — configuration

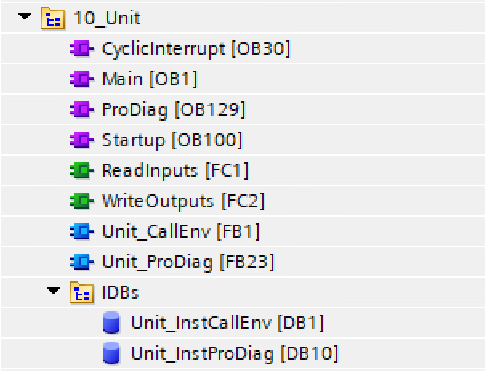
"Hardware".equipmentC — hardware

ENO —

2.1.5 Units

A machine is represented by a unit in the Machine Template project. All unit specific OBs / FBs / FCs / DBs that handle the logic related to the unit level can be found in the unit folder.

Figure 2-13: Content of the group 10_Unit



- 10_Unit
  - CyclicInterrupt [OB30]
  - Main [OB1]
  - ProDiag [OB129]
  - Startup [OB100]
  - ReadInputs [FC1]
  - WriteOutputs [FC2]
  - Unit_CallEnv [FB1]
  - Unit_ProDiag [FB23]
  - IDBs
    - Unit_InstCallEnv [DB1]
    - Unit_InstProDiag [DB10]

CyclicInterrupt OB

- Should be used to ensure stable task execution of the main process and to avoid communication load problems.
- Within the CyclicInterrupt OB, the ReadInputs FC is called first, then all CallEnv FBs and finally the WriteOutputs FC.

Startup OB

The Startup OB can be used to perform one-time configuration, initialization, or assignments during the start-up of the PLC.

ReadInputs FC

- Reads all inputs of the machine and writes the information to the corresponding control module in the Hardware DB.
- Must be called in the beginning of the cycle.

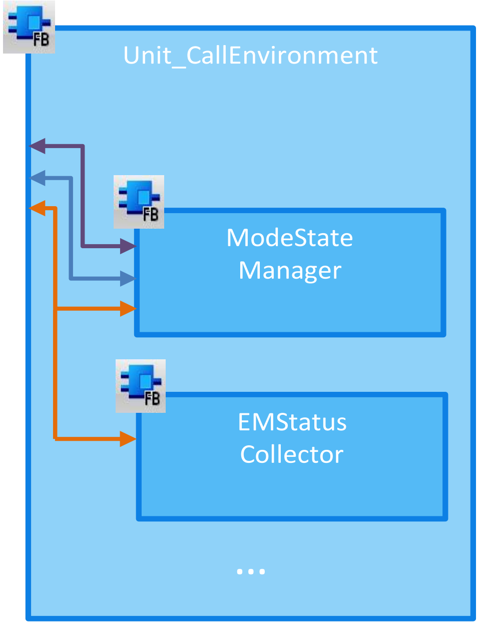
WriteOutputs FC

- Transfers the information for output handling from the corresponding control module / unit Hardware DB and sets PLC output tags.
- Must be called at the end of the cycle.

Unit_CallEnv FB

Call of all logic related to the unit and contains the Unit Mode State Manager and the required functional extensions, like stack light control FB.

Figure 2-14: Unit_CallEnvironment FB



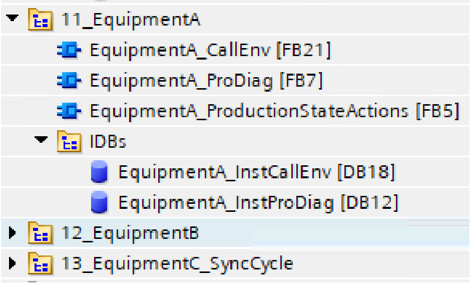
Unit_ProDiag FB

For each unit, a corresponding ProDiag FB must be created. This FB calls all supervisions and generates diagnostic information for the unit.

2.1.6 Equipments

The equipment folder contains the program to handle all logic related to the equipment module. In this template it and provides an example for calling the control modules, ProDiag FB and ProductionStateAction FB

Figure 2-15: Content of the group Equipment x

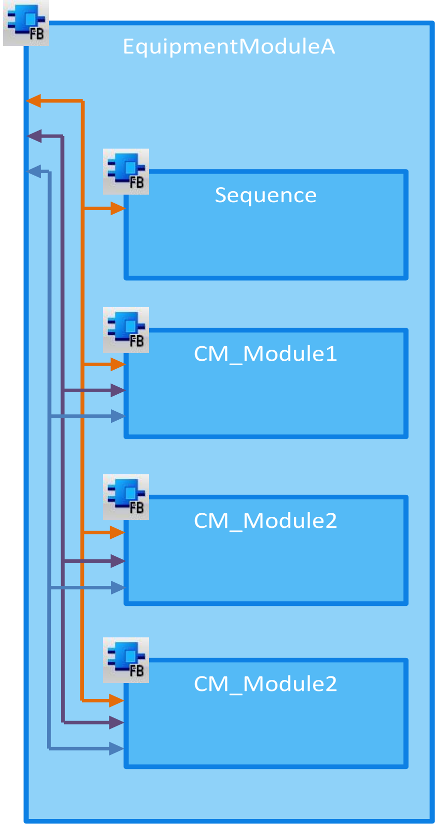


- 11_EquipmentA
  - EquipmentA_CallEnv [FB21]
  - EquipmentA_ProDiag [FB7]
  - EquipmentA_ProductionStateActions [FB5]
  - IDBs
    - EquipmentA_InstCallEnv [DB18]
    - EquipmentA_InstProDiag [DB12]
- 12_EquipmentB
- 13_EquipmentC_SyncCycle

Equipment_CallEnv FB

Call of all logic related to the corresponding equipment module, like the call of the sequences and control modules.

Figure 2-16: EquipmentModule FB



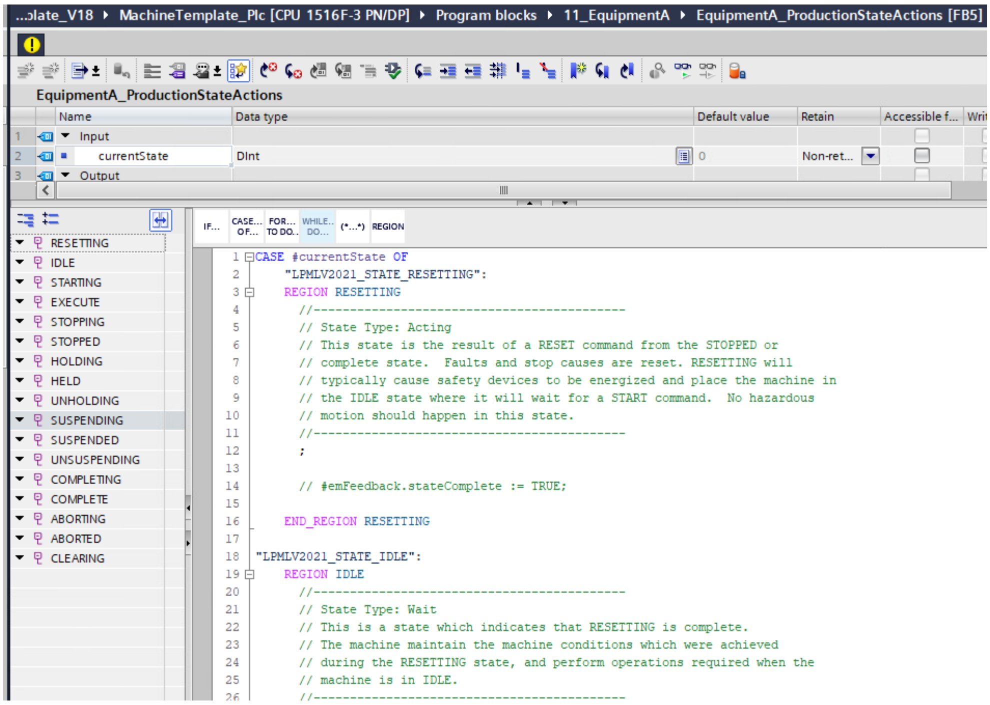
Equipment_ProDiag FB

For each equipment module, a corresponding ProDiag FB must be created. This FB calls all supervisions and generates diagnostic information for the equipment module.

Equipment_ProductionStateActions FB

The Equipment_ProductionStateActions FB provides the call of the sequence for the production mode and represents a functional implementation for all standard states.

Figure 2-17: ProductionStateActions FB



...plate_V18 ▸ MachineTemplate_Plc [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ 11_EquipmentA ▸ EquipmentA_ProductionStateActions [FB5]

EquipmentA_ProductionStateActions

	Name	Data type	Default value	Retain	Accessible f...	Wri
1	▼ Input				□	□
2	currentState	DInt	0	Non-ret...	□	□
3	▼ Output				□	□

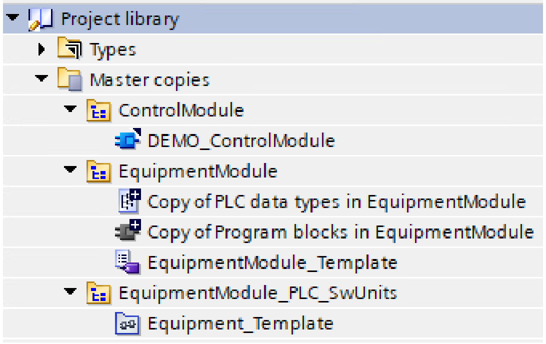
```
CASE #currentState OF
    "LPMLV2021_STATE_RESETTING":
    REGION RESETTING
      //-------------------------------------------
      // State Type: Acting
      // This state is the result of a RESET command from the STOPPED or
      // complete state.  Faults and stop causes are reset. RESETTING 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.
      //-------------------------------------------
      ;
      
      // #emFeedback.stateComplete := TRUE;
      
    END_REGION RESETTING
    
"LPMLV2021_STATE_IDLE":
    REGION IDLE
      //-------------------------------------------
      // State Type: Wait
      // This is a state which indicates that RESETTING is complete.
      // The machine maintain the machine conditions which were achieved
      // during the RESETTING state, and perform operations required when the
      // machine is in IDLE.
      //-------------------------------------------
```

2.1.7 Master copies

The master copies in the project library contain templates for a control module and equipment modules for the standard PLC structure and the PLC structure with software units.

These can be used to extend the project.

Figure 2-18: Master copies in the project library



- Project library
  - Types
  - Master copies
    - ControlModule
      - DEMO_ControlModule
    - EquipmentModule
      - Copy of PLC data types in EquipmentModule
      - Copy of Program blocks in EquipmentModule
      - EquipmentModule_Template
    - EquipmentModule_PLC_SwUnits
      - Equipment_Template

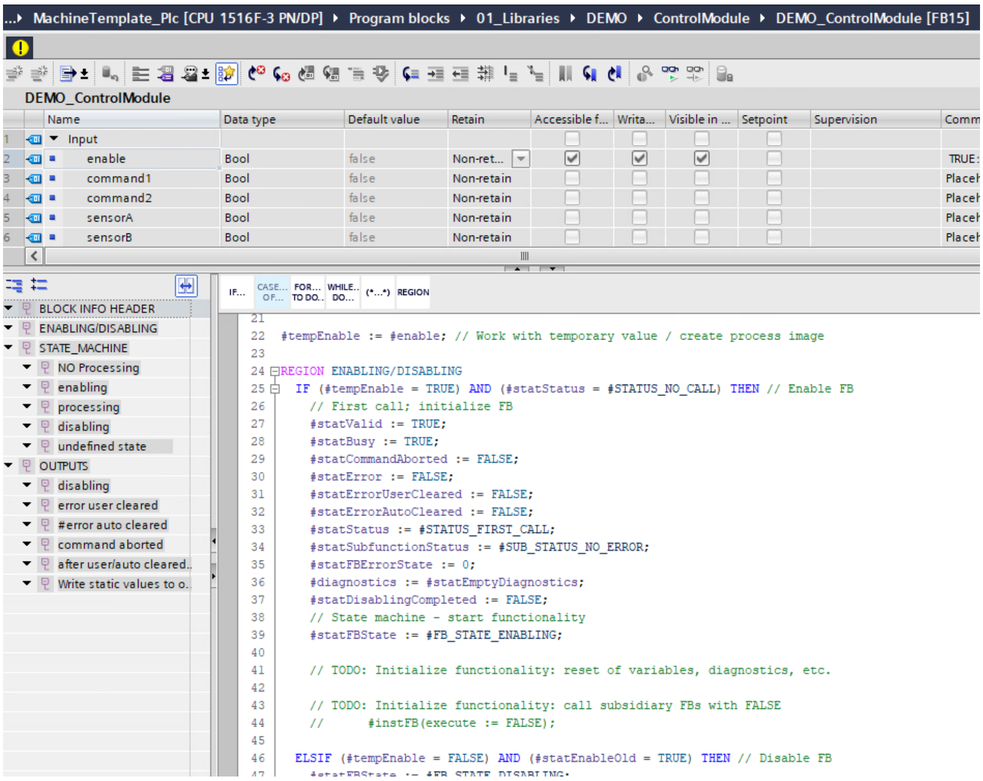
DEMO_ControlModule

The Machine Template provides a DEMO_ControlModule FB, that can be used to extend the project.

This FB can be found in the libraries > DEMO > ControlModule or in the Master copies.

Control modules can also be used directly from any stantard library, for example the Library of Basic Control (LBC).

Figure 2-19: DEMO_ControlModule FB



...▸ MachineTemplate_Plc [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ 01_Libraries ▸ DEMO ▸ ControlModule ▸ DEMO_ControlModule [FB15]

DEMO_ControlModule

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comm
1	▼ Input				□	□	□	□		
2	enable	Bool	false	Non-ret...	☑	☑	☑	□		TRUE:
3	command1	Bool	false	Non-retain	□	□	□	□		Placeh
4	command2	Bool	false	Non-retain	□	□	□	□		Placeh
5	sensorA	Bool	false	Non-retain	□	□	□	□		Placeh
6	sensorB	Bool	false	Non-retain	□	□	□	□		Placeh

- BLOCK INFO HEADER
- ENABLING/DISABLING
- STATE_MACHINE
  - NO Processing
  - enabling
  - processing
  - disabling
  - undefined state
- OUTPUTS
  - disabling
  - error user cleared
  - #error auto cleared
  - command aborted
  - after user/auto cleared..
  - Write static values to o..

```

#tempEnable := #enable; // Work with temporary value / create process image

REGION ENABLING/DISABLING
  IF (#tempEnable = TRUE) AND (#statStatus = #STATUS_NO_CALL) THEN // Enable FB
    // First call; initialize FB
    #statValid := TRUE;
    #statBusy := TRUE;
    #statCommandAborted := FALSE;
    #statError := FALSE;
    #statErrorUserCleared := FALSE;
    #statErrorAutoCleared := FALSE;
    #statStatus := #STATUS_FIRST_CALL;
    #statSubfunctionStatus := #SUB_STATUS_NO_ERROR;
    #statFBErrorState := 0;
    #diagnostics := #statEmptyDiagnostics;
    #statDisablingCompleted := FALSE;
    // State machine - start functionality
    #statFBState := #FB_STATE_ENABLING;

    // TODO: Initialize functionality: reset of variables, diagnostics, etc.

    // TODO: Initialize functionality: call subsidiary FBs with FALSE
    //      #instFB(execute := FALSE);

  ELSIF (#tempEnable = FALSE) AND (#statEnableOld = TRUE) THEN // Disable FB
    #statFBState := #FB_STATE_DISABLING;
```

NOTE

The following application example provides a description of the implemented mode/state machine function related LPMLV30 blocks.
https://support.industry.siemens.com/cs/ww/en/view/49970441

2.2 PLC Structure with Software Units

Using software units, allows a much more modular structure of the PLC program. We recommend this approach.

Software units allow breaking down the user program into separately loadable units. You can load changes in different software units independently of one another. This is especially helpful when these changes were made by different users.

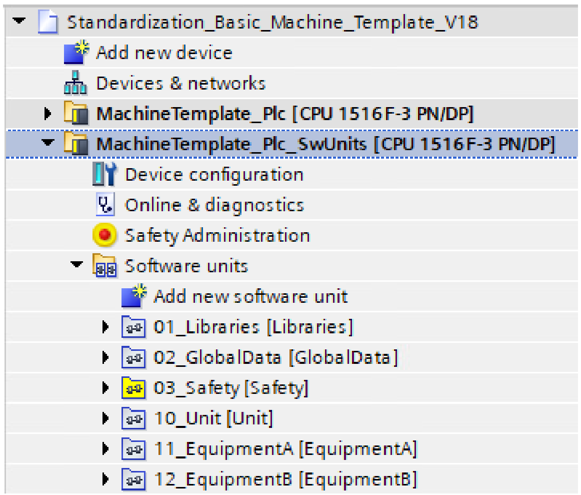
NOTE

Detailed information about software units can be found in STEP 7 manual

https://support.industry.siemens.com/cs/ww/en/view/109815056/160160660107

The PLC program with software units in TIA Portal V18 has the following structure.

Figure 2-20: PLC Program structure with software units



```
▼ Standardization_Basic_Machine_Template_V18
    Add new device
    Devices & networks
  ► MachineTemplate_Plc [CPU 1516F-3 PN/DP]
  ▼ MachineTemplate_Plc_SwUnits [CPU 1516F-3 PN/DP]
      Device configuration
      Online & diagnostics
      Safety Administration
    ▼ Software units
        Add new software unit
      ► 01_Libraries [Libraries]
      ► 02_GlobalData [GlobalData]
      ► 03_Safety [Safety]
      ► 10_Unit [Unit]
      ► 11_EquipmentA [EquipmentA]
      ► 12_EquipmentB [EquipmentB]
```

Software units in TIA Portal V18 allow namespaces to structure the program.

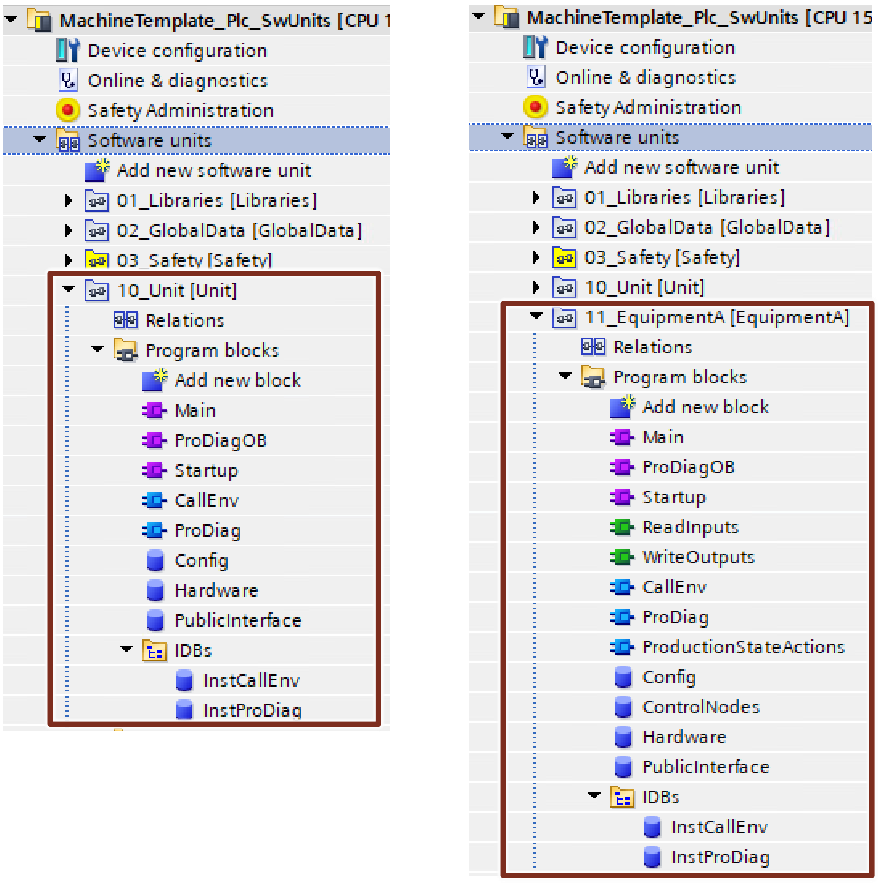
Namespaces offer the following advantages:

- Program elements can have the same name if they are in different namespaces.
- Clear representation of elements in the project tree and of operands in the program code.

The main advantage compared to the standard PLC structure without software units is that blocks can be assigned to the corresponding software unit.

As the following figure shows, all related blocks are assigned to the desired unit / equipment module. Additionally, OBs, FBs and DBs in the same project have the same names because they are assigned to different namespaces.

Figure 2-21: Content of software units “Unit” and “Equipment x”



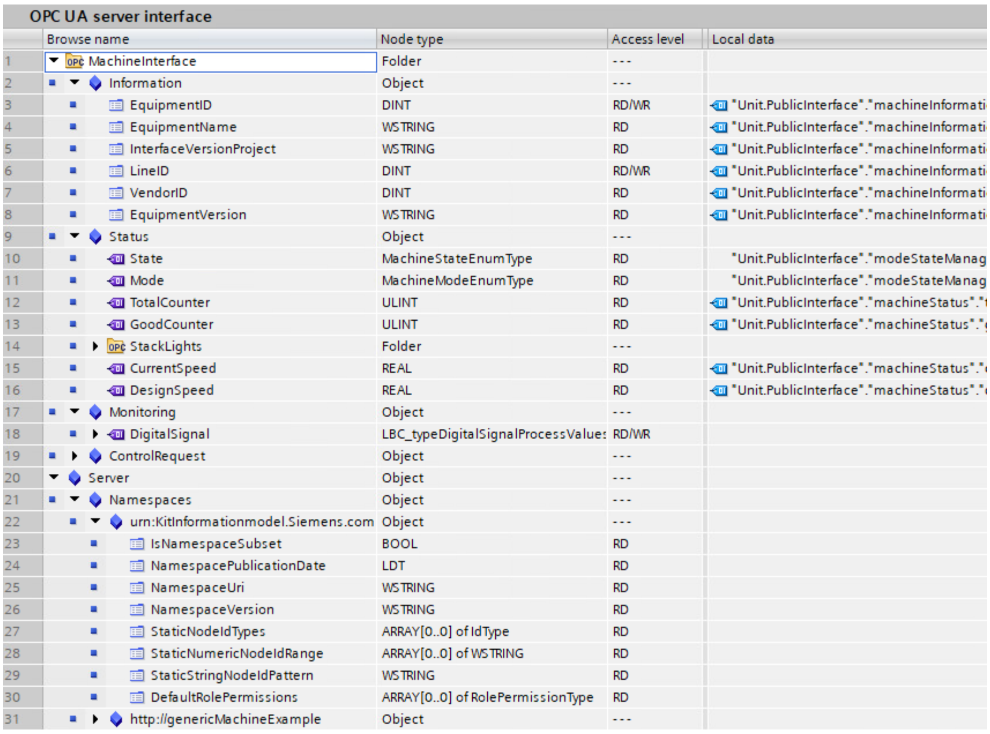
```
MachineTemplate_Plc_SwUnits [CPU 1
  Device configuration
  Online & diagnostics
  Safety Administration
  Software units
    Add new software unit
    01_Libraries [Libraries]
    02_GlobalData [GlobalData]
    03_Safety [Safety]
    10_Unit [Unit]
      Relations
      Program blocks
        Add new block
        Main
        ProDiagOB
        Startup
        CallEnv
        ProDiag
        Config
        Hardware
        PublicInterface
        IDBs
          InstCallEnv
          InstProDiag
```

```
MachineTemplate_Plc_SwUnits [CPU 15
  Device configuration
  Online & diagnostics
  Safety Administration
  Software units
    Add new software unit
    01_Libraries [Libraries]
    02_GlobalData [GlobalData]
    03_Safety [Safety]
    10_Unit [Unit]
    11_EquipmentA [EquipmentA]
      Relations
      Program blocks
        Add new block
        Main
        ProDiagOB
        Startup
        ReadInputs
        WriteOutputs
        CallEnv
        ProDiag
        ProductionStateActions
        Config
        ControlNodes
        Hardware
        PublicInterface
        IDBs
          InstCallEnv
          InstProDiag
```

2.3 OPC UA communication

The Machine Template provides an interface to control and monitor the machine. The structure used in this template is shown in the following figure. It is defined in the OPC UA communication configuration as an OPC UA server interface.

Figure 2-22: OPC UA server interface



OPC UA server interface

	Browse name	Node type	Access level	Local data
1	MachineInterface	Folder	---	
2	Information	Object	---	
3	EquipmentID	DINT	RD/WR	"Unit.PublicInterface"."machineInformati
4	EquipmentName	WSTRING	RD	"Unit.PublicInterface"."machineInformati
5	InterfaceVersionProject	WSTRING	RD	"Unit.PublicInterface"."machineInformati
6	LineID	DINT	RD/WR	"Unit.PublicInterface"."machineInformati
7	VendorID	DINT	RD	"Unit.PublicInterface"."machineInformati
8	EquipmentVersion	WSTRING	RD	"Unit.PublicInterface"."machineInformati
9	Status	Object	---	
10	State	MachineStateEnumType	RD	"Unit.PublicInterface"."modeStateManag
11	Mode	MachineModeEnumType	RD	"Unit.PublicInterface"."modeStateManag
12	TotalCounter	ULINT	RD	"Unit.PublicInterface"."machineStatus"."
13	GoodCounter	ULINT	RD	"Unit.PublicInterface"."machineStatus"."
14	StackLights	Folder	---	
15	CurrentSpeed	REAL	RD	"Unit.PublicInterface"."machineStatus"."
16	DesignSpeed	REAL	RD	"Unit.PublicInterface"."machineStatus"."
17	Monitoring	Object	---	
18	DigitalSignal	LBC_typeDigitalSignalProcessValue:	RD/WR	
19	ControlRequest	Object	---	
20	Server	Object	---	
21	Namespaces	Object	---	
22	urn:KitInformationmodel.Siemens.com	Object	---	
23	IsNamespaceSubset	BOOL	RD	
24	NamespacePublicationDate	LDT	RD	
25	NamespaceUri	WSTRING	RD	
26	NamespaceVersion	WSTRING	RD	
27	StaticNodeIdTypes	ARRAY[0..0] of IdType	RD	
28	StaticNumericNodeIdRange	ARRAY[0..0] of WSTRING	RD	
29	StaticStringNodeIdPattern	WSTRING	RD	
30	DefaultRolePermissions	ARRAY[0..0] of RolePermissionType	RD	
31	http://genericMachineExample	Object	---	

NOTE

The OPC UA communication protocol enables you to create your own standards for an OPC UA information model. You can generate a standardized user-specific server interface.

The application example provides an OPC UA information model which provides OPC UA objects for main manufacturing use cases
https://support.industry.siemens.com/cs/ww/en/view/109755133

2.4 Alarm Handling

ProDiag is used in the Machine Template to detect errors in time, to avoid production downtimes and to guarantee high availability.

Each unit and equipment module in this Machine Template has its own ProDiag FB and provides its diagnostic information.

As an example, a ProDiag supervision is preconfigured in the DEMO_ControlModule FB as this is instantiated in the Unit_CallEnv FB.

When an error occurs in this FB, a ProDiag supervision is triggered, which displays the diagnostic information of the relevant module. The PLC integrated alarm server receives this alarm and provides it to all clients (TIA Portal, PLC Display, HMI, SIMATIC web server, etc.).

2.4.1 Pre-Configuration

The messages in the Machine Template are generated in English. Multilingual texts are possible.

To display the messages correctly,

- the correct project languages must be selected,
- the languages must be configured correctly in the device configuration,
- the corresponding texts must be available in this language.

2.4.2 Error Detection

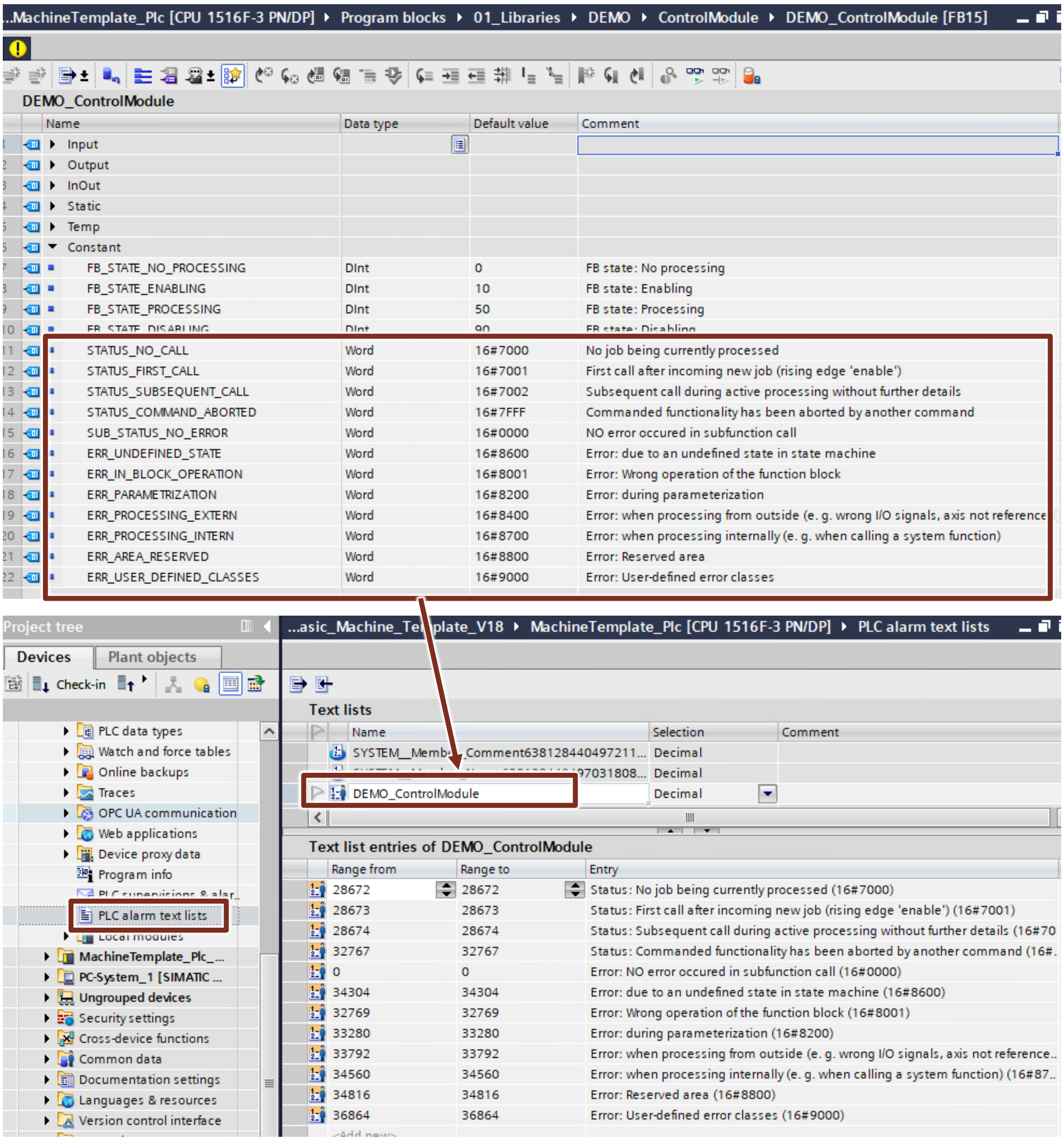
When an error occurs, an alarm should be generated, and the correct alarm message should be displayed.

If an error occurs (e.g. error in block operation), the error must be output and the status is set (16#8001).

Constants are assigned to the status code, which contain the specific information about the error (e.g. 16#8001: Error: Wrong operation of the function block).

These constants are included in a PLC alarm text list DEMO_ControlModule.

Figure 2-23: Diagnostic information of DEMO_ControlModule represented in a PLC alarm text list



DEMO_ControlModule

Name	Data type	Default value	Comment
▸ Input			
▸ Output			
▸ InOut			
▸ Static			
▸ Temp			
▾ Constant			
FB_STATE_NO_PROCESSING	DInt	0	FB state: No processing
FB_STATE_ENABLING	DInt	10	FB state: Enabling
FB_STATE_PROCESSING	DInt	50	FB state: Processing
FB_STATE_DISABLING	DInt	90	FB state: Disabling
STATUS_NO_CALL	Word	16#7000	No job being currently processed
STATUS_FIRST_CALL	Word	16#7001	First call after incoming new job (rising edge 'enable')
STATUS_SUBSEQUENT_CALL	Word	16#7002	Subsequent call during active processing without further details
STATUS_COMMAND_ABORTED	Word	16#7FFF	Commanded functionality has been aborted by another command
SUB_STATUS_NO_ERROR	Word	16#0000	NO error occured in subfunction call
ERR_UNDEFINED_STATE	Word	16#8600	Error: due to an undefined state in state machine
ERR_IN_BLOCK_OPERATION	Word	16#8001	Error: Wrong operation of the function block
ERR_PARAMETRIZATION	Word	16#8200	Error: during parameterization
ERR_PROCESSING_EXTERN	Word	16#8400	Error: when processing from outside (e. g. wrong I/O signals, axis not reference
ERR_PROCESSING_INTERN	Word	16#8700	Error: when processing internally (e. g. when calling a system function)
ERR_AREA_RESERVED	Word	16#8800	Error: Reserved area
ERR_USER_DEFINED_CLASSES	Word	16#9000	Error: User-defined error classes

Text lists

Name	Selection	Comment
SYSTEM__Memb_Comment638128440497211...	Decimal	
...97031808...	Decimal	
DEMO_ControlModule	Decimal	

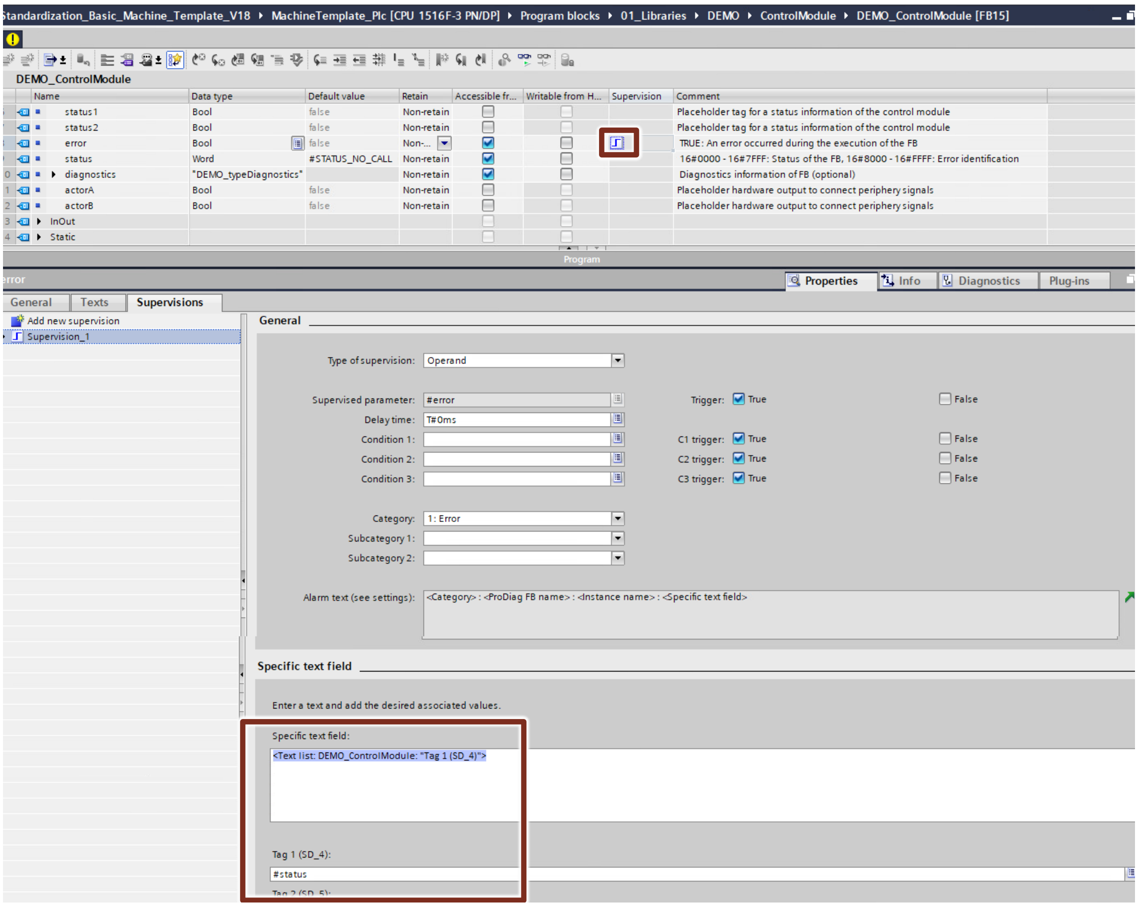
Text list entries of DEMO_ControlModule

Range from	Range to	Entry
28672	28672	Status: No job being currently processed (16#7000)
28673	28673	Status: First call after incoming new job (rising edge 'enable') (16#7001)
28674	28674	Status: Subsequent call during active processing without further details (16#70
32767	32767	Status: Commanded functionality has been aborted by another command (16#.
0	0	Error: NO error occured in subfunction call (16#0000)
34304	34304	Error: due to an undefined state in state machine (16#8600)
32769	32769	Error: Wrong operation of the function block (16#8001)
33280	33280	Error: during parameterization (16#8200)
33792	33792	Error: when processing from outside (e. g. wrong I/O signals, axis not reference..
34560	34560	Error: when processing internally (e. g. when calling a system function) (16#87..
34816	34816	Error: Reserved area (16#8800)
36864	36864	Error: User-defined error classes (16#9000)
<Add new>		

2.4.3 Alarm Generating

The error bit in the output area of the DEMO_ControlModule FB is configured with a ProDiag supervision and the status refers the corresponding alarm message.

Figure 2-24: ProDiag supervision configuration



Standardization_Basic_Machine_Template_V18 ▸ MachineTemplate_Plc [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ 01_Libraries ▸ DEMO ▸ ControlModule ▸ DEMO_ControlModule [FB15]

DEMO_ControlModule

Name	Data type	Default value	Retain	Accessible fr...	Writable from H...	Supervision	Comment
status1	Bool	false	Non-retain	☐	☐		Placeholder tag for a status information of the control module
status2	Bool	false	Non-retain	☐	☐		Placeholder tag for a status information of the control module
error	Bool	false	Non-...	☑	☐	⎍	TRUE: An error occurred during the execution of the FB
status	Word	#STATUS_NO_CALL	Non-retain	☑	☐		16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification
▸ diagnostics	"DEMO_typeDiagnostics"		Non-retain	☑	☐		Diagnostics information of FB (optional)
actorA	Bool	false	Non-retain	☐	☐		Placeholder hardware output to connect periphery signals
actorB	Bool	false	Non-retain	☐	☐		Placeholder hardware output to connect periphery signals
▸ InOut				☐	☐		
▸ Static				☐	☐		

error

General | Texts | Supervisions

Add new supervision
Supervision_1

General

Type of supervision: Operand

Supervised parameter: #error — Trigger: ☑ True ☐ False
Delay time: T#0ms
Condition 1: — C1 trigger: ☑ True ☐ False
Condition 2: — C2 trigger: ☑ True ☐ False
Condition 3: — C3 trigger: ☑ True ☐ False

Category: 1: Error
Subcategory 1:
Subcategory 2:

Alarm text (see settings): <Category> : <ProDiag FB name> : <Instance name> : <Specific text field>

Specific text field

Enter a text and add the desired associated values.

Specific text field: <Text list: DEMO_ControlModule: "Tag 1 (SD_4)">

Tag 1 (SD_4): #status

Tag 2 (SD_5):

In the properties of the selected ProDiag supervision, specific information about the alarm text is shown and can be configured.

In this example, the displayed alarm message includes the category of the supervision, the ProDiag FB name, the instance name, and the specific text field.

In the specific text field, associated values are used. In the Machine Template, a dynamic parameter (text list) is configured.

<Text list: DEMO_ControlModule: "Tag 1 (SD_4)">

The alarm message is generated using a provided text list in combination with the status word of the FB.

NOTE More information about ProDiag can be found in the following entry https://support.industry.siemens.com/cs/ww/en/view/109740151

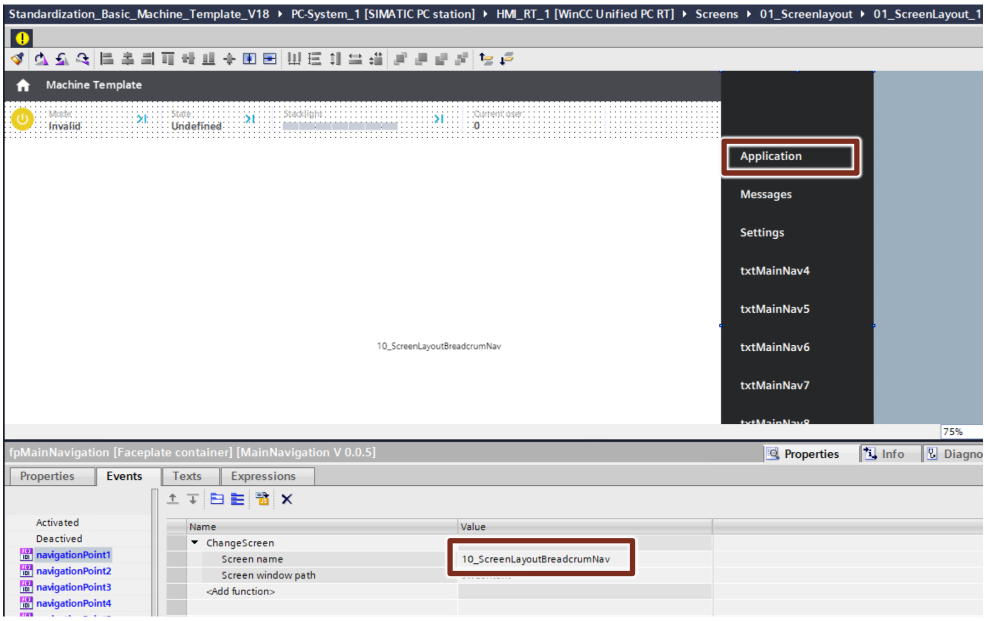
2.5 HMI in WinCC Unified

The Machine Template provides a customizable HMI Template for WinCC Unified with the goal of not starting the development of the HMI interface from scratch and reducing the programming effort.

2.5.1 Main navigation

The screen layout "01_ScreenLayout_1" is the base screen of the HMI Application in the Machine Template and calls the screen "10_ScreenLayoutBreadcrumNav" to change the screen content based on a defined navigation file.

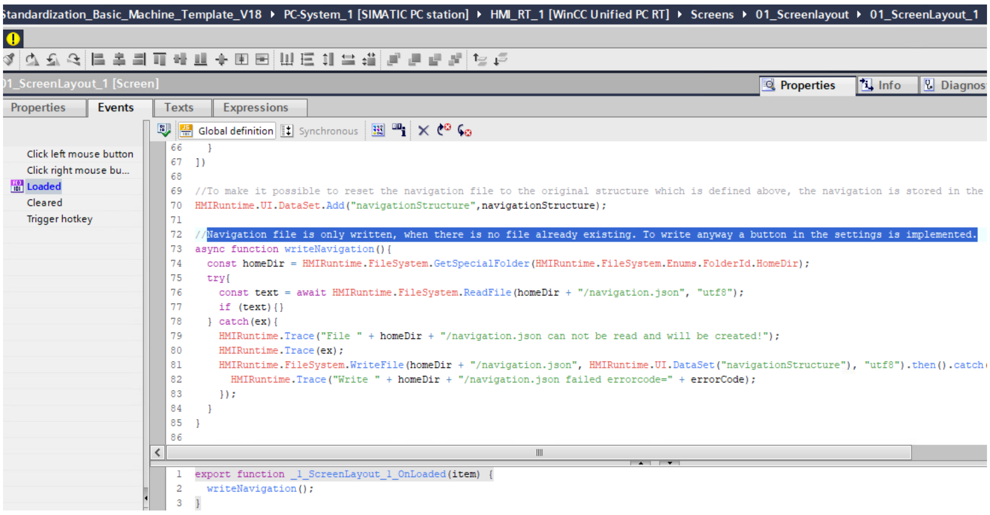
Figure 2-25: 01_ScreenLayout_1



ChangeScreen
Screen name: 10_ScreenLayoutBreadcrumNav
Screen window path
<Add function>

The navigation file is created in the global definition of the "01_ScreenLayout_1" and called via a function "writeNavigation" as a loaded Event.

Figure 2-26: 01_ScreenLayout_1 Event

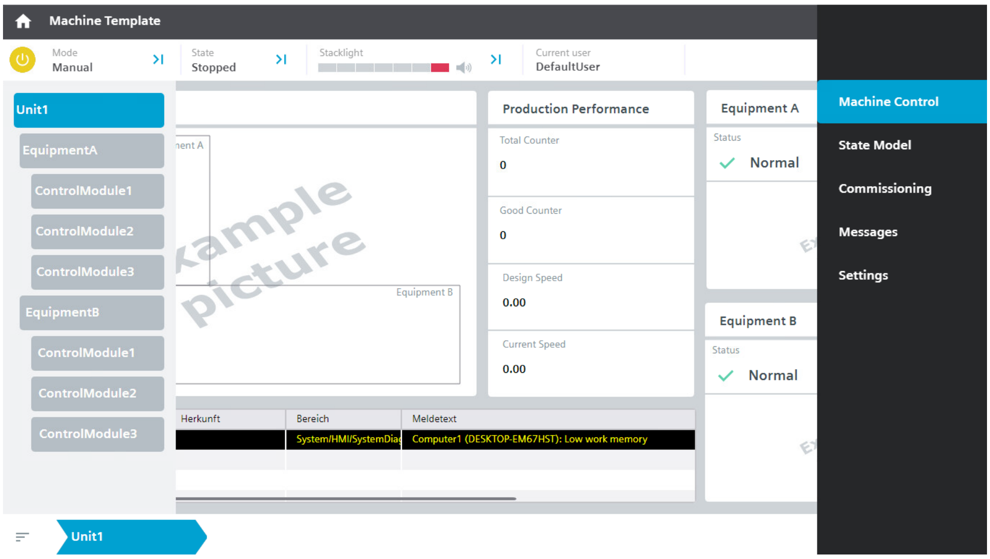

Standardization_Basic_Machine_Template_V18 ▸ PC-System_1 [SIMATIC PC station] ▸ HMI_RT_1 [WinCC Unified PC RT] ▸ Screens ▸ 01_Screenlayout ▸ 01_ScreenLayout_1
01_ScreenLayout_1 [Screen]
Properties
Info
Diagnos
Properties
Events
Texts
Expressions
Click left mouse button
Click right mouse bu...
Loaded
Cleared
Trigger hotkey
Global definition
Synchronous

```
  }
])

//To make it possible to reset the navigation file to the original structure which is defined above, the navigation is stored in the
HMIRuntime.UI.DataSet.Add("navigationStructure",navigationStructure);

//Navigation file is only written, when there is no file already existing. To write anyway a button in the settings is implemented.
async function writeNavigation(){
  const homeDir = HMIRuntime.FileSystem.GetSpecialFolder(HMIRuntime.FileSystem.Enums.FolderId.HomeDir);
  try{
    const text = await HMIRuntime.FileSystem.ReadFile(homeDir + "/navigation.json", "utf8");
    if (text){}
  } catch(ex){
    HMIRuntime.Trace("File " + homeDir + "/navigation.json can not be read and will be created!");
    HMIRuntime.Trace(ex);
    HMIRuntime.FileSystem.WriteFile(homeDir + "/navigation.json", HMIRuntime.UI.DataSet("navigationStructure"), "utf8").then().catch
      HMIRuntime.Trace("Write " + homeDir + "/navigation.json failed errorcode=" + errorCode);
    });
  }
}

```

```
export function _1_ScreenLayout_1_OnLoaded(item) {
  writeNavigation();
}
```

After the execution of the navigation file, the main navigation looks like this with the right screen content.

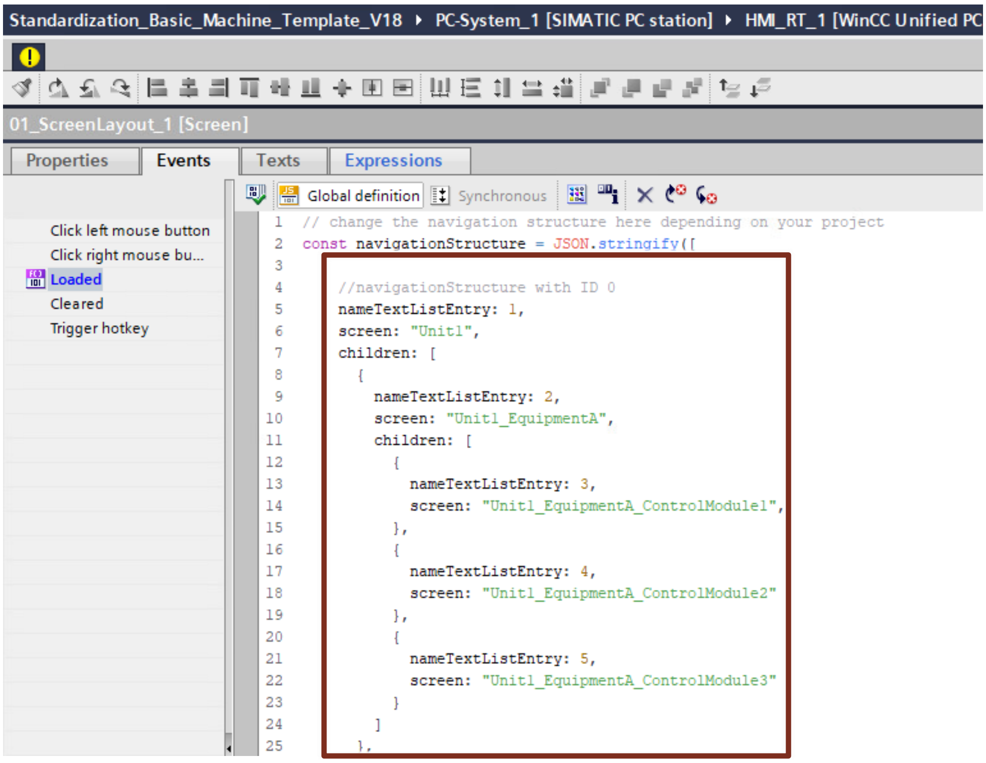
Figure 2-27: Main navigation



2.5.1.1 Object oriented navigation

An object-oriented navigation structure is created in the global definition of the "01_ScreenLayout_1" and defines the setup of the navigation. Based on this structure and the "wirteNavigation" function, the navigation with the right screen content (Unit, Equipment Module, Control Module) is called.

Figure 2-28: Navigation Structure



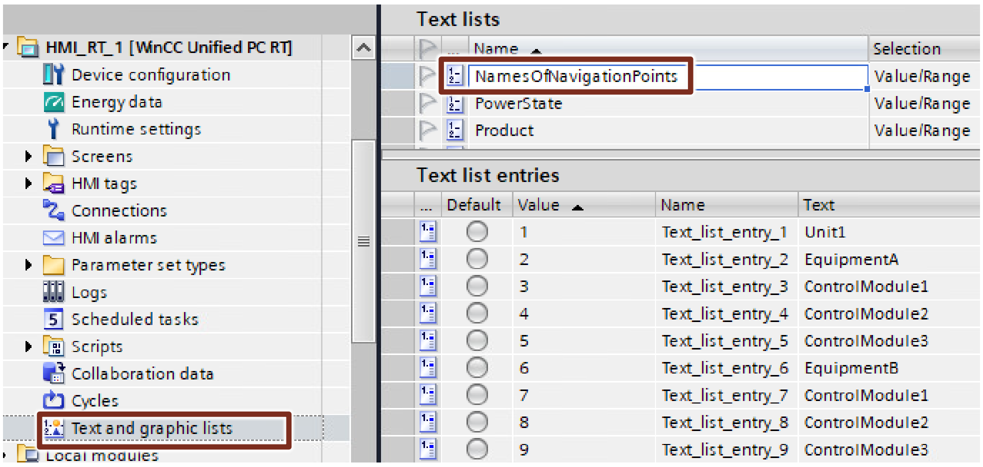
```
// change the navigation structure here depending on your project
const navigationStructure = JSON.stringify([

    //navigationStructure with ID 0
    nameTextListEntry: 1,
    screen: "Unit1",
    children: [
      {
        nameTextListEntry: 2,
        screen: "Unit1_EquipmentA",
        children: [
          {
            nameTextListEntry: 3,
            screen: "Unit1_EquipmentA_ControlModule1",
          },
          {
            nameTextListEntry: 4,
            screen: "Unit1_EquipmentA_ControlModule2"
          },
          {
            nameTextListEntry: 5,
            screen: "Unit1_EquipmentA_ControlModule3"
          }
        ]
      },
```

The navigation structure is an Array and includes children. For example, the navigation structure with the ID (0) represents in the application the "Unit1".

The navigation structure is based on the screen ID, which is assigned following an ascending order according to its position on each navigation level.

The "nameTextListEntry" in the structure defines the name of the navigation point and this name is included in a text list (NamesOfNavigationPoints). The name of the "Unit1" can be shown in the text list with the entry 1.

Figure 2-29: Text and graphic list



Text lists

Name	Selection
NamesOfNavigationPoints	Value/Range
PowerState	Value/Range
Product	Value/Range

Text list entries

Default	Value	Name	Text
○	1	Text_list_entry_1	Unit1
○	2	Text_list_entry_2	EquipmentA
○	3	Text_list_entry_3	ControlModule1
○	4	Text_list_entry_4	ControlModule2
○	5	Text_list_entry_5	ControlModule3
○	6	Text_list_entry_6	EquipmentB
○	7	Text_list_entry_7	ControlModule1
○	8	Text_list_entry_8	ControlModule2
○	9	Text_list_entry_9	ControlModule3

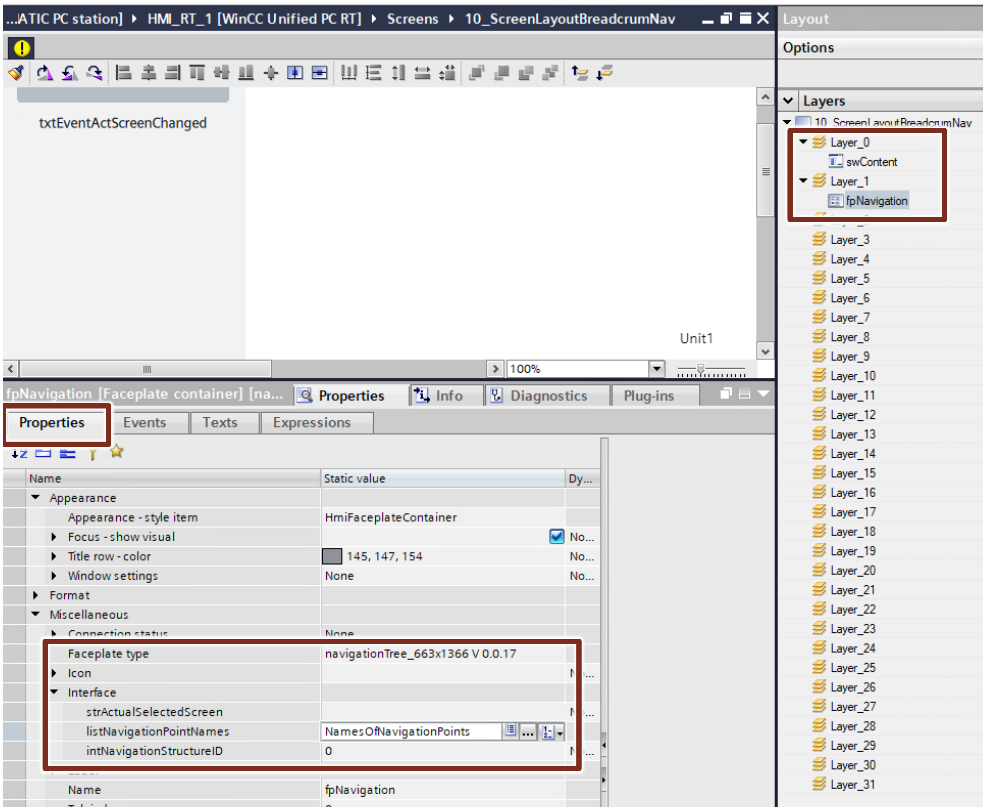
NOTE

The navigation file is only written when there is no file already existing. To be able to write a new navigation file a rebuild button "Rebuild Navigation file" in the settings is implemented.

The screen "10_ScreenLayoutBreadcrumNav" includes a faceplate for the navigation (fpNavigation) and a content screen window (swContent).

In the properties of the "fpNavigation" the faceplate, the name of the text list and the screen ID of the navigation structure are connected.

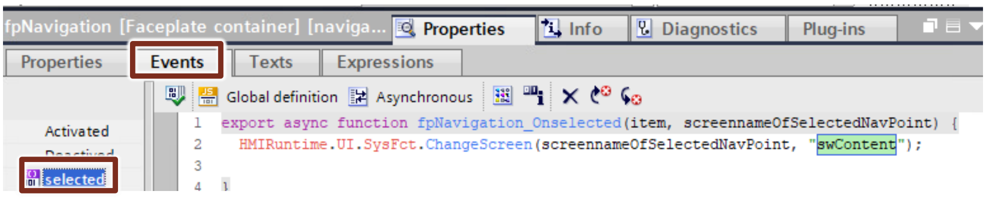
Figure 2-30: Faceplate properties



Name	Static value	Dy...
Appearance		
Appearance - style item	HmiFaceplateContainer	
Focus - show visual	☑	No...
Title row - color	▨ 145, 147, 154	No...
Window settings	None	No...
Format		
Miscellaneous		
Connection status	None	
Faceplate type	navigationTree_663x1366 V 0.0.17	
Icon		N...
Interface		
strActualSelectedScreen		N...
listNavigationPointNames	NamesOfNavigationPoints	
intNavigationStructureID	0	N...
Name	fpNavigation	

In the Events, the function changes the screen window content depending on the selected screen name.

Figure 2-31: Faceplate events

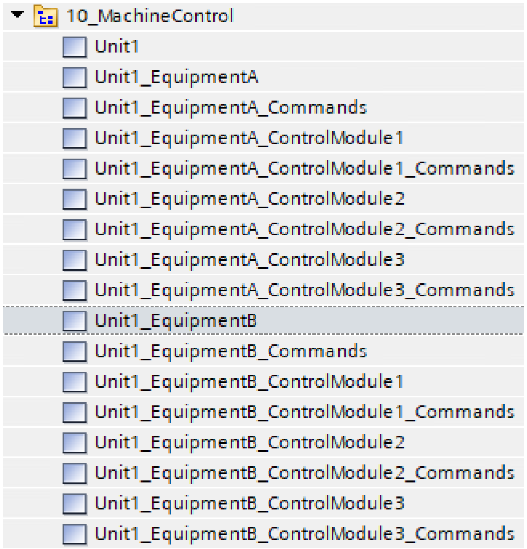


```
export async function fpNavigation_Onselected(item, screennameOfSelectedNavPoint) {
  HMIRuntime.UI.SysFct.ChangeScreen(screennameOfSelectedNavPoint, "swContent");

}
```

2.5.1.2 Screens

The "MachineControl" folder contains the screens showing the structure of the Unit, Equipment Module and Control Module in the PLC.

Figure 2-32: Content of folder 10_MachineControl



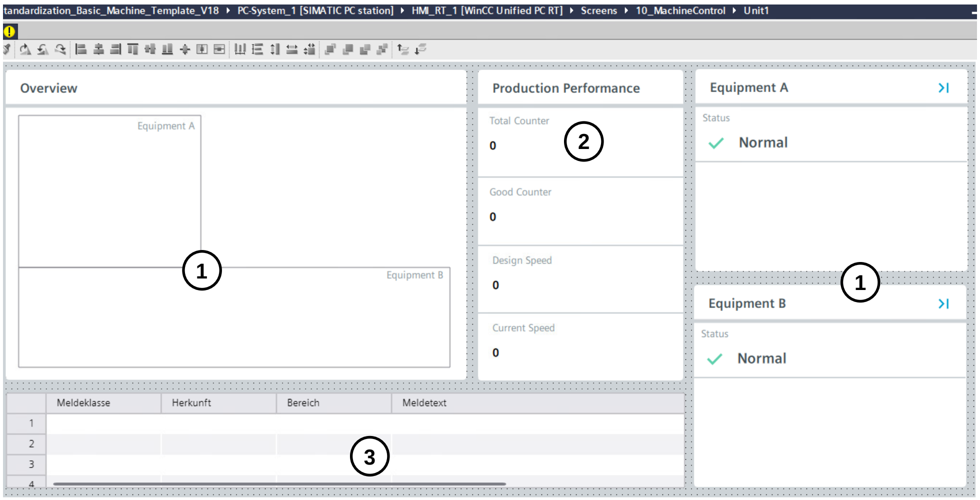
- 10_MachineControl
  - Unit1
  - Unit1_EquipmentA
  - Unit1_EquipmentA_Commands
  - Unit1_EquipmentA_ControlModule1
  - Unit1_EquipmentA_ControlModule1_Commands
  - Unit1_EquipmentA_ControlModule2
  - Unit1_EquipmentA_ControlModule2_Commands
  - Unit1_EquipmentA_ControlModule3
  - Unit1_EquipmentA_ControlModule3_Commands
  - Unit1_EquipmentB
  - Unit1_EquipmentB_Commands
  - Unit1_EquipmentB_ControlModule1
  - Unit1_EquipmentB_ControlModule1_Commands
  - Unit1_EquipmentB_ControlModule2
  - Unit1_EquipmentB_ControlModule2_Commands
  - Unit1_EquipmentB_ControlModule3
  - Unit1_EquipmentB_ControlModule3_Commands

Unit screen

The Unit screen contains

1. an overview of the related Equipment Modules,
2. an area which displays production performance of the Unit,
3. an alarm view displaying alarms associated to the Unit

Figure 2-33: Unit screen



tandardization_Basic_Machine_Template_V18 ▸ PC-System_1 [SIMATIC PC station] ▸ HMI_RT_1 [WinCC Unified PC RT] ▸ Screens ▸ 10_MachineControl ▸ Unit1

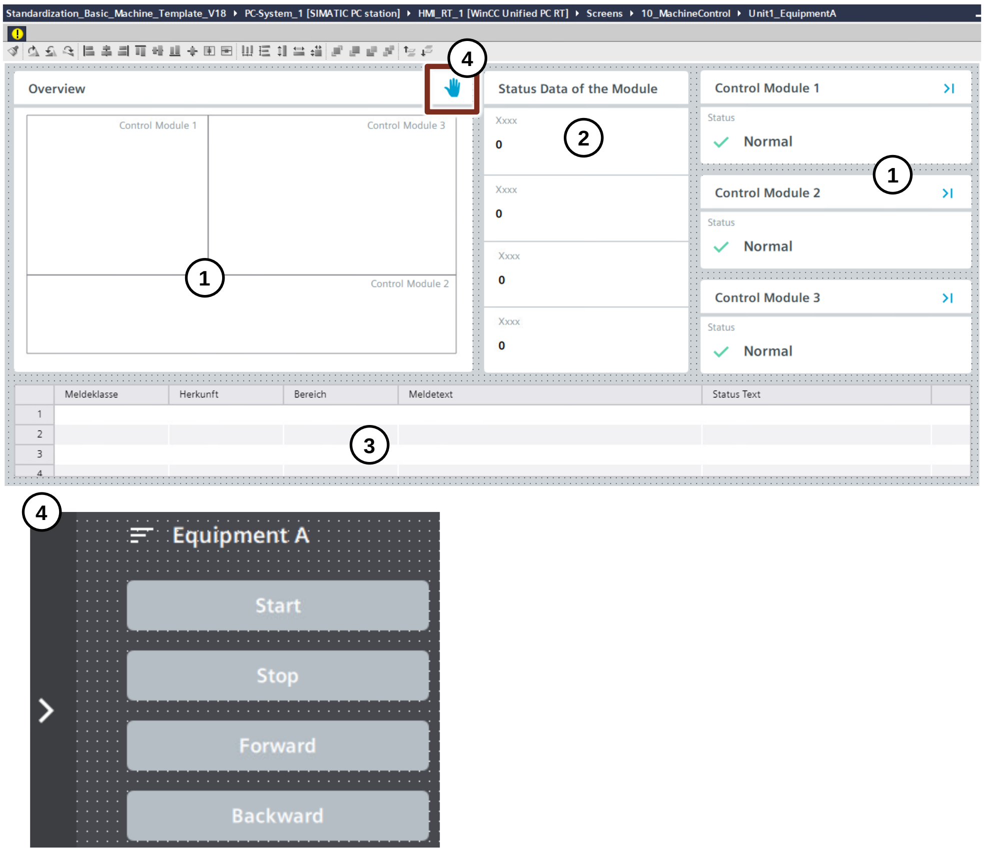
	Meldeklasse	Herkunft	Bereich	Meldetext
1				
2				
3				
4				

Equipment Module screen

The Equipment Module screen contains

1. an overview of the related Control Modules,
2. an area which displays status data of the Equipment Module,
3. an alarm view displaying alarms associated to the Equipment Module
4. a command for manual operation

Figure 2-34: Equipment module screen

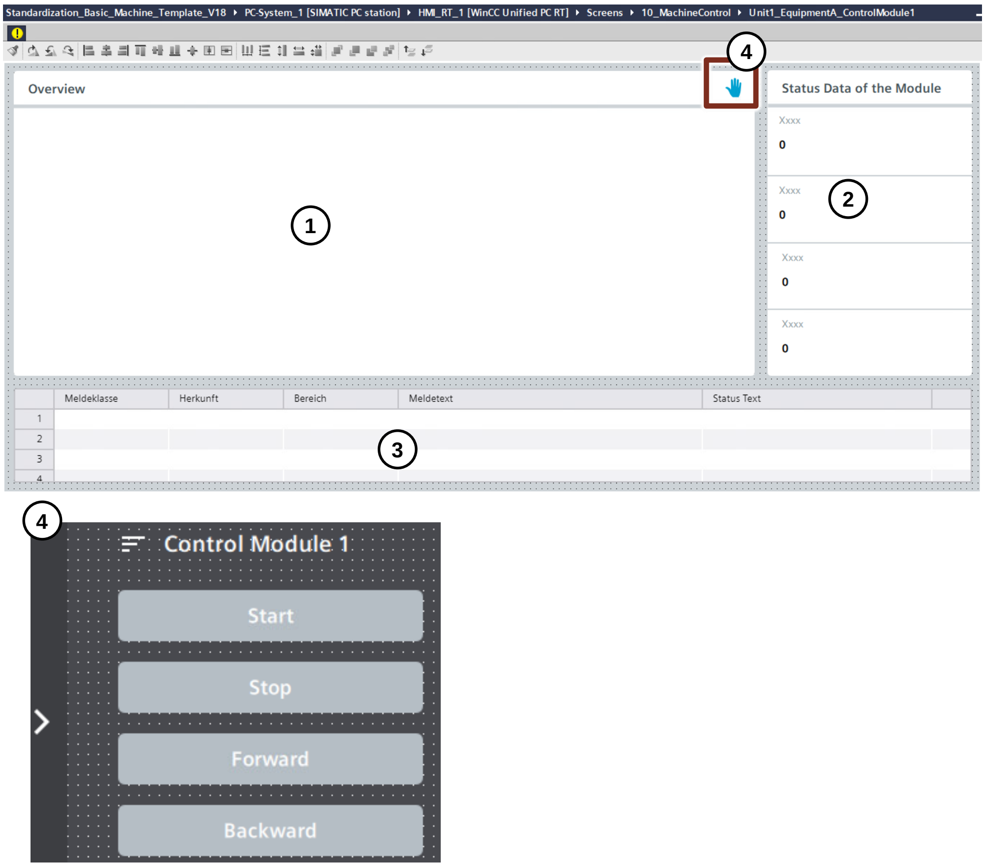


Control Module screen

The Control Module screen contains

1. an overview of the Control Modules itself, which can be designed as desired
2. an area which displays status data of the Control Module
3. an alarm view displaying alarms associated to the Control Module
4. a command for manual operation

Figure 2-35: Control module screen

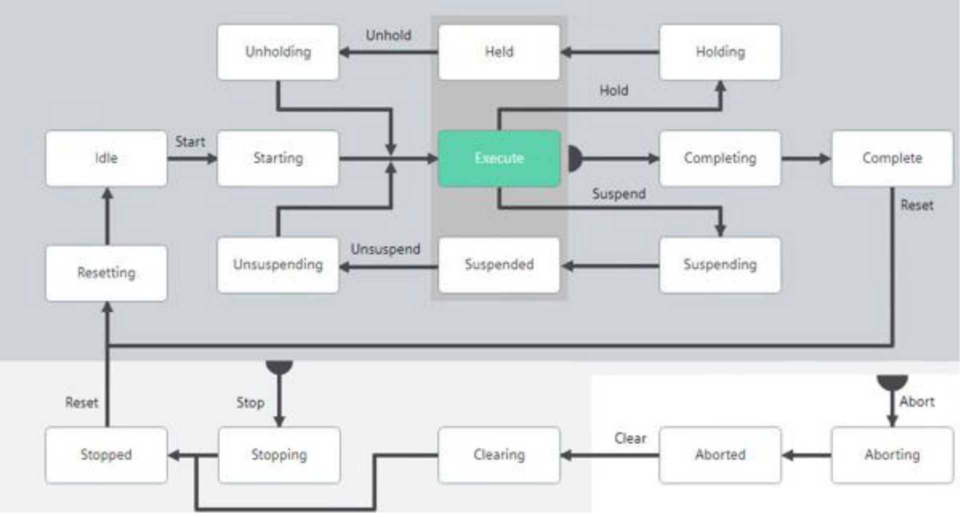

Standardization_Basic_Machine_Template_V18 ▸ PC-System_1 [SIMATIC PC station] ▸ HMI_RT_1 [WinCC Unified PC RT] ▸ Screens ▸ 10_MachineControl ▸ Unit1_EquipmentA_ControlModule1
Overview
4
Status Data of the Module
Xxxx
0
Xxxx
0
2
Xxxx
0
Xxxx
0
1
Meldeklasse
Herkunft
Bereich
Meldetext
Status Text
1
2
3
4
3
4
Control Module 1
Start
Stop
Forward
Backward

2.5.2 State Model

The state model helps operator understand what the machine is doing and what is needed to move into a more productive state.

The Machine Template provides a state model with a set of predefined states based on the OMAC model as an example.

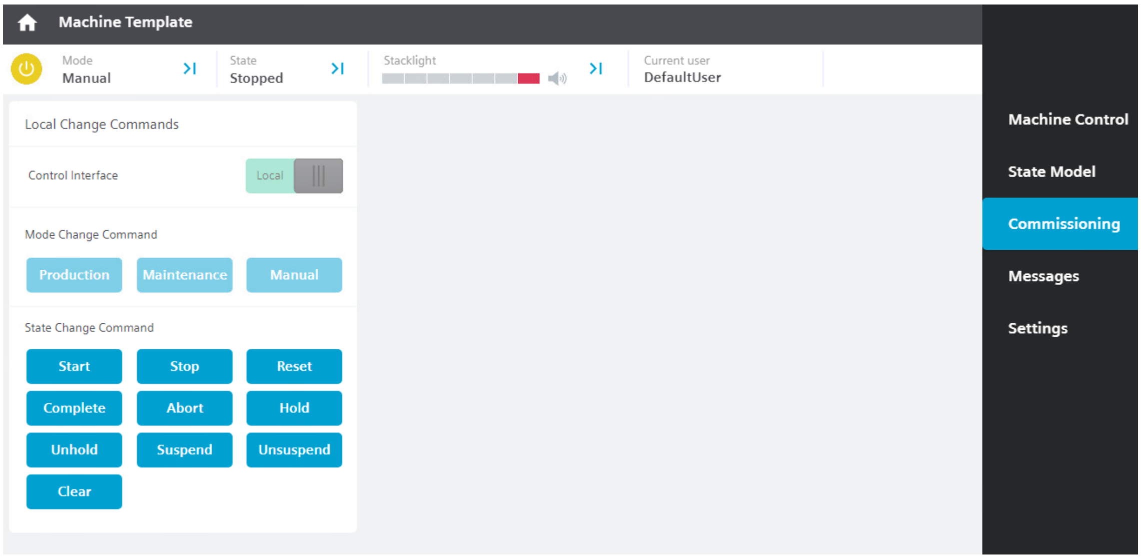
Figure 2-36: State model screen



2.5.3 Commissioning

The commissioning area offers the maintenance engineer the possibility to switch between the different modes and states and to control the machine manually. There are two types of control interfaces (Local and Remote).

Figure 2-37: Commissioning screen



Mode
Manual

State
Stopped

Current user
DefaultUser

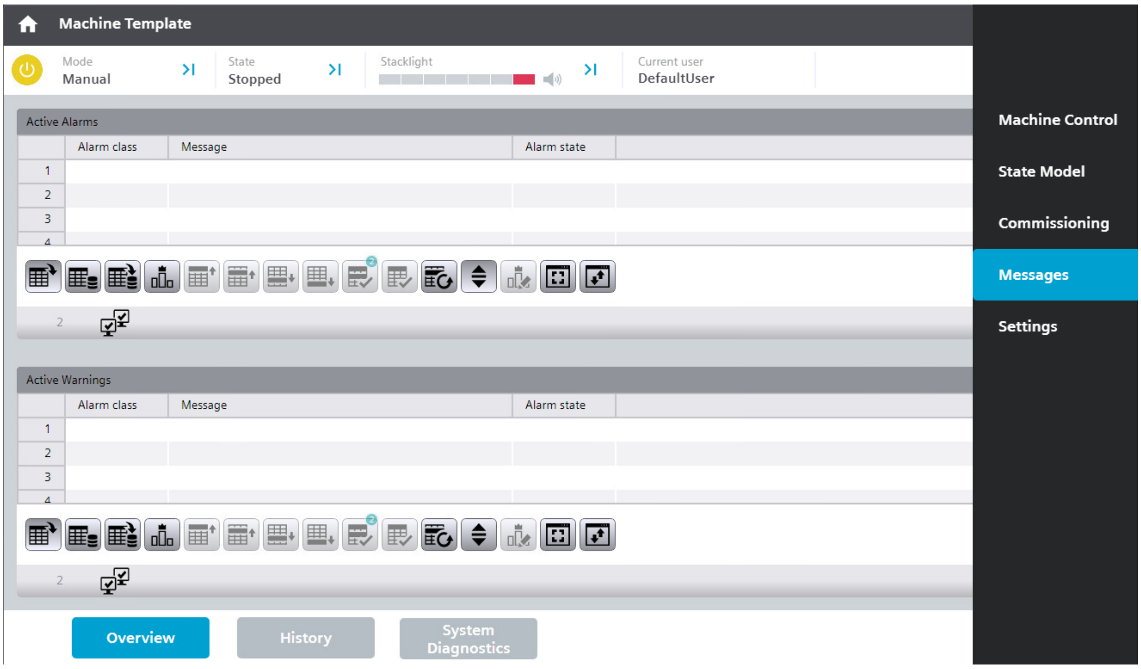
Control Interface Local

2.5.4 Messages

The current alarms and warnings are displayed in the sub-navigation under "Overview".

Under "History" you can see the archived alarms and warnings, and the system diagnosis is also displayed.

Figure 2-38: Message screen



Mode: Manual

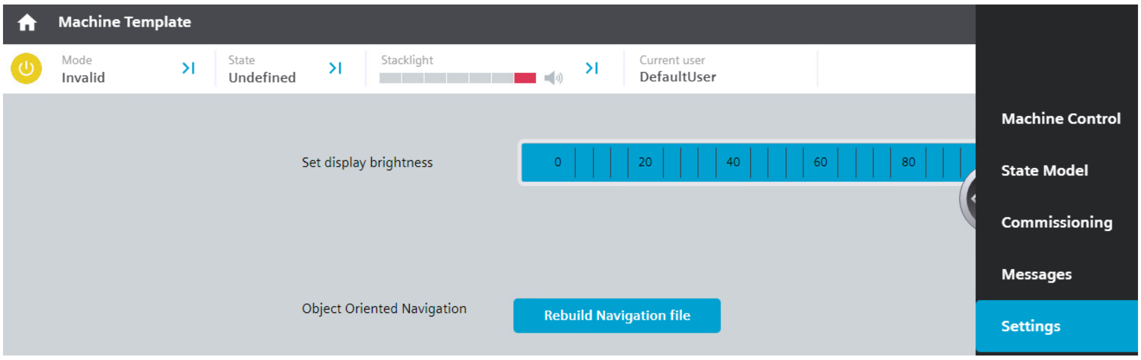
State: Stopped

Current user: DefaultUser

2.5.5 Settings

A rebuild button "Rebuild Navigation file" is implemented, to allow a new creation of the navigation file.

Figure 2-39: Settings screen



Mode: Invalid

State: Undefined

Current user: DefaultUser

Set display brightness

Object Oriented Navigation

3 How to implement in PLC

If additional control modules or equipment modules are needed for the machine project, these modules can be created from the templates. The following chapters describe the options how to do this in the project.

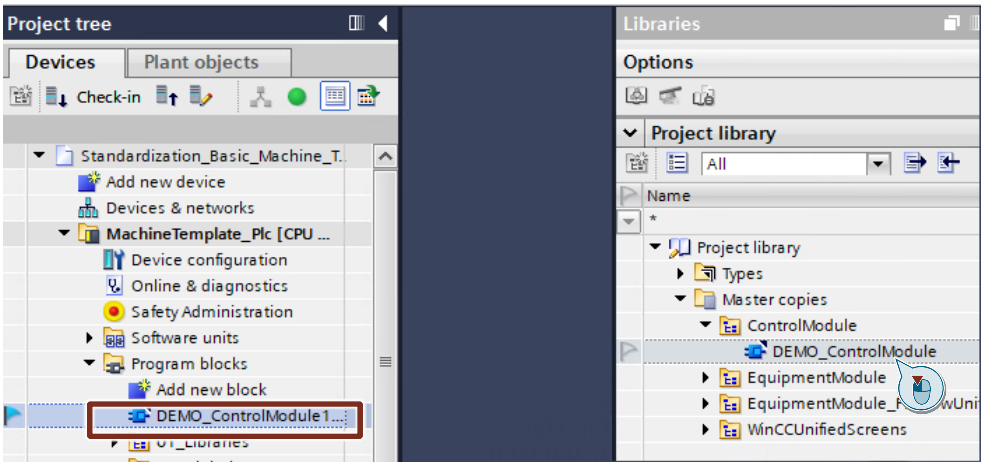
3.1 Implementation in Standard PLC Structure

This chapter describes step by step the implementation of new control modules and equipment modules in the machine project without software units.

3.1.1 Implement a new control module

To implement a new control nodule, drag and drop the file DEMO_ControlModule from the master copies into your program blocks and rename it as desired.

Figure 3-1: Add DEMO_ControlModule to Program blocks

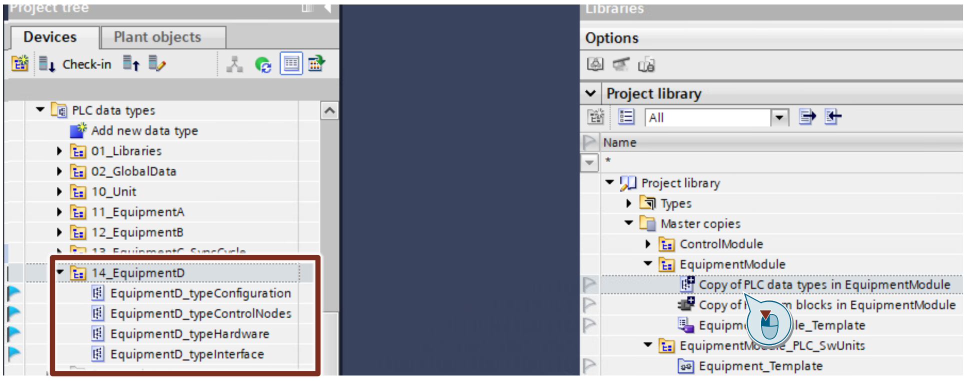


3.1.2 Implement a new equipment module

To implement a new equipment module, proceed as follows:

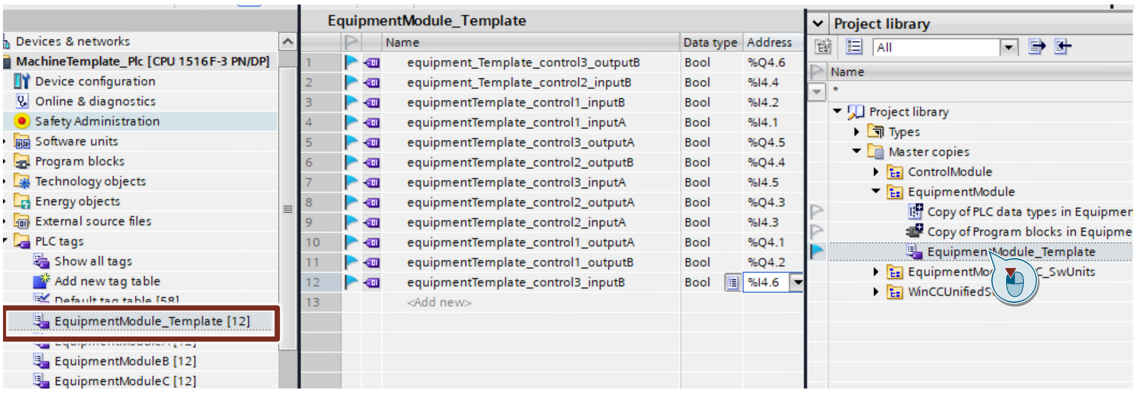
1. Add PLC data types
   Drag and drop the file "Copy of PLC data types in EquipmentModule" from the master copies into your PLC data types and rename it as desired (e.g., EquipmentD).

Figure 3-2: Add PLC data types



2. Add PLC tags
   - Drag and drop the tag table "EquipmentModule_Template" from the master copies into your PLC tags and rename it as desired
   - Insert the desired tags and adjust the input/output addresses

Figure 3-3: Add PLC tags

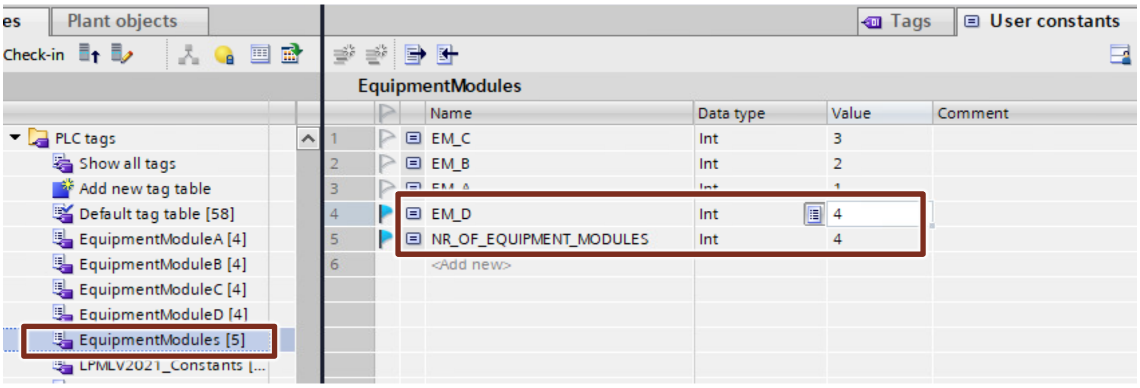


EquipmentModule_Template

	Name	Data type	Address
1	equipment_Template_control3_outputB	Bool	%Q4.6
2	equipment_Template_control2_inputB	Bool	%I4.4
3	equipmentTemplate_control1_inputB	Bool	%I4.2
4	equipmentTemplate_control1_inputA	Bool	%I4.1
5	equipmentTemplate_control3_outputA	Bool	%Q4.5
6	equipmentTemplate_control2_outputB	Bool	%Q4.4
7	equipmentTemplate_control3_inputA	Bool	%I4.5
8	equipmentTemplate_control2_outputA	Bool	%Q4.3
9	equipmentTemplate_control2_inputA	Bool	%I4.3
10	equipmentTemplate_control1_outputA	Bool	%Q4.1
11	equipmentTemplate_control1_outputB	Bool	%Q4.2
12	equipmentTemplate_control3_inputB	Bool	%I4.6
13	<Add new>		

3. Define user constants
   - Open the EquipmentModules tag table, switch to the user constants and increase the number of equipment modules by one.
   - Add a new user constant with the required value (e.g., EM_D)

Figure 3-4: Adjust user constants



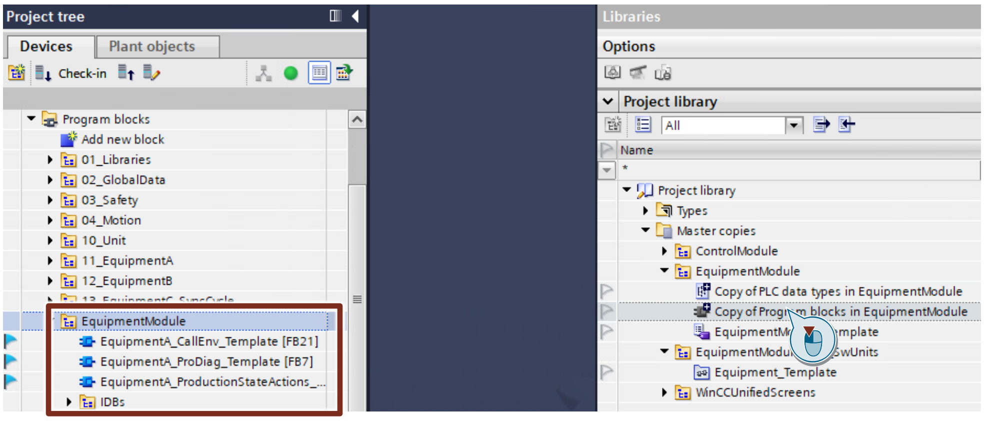
EquipmentModules

	Name	Data type	Value	Comment
1	EM_C	Int	3	
2	EM_B	Int	2	
3	EM_A	Int	1	
4	EM_D	Int	4	
5	NR_OF_EQUIPMENT_MODULES	Int	4	
6	<Add new>			

4. Add new equipment module group in program blocks

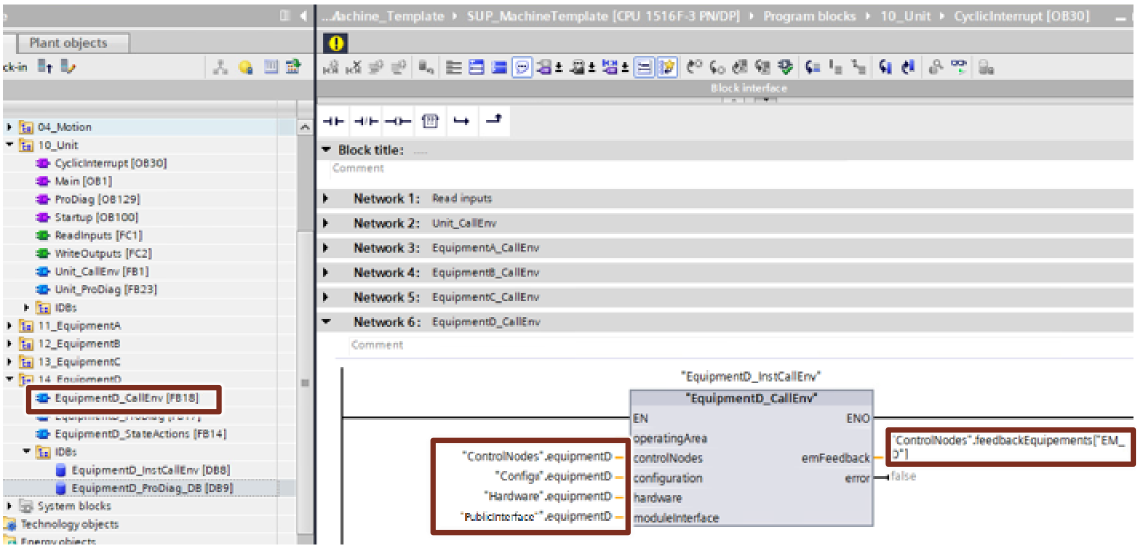
   Drag and drop the file "Copy of Program blocks in EquipmentModule" from the master copies into your program blocks and rename it as desired. (e.g., EquipmentModuleD).

Figure 3-5: Add new equipment module



5. Call equipment module in the user program
   - Drag and drop the Equipment_CallEnv FB into the CyclicInterrupt OB and instantiate it.
   - Add the newly implemented equipment module to the global data DBs (ControlNodes, Config, Hardware, PublicInterface).
   - Connect the InOut parameters in the OB to the newly added equipment module in the global data DBs.
   - The response of the device module must be written to the feedbackEquipements array in the ControlNodes.

Figure 3-6: Call equipment module in the user program



3.1.3 Reuse an existing equipment module

You can copy and paste an existing equipment module with the dependent control modules, or add a new one from the master copies, rename it, and adapt all steps and parameters described in the previous chapter.

3.1.4 Add an additional own equipment module

You can add your own equipment module.

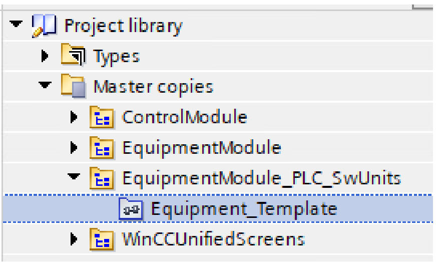
Keep in mind that the structure of the new implemented equipment module must have the same as the structure of the Machine Template to ensure modularization.

3.2 Implementation in PLC with Software Units

The implementation of a new control module in the PLC with software unit is the same as the implementation in the Standard PLC Structure. The only difference is that the desired control module must be added in the corresponding software unit.

To implement a new equipment module in the PLC with software units, drag and drop the "Equipment_Template" from the folder "EquipmentModule_PLC_SwUnits" in the master copies into the PLC and rename it as desired.

Figure 3-7: Equipment module template

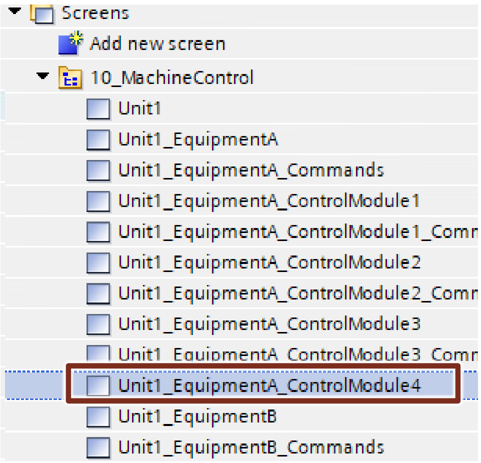


4 How to implement in HMI

Proceed as follows to extend the object-oriented navigation with a new unit / equipment module or control module.

1. Create a new screen e.g., for a new control module (*ControlModule4)

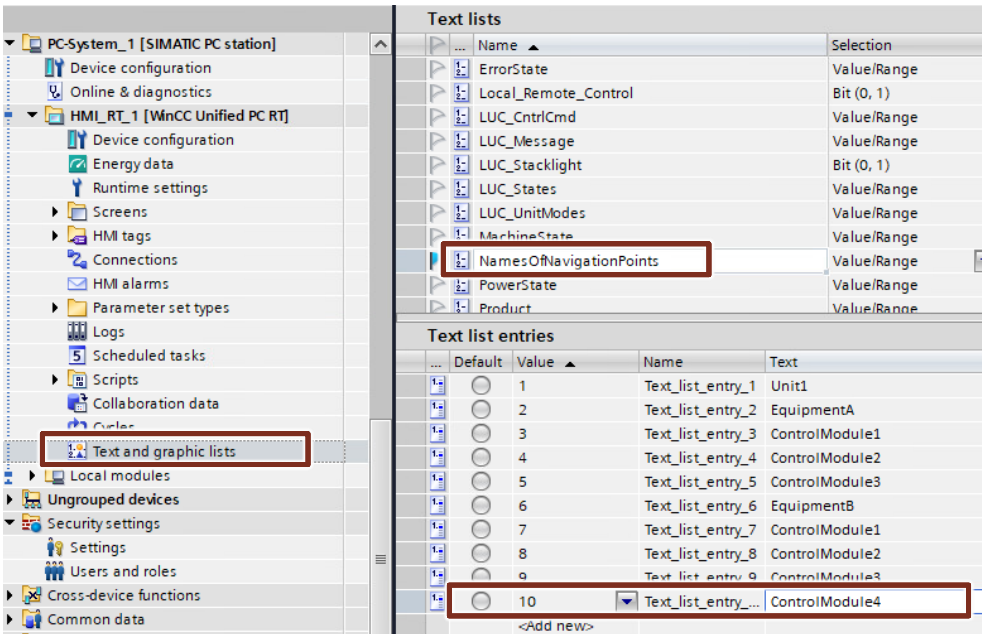
Figure 4-1: Add new control module screen



- Screens
  - Add new screen
  - 10_MachineControl
    - Unit1
    - Unit1_EquipmentA
    - Unit1_EquipmentA_Commands
    - Unit1_EquipmentA_ControlModule1
    - Unit1_EquipmentA_ControlModule1_Comm
    - Unit1_EquipmentA_ControlModule2
    - Unit1_EquipmentA_ControlModule2_Comm
    - Unit1_EquipmentA_ControlModule3
    - Unit1_EquipmentA_ControlModule3_Comm
    - Unit1_EquipmentA_ControlModule4
    - Unit1_EquipmentB
    - Unit1_EquipmentB_Commands

2. Add a new entry in the text and graphic list "NamesOfNavigationPoints"

Figure 4-2: Update text list



Text lists

Name	Selection
ErrorState	Value/Range
Local_Remote_Control	Bit (0, 1)
LUC_CntrlCmd	Value/Range
LUC_Message	Value/Range
LUC_Stacklight	Bit (0, 1)
LUC_States	Value/Range
LUC_UnitModes	Value/Range
MachineState	Value/Range
NamesOfNavigationPoints	Value/Range
PowerState	Value/Range
Product	Value/Range

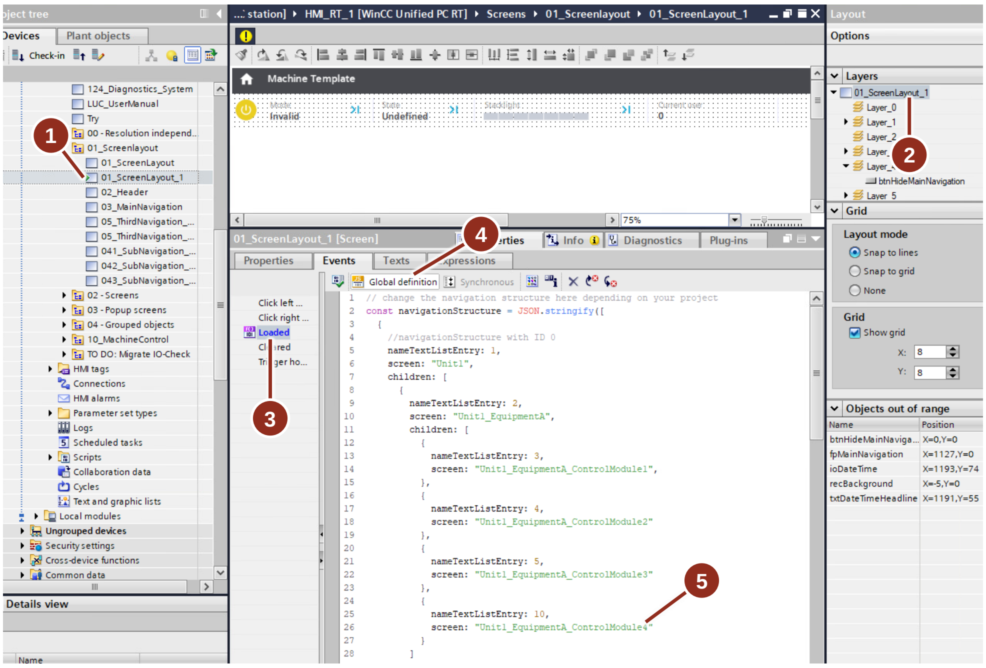
Text list entries

Default	Value	Name	Text
○	1	Text_list_entry_1	Unit1
○	2	Text_list_entry_2	EquipmentA
○	3	Text_list_entry_3	ControlModule1
○	4	Text_list_entry_4	ControlModule2
○	5	Text_list_entry_5	ControlModule3
○	6	Text_list_entry_6	EquipmentB
○	7	Text_list_entry_7	ControlModule1
○	8	Text_list_entry_8	ControlModule2
○	9	Text_list_entry_9	ControlModule3
○	10	Text_list_entry_...	ControlModule4
	<Add new>		

3. To extend the navigation structure with the inserted screen
   1. Open the screen "01_ScreenLayout_1"
   2. Go to Layers and select "01_ScreenLayout_1"
   3. Klick on Events and select Loaded
   4. Open the Global definition

5. Extend the structure with the newly created screen
   screen: screen name of the new added screen
   nameTextListEntry: the ID in the text list

Figure 4-3: Update the script for the navigation structure



```
// change the navigation structure here depending on your project
const navigationStructure = JSON.stringify([
  {
    //navigationStructure with ID 0
    nameTextListEntry: 1,
    screen: "Unit1",
    children: [
      {
        nameTextListEntry: 2,
        screen: "Unit1_EquipmentA",
        children: [
          {
            nameTextListEntry: 3,
            screen: "Unit1_EquipmentA_ControlModule1",
          },
          {
            nameTextListEntry: 4,
            screen: "Unit1_EquipmentA_ControlModule2"
          },
          {
            nameTextListEntry: 5,
            screen: "Unit1_EquipmentA_ControlModule3"
          },
          {
            nameTextListEntry: 10,
            screen: "Unit1_EquipmentA_ControlModule4"
          }
        ]
```

Name	Position
btnHideMainNaviga...	X=0,Y=0
fpMainNavigation	X=1127,Y=0
ioDateTime	X=1193,Y=74
recBackground	X=-5,Y=0
txtDateTimeHeadline	X=1191,Y=55

NOTE

To add a new unit or equipment module, the same procedure is to be followed. The only difference is the position of the desired screen in the script.

5 Annex

5.1 Used Libraries

This chapter provides a general overview of the used libraries.

DEMO ControlModule

Demo is a template for control module and not a library, this can be customized as needed and is included in the master copies as template.

LGF

The Library of General Functions (LGF) contains additional functions for STEP 7 (TIA Portal) which are often required in automation processes.

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

https://support.industry.siemens.com/cs/ww/en/view/49970441

LUC (coming soon)

The Library of Unit Control provides helpful function blocks on machine level. This enables the user to implement a global state and mode manager and extends its functionality with different functions, like OPC UA Methods, stack light controller etc.

6 Appendix

6.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers
– ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:

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

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

support.industry.siemens.com/cs/sc

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:

support.industry.siemens.com/cs/ww/en/sc/2067

6.2 Industry Mall

The Siemens Industry Mall is the platform on which the entire siemens Industry product portfolio is accessible. From the selection of products to the order and the delivery tracking, the Industry Mall enables the complete purchasing processing – directly and independently of time and location:
mall.industry.siemens.com

6.3 Links and literature

Table 6-1

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109817223
\3\	Programming Guidelines and Programming Style guide for SIMATIC S7-1200 and S7-1500 https://support.industry.siemens.com/cs/ww/en/view/81318674
\4\	Guideline on Library Handling in Tia Portal https://support.industry.siemens.com/cs/ww/en/view/109747503
\5\	Libraries in the TIA Portal https://support.industry.siemens.com/cs/ww/en/view/109738702

6.4 Change documentation

Table 6-2

Version	Date	Modifications
V1.0.0	03/2023	First version
V1.0.1	03/2023	Revised and improved text