Selmo Solution

For
software
that never
lets you
down!

Selmo solution

Selmo in Use

1 Selmo in Use

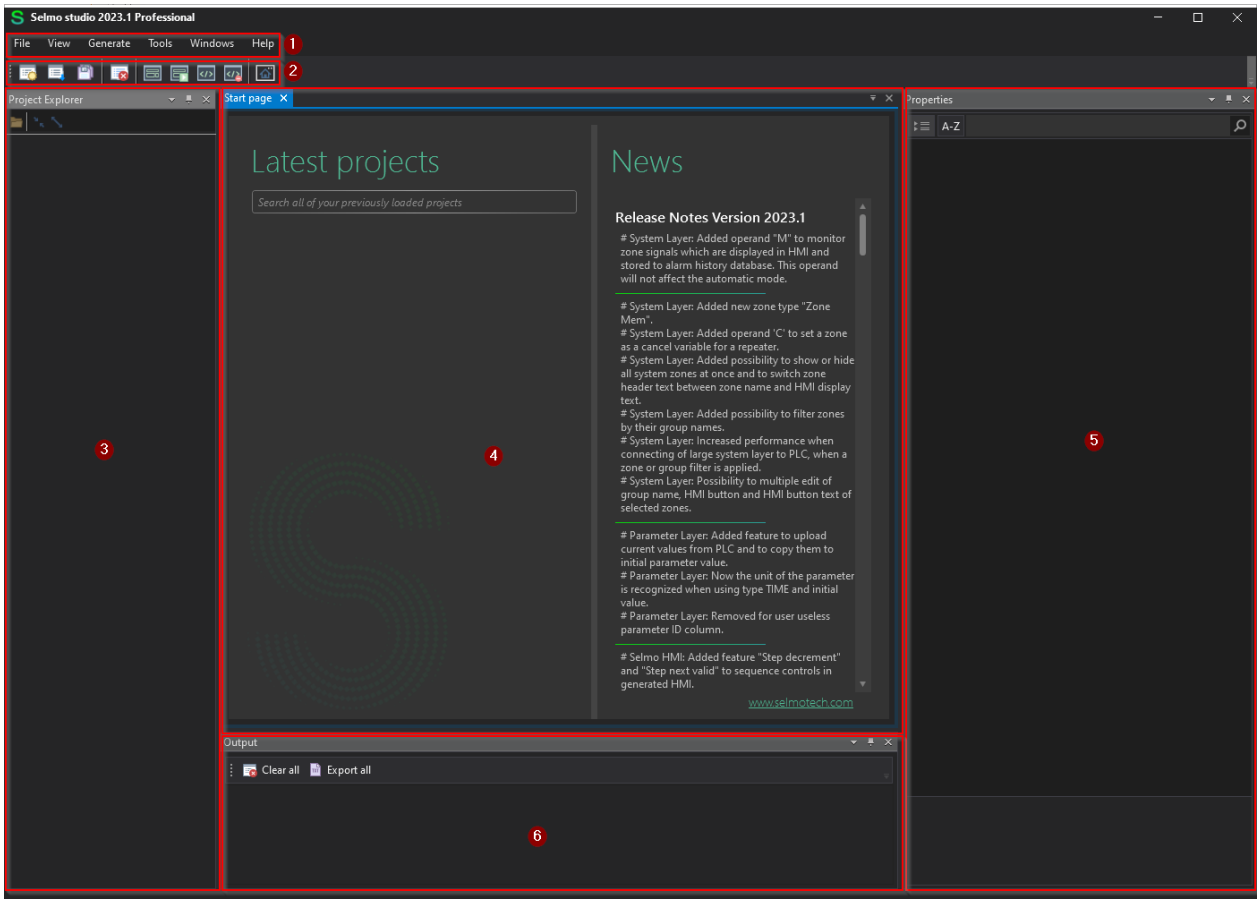
Selmo in Use describes every step in Selmo Studio, from project creation to application in operation. The focus is on the use of Selmo Studio, how stepping circuits and signals are modeled and defined here.

The documentation refers to Selmo Studio version 2023.1 SP1.

Studio

A detailed description of all features can be found in the corresponding chapter Selmo Studio. In the following, only the single steps are explained, which are essential to create a project. The start page contains several controls, which are listed below.

1. Menu
2. Quick Access Bar
3. Project Explorer
4. Editor window
5. Properties window
6. Output window



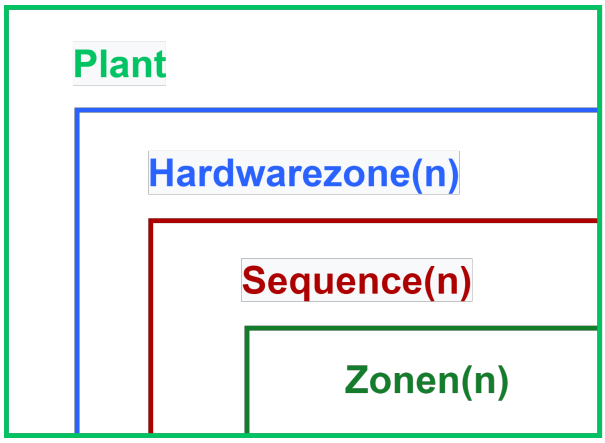
Studio

1.1 Structure plant

The plant structure describes the hierarchical organization of a technical plant into logical and functional units. The plant structure can vary depending on the application, but typically it consists of three levels: Plant, Hardware Zone and Sequence. The Plant level is the top level of the plant structure and represents the entire plant. The Plant level is used to clearly represent and organize the plant and to assign global parameters and functions. The hardware zone level is the middle level of the plant structure and divides the plant level into smaller sections, each containing a specific hardware component or module. The hardware zone level is used to define and configure hardware interfaces and properties and to manage resources and security aspects. Each hardware zone has independent automatic and manual controls. The Sequence level is the lowest level of the system structure and describes the sequences and processes within a hardware zone. The sequence level is used for programming and controlling the logical functions and algorithms as well as for monitoring and diagnosing the operating status. The sequence level is divided into different layers.

Structure of the Selmo project

The system structure is always the same:



This consists of different layers. The top level is called the Plant and represents the overall system. The Plant always contains at least one sublevel, the hardware zone(s). The hardware zone can contain one or more Sequences. A Sequence is always assigned to only one hardware zone. A Sequence is a subsystem that defines a logical process description. This consists of a logical process description in the form of a step chain (Logic Layer), the definition of states by zones (System Layer), the interlocks in manual mode (MXIC) and constantly monitored zones (CMZ).

Plant

In industry, the term "Plant" usually refers to a facility or equipment used to manufacture or process products or raw materials. For example, this may be a factory, plant, power station, or refinery. A Plant often includes multiple units that operate in a specific sequence to produce the desired product or process the raw material. These may include manufacturing equipment, machinery, chemical reactors, storage and supply facilities, or control systems. Plants can be very large and complex often requiring specialized knowledge and skills to operate and maintain. Therefore, multiple professionals such as engineers, technicians, and laborers typically work together to operate a Plant and achieve optimal performance.

Hardware Zone (HWZ)

A hardware zone (HWZ for short) is a specific area of a machine or cell area with separate automatic and manual controls. Each hardware zone has its own specific overview. The size of a hardware zone depends on the configuration of the machine or cell and can contain and control one or more step sequences.

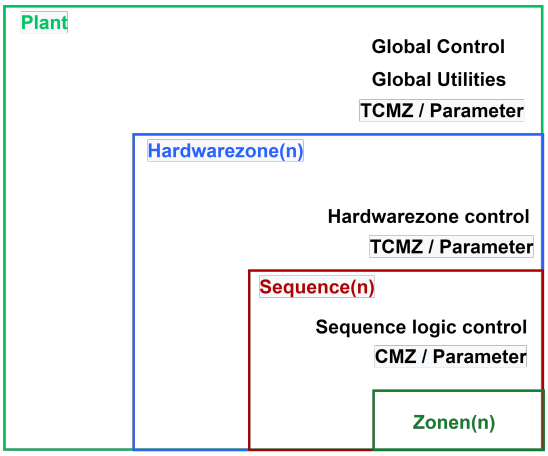
Sequence

A sequence control is a type of control system that executes a sequence of predefined steps or states. An example of a sequence controller is a step controller, which consists of

a series of memory elements, each representing a step in the sequence. A step controller can be controlled by external input signals or internal logic. A sequence controller can be used to automate or synchronize complex processes, such as manufacturing in a factory or communications in a network.

Elements of Plant, Hardware Zone and Sequence

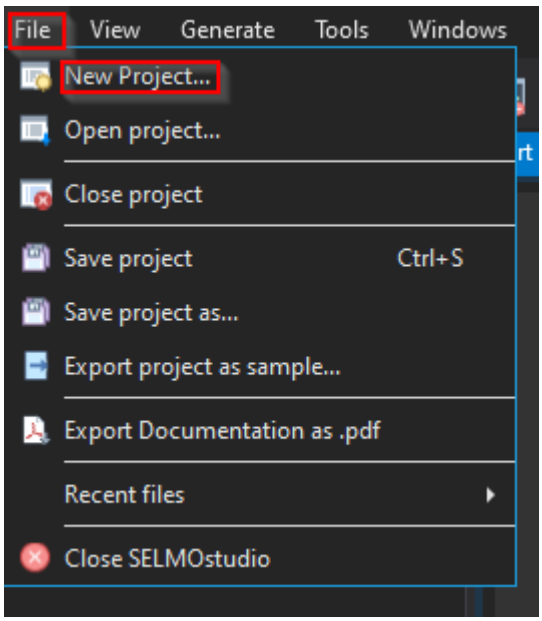
Each level of a Selmo project has certain properties and functions that apply in the effective area. The elements and its effective area are described in the following figure:



1.1.1 Embed step chain (Sequence)

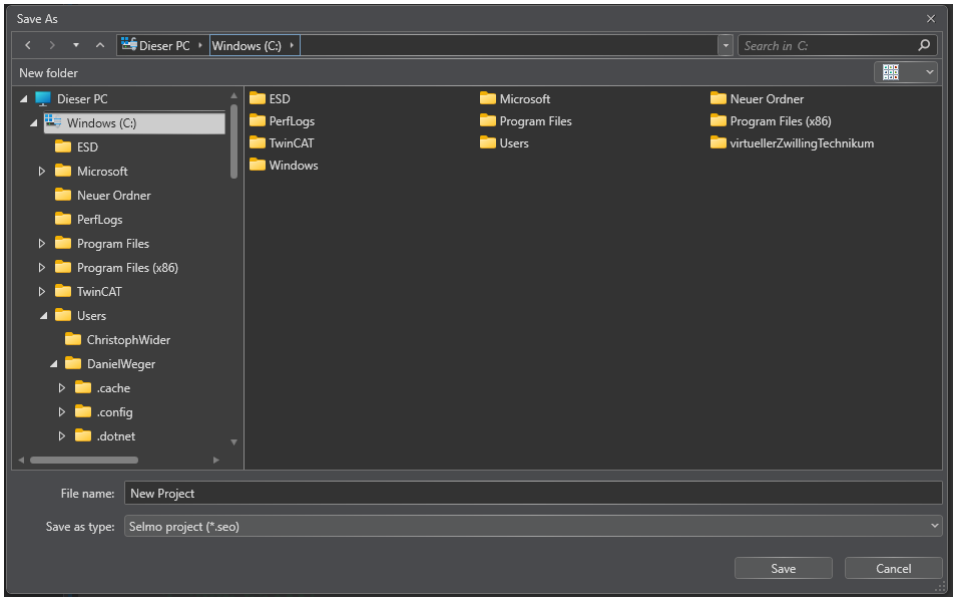
To embed a step chain in a plant, you must first create the plant. To do this, select the "New Project" option in the "File" context menu and enter a name.

A new Project can be created under File, New Project.



Studio

Select project path and assign a unique name. Each project is created as a Selmo pro-ject (*.seo).



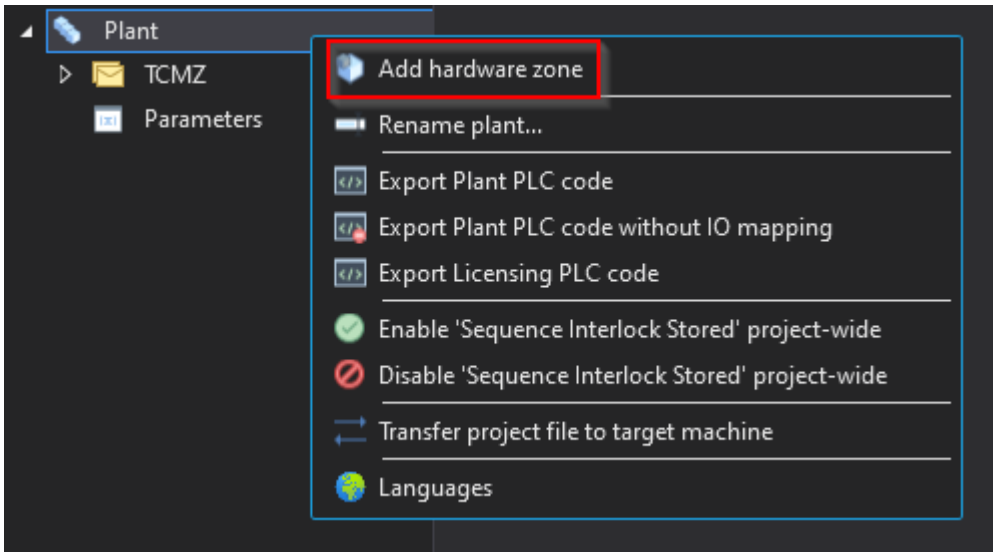
Studio

Next, you need to define the hardware zone where the step chain will be executed.

A hardware zone (HWZ for short) is a specific area of a machine or cell area with separate automatic and manual controls. Each hardware zone has its own specific overview. The size of a hardware zone depends on the configuration of the machine or cell and can contain and control one or more step sequences.

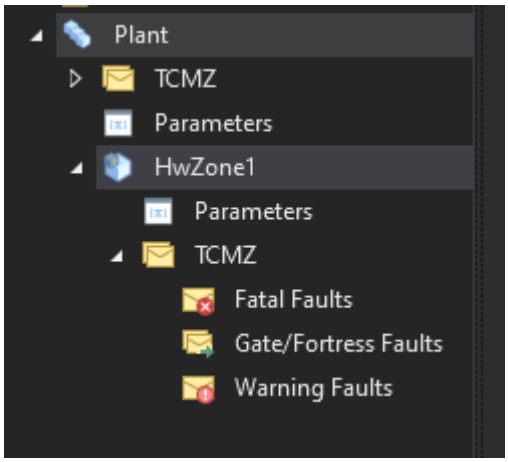
In Selmo Studio, right-clicking on Plant, Add Hardware Zone will add a new machine zone.

Any number of hardware zones can be added to a Plant, but it depends on the performance of the hardware (PLC).



Studio

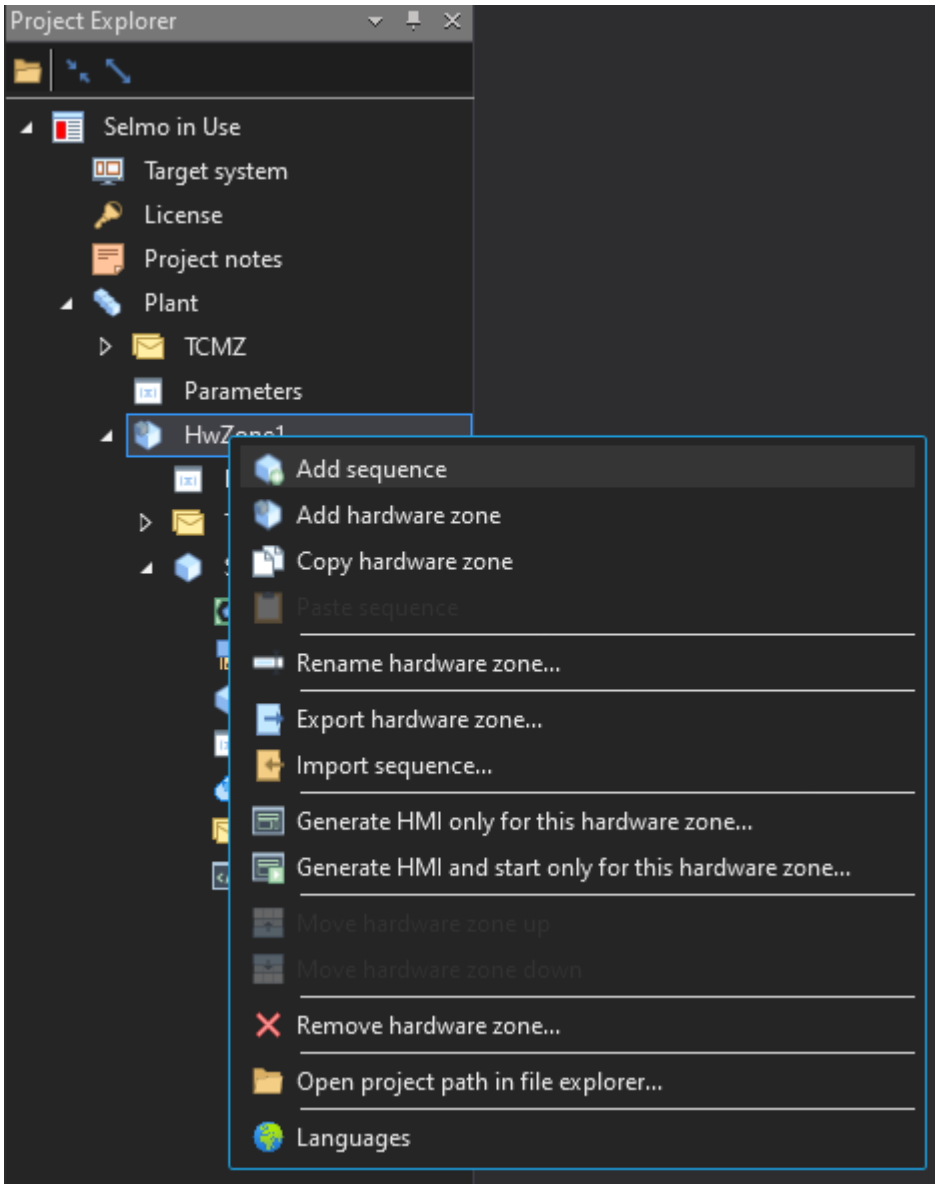
This creates the following structure:



Studio

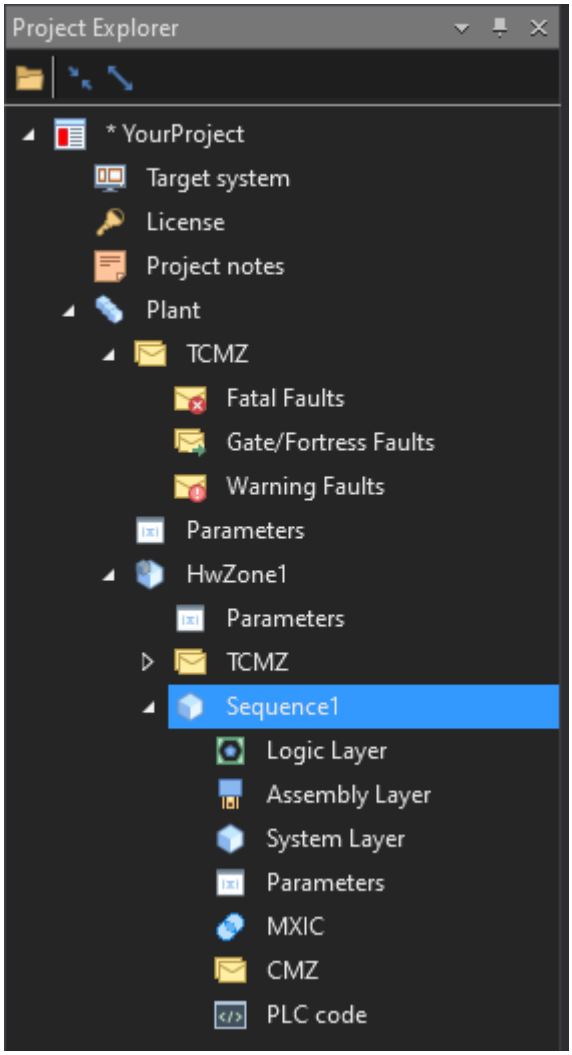
Finally, you must add the Sequence, which contains the individual steps of the sequence.

A step chain is called a Sequence and represents a process flow in single steps. You add a Sequence by right-clicking on a HWZ. It is also possible to import or copy a sequence. Each sequence can only be assigned to one hardware zone, but you can assign several sequences to one hardware zone.



Studio

The structure of a Sequence is always the same and consists of several layers as shown in the following picture.



Studio

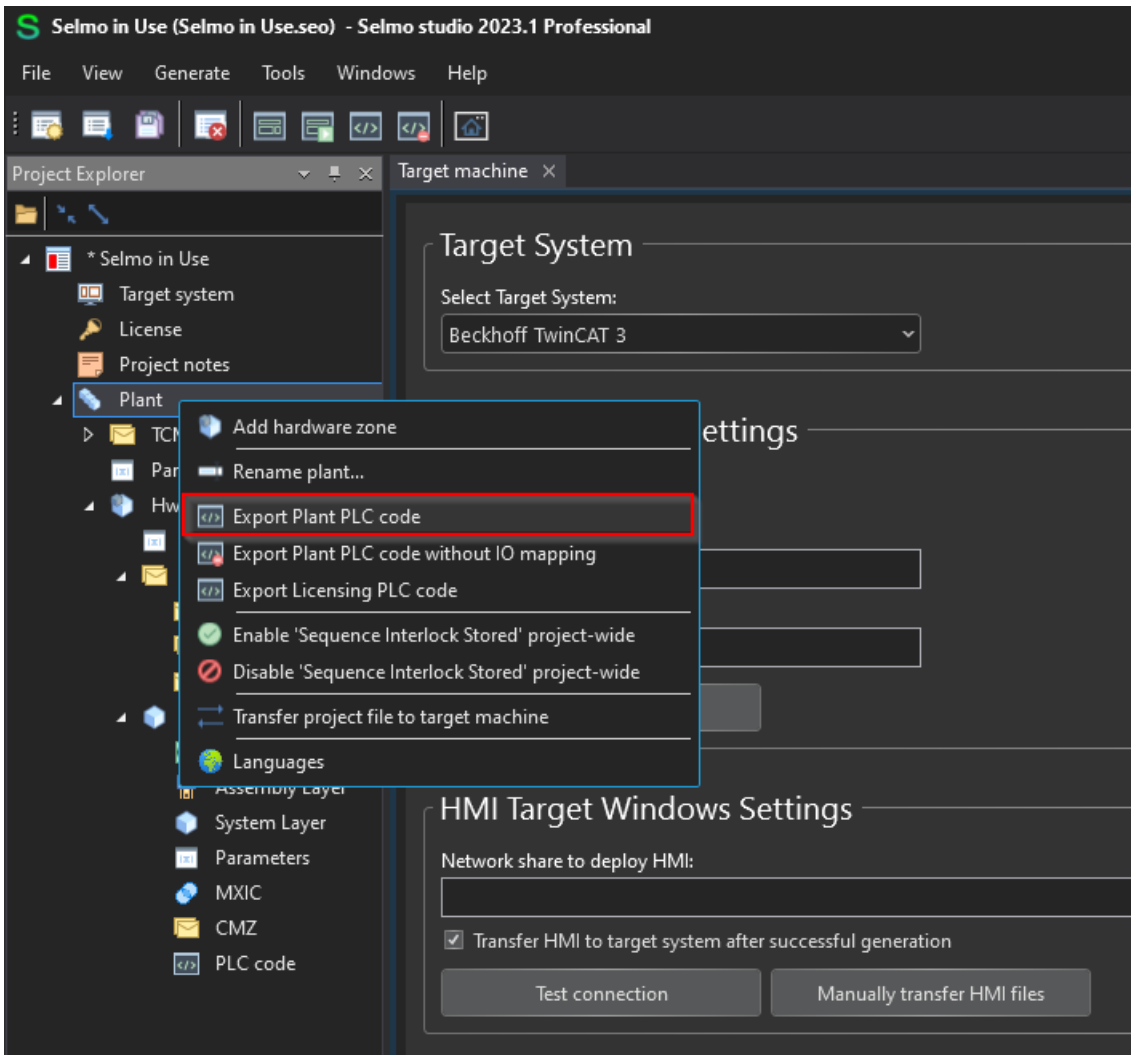
1.1.2 What happens in the PLC/HMI

The Selmo Solution is a software solution that allowsto create and control processes of automated plants with a graphical user interface. To convert the Selmo Solution into PLC code that the plant can execute, you need to perform the following steps:

- Open the Selmo Solution in the Selmo Studio application.
- Select the Plant you want to export by clicking on it in the Project Explorer.
- Right-click on the Plant and select the "Export Plant PLC Code" option from the context menu.
- Specify a name and location for the exported file and click "Save".
- The Selmo Solution will now be converted to PLC code and saved to the specified file.

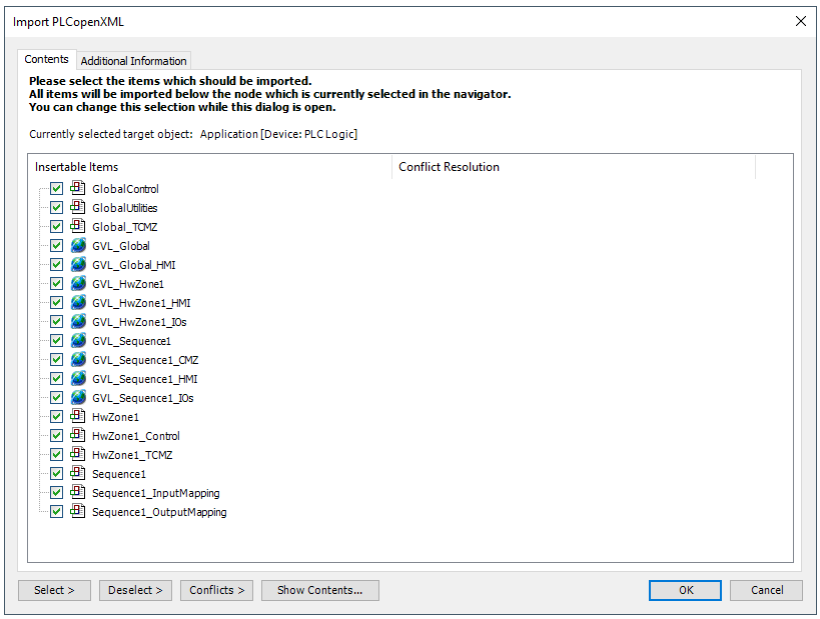
Hint:

With Selmo, the possibility has been created to generate a PLCopen-compliant XML by using an export function and importing it into the selected controller. This means that the program, libraries, and projects created with IEC 61131-3 can be saved in a standardized XML format that can be read and edited by various software tools. The PLCopen XML format is part of the IEC 61131 series of standards and is to as IEC 61131-10. It provides an open interface for exchanging information between development environments and other platforms. The PLCopen XML format is described in the PLCopen technical documentation.



Studio

You can then open the exported file in your preferred PLC programming environment and transfer it to the plant.



PLC

This PLC program contains the following elements that are necessary for control and diagnostics:

The **Plant** elements are the basic building blocks for configuring and controlling the plant. They have the following properties:

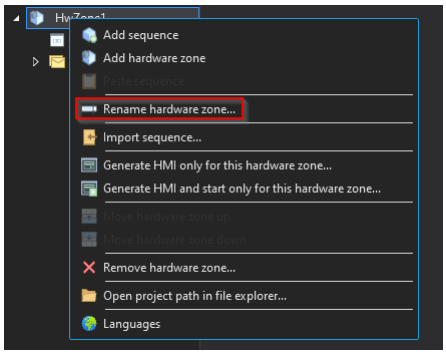
- They act **GLOBALLY**, that is, they affect all hardware zones and all licensed functions of the plant.
- **Global Control** is an element that enables general calling of hardware zones and licensing.
- **Global Utilities** is an element that provides some functional elements useful for diagnostics and maintenance of the installation. For example: Lamp Test, Global Release, Global Reset, etc.
- The plant has a **Global TCMZ** (Total Constantly Monitored Zone) system that constantly monitors all the important parameters of the plant, such as the compressed air or lubricant supply. The Global TCMZ system acts on all hardware zones of the plant and can interrupt the automation in any hardware zone in case of fault detection. This increases the safety and efficiency of the plant.
- **GVL_Global_*** is an important resource for programming plants with different elements. It allows us to define and manage globally the variables needed for the function and control of the elements. This allows the elements to communicate with each other and exchange data. The GVL_* contains the names, types, and values of the variables assigned to the elements.

The ***hardware zones*** are necessary to control and monitor the individual machine areas. They make it possible to switch between manual and automatic operating modes and to execute the defined sequences.

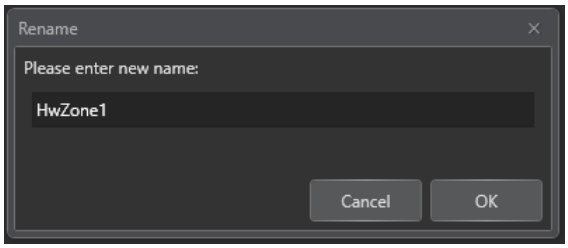
- **<HwZone1>** is an element that calls and controls all sequences in the hardware zone.
- **<HwZone1>_Control** is an element that provides additional functions such as mode selection, EOC mode or status information of the Sequences.
- **<HwZone1>TCMZ** is an element that constantly monitors the hardware zone and interrupts the automatic in case of an error.
- **GVL_<HwZone1>_*** is an important resource for programming machine zones with different elements. It makes it possible to globally define and manage the variables needed for the function and control of the elements. This allows the elements to communicate with each other and exchange data. The GVL_* contains the names, types and values of the variables associated with the elements.

Attention:

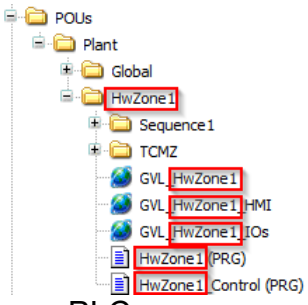
Renaming the hardware zone changes the name<HwZone1> of the entire structure including the Global Variable List, PLC program and HMI (see PLC code).



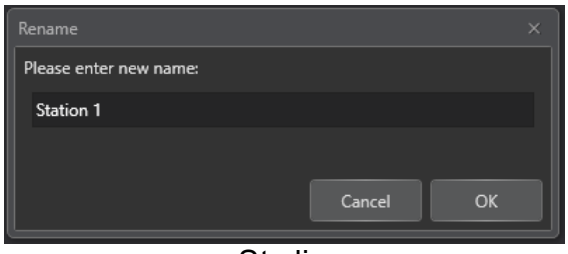
Studio



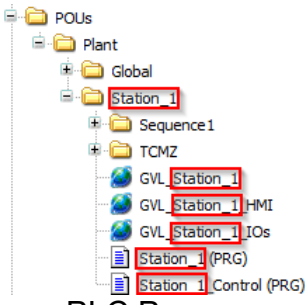
Studio



PLC program



Studio



PLC Program

The ***Sequence*** elements describe the sequences and processes within a hardware zone. Any number of Sequences can be defined in a hardware zone and always consist of the following elements:

- **<Sequence1>** is the basis of a logical process sequence and is divided into the following sections:

  Standard Beginning

  Sequence Logic Control

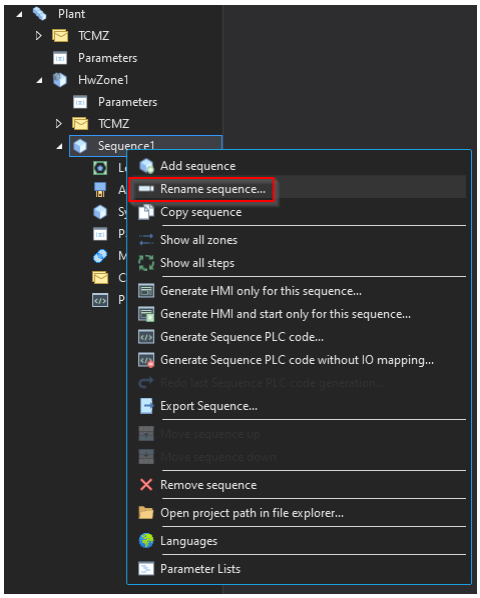
  CMZ

  Standard End

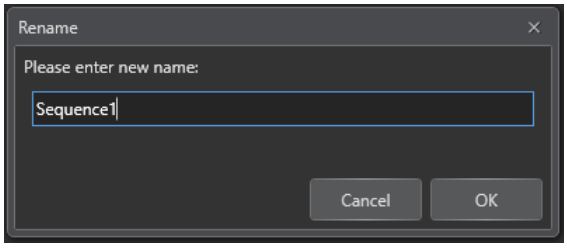
- In the **<Sequence1>_InputMapping**, input signals of the real world are assigned to the sequence.
- In the **<Sequence1>_OutputMapping**, output signals are assigned to the real world of the sequence.
- **GVL_<Sequence1>*** is an important resource for programming logical process sequences with various elements. It allows us to define and manage globally the variables needed for the function and control of the elements. This allows the elements to communicate with each other and exchange data. The GVL_* contains the names, types and values of the variables assigned to the elements.

Attention:

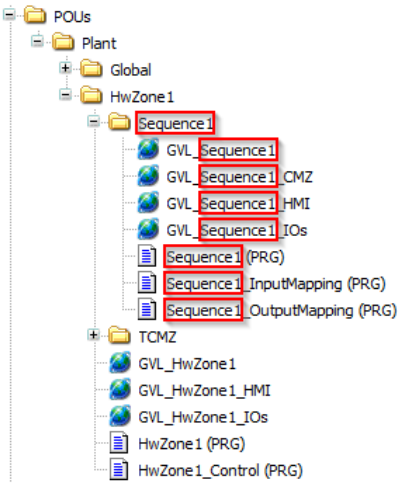
Renaming the Sequence changes the name<Sequence1> of the whole structure including the global variable list, PLC program and HMI (see PLC code).



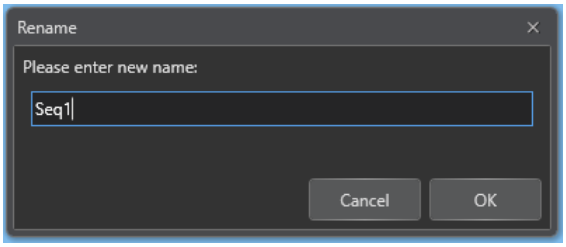
Studio



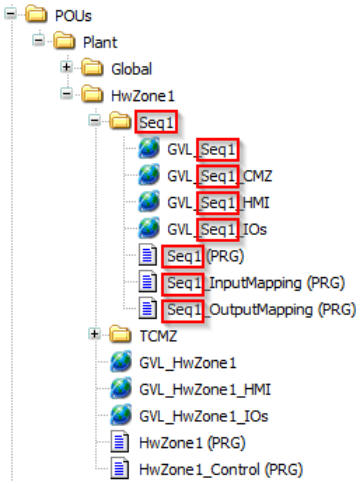
Studio



PLC program



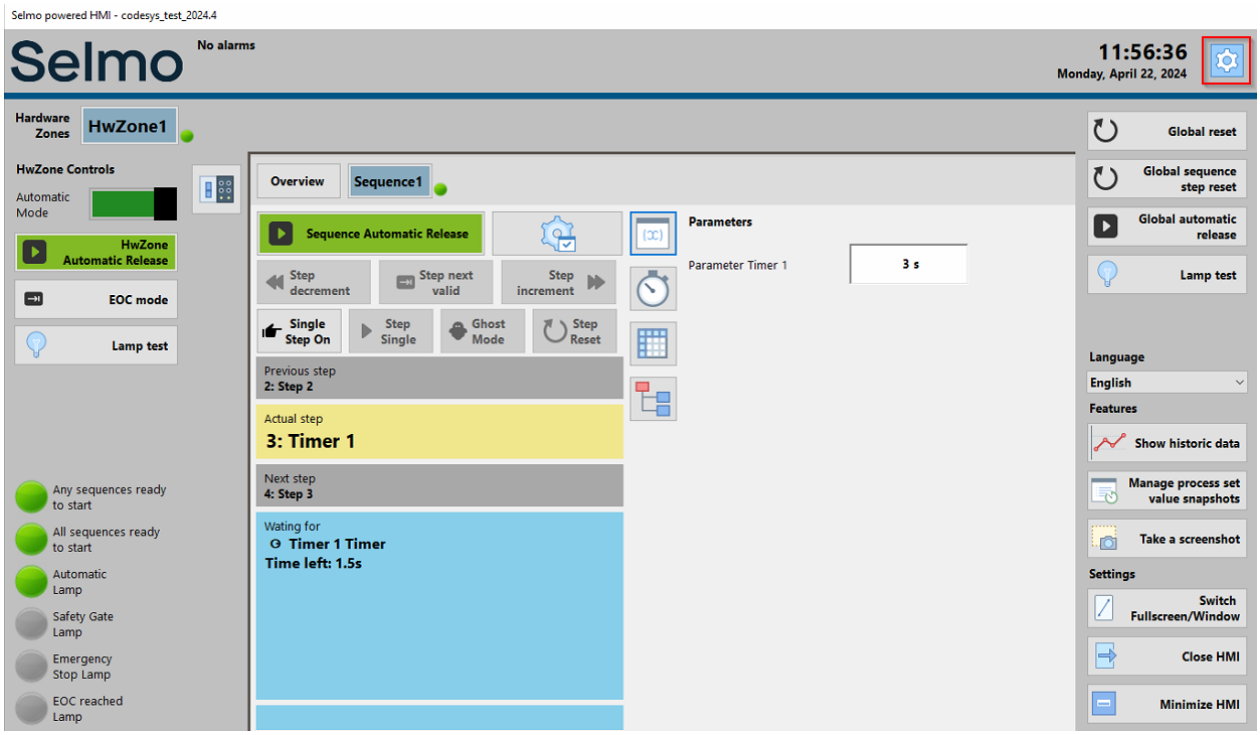
Studio



PLC program

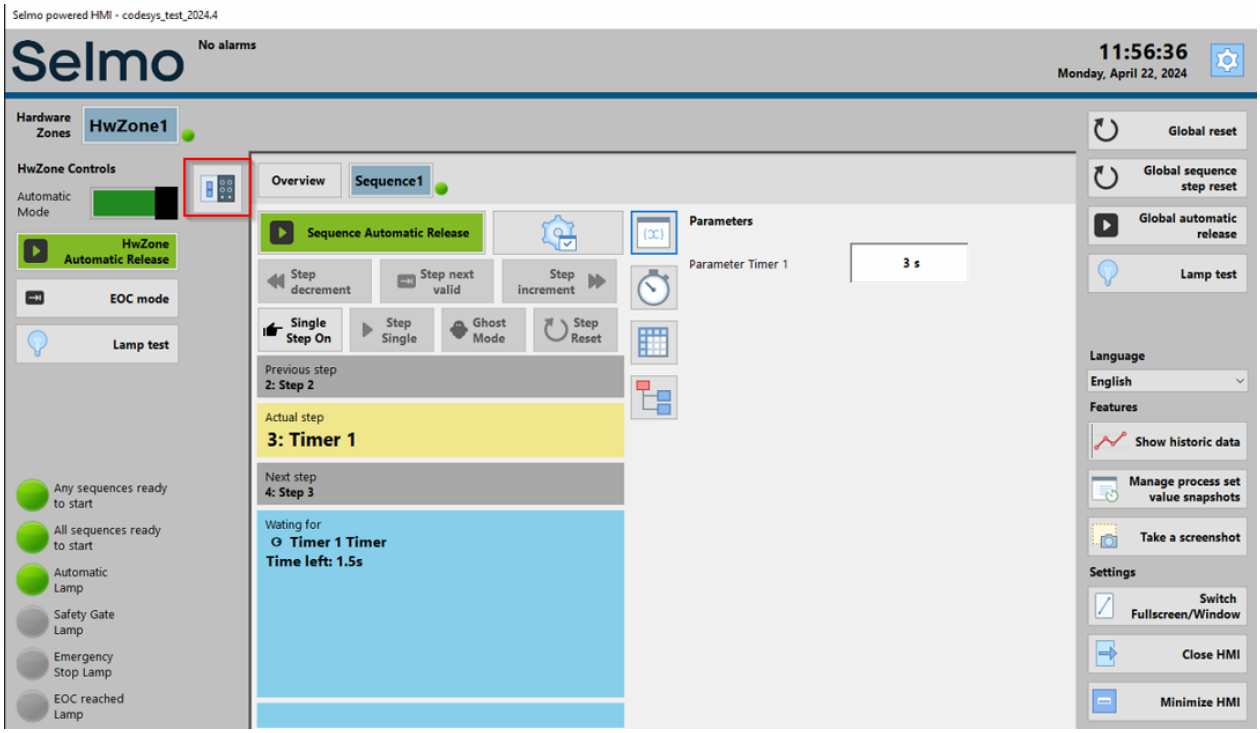
After exporting the PLC code of the entire Plant, an HMI can be generated which has the same structure as the PLC program:

- Plant: This is the top level representing the entire plant. Here the user can get an overview of the plant's status and performance.



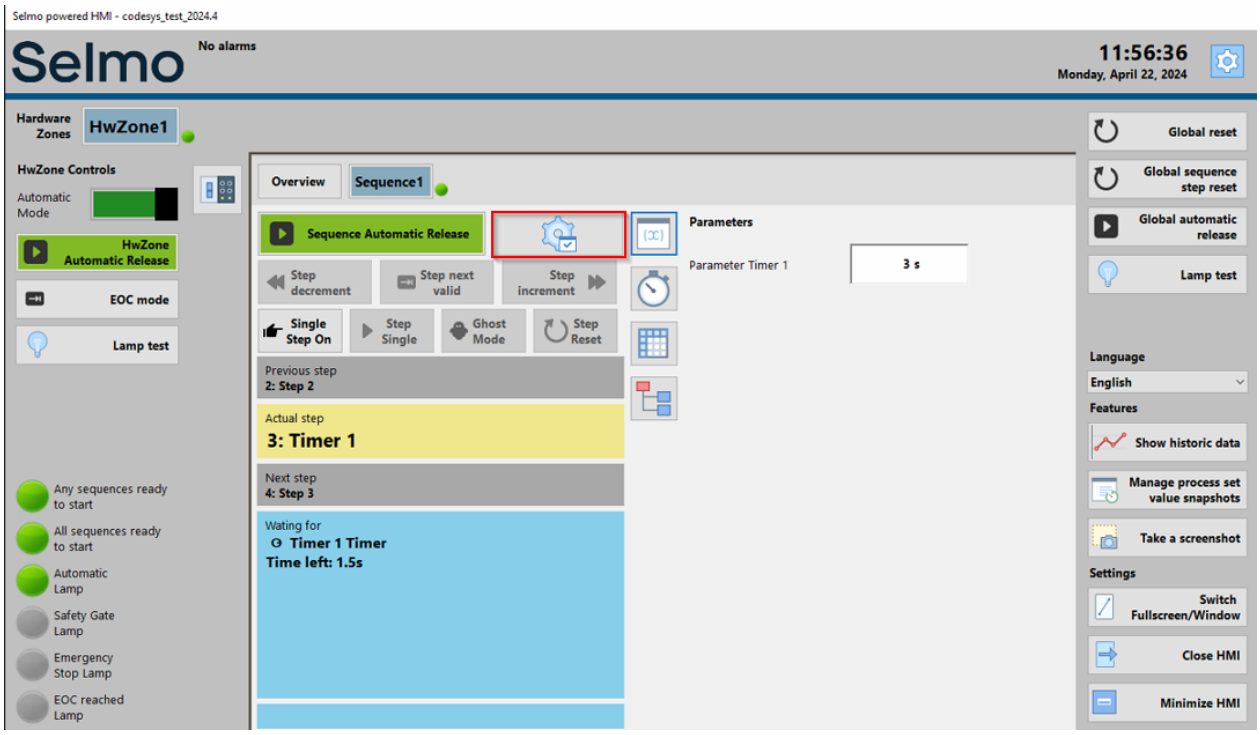
HMI

- Hardware Zones: These are the lower levels representing the different areas of the plant. Here the user can control and monitor the individual components and devices of the plant.



HMI

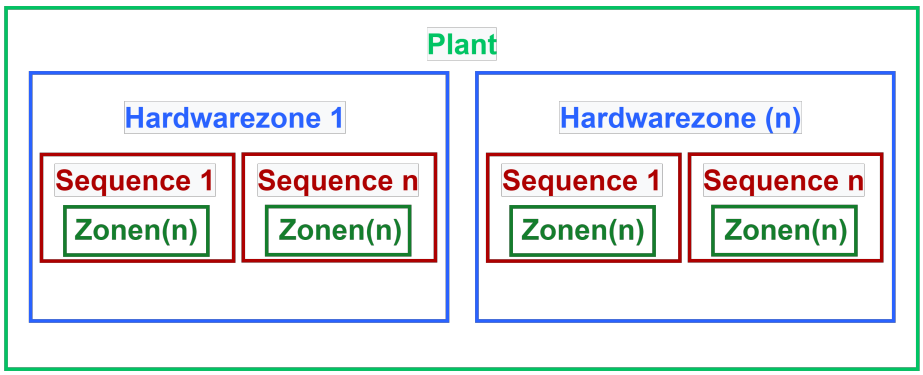
- Sequence: This is the lowest level, which describes the sequences and processes of the plant. Here the user can start, stop, pause, and modify the sequences required for the production or operation of the plant.



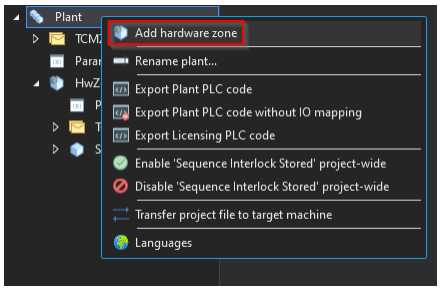
HMI

1.1.3 Scenario multiple HWZ

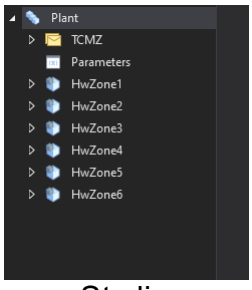
A Plant can contain several hardware zones. These hardware zones can contain several sequences. Signals can be connected to zones in these sequences. Accordingly, a plant can have the following structure:



When multiple hardware zones are added, the structure of the Plant expands to include the elements of the previously mentioned hardware zones. There is no predetermined upper limit, but the performance of the hardware in use influences the expandability.

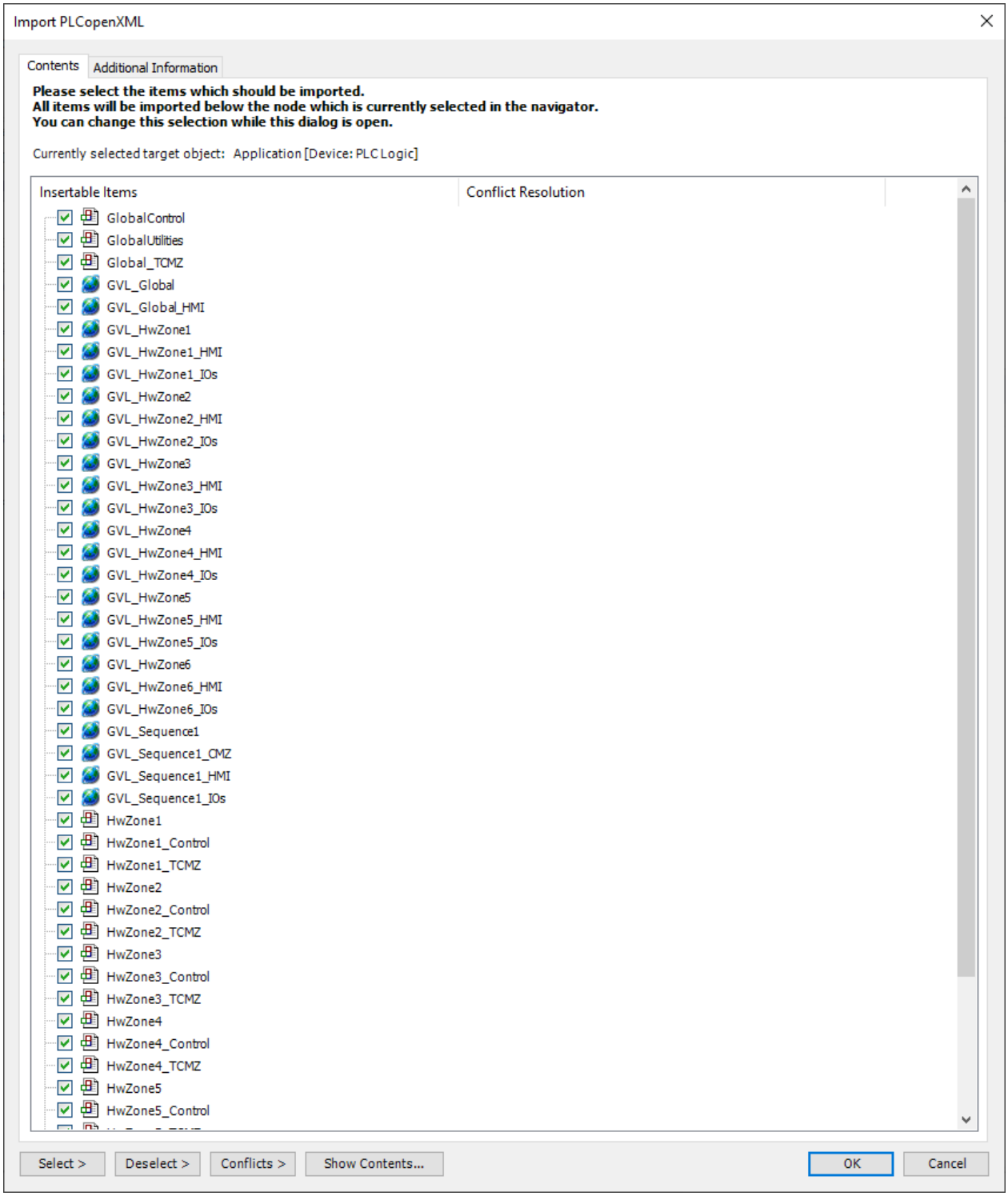


Studio



Studio

The following PLC program elements are created by re-export:



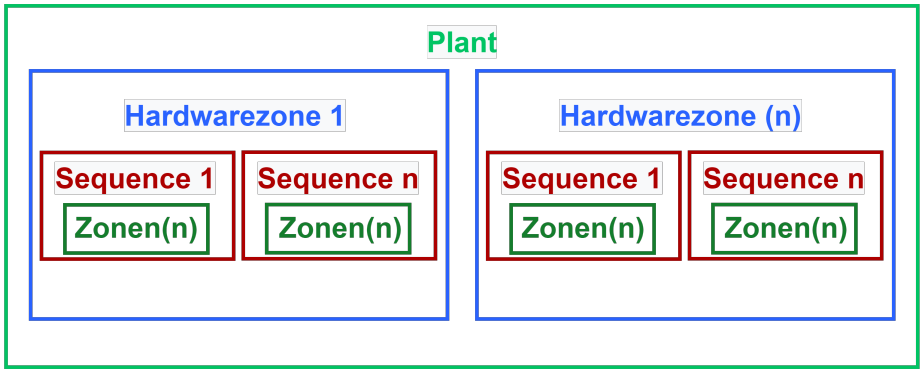
PLC

The following HMI elements are created:

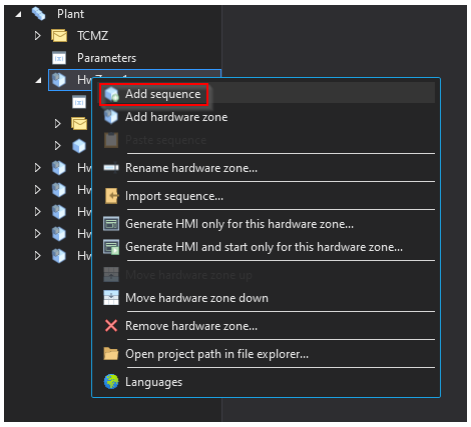
HMI

1.1.4 Scenario multiple sequences

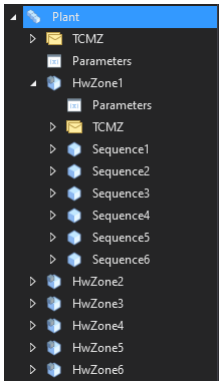
Hardware zones can contain several sequences. In these Sequences signals can be connected with zones. Accordingly, a plant can have the following structure:



If several Sequences are added, the structure of the Plant expands by the elements of the previously mentioned Sequence. There is no predetermined upper limit, but the performance of the hardware in use influences the expandability.

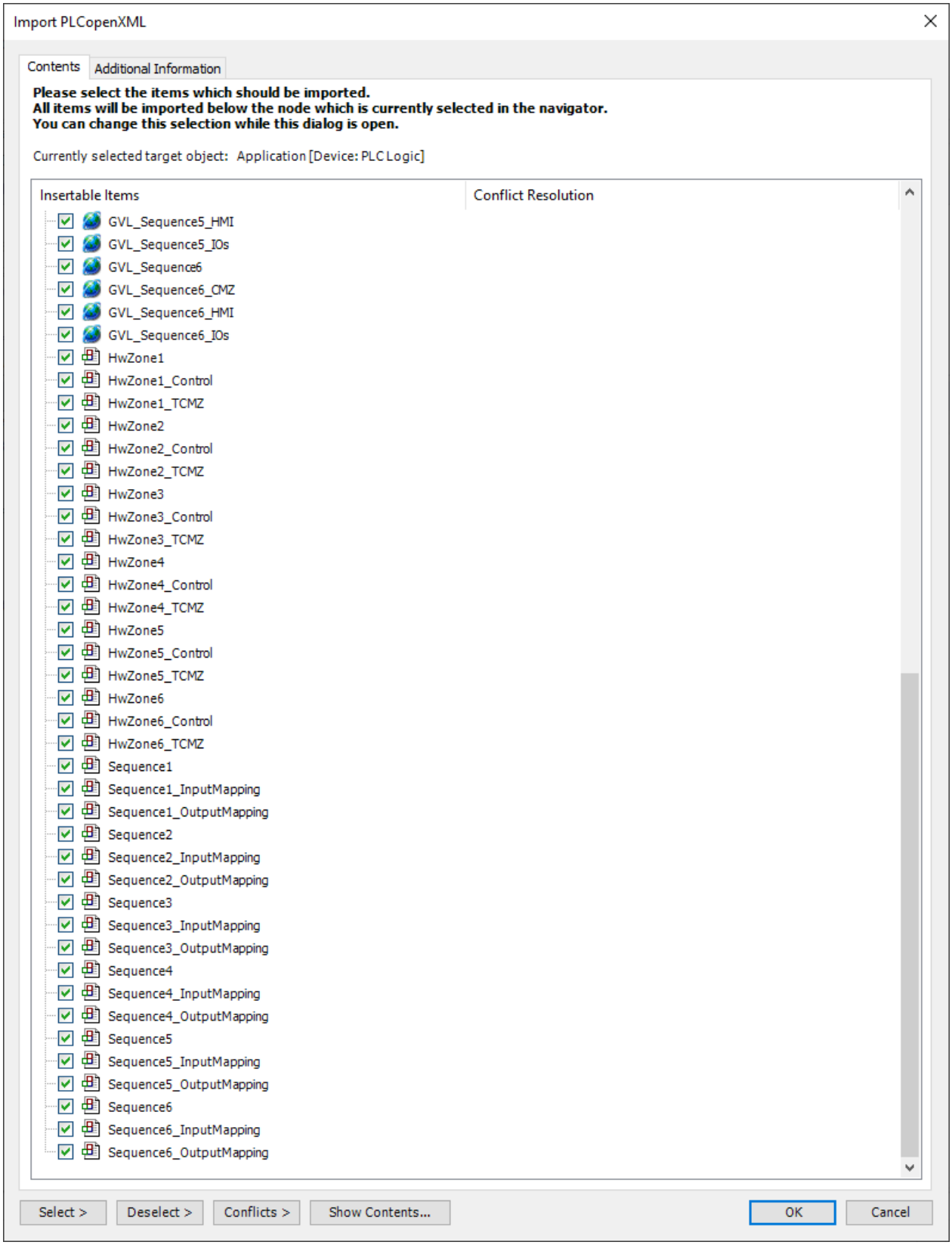


Studio



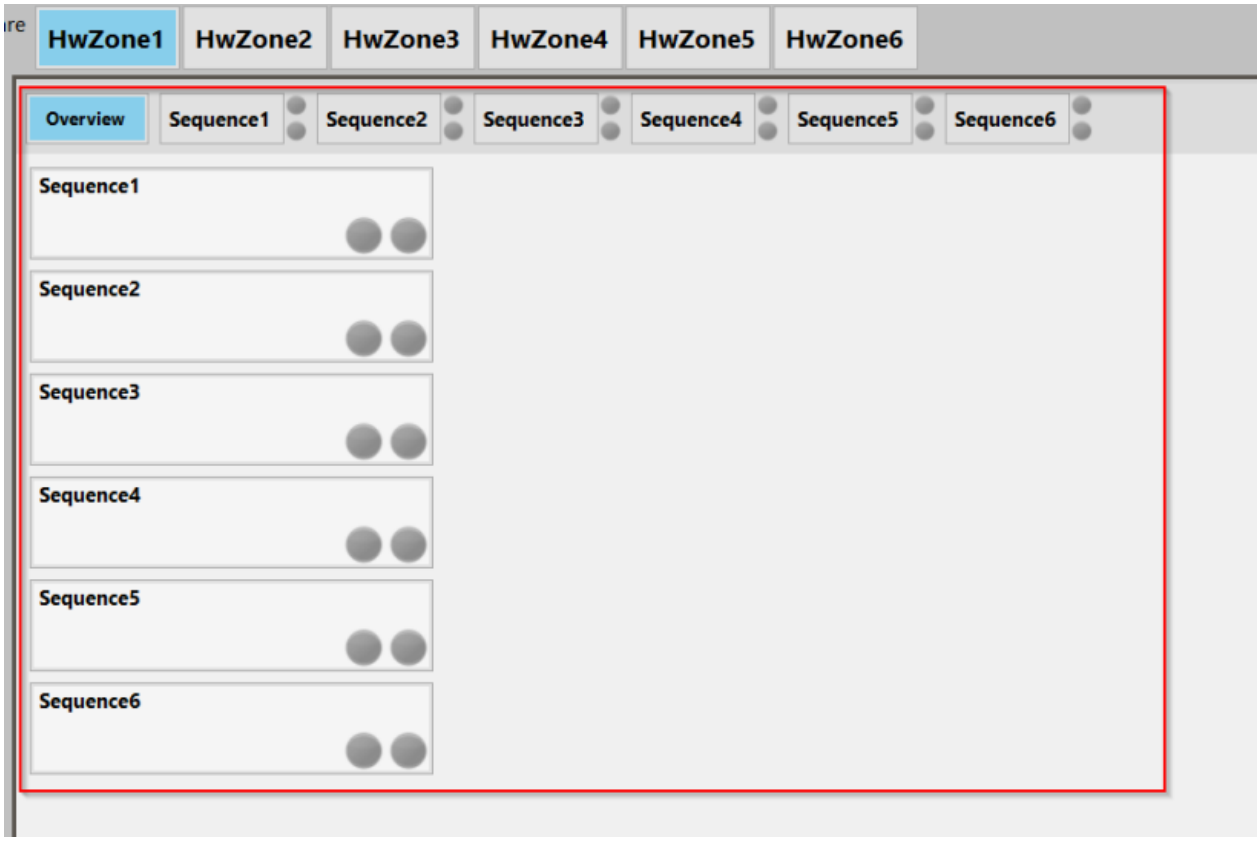
Studio

The following PLC program elements are created by re-export:



PLC

The following HMI elements are created:



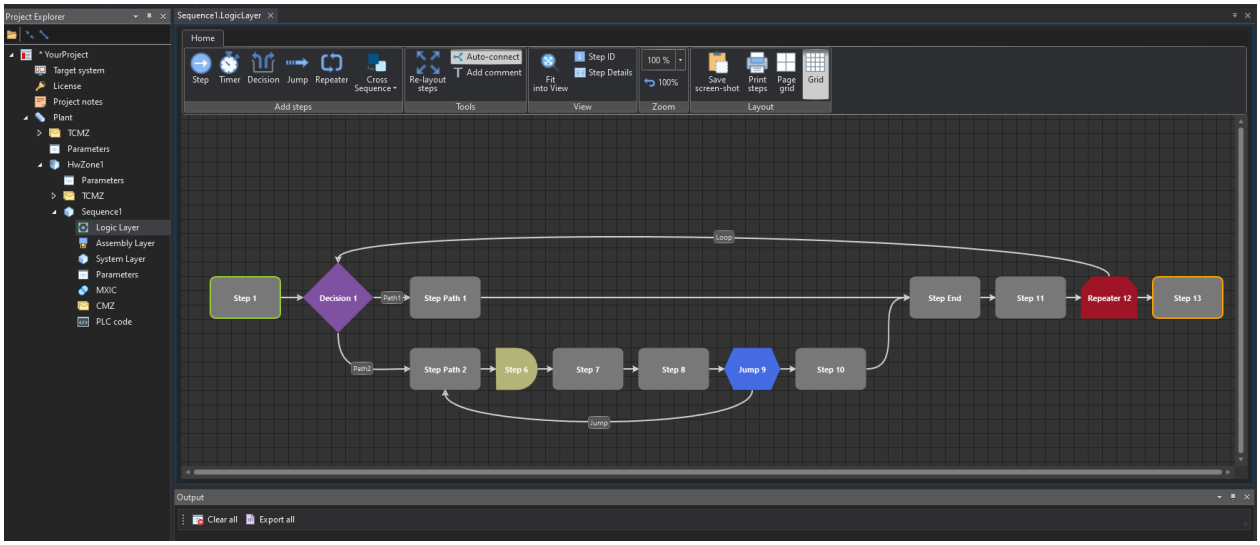
HMI

1.2 Create Logic

A process model is an abstract representation of a real or planned process executed by a manufacturing machine. A process model consists of a set of elements that describe the activities, resources, events, and decisions that make up the process.

The logic of a manufacturing machine can be divided into two layers: the Logic Layer and the System Layer. The Logic Layer contains the technical requirements of the machine. The System Layer contains the machine's technical aspects and the interfaces to other systems.

The logical process is defined in the logic layer. Five modeling elements are available to describe this process. Logic elements describe the relationships between the individual steps.

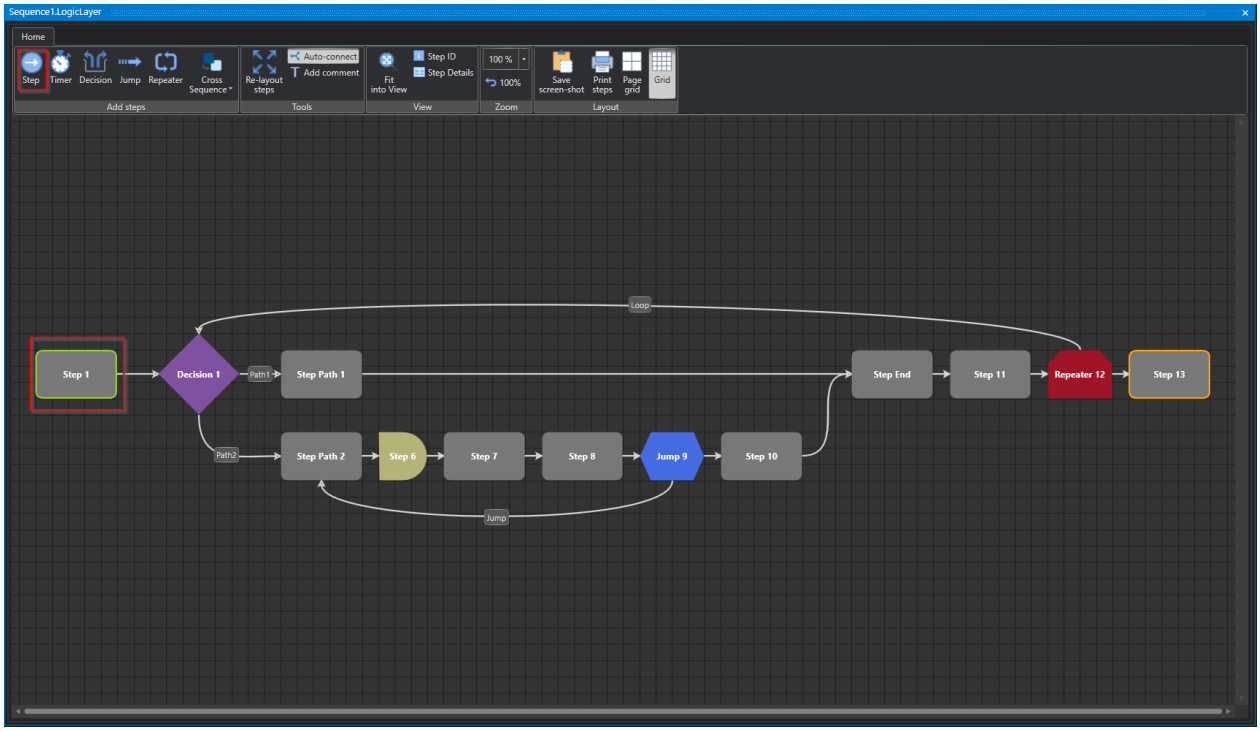


Studio

1.2.1 Elements of modeling

Step

The step is the basic element for modeling, which is used to describe the individual discrete process steps or plant states. Each step is assigned a unique ID. A step is defined as an initial state with ID 1 via the "Start Shape" property.



Studio

Properties

Studio

Monitoring

Step time monitoring is a function that monitors the execution time of each step in a program. It can be activated or deactivated via the HMI. The monitoring is "cycle-accurate", i.e. the time is measured in each cycle and compared with a preset limit value. If the time exceeds the limit value, an alarm is triggered. The step time monitoring is displayed in a window in the HMI, which shows the current time, the limit value, and the status of each step. An additional time can be defined, which is added to the measured time when determining the limit value. The step time alarm is triggered when the current step time exceeds the limit value formed by the measured step time plus the additional step time.

Disable Timeout

In this section you have the option to disable the step time monitoring for the selected step. Note, however, that step time monitoring is enabled by default.

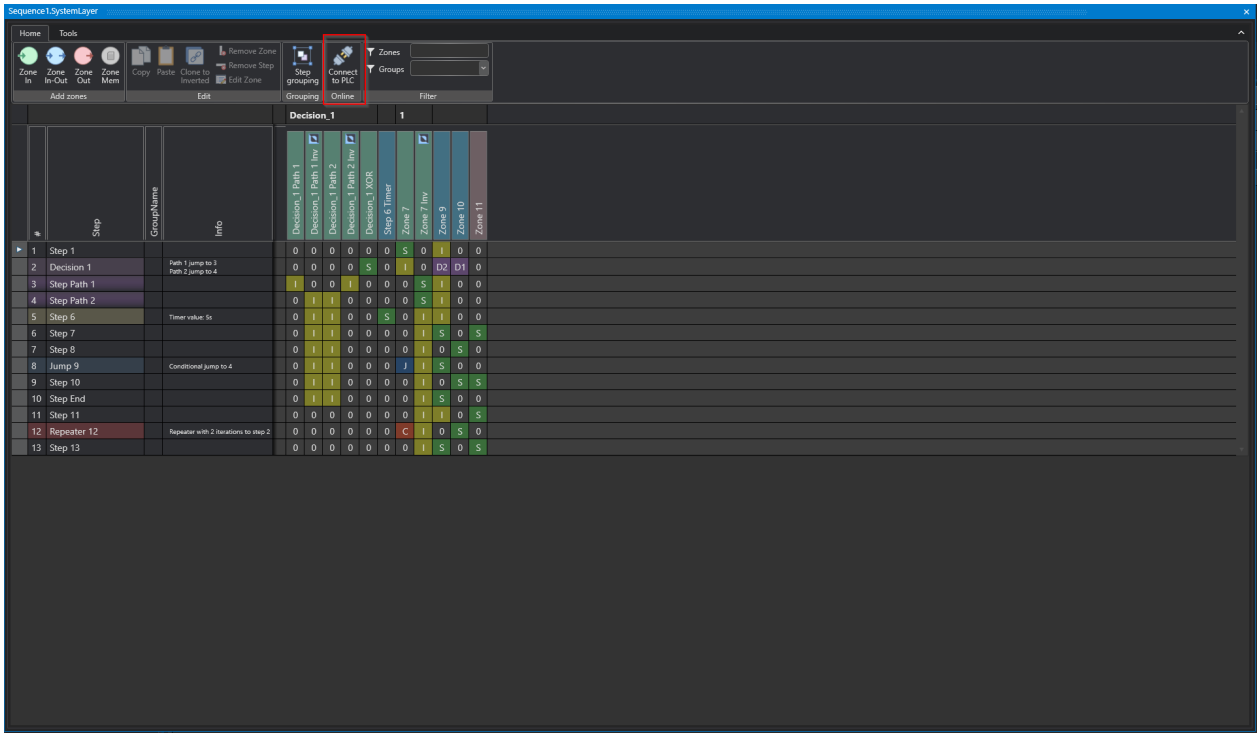
Timeout

In the HMI (Human-Machine Interface), a warning is displayed if there is no change of state within a certain period. Please specify during which period, without a state change, a warning should appear on the HMI.

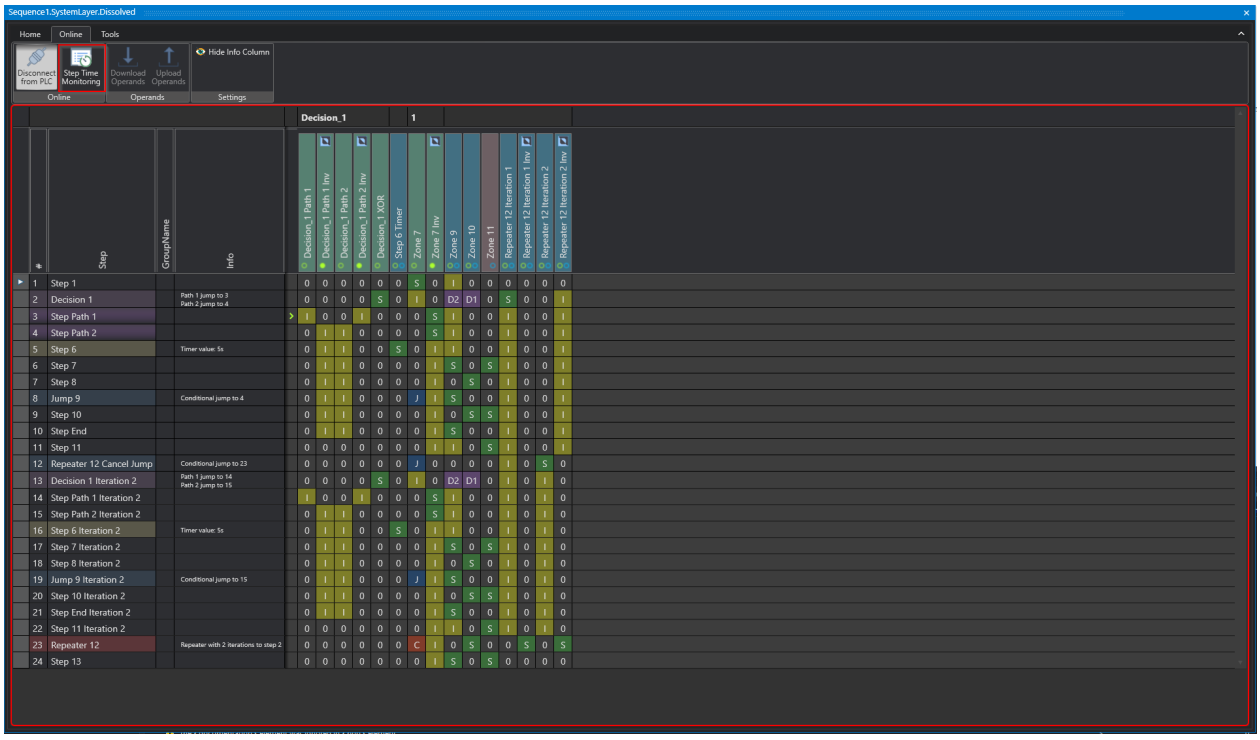
Timeout Additional

In teach mode, the step times can be measured, and this time will be added to the measured time. A warning will be issued on the HMI, if a step exceeds the measured time plus a specified tolerance time. Please note that the exact settings and parameters for measuring the step times, the tolerance time, and the warning on the HMI depend on the specific application or system and must be configured accordingly.

To access the online monitoring in the Studio in the System Layer, you can use the following icons:

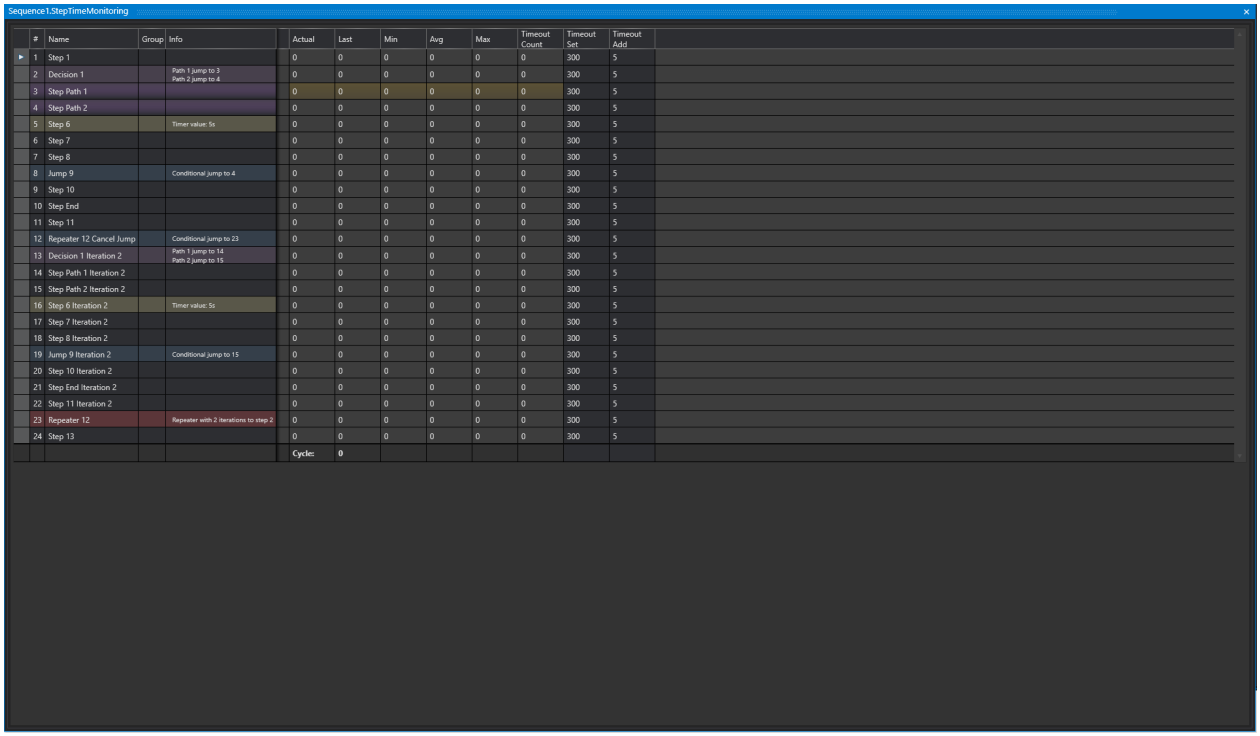


Studio



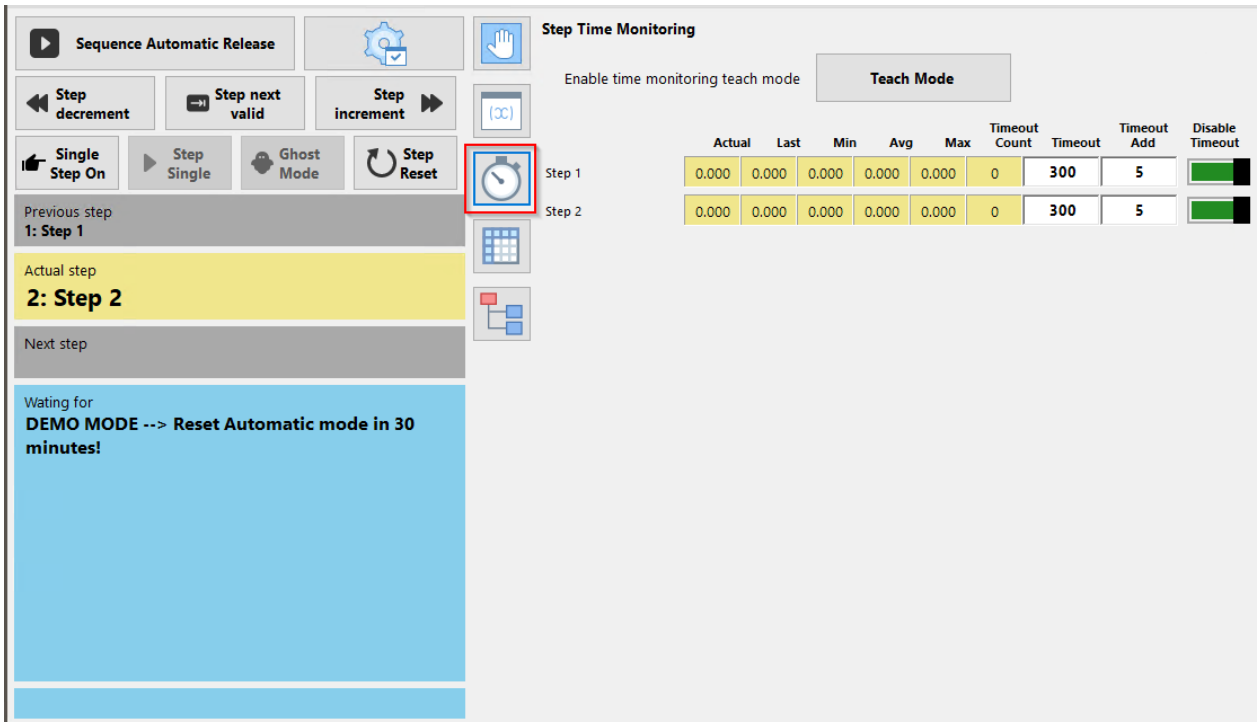
Studio

At this point all step related data is displayed in different formats.



Studio

Step time monitoring is also available in the HMI.



HMI

Step Control

End Of Cycle (EOC) is an operating mode that puts the plant in a safe state when production is finished. The plant runs until the defined EOC step is reached, which can vary depending on the plant type. The EOC mode can be safely switched on via the safety key switch on the operator console. The display on the HMI shows the current status of the EOC mode. Each Sequence has its own EOC step, which shuts down the corresponding components or moves them to a park position.

End Of Cycle

This setting can be used to specify that the sequence may safely stop at the specified step when stopped after the end of the cycle.

Studio

Common

Group Name

It is possible to group steps by a common group name.

ID

The step ID is the unique ID within a sequence, this is automatically assigned by the system depending on the start step.

Name

The name is displayed in the HMI layer as well as in the system layer. If no text has been entered for the "HMI Display Text" property, this name will be used instead.

HMI

HMI

HMI Display Text

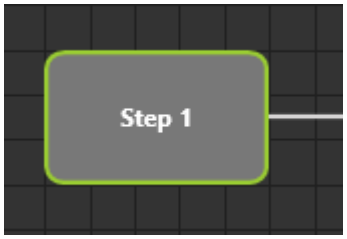
The "HMI Display Text" is the text that is displayed in the HMI for this step and can be used as an alternative to the name.

HMI

Logic Layer

Start Shape

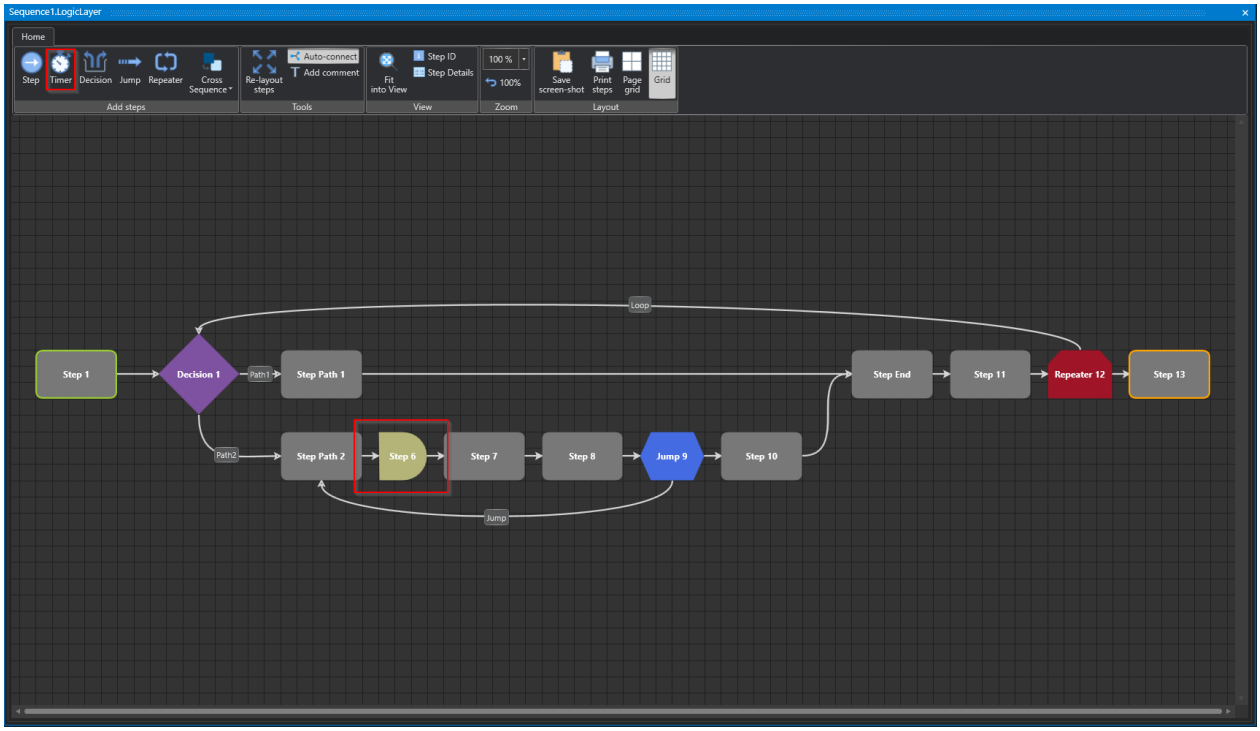
Defines the first step of the sequence.



Studio

Timer Step

The Timer element allows the modeling of time-dependent steps. A timer step can only change to the following step when a specific time has elapsed. This time is specified by a timer parameter. An example would be a timed stirring process in a tank, where the step does not move to the next step until the predefined stirring time has elapsed.



Studio

Properties

Studio

System Zones

System Zones are the zones, automatically generated with a step and are essential for the step to function.

Show System Zones

It is possible to show or hide the system zones in the System Layer.

Timer

Timer Parameters

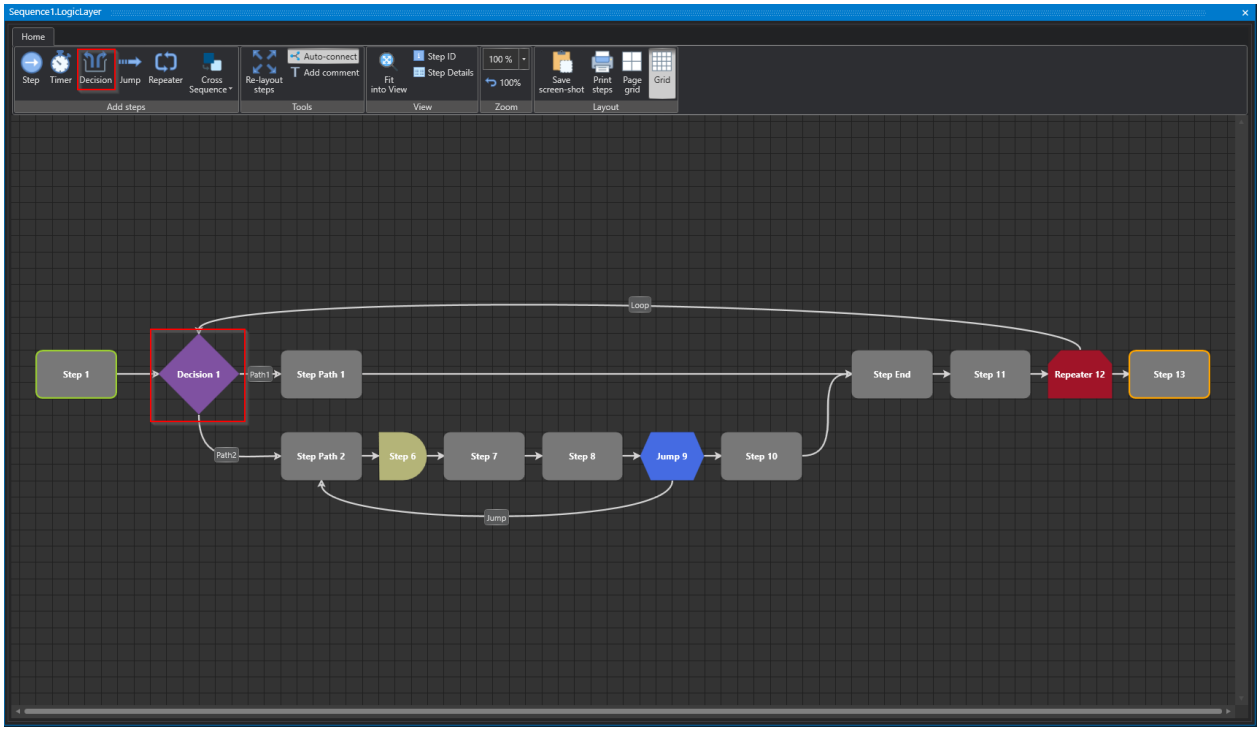
It is possible to use a parameter previously created in Parameters as a time base.

Timer Value

It is possible to enter a fixed time for this step.

Decision Step

The Decision step is used to make decisions in the process. Depending on the fulfillment of a criterion, it is decided which path will be continued in the Sequence. Two paths can be selected, as shown in the figure below.



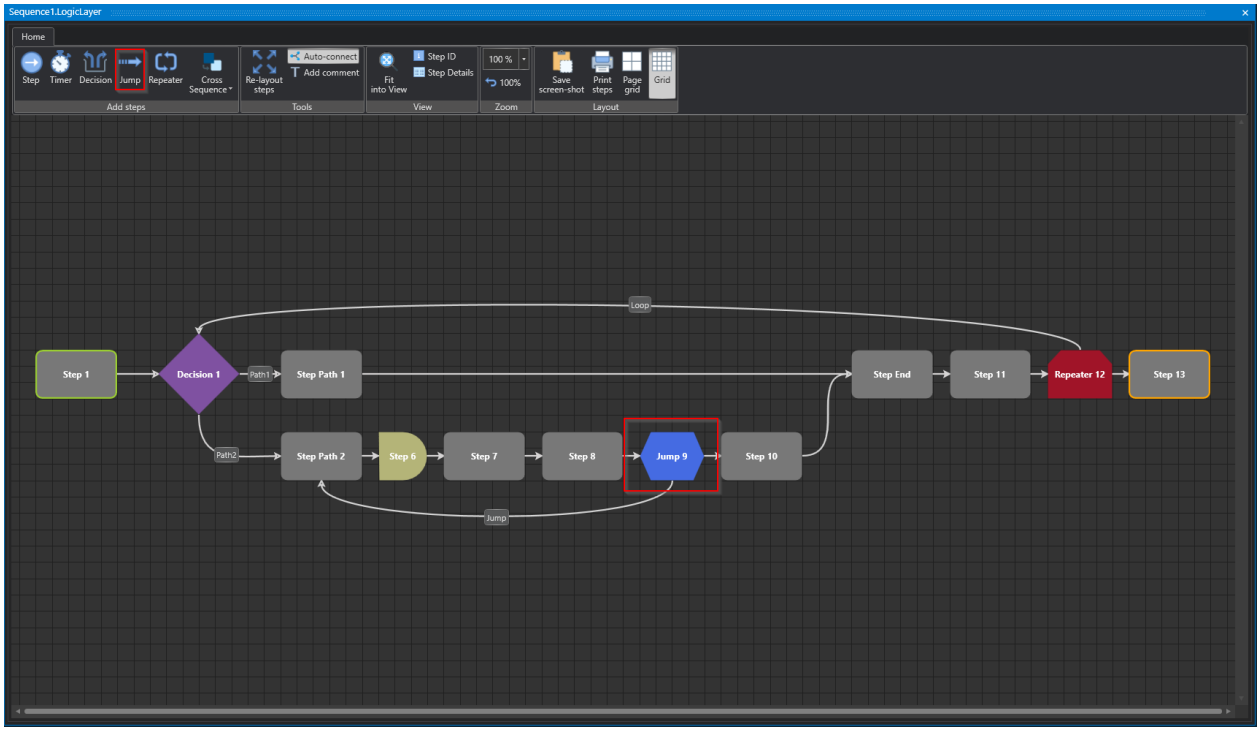
Studio

Properties

Studio

Jump Step

A jump in process modeling allows you to jump from a specific step to another step in the step chain to affect program flow. Jumps can be used to implement conditional flows or to skip or bypass certain steps based on special conditions or events. The use of jumps requires careful planning and modeling to ensure that the flow of the step chain works correctly and reliably. Improper use of jumps can lead to errors and make program logic unpredictable.



Studio

Properties

Studio

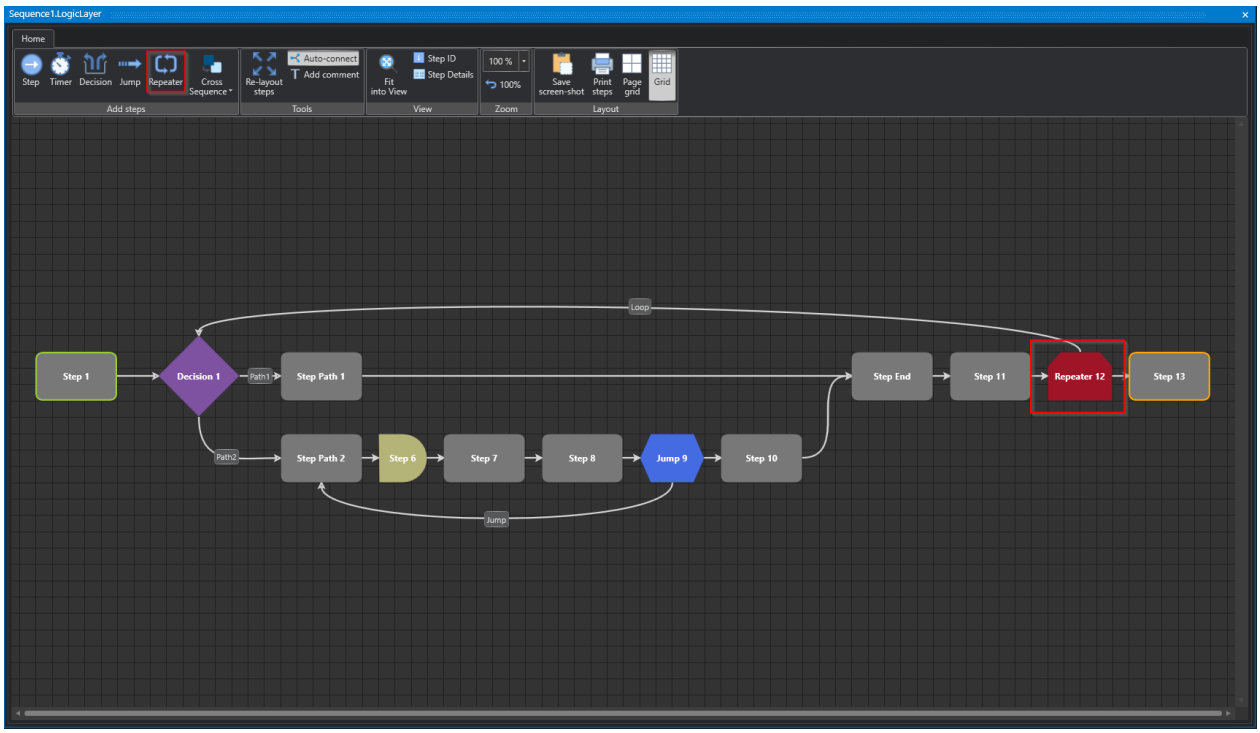
Jump

Jump Step

Determination of the jump target, i.e. the step to jump to, based on specific conditions stored in the system.

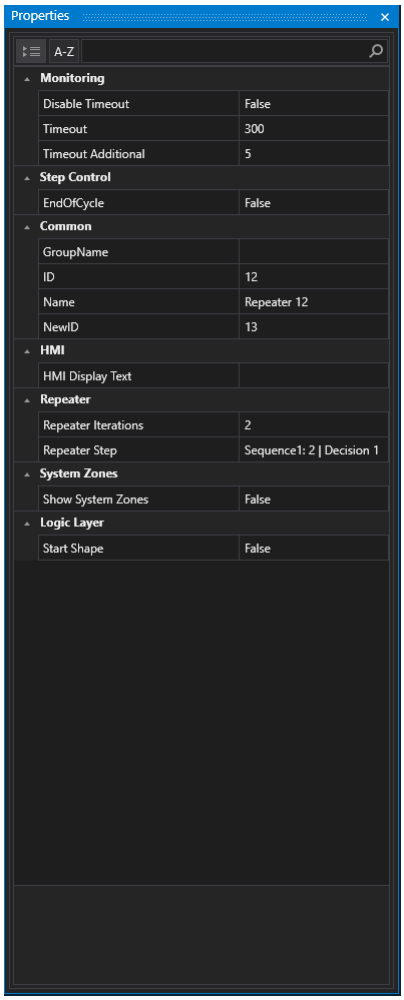
Repeater Step

The repeater allows repeated execution of specific steps previously modeled by copying the steps defined in the loop up to the specified number of repeater iterations. This is a finite sequence of iterations rather than traditional loop programming. The repeater also provides the ability to terminate loop execution before reaching the maximum iteration by using a termination variable.



Studio

Properties



Studio

Repeater

Repeater Iterations

Determines the number of duplications of steps in the loop.

Repeater Step

Determines the target step to jump to.

1.2.2 Cross Sequence

The Cross Sequence is composed of Cross Sequence Sender and Cross Sequence Receiver. The Cross Sequence is used to organize the shares between two sequences. Through these shares it is possible to synchronize, arrange, or time Sequences. In the following example, a Cross Sequence Receiver in Sequence 2 is shared by a Cross Sequence Sender. When the Cross Sequence Sender is created, two system zones are created to control the release. To do this, the Cross Sequence Receiver must be assigned to a Master. To do this, select the properties of the Cross Sequence Receiver and assign a Cross Sequence Transmitter. In this example, the Cross Sequence Receiver of Sequence 1 was selected. The Cross Sequence Receiver Getter is released by the selected Cross Sequence Sender, see the following figure. There is still the possibility creating a Cross Sequence Multi Sender to release of several Cross Sequence Senders. In the system layer the corresponding assignment is entered in the info field.

Important basics

Each **Receiver Getter** is automatically linked to a Receiver Setter.

Each **Transmitter Setter** is automatically linked to a Transmitter Getter.

A **Receiver Getter** can select (only) one Sender Setter for release. The feedback of the receiver setter is automatically assigned to the transmitter getter.

A **Transmitter Setter** can be assigned (several) receiver getters for release. The feedback of the receiver setter is automatically assigned to the transmitter getter.

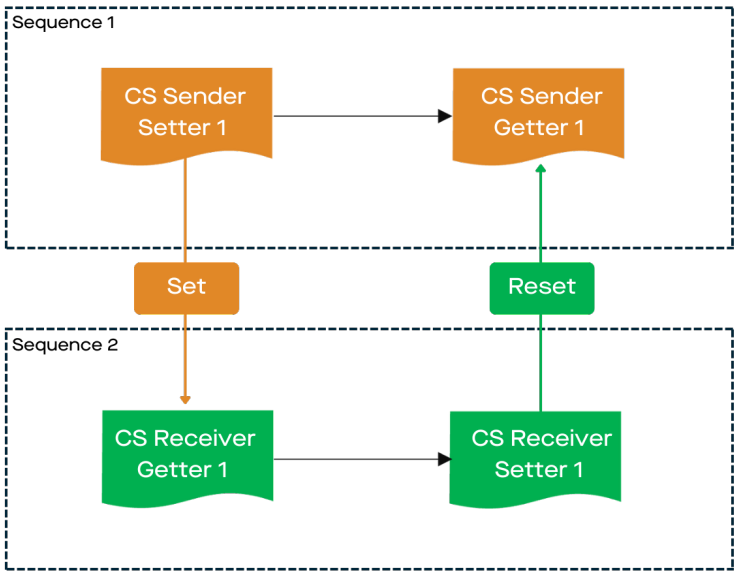
Each **Multi Sender Setter** is automatically linked to a Multi Sender Getter.

A **Multi Sender Setter** can select (only) one Sender Setter for release. Automatically, the Receiver Setter feedback is assigned to the Transmitter Getter. The Sender Getter reports back to the Multi Sender Getter.

A Transmitter **Setter** can be released multiple times by the **Multi Transmitter Setter**.

Cross Sequence Transmitter-Receiver Enable

The Cross Sequence Transmitter-Receiver release is shown in the following figure:



The process in detail is as explained in the following steps:

Step 1 in Sequence 1: Transmitter Setter 1 enables the assigned Receiver Getter 1. Step Forwarding.

Step 1 in Sequence 2: Wait for release of Transmitter Setter 1.

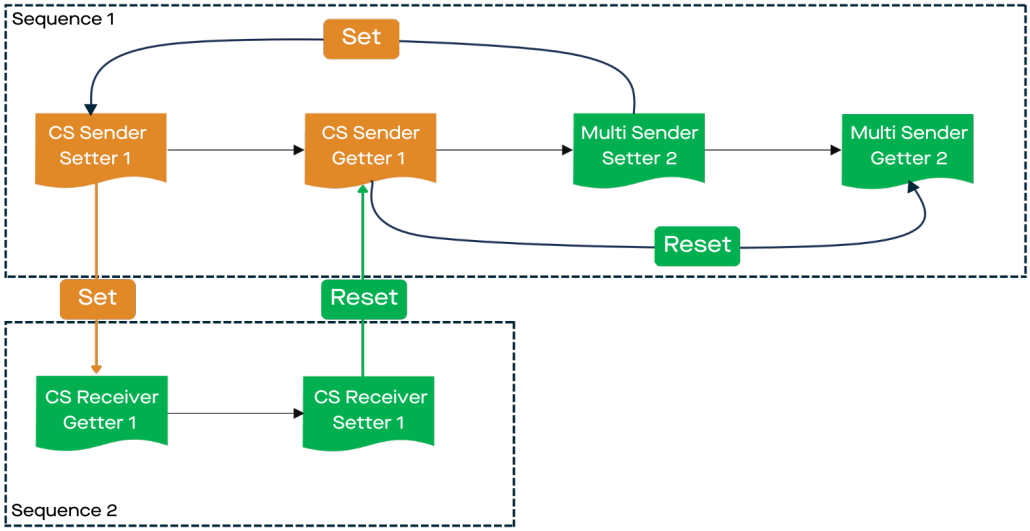
Step 1 in Sequence 2: Release received from transmitter Setter 1. Step changeover.

Step 2 in Sequence 1: Wait for the release of Receiver Getter 1 of Sequence 2.

Step 2 in Sequence 2: Feedback from Receiver Setter 1 to Transmitter Getter 1. Step switching.

Step 2 in Sequence 1: Release received from Receiver Getter 1. Step switching.

Cross Sequence Multi Transmitter-Receiver Enable



(Abbreviated step sequence) The sequence of the two Cross Sequences with Multi Sender release is as follows:

Step 1 & 2 in Sequence 1 & 2 are done as described in the Cross Sequence Multi Sender-Receiver Enable screen.

Step 3 in Sequence 1: Multi Sender Setter 1 releases the assigned Transmitter Setter 1. Step chain.

The step chain of the transmitter-receiver release starts again.

Feedback from Receiver Setter 1 to Transmitter Getter 1. Feedback from Transmitter Getter 1 to Multi Transmitter Getter 2. Step changeover

Step 4 in Sequence 1: Waiting for release of transmitter Getter 1. step-by-step switching

(Complete step sequence) The course of the two Cross Sequences with Multi Sender release looks as follows:

Step 1 in Sequence 1: Transmitter Setter 1 releases the assigned Receiver Getter 1. Step sequence.

Step 1 in Sequence 2: Wait for release from Transmitter Setter 1. Release received from Transmitter Setter 1. Step switching.

Step 2 in Sequence 1: Waiting for release of Receiver Getter 1 of Sequence 2. Release received from Receiver Getter 1. Step switching.

Step 2 in Sequence 2: Feedback from Receiver Setter 1 to Transmitter Getter 1. Step switching.

Step 3 in Sequence 1: Multi Transmitter Setter 1 releases the assigned Transmitter Setter 1. Step changeover.

Step 1 in Sequence 1: Transmitter Setter 1 releases the assigned Receiver Getter 1. Step switching.

Step 1 in Sequence 2: Wait for release of transmitter Setter 1. Release of transmitter Setter 1 received. Step switching.

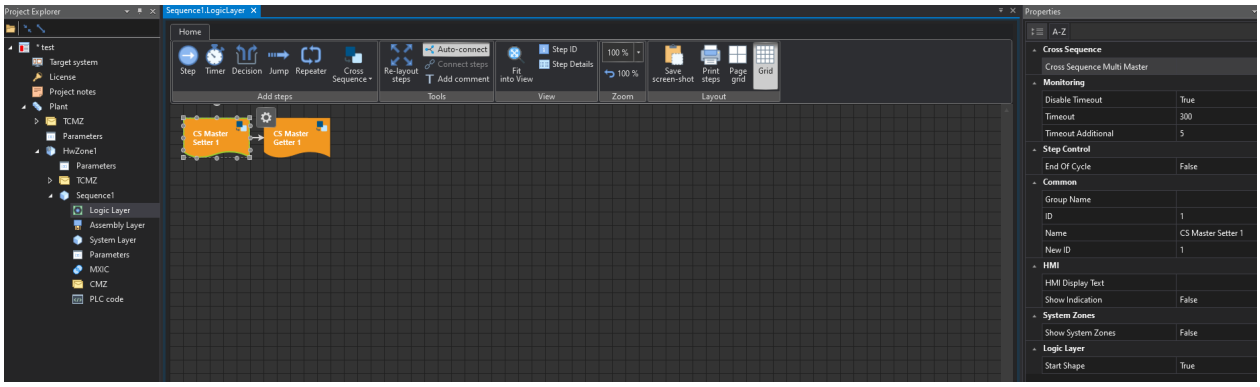
Step 2 in Sequence 1: Waiting for release of Receiver Getter 1 of Sequence 2. Release received from Receiver Getter 1. Step switching.

Step 2 in Sequence 2: Feedback from Receiver Setter 1 to Transmitter Getter 1. Feedback from Transmitter Getter 1 to Multi Transmitter Getter 2. Step switching.

Step 4 in Sequence 1: Waiting for release of transmitter Getter 1. Step switching.

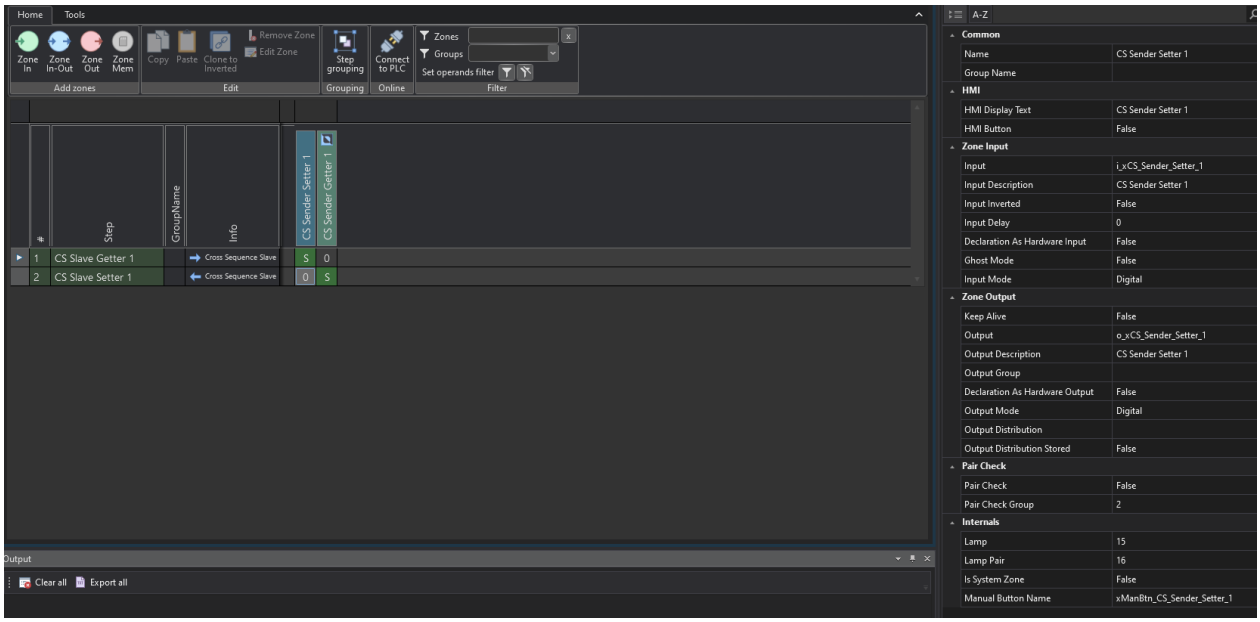
The Cross Sequence is composed of Cross Sequence Transmitter and Cross Sequence Receiver. The Cross Sequence is used to organize the releases between two sequences. Through these shares it is possible to synchronize, arrange or time Sequences.

In the following example, a Coss Sequence Receiver in Sequence 2 is shared by a Cross Sequence Sender.



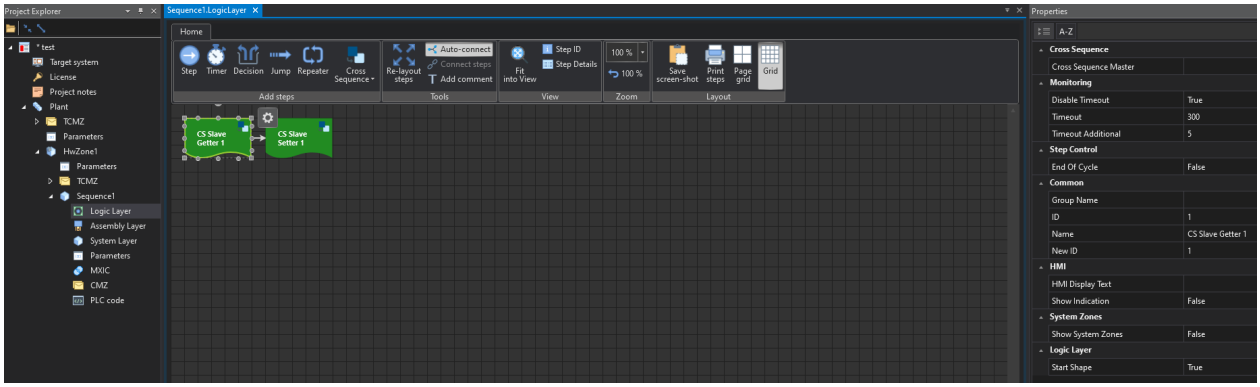
Studio

When the Cross Sequence Sender is created, two system zones are created that control the share.



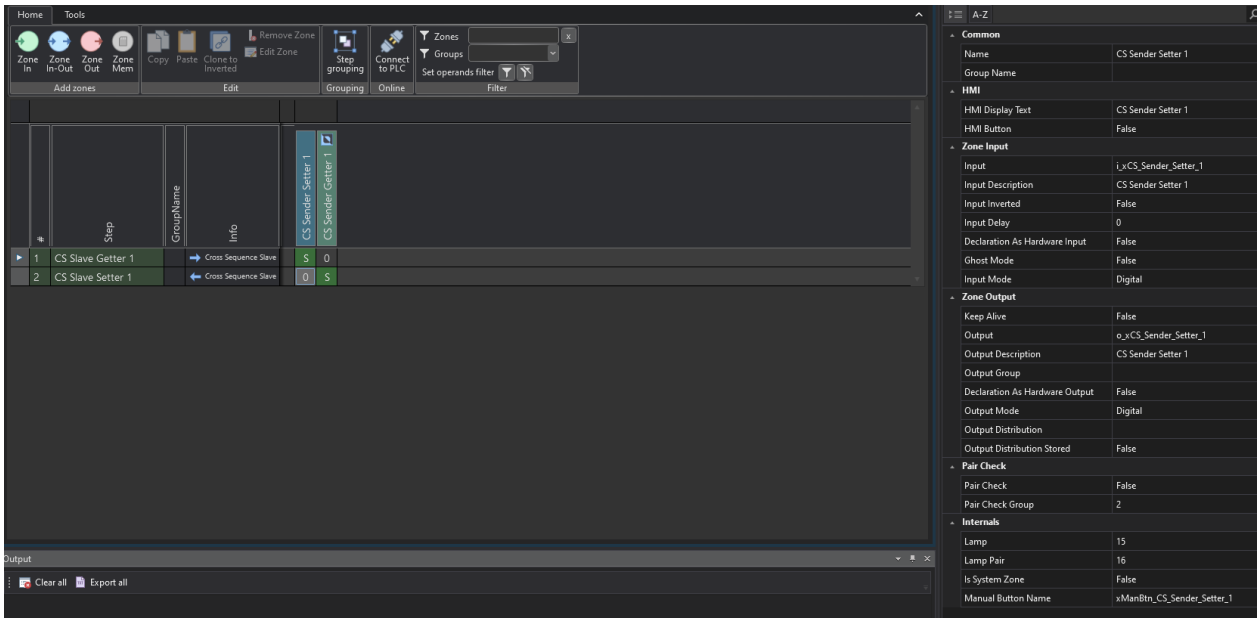
Studio

To do this, the Cross Sequence Receiver must be assigned to a transmitter. To do this, select the properties of the Cross Sequence Receiver and assign a Cross Sequence Transmitter. In this example, the Cross Sequence Receiver has been assigned to Sequence 1.

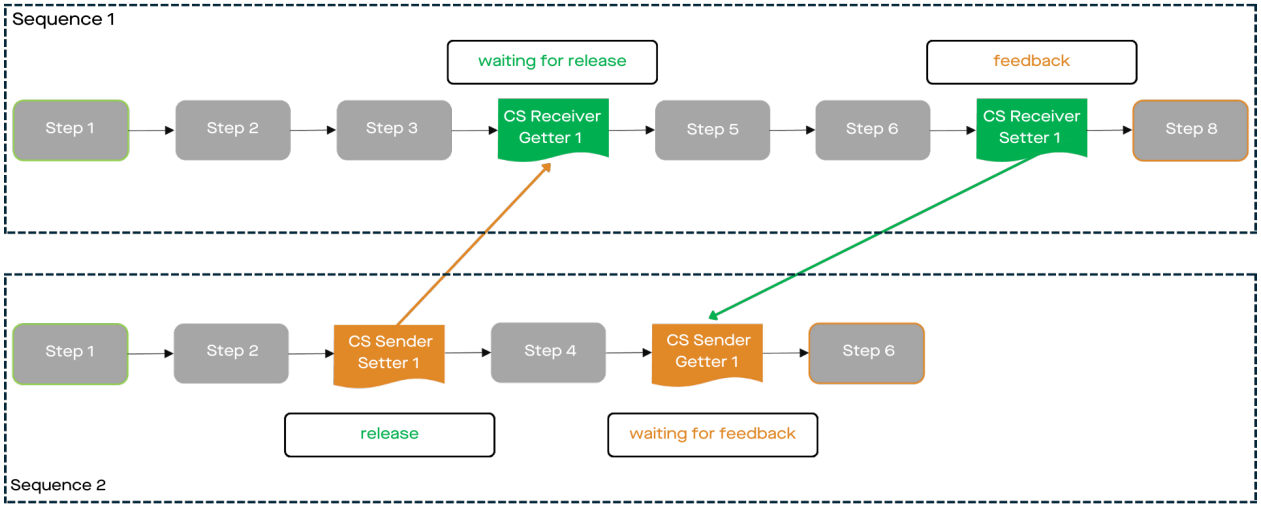


Studio

The Cross Sequence Receiver Getter is released by the selected Cross Sequence Transmitter, see the following picture.



Studio



Studio

The Cross Sequence is controlled by Setter/Getter logic:

- **Cross Sequence Transmitter Setter**

sets the enable bit

- **Cross Sequence Sender Getter**

waits until the enable bit is reset by the receiver

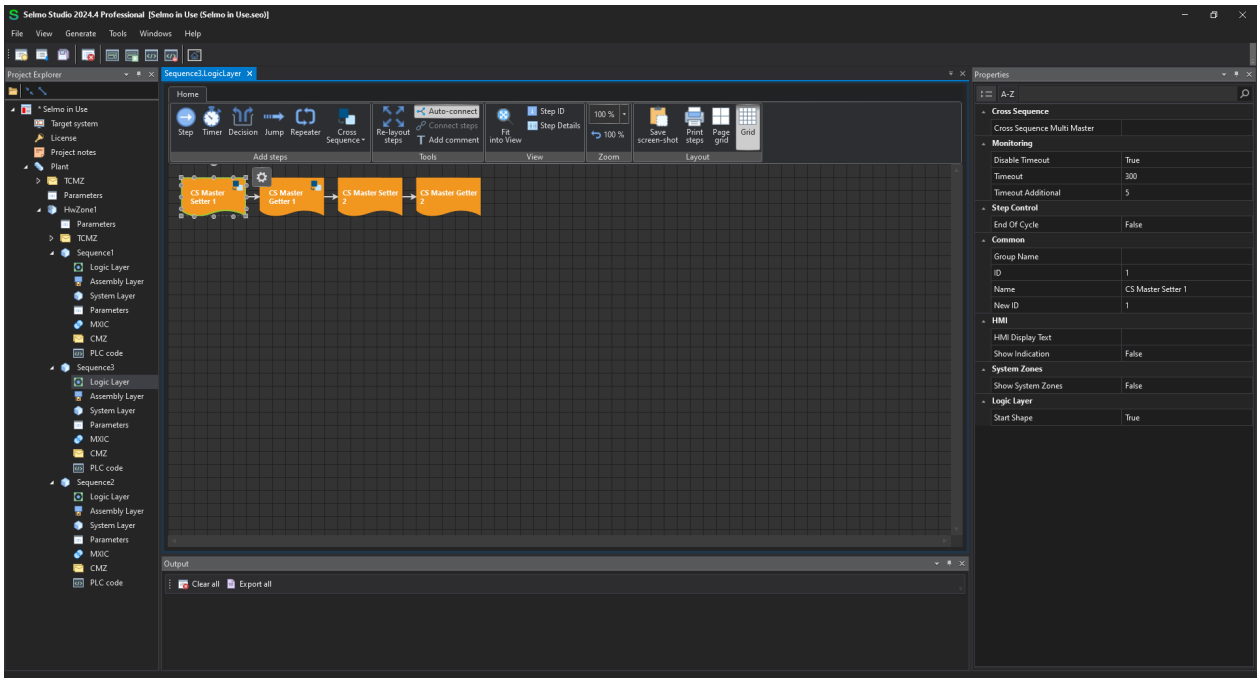
- **Cross Sequence Receiver Getter**

waits for the enable bit from the transmitter

- **Cross Sequence Receiver Setter**

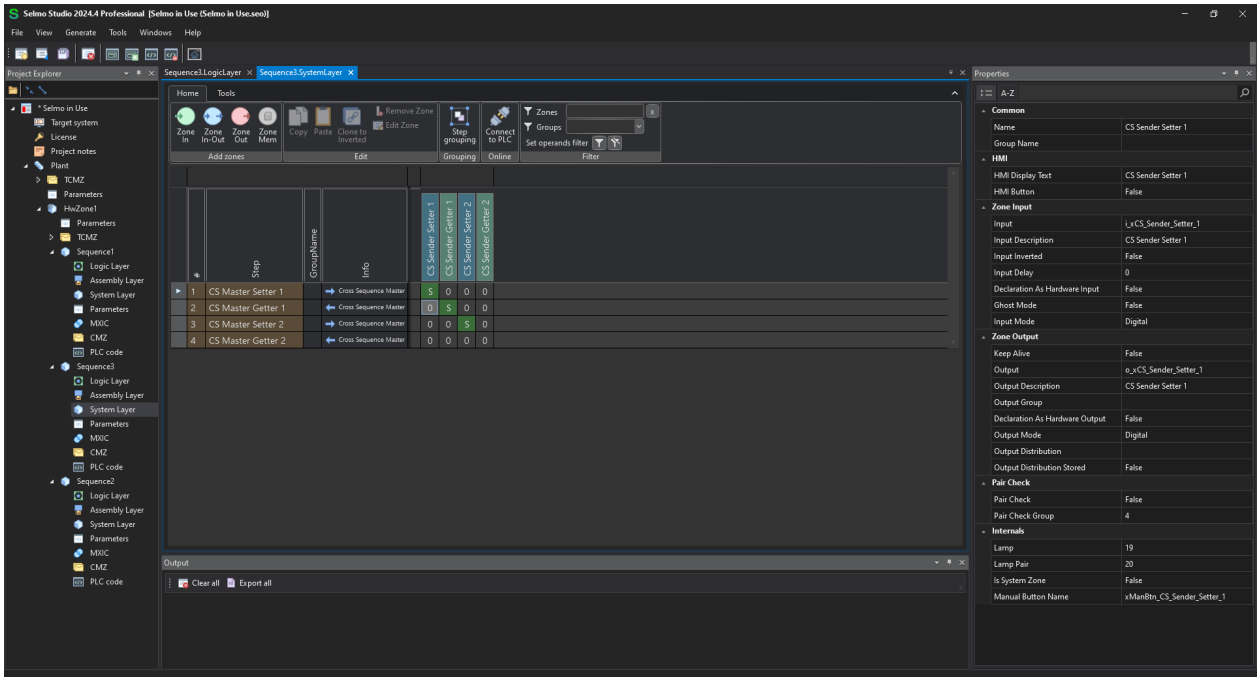
resets the enable bit

There is also the possibility to create a Cross Sequence Multi Sender for the release of several Cross Sequence Transmitters.



Studio

In the System Layer the corresponding assignment is entered in the info field.

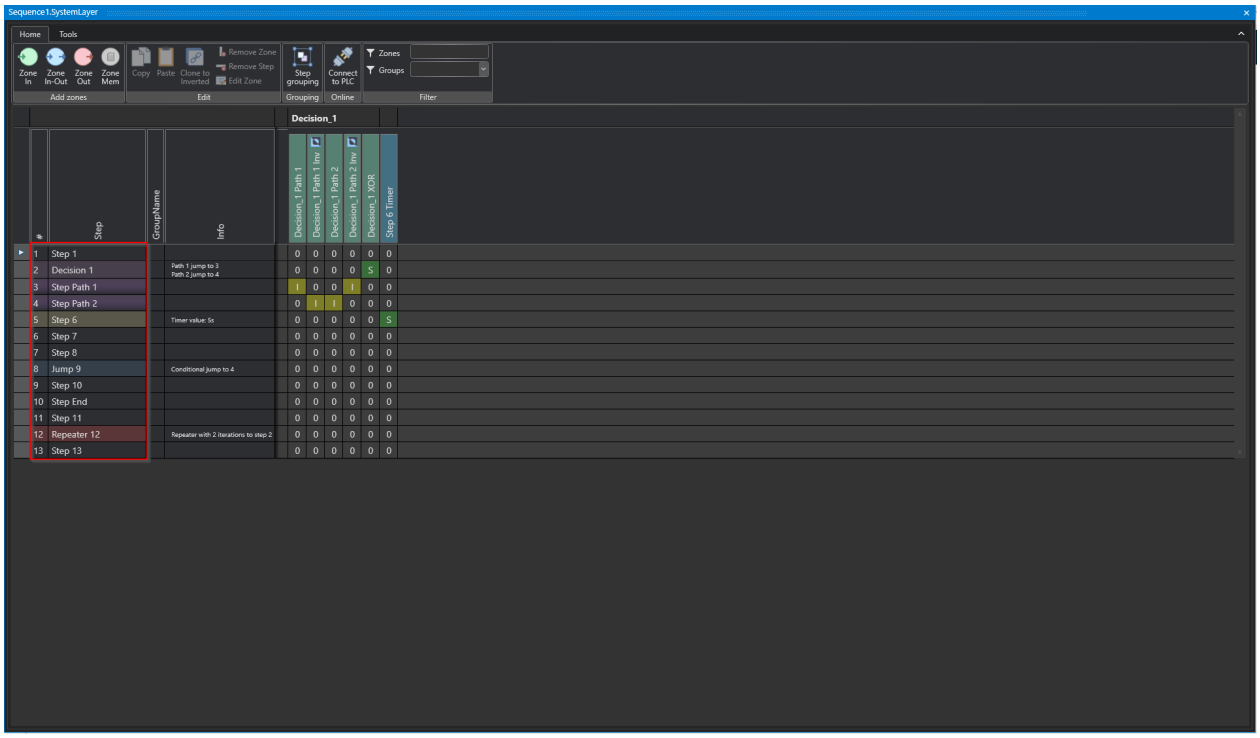


Studio

1.2.3 Conversion to system layer

The logic layer is displayed in the system layer as a column on the left. In the system layer the program is modeled, the zones are defined, and the machine states or monitoring are determined.

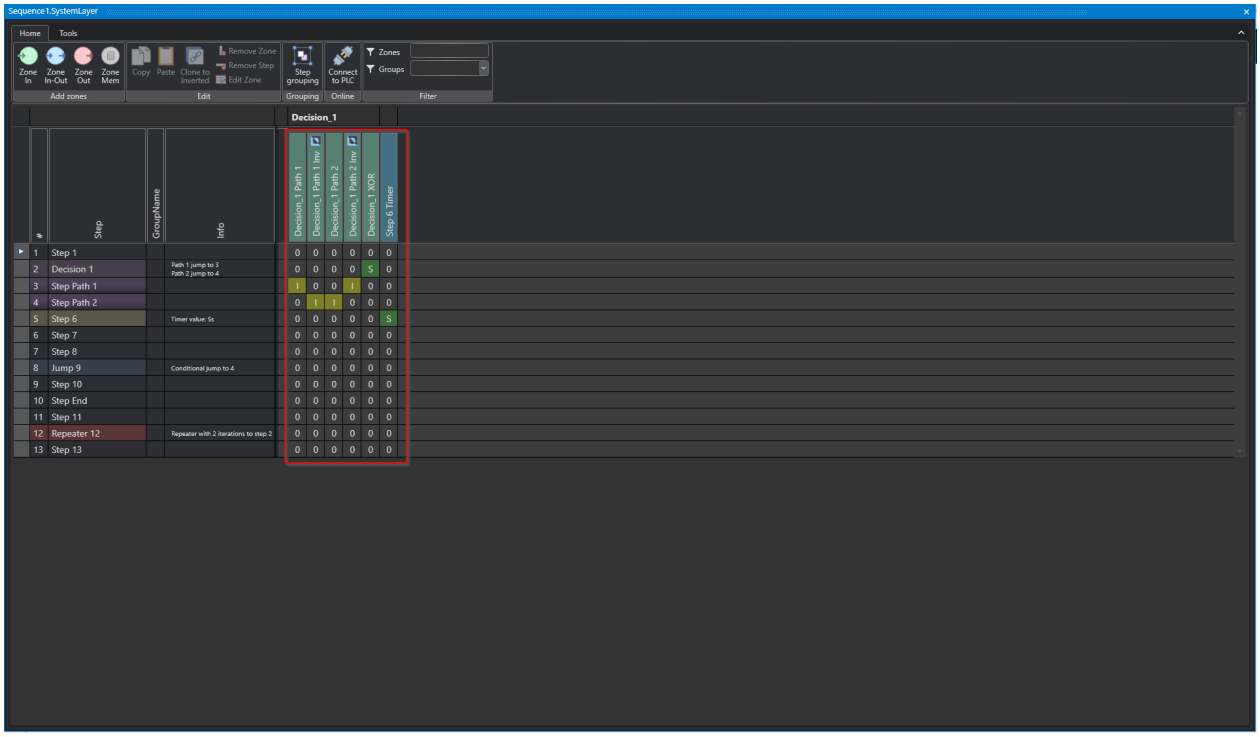
In each step the states of the zones are described with the operands '0', 'I', 'S', 'M', 'D1', 'D2', 'J', 'C'.



Studio

1.3 Create signals

The system layer is the layer responsible for processing and controlling process signals. The process signals are binary values that represent the state of sensors or actuators. The system layer uses bit-controlled logic functions to link the process signals and control corresponding outputs. The system layer is divided into zones, each of which monitors or controls a part of the overall process.



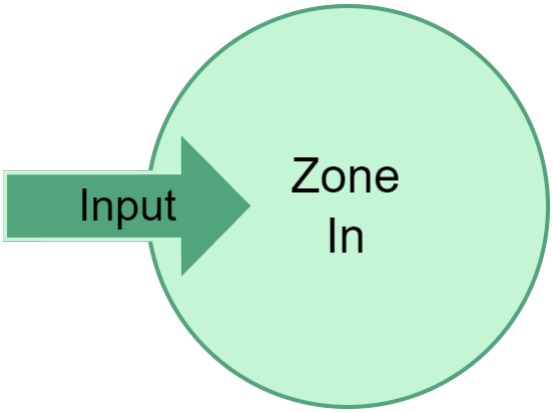
Studio

One possible subdivision of signals in an automated system is the distinction between Constantly Monitored Zone (CMZ) signals and process signals. CMZs include all signals that are relevant to the safety and protection of the plant, such as fuses, monitoring, emergency stop switches, etc. These signals are constantly monitored and evaluated to trigger a shutdown or a warning if necessary. Process signals, on the other hand, are all signals that control or regulate the actual process sequence, such as sensors, actuators, measured values, etc. The division of signals into CMZ and process signals helps to define and simplify the requirements for sensor technology, signal processing and communication.

1.3.1 Zone

Zones allow us to define and control different areas of a system. Zones can be configured differently depending on the application and requirements. It is important to understand the functions and properties of the different zone types. In this section, the zone types In, InOut, Out. and Mem are explained, and their possible applications are shown. It also describes the operands that can be used in a zone to trigger or check certain actions.

Include sensor



To include a sensor in your application, you must use the Zone-In. The sensor sends only one input signal to the system; this is monitored with the Zone-In and the negated Zone-In on high and low.

Examples of sensors that can be integrated with a Zone-In:


Inductive sensor

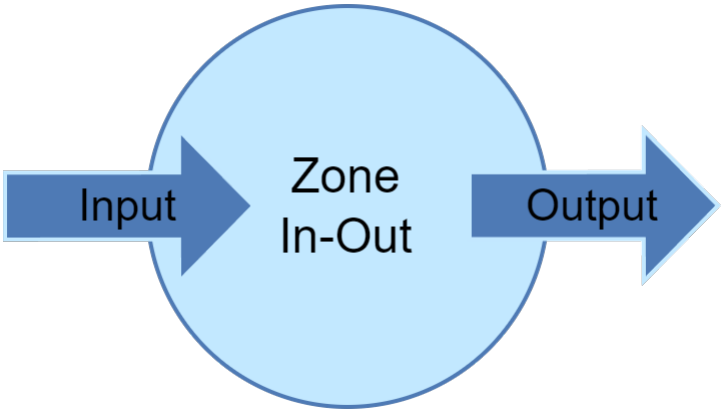

Switch


Push button


End position sensor


Light barrier

Integrate actuator with feedback



The Zone-InOut switches an output and expects feedback that the function controlled by the output has been executed.

Examples:

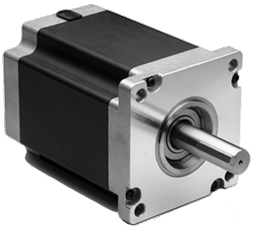
- Cylinder valves "Extend" is switched: The Zone-InOut expects the end position sensor of the cylinder as input.
- Servo motor "Move to position 1" is switched: The Zone-InOut expects the signal that the axis is in position 1 as input.
- Frequency converter "Speed 1" is switched: Zone-InOut expects as input the signal that the frequency inverter has reached the speed.

Examples for sensors that can be integrated with a Zone-InOut:

Frequency inverter

Cylinder

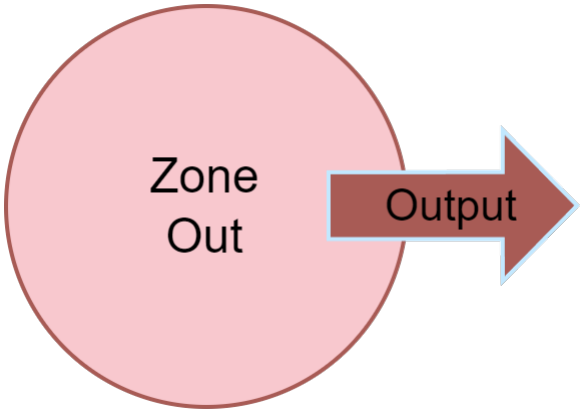
Servo motor



Stepper motor

Regulation

Integrate actuator without feedback



The Zone-Out switches actuators and does not expect feedback, i.e., it operates pure switching logic. This zone type is mainly used for lights and signals, but also for actuators that have no feedback. These, however, should be avoided, if possible, through technical means as they lead to uncertainties in the process.

Examples of sensors that can be integrated with a Zone-Out:

Alarm light

Signal horn

Staff light

Signal lamps

Cylinder without feedback

Integrating flags

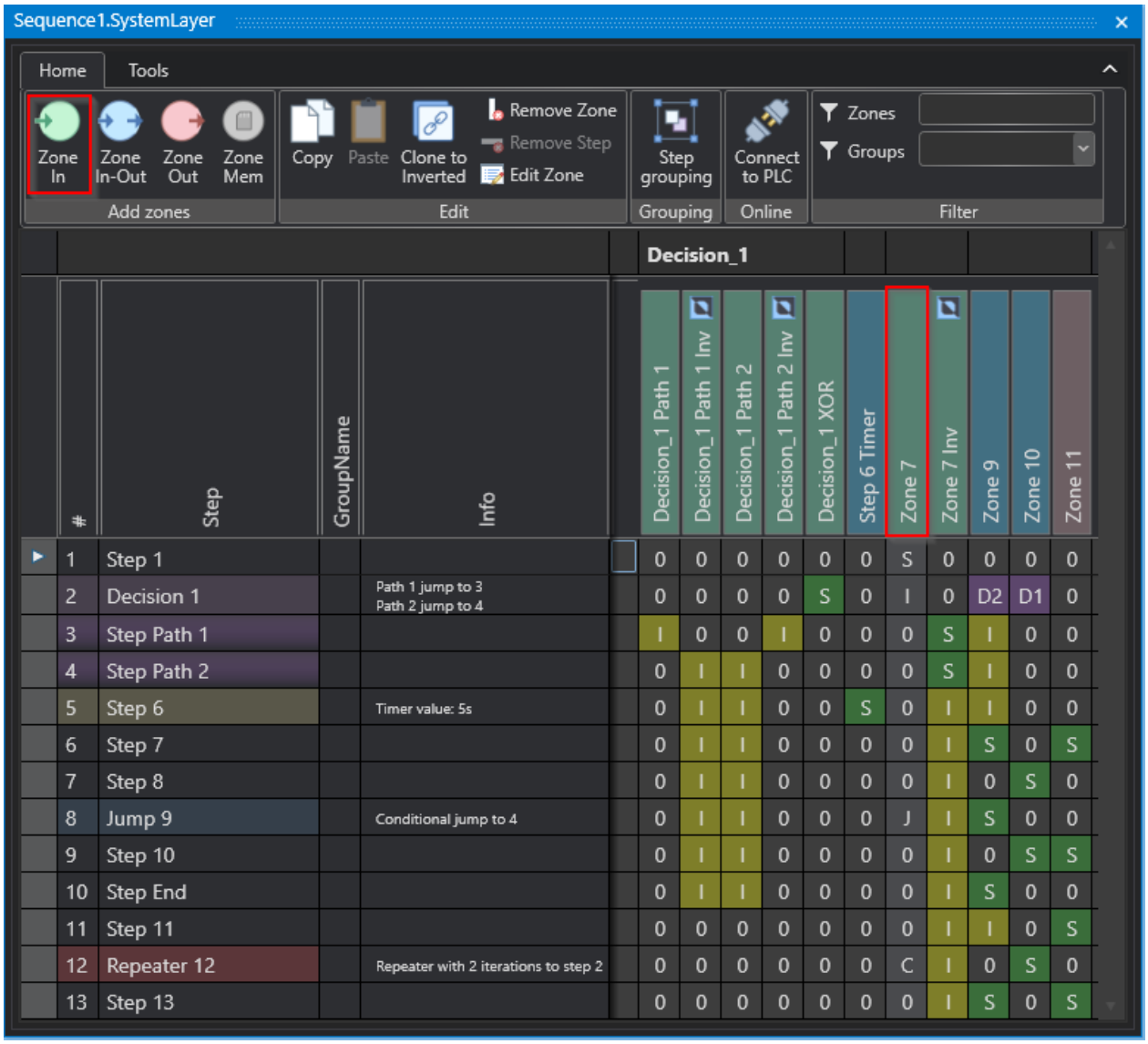
The zone mem is used as a flag to remember executed process steps. It always consists of two zones, the first sets and checks the flag to high, the second resets the flag and checks the flag to low.

1.3.1.1 Zone In

The **Zone-In** can be used to process input information from e.g., sensors. If the Sequence check is set in a Zone-In in a step, a state change of this Zone-In is expected for this step. An interlock check of an input zone ensures that the expected state of this zone-in this step is maintained. In practice, for example, a switch can be monitored by a Zone-In. If a switching operation is required, the signal of the switch must be linked to the Zone-In and a sequence check must be set in the corresponding step.

Create Zone-In

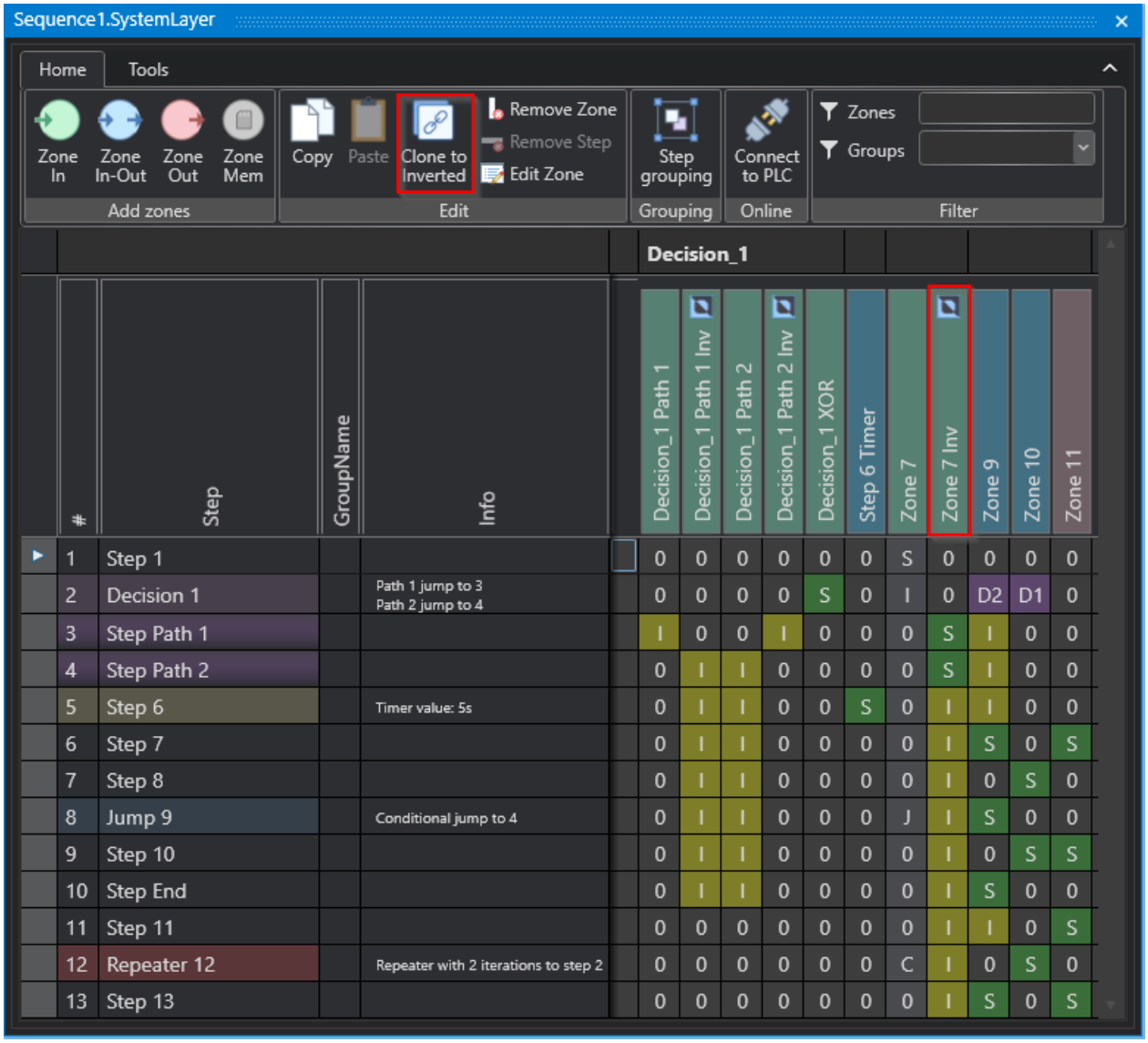
To create a Zone-In, click on the corresponding icon.



Studio

Inverted Zone-In

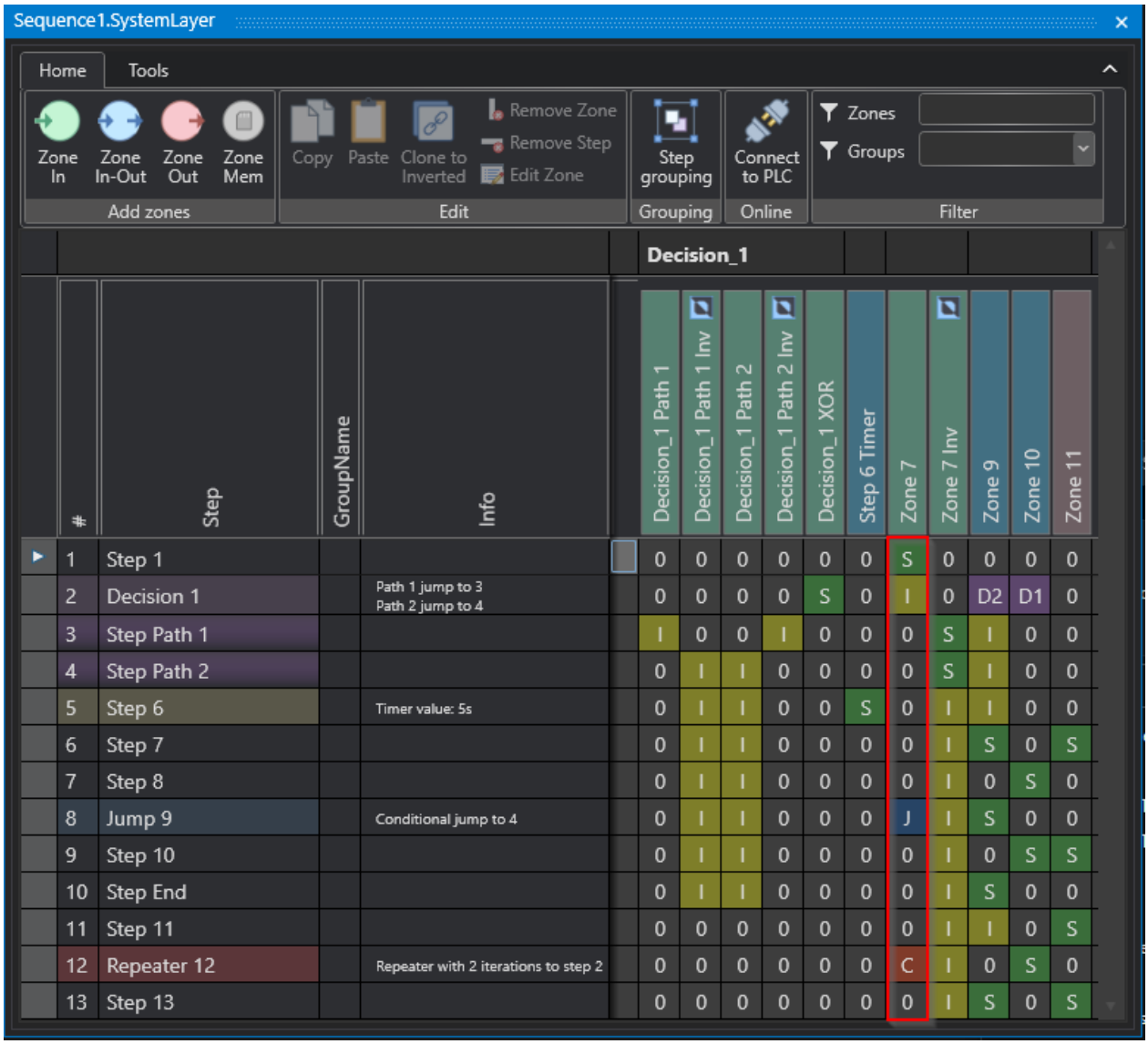
To completely protect a signal, an inverted Zone-In can be inserted. The inverted zone monitors the safe transition of the signal associated with the zone from true to false. For example, pushbuttons, sensors, etc., are monitored with it. This can be used to prevent incorrect operation. The button Clone to inverted inserts an inverted zone of the selected Zone-In.



Studio

Operands of the Zone-In

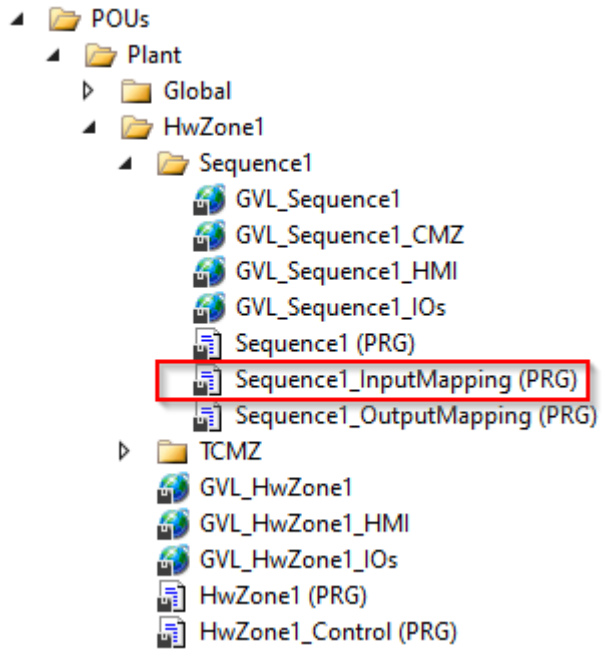
If a zone-in is to be monitored in the step, it is necessary to enter the operand 'I'. The operand 'S' is used to wait for a change of state of the zone. The Zone-In is used to process input signals like buttons, sensors etc.



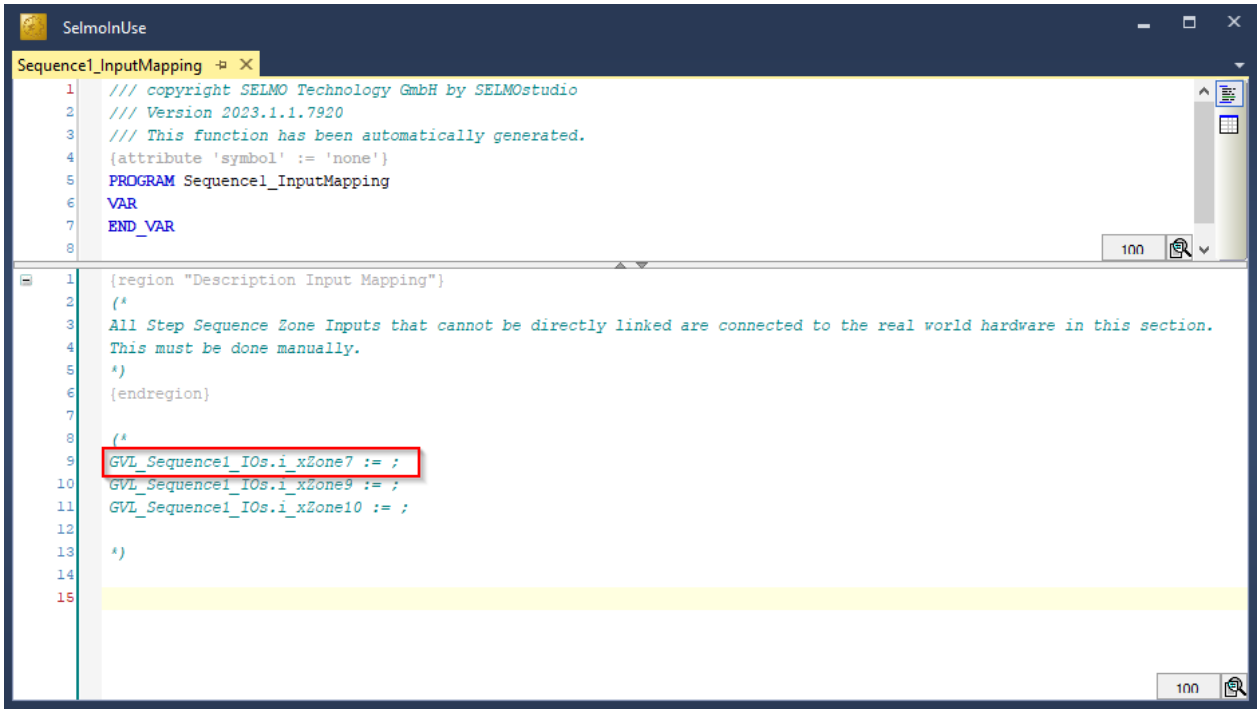
Studio

Assignment of signals to the Zone-In

Further the input of the Zone-In is generated in the code and can be linked to a signal, see code below:

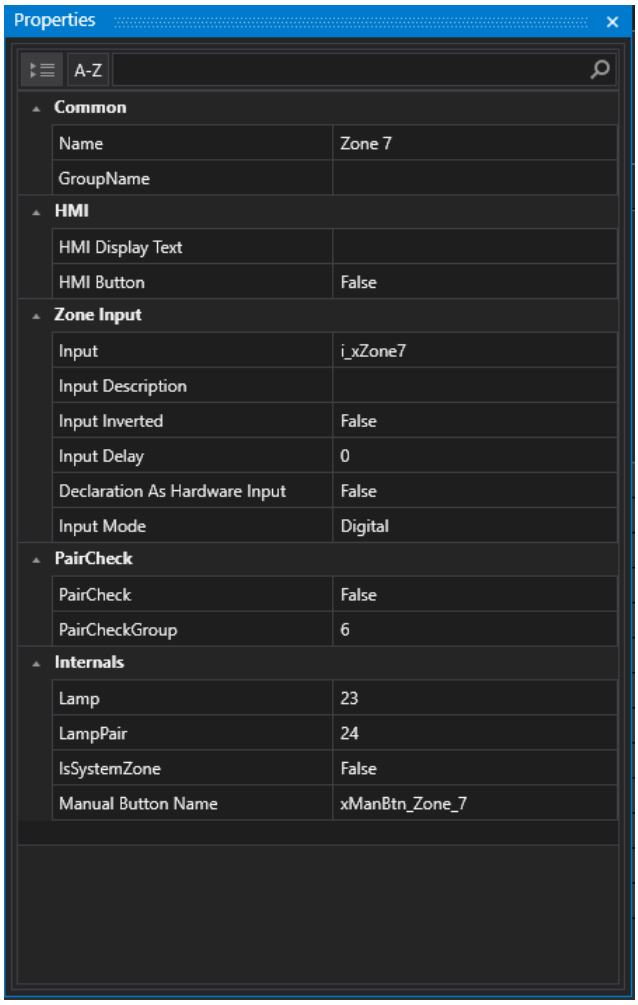


PLC



PLC

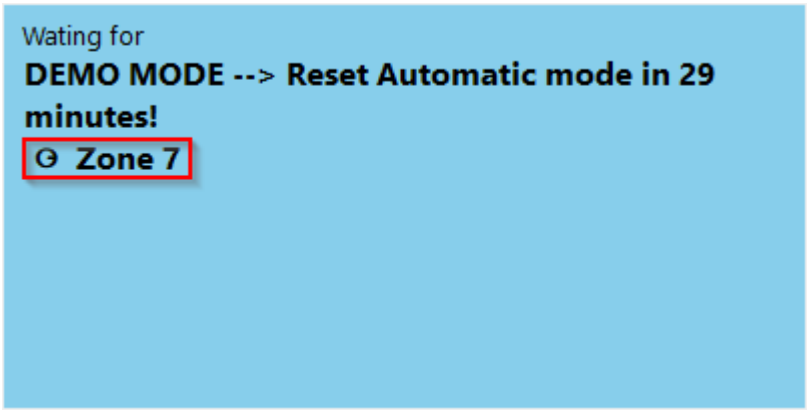
Properties



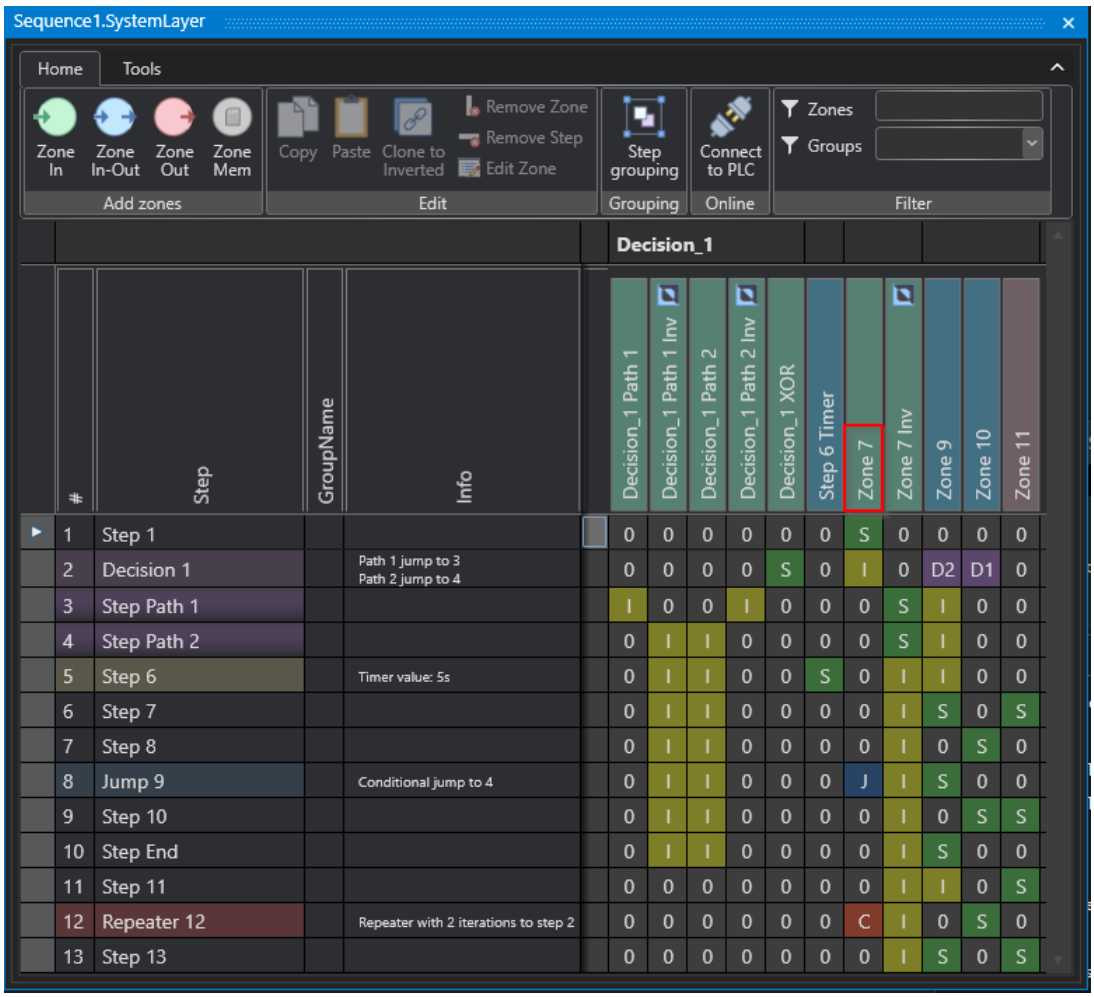
Studio

Name

The name serves as an identifier of the zone and is displayed in both the HMI and the system layers.



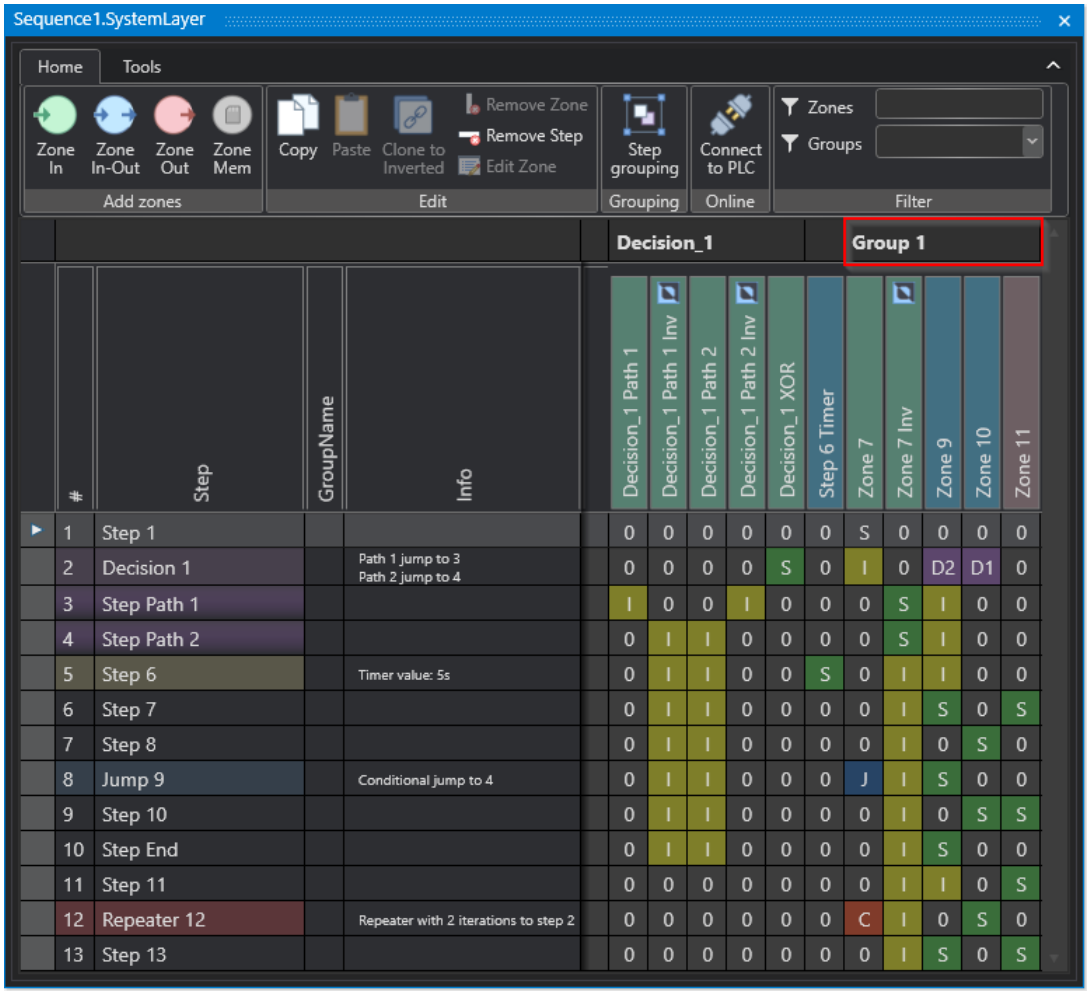
HMI



Studio

Group Name

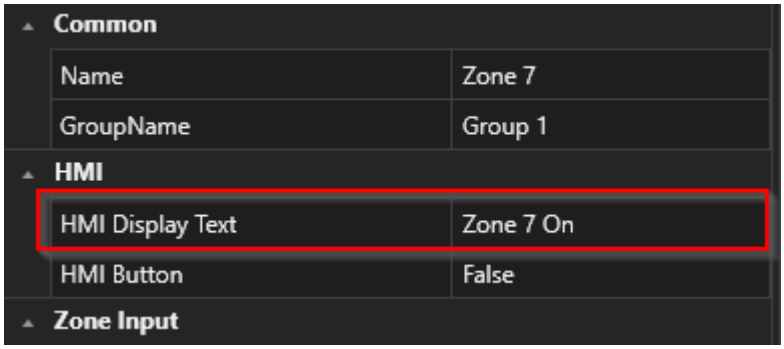
The GroupName property can be used to group zones together. It also serves as a filter name for the filter function.



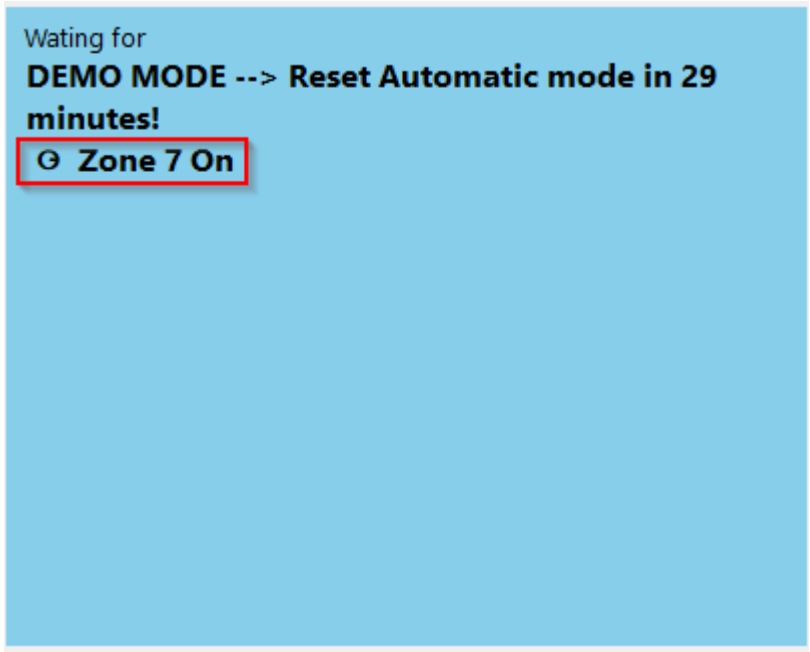
Studio

HMI Display Text

If a text is entered here, the text is only effective in the HMI and is adopted as the display text. The property Name is overwritten in the HMI.



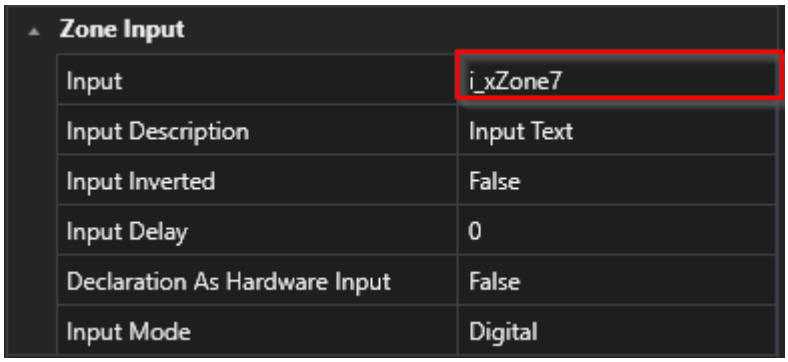
Studio



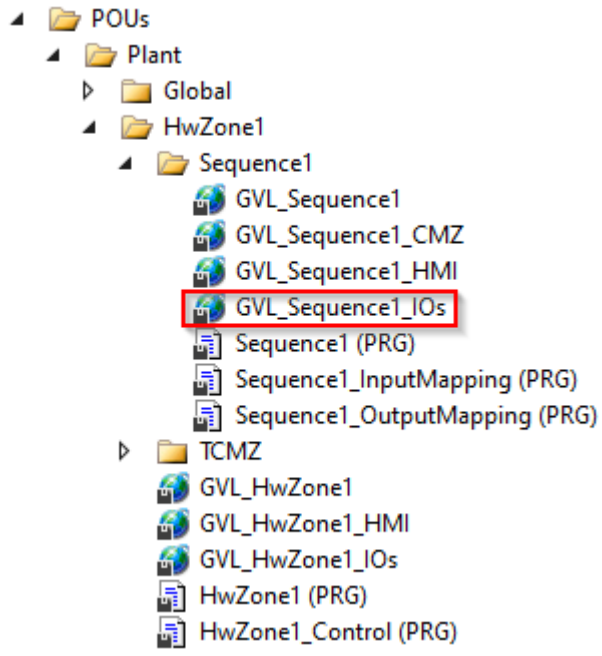
HMI

Input

In the context of PLC programming, the term "input variable name" refers to the designation or name of an input variable that is defined and used in the programmable logic controller (PLC) program code. For example, an input variable may be a signal from a sensor or other source that is processed by the PLC to perform a specific action. The name of the input variable in the PLC program is an important aspect of controller programming because it helps make the code easier to read, understand, and maintain. Therefore, the name should be carefully chosen and descriptive to clarify the function of the input variable and increase the readability of the code.



Studio



PLC

SelmoInUse

GVL_Sequence1_IOs

```
/// copyright SELMO Technology GmbH by SELMOstudio
/// Version 2023.1.1.7920
/// This function has been automatically generated.
///
/// ### PLEASE DO NOT MAKE ANY CHANGES MANUALLY HERE! Make the changes only in SELMOstudio ###
{attribute 'qualified_only'}
{attribute 'symbol' := 'readwrite'}
VAR_GLOBAL
    ///Input: no description
    i_xDecision_1Path1: BOOL;
    ///Input: no description
    i_xDecision_1Path2: BOOL;
    ///Input: Input Text
    i_xZone7: BOOL;
    ///Input: no description
    i_xZone9: BOOL;
    ///Input: no description
    i_xZone10: BOOL;
    ///Output: no description
    o_xZone_6: BOOL;
    ///Output: no description
    o_xZone9: BOOL;
    ///Output: no description
    o_xZone10: BOOL;
    ///Output: no description
    o_xZone11: BOOL;
    ///Output: no description
    o_xZone_12RepeaterIteration1: BOOL;
    ///Output: no description
    o_xZone_14RepeaterIteration2: BOOL;
END_VAR
```

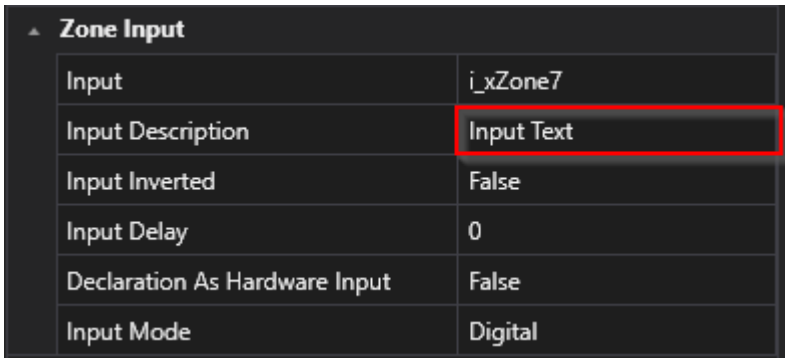
100

PLC

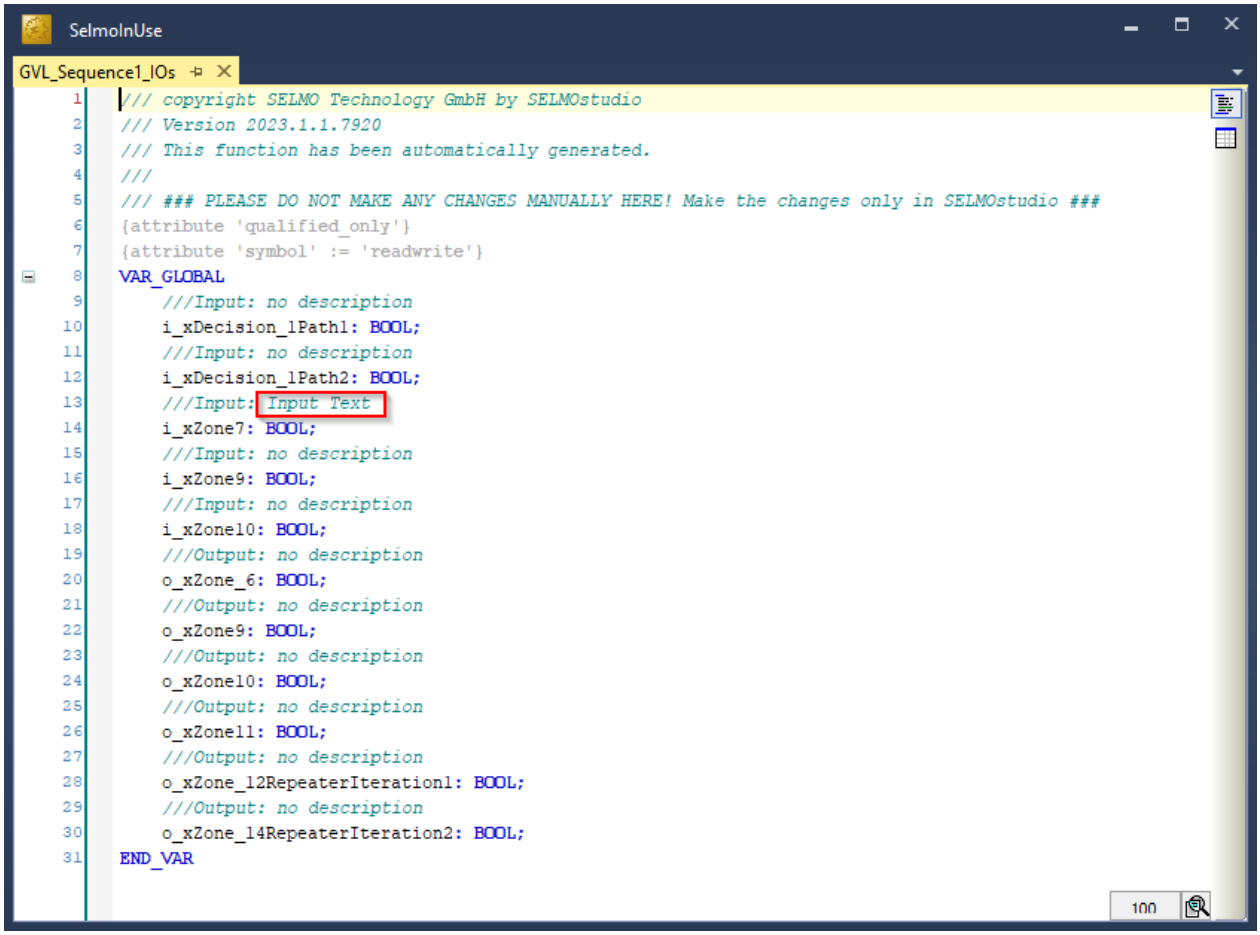
Input Description

In PLC programming, it is important that each input variable in the code has a meaningful description. Such a description helps other programmers or maintenance personnel to understand and edit the code more easily. The input variable description should ideally describe the purpose and operation of the variable.

A clear and concise description of the input variable can also ensure that it is properly configured and calibrated before it is integrated into the PLC program. It also aids in debugging and troubleshooting the code by allowing the programmer to quickly determine which input variables are affected. The input variable description should therefore be considered an important part of the PLC program documentation to increase code efficiency, maintainability, and accuracy.



Studio



PLC

Input Inverted

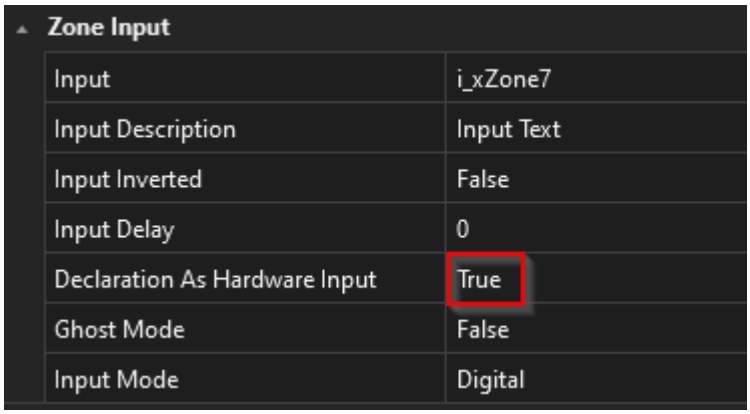
Please specify whether the input signal is inverted or not. This means that you need to decide whether the signal entering a particular system is inverted in polarity or not. An inverted signal polarity means that the signal is reversed in terms of its positive and negative polarity. It is important to make this decision because it affects the way the signal is processed in the system.

Input Delay

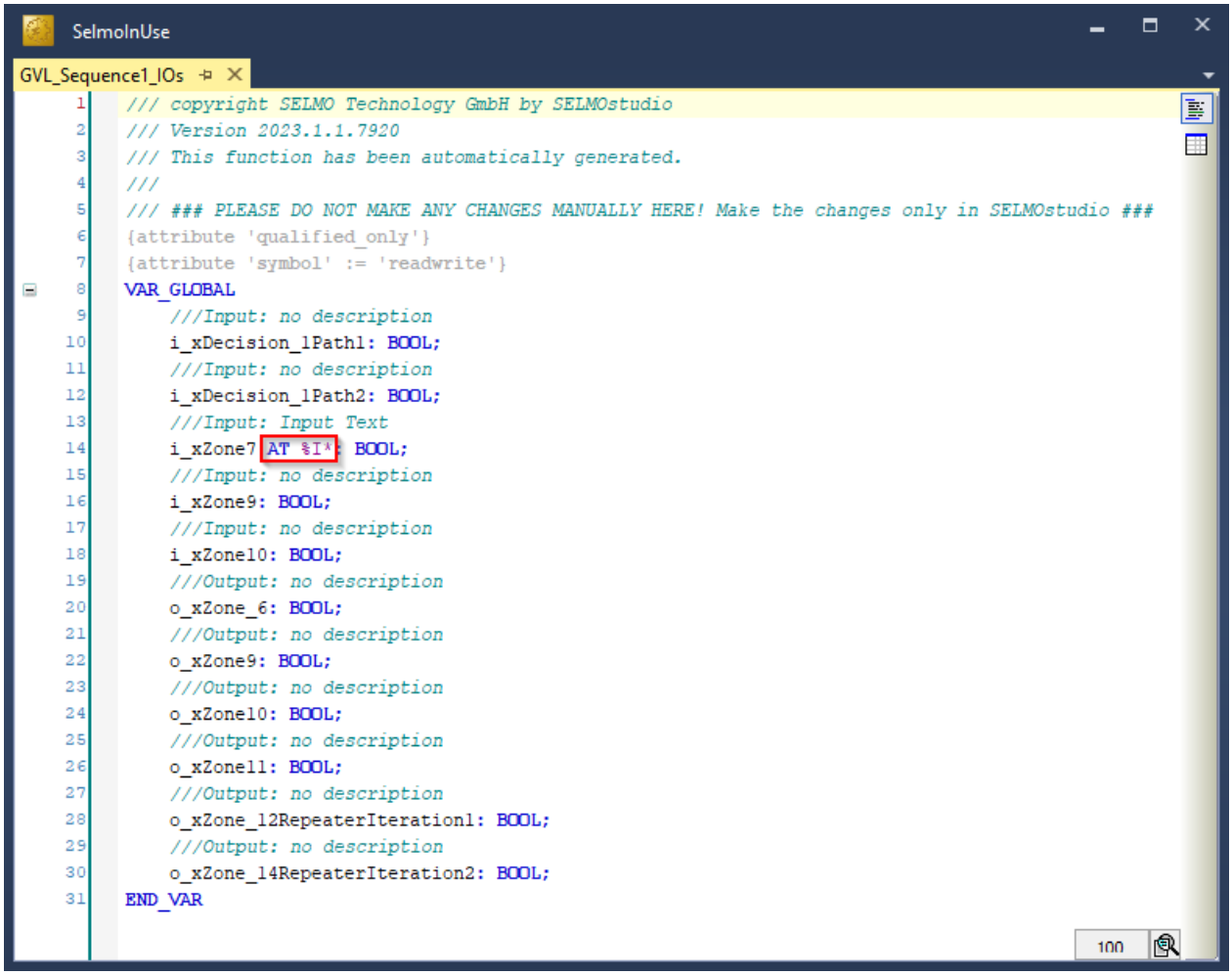
Please specify by how many milliseconds the input signal should be delayed. The delay refers to the time difference between when the signal is received and when it is processed in the system. A delay may be intended to modify the signal in a particular way or to ensure it is processed synchronously with other signals. The exact amount of time to delay the signal depends on the requirements of the system and the type of signal being processed.

Declaration as Hardware Input

If you declare the input as true, it will be declared as hardware input and included with the "AT %I*" attribute in the programming logic. This means that the variable receives a signal or value from a physical input to the system, such as a sensor or switch.



Studio

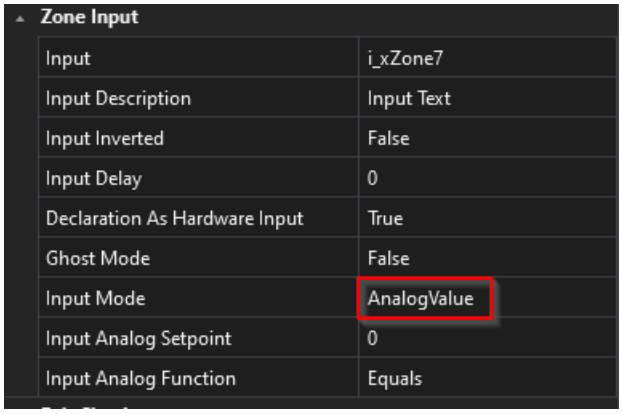


PLC

Input Mode

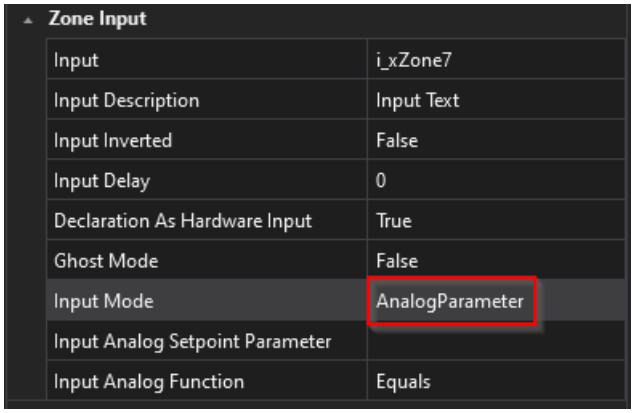
The type of signal is determined by the mode of the input. This can be either a digital or analog signal or a parameter. The mode of the input thus specifies what kind of signal is expected and how this signal should be interpreted. For example, if the input mode is set to "digital", the system expects a signal consisting of discrete values, while a continuous signal is expected in an analog input mode. In parameter mode, on the other hand, a value representing a specific parameter is expected. Overall, the type of signal that a system receives, and processes thus depends largely on the set input mode.

AnalogValue



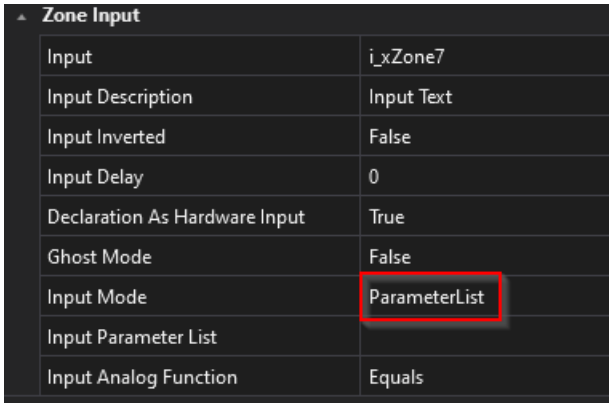
If 'AnalogValue' is selected, the zone waits internally for a digital signal while an analog value is selected externally. Forwarding takes place as soon as this function is activated and a specific condition is met. This function can be found in the PLC code under the corresponding zone.

AnalogParameter



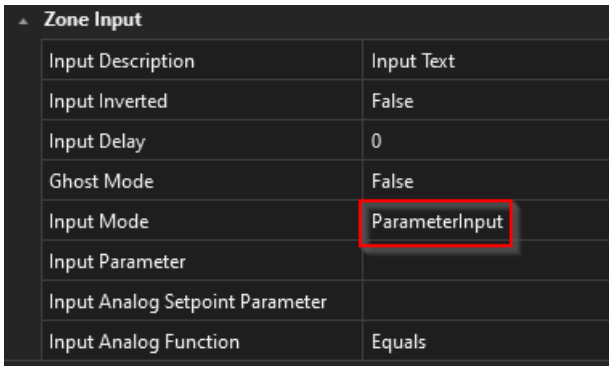
The principle corresponds to that of 'AnalogValue', but the analog step differs in that it is no longer a static value, but a parameter. The corresponding name of the selected parameter is then displayed under 'Input Analog Setpoint Parameter'. This function can also be found in the PLC code under the corresponding zone.

ParameterList



The 'ParameterList' is compared with an input signal, whereby an analog value is read in again. The difference, however, is that several comparisons can be carried out per step. To be able to use this function, a parameter list must already exist or be created so that it can be selected in the desired zone. The selected parameter list is then in 'Input Parameter List'. Wherever you want to make a comparison in the parameter list, a 'Sequence Check' must be set in the system layer.

ParameterInput



With this function, the input signal is omitted; instead, two parameters are compared with each other. In the PLC code, the IO location of the zone is no longer used under the corresponding zone, but a parameter is also specified here.

PairCheck

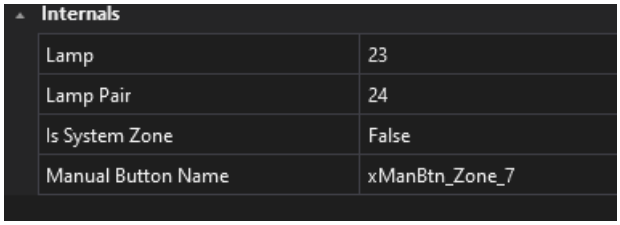
When PairCheck is active, the zone is checked with other zones to ensure that certain conditions are met, such as the presence of signal 1 and signal 2, which must not occur simultaneously.

PairCheckGroup

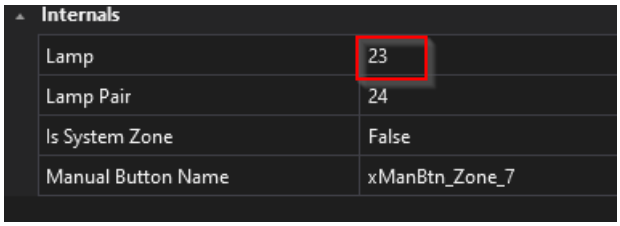
The PairCheckGroup number is used to specify whether a zone-in a check should be grouped with other zones in the same group. This grouping allows certain checks to be

applied to multiple zones that have the same PairCheckGroup number to ensure that the results are consistent.

Internals

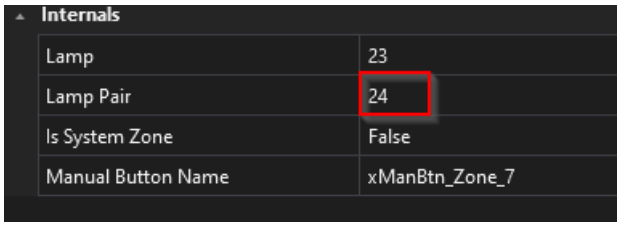


Lamp



'Lamp' refers to the index of the respective array. The zone information is stored in the array via this index. This index is then used to access the array via the HMI and the texts are displayed in the 'Waiting for' window.

LampPair

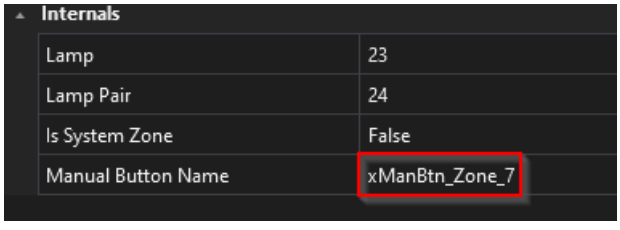


Same function as 'Lamp'.

Is System Zone

If the function is set to True, this serves as an indicator that it is a system zone that is created automatically (e.g. timer).

ManualButtonName



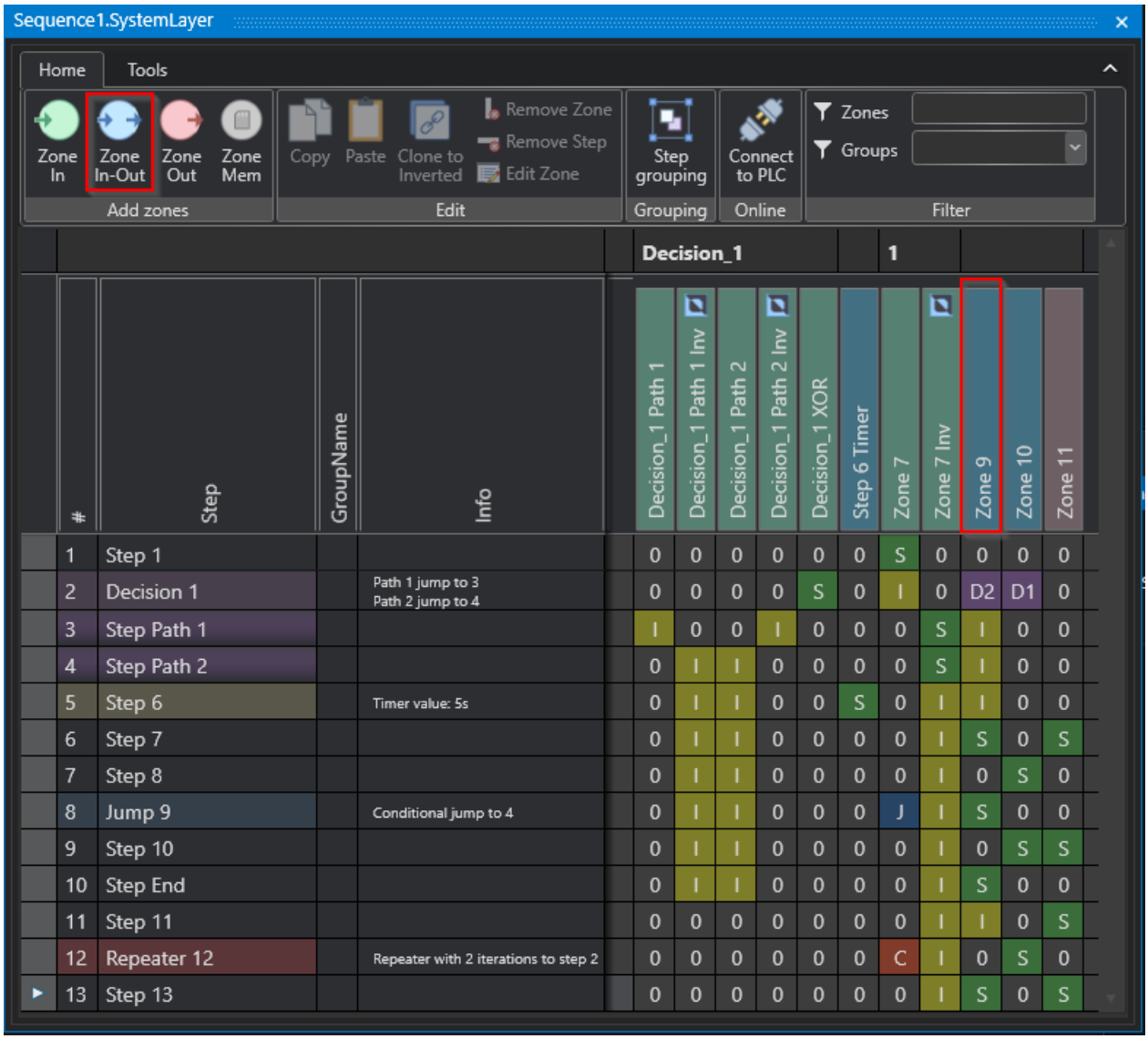
This is text-based information on how the manual button was named in the PLC.

1.3.1.2 Zone InOut

The **Zone-InOut** combines an input and output zone. To control the output, it is necessary to enter the operand 'S', so the output of the zone is controlled until the input of the zone reports the logical value 1. With this the condition is fulfilled and it can be switched to the next step. If the input of the zone is to be monitored, this is possible by entering the operand 'I'. For example, a valve of a pneumatic cylinder is controlled with the output of the zone and the end position of the cylinder is monitored with the corresponding input of the zone.

Create Zone-InOut

To create a Zone-InOut, click on the corresponding symbol.

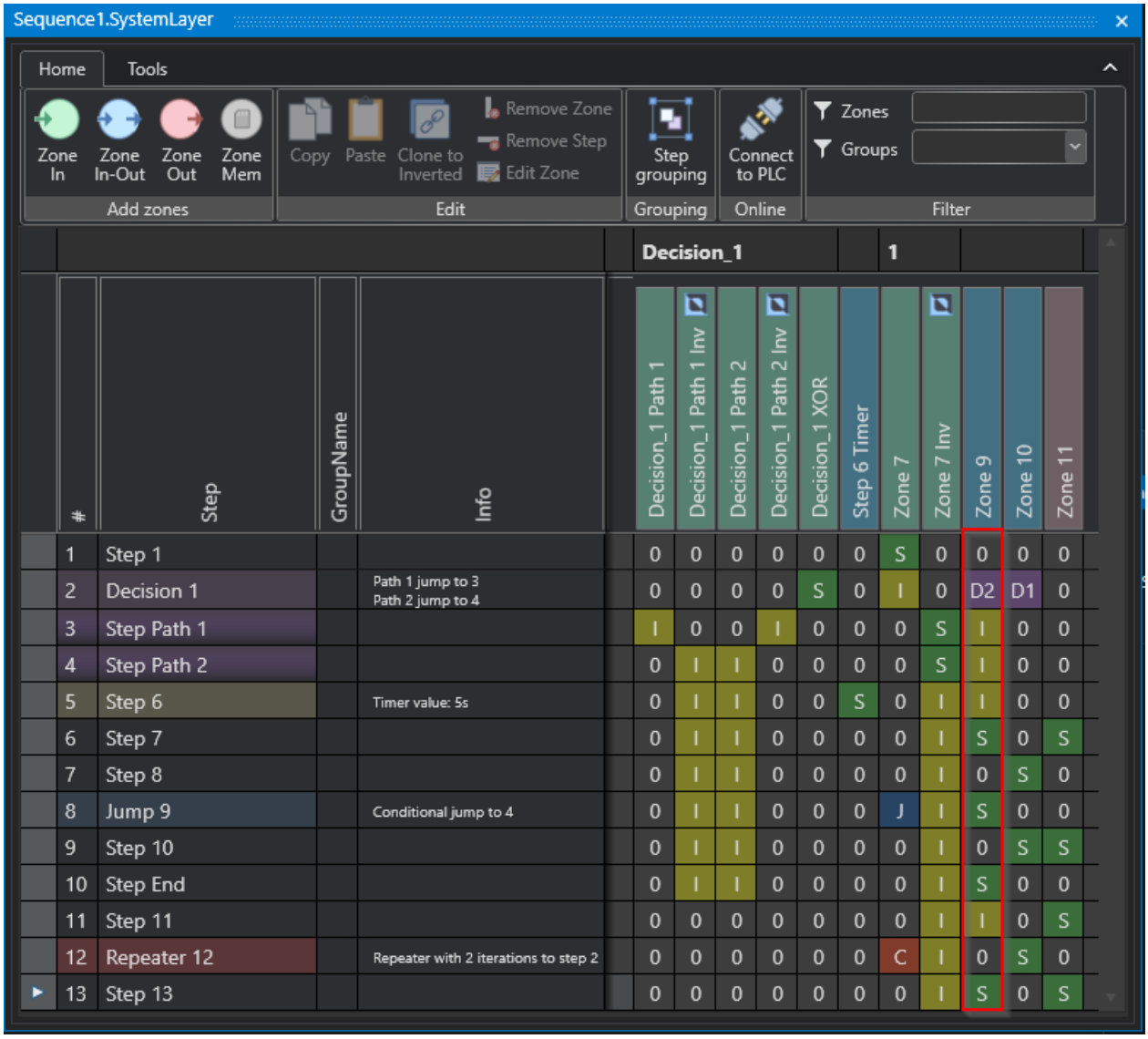


Studio

Operands of the Zone-InOut

The Zone-InOut is defined as an output with corresponding feedback via an input. Thus, outputs can be controlled until corresponding feedback is reported via the input. For example, a valve with position detection can be controlled by this. The position detection of the valve is designed in such a way that the position open or closed can be detected. To start the action (open valve), a sequence check is set. Now the output of the valve linked to the Zone-InOut is controlled until the target position of the valve is reached and the feedback is given via the linked input of the position detection. The interlock check can be used to check whether the inputs linked to the Zone-InOut remain in this state. If a "don't care" is set over the value 0 of the operand in a Zone-InOut, information of the linked input is ignored, and the linked output is not used.

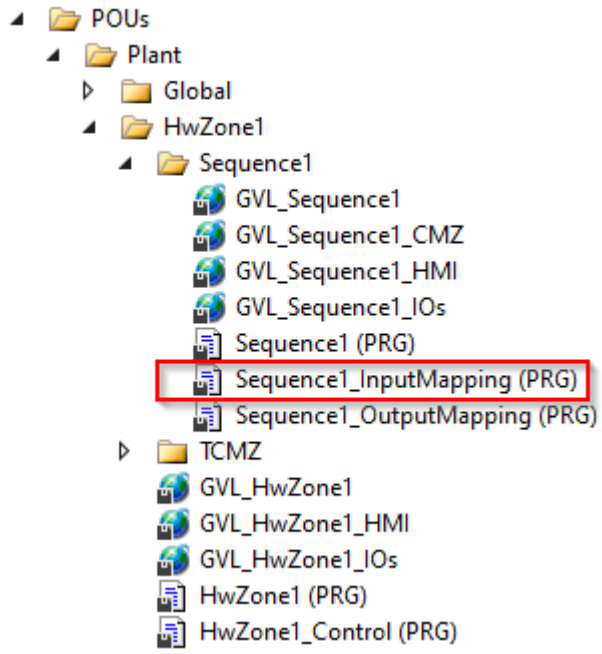
If a Zone-InOut is to be monitored in the step, it is necessary to enter the operand 'I'. With the operands 'S' a change of state from logical 0 to logical 1 is waited for. The Zone-InOut is used to process input signals and output signals such as cylinders, frequency converters, contactors etc.



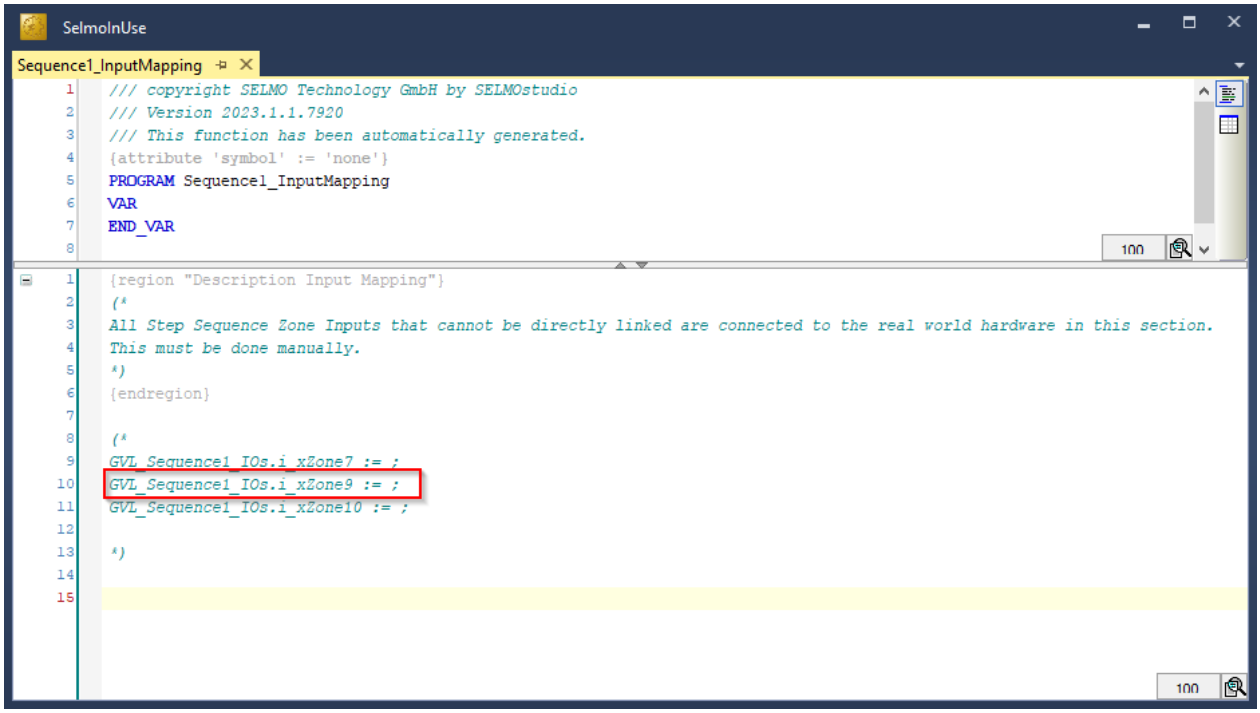
Studio

Assignment of signals to the Zone-InOut

Further the input of the Zone-InOut is entered in the code and can be linked to a signal, see following code:

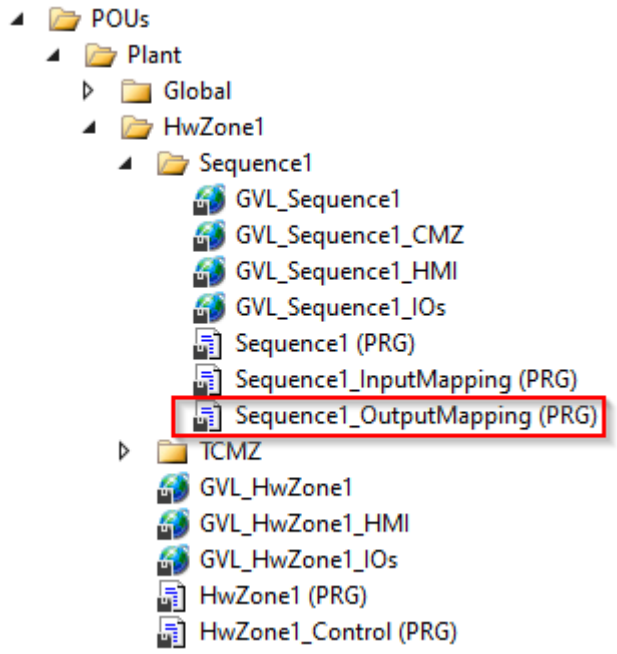


PLC

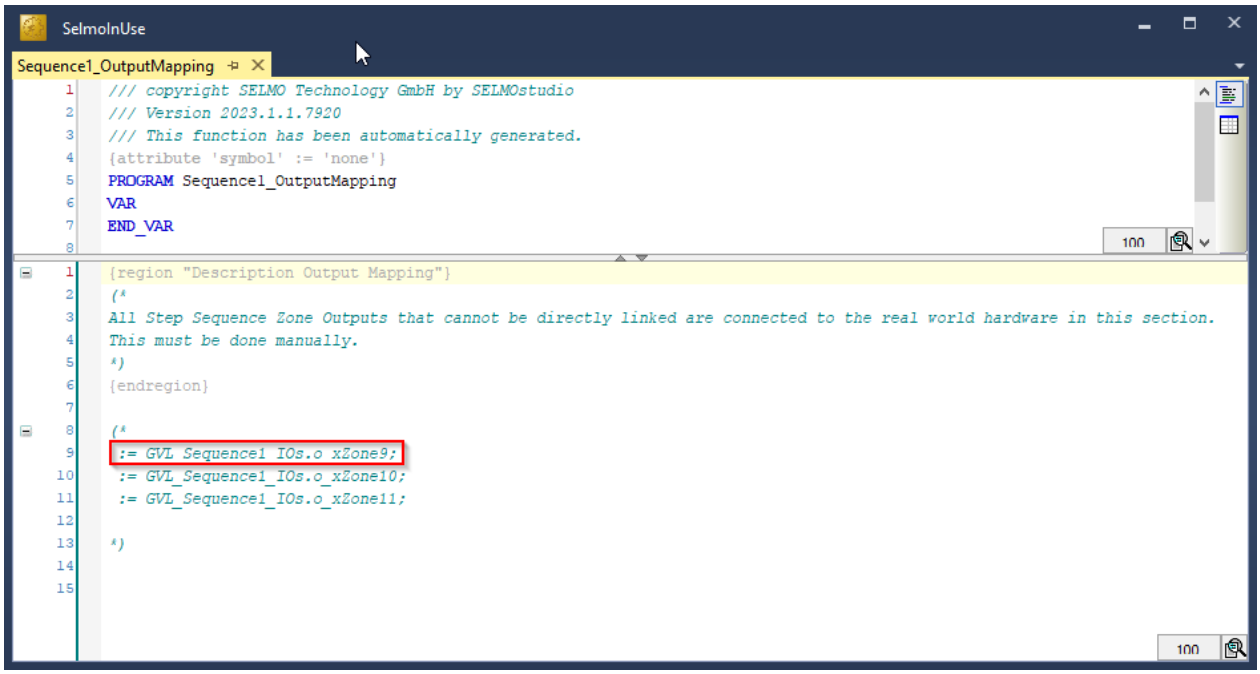


PLC

Furthermore, the output of the Zone-InOut is entered in the code and can be linked to a signal, see the following code:

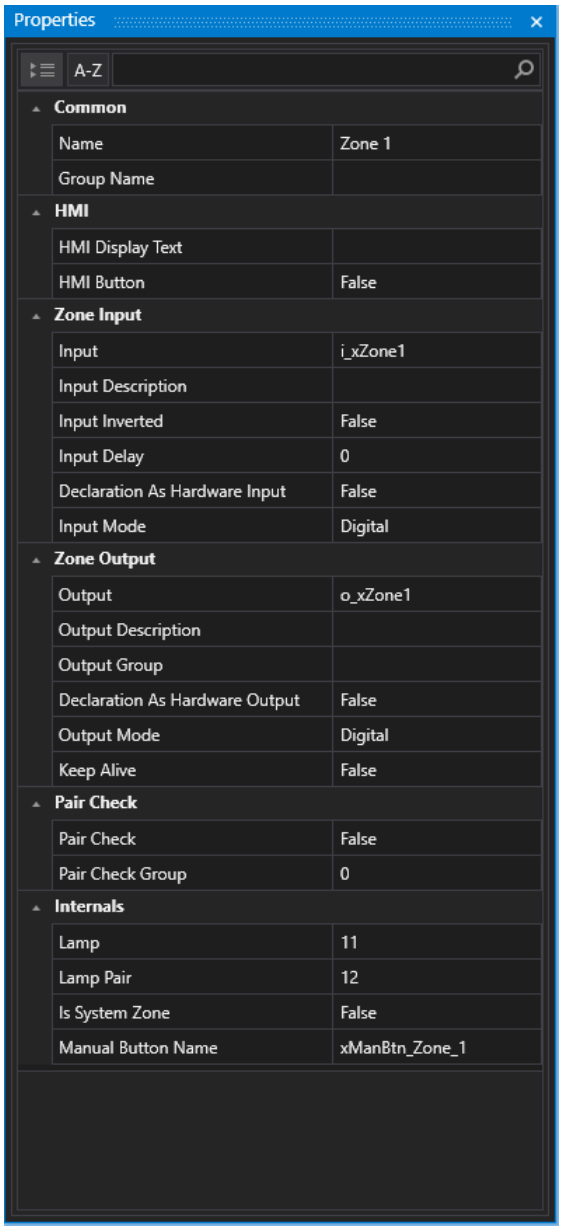


PLC



PLC

Properties



Studio

- **Name, Group Name, HMI Display Text, HMI Button, HMI Button Text, Input, Input Description, Input Inverted, Input Delay, Declaration as Hardware Input, Ghost Mode, Input Mode, Keep Alive, Output, Output Description, Output Groub, Declaration as Hardware Output, Output Distribution, Output Distribution Stored, PairCheck, PairCheckGroup, Lamp, Lamp Pair, Is System Zone, Manual Button Name**

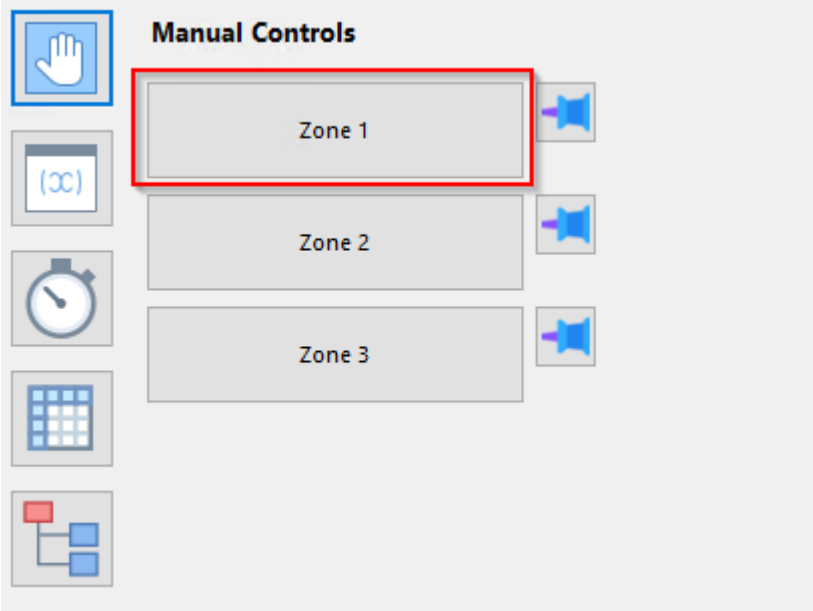
See Zone-In

HMI Button

The HMI Button is a user interface that functions only in Manual mode and is used to control the output. Unlike the automatic mode, where the system operates automatically, the manual mode gives the user the ability to make manual interventions.

The HMI button is pressed to control the output until a feedback signal (input of the zone) becomes active. The feedback signal provides feedback to the system as to whether the desired state has been achieved or if changes need to be made. When MXIC cross-locking is active, the output cannot be controlled, and the HMI displays information about the zone with which the interlocking is taking place.

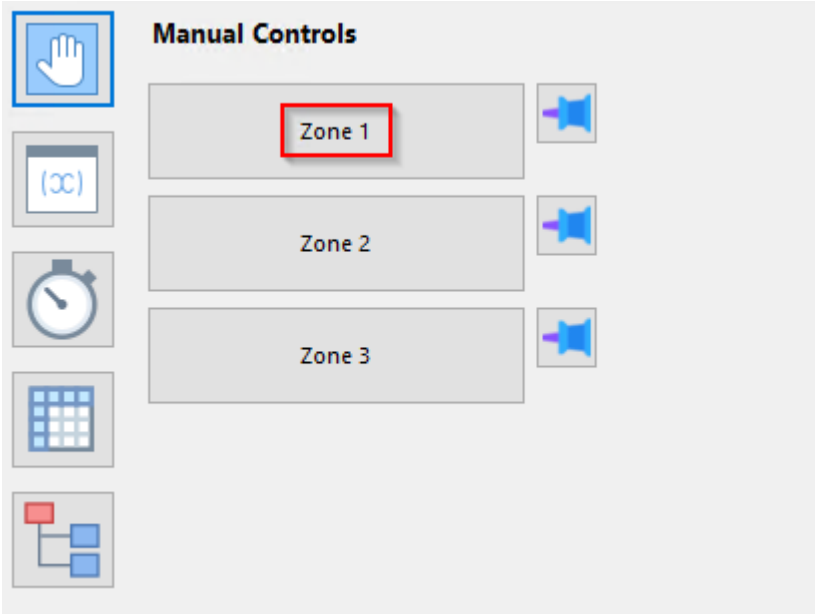
MXIC cross-locking is a safety function that prevents the zone from being activated unless certain conditions are met (zones x y are in a defined state). This is especially important in critical applications where the simultaneous occurrence of multiple events can lead to dangerous situations. Using the HMI button combined with the MXIC cross-lock ensures that only the desired output is activated and potential hazards are avoided.



HMI

HMI Button Text

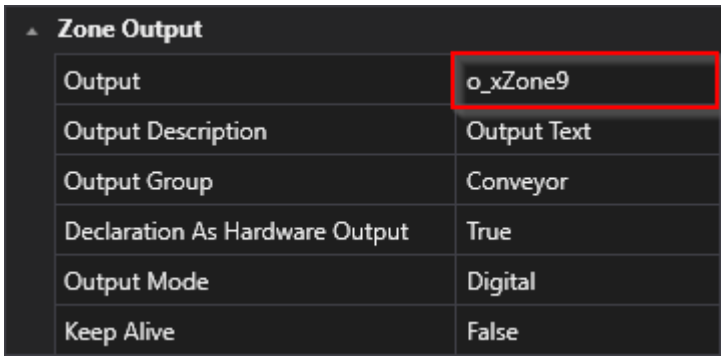
The HMI Button Text is the identifier of the zone's manual operation button and provides a clear functional designation of the zone to provide intuitive operation for the user.



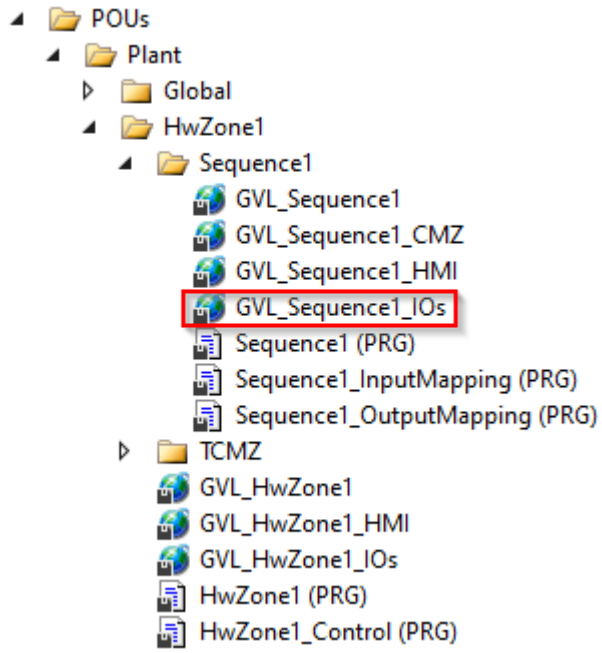
HMI

Output

In the context of PLC programming, the term "output variable name" refers to the designation or name of an output variable defined and used in the programmable logic controller (PLC) program code. For example, an output variable may be a signal from an actuator or other sink that the PLC processes to perform a specific action. The name of the output variable in the PLC program is an important aspect of controller programming because it helps make the code easier to read, understand, and maintain. Therefore, the name should be carefully chosen and descriptive to clarify the function of the output variable and increase the readability of the code.



Studio



PLC

SelmoInUse

GVL_Sequence1_IOs

```
1  /// copyright SELMO Technology GmbH by SELMOstudio
2  /// Version 2023.1.1.7920
3  /// This function has been automatically generated.
4  ///
5  /// ### PLEASE DO NOT MAKE ANY CHANGES MANUALLY HERE! Make the changes only in SELMOstudio ###
6  {attribute 'qualified_only'}
7  {attribute 'symbol' := 'readwrite'}
8  VAR_GLOBAL
9      ///Input: no description
10     i_xDecision_1Path1: BOOL;
11     ///Input: no description
12     i_xDecision_1Path2: BOOL;
13     ///Input: Input Text
14     i_xZone7 AT %I*: BOOL;
15     ///Input: no description
16     i_xZone9: BOOL;
17     ///Input: no description
18     i_xZone10: BOOL;
19     ///Output: no description
20     o_xZone_6: BOOL;
21     ///Output: Output Text
22     o_xZone9 AT %Q*: BOOL;
23     ///Output: no description
24     o_xZone10: BOOL;
25     ///Output: no description
26     o_xZone11: BOOL;
27     ///Output: no description
28     o_xZone_12RepeaterIteration1: BOOL;
29     ///Output: no description
30     o_xZone_14RepeaterIteration2: BOOL;
31 END_VAR
```

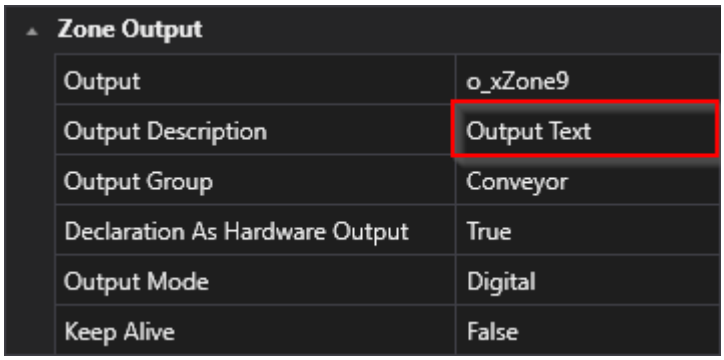
100

PLC

Output Description

In PLC programming, it is important that each output variable in the code has a meaningful description. Such a description helps other programmers or maintenance personnel to understand and edit the code more easily. The output variable description should ideally describe the purpose and operation of the variable.

A clear and concise description of the output variable can also ensure that it is properly configured and calibrated before it is integrated into the PLC program. It also aids in debugging and troubleshooting the code by allowing the programmer to quickly determine which output variables are affected. Therefore. the output variable description should be considered an important part of the PLC program documentation to increase code efficiency, maintainability and freedom from errors.



Studio

SelmoInUse

GVL_Sequence1_IOs

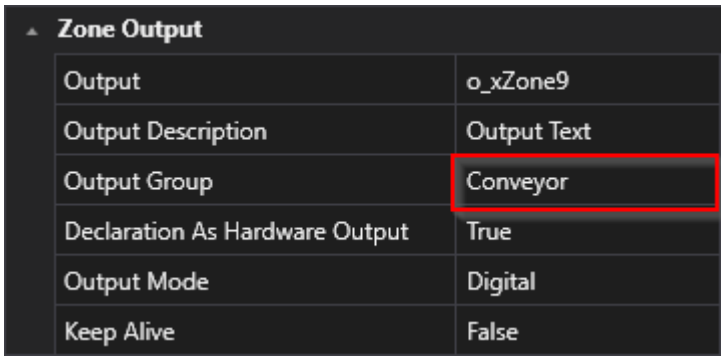
```
1  /// copyright SELMO Technology GmbH by SELMOstudio
2  /// Version 2023.1.1.7920
3  /// This function has been automatically generated.
4  ///
5  /// ### PLEASE DO NOT MAKE ANY CHANGES MANUALLY HERE! Make the changes only in SELMOstudio ###
6  {attribute 'qualified_only'}
7  {attribute 'symbol' := 'readwrite'}
8  VAR_GLOBAL
9      ///Input: no description
10     i_xDecision_1Path1: BOOL;
11     ///Input: no description
12     i_xDecision_1Path2: BOOL;
13     ///Input: Input Text
14     i_xZone7 AT %I*: BOOL;
15     ///Input: no description
16     i_xZone9: BOOL;
17     ///Input: no description
18     i_xZone10: BOOL;
19     ///Output: no description
20     o_xZone_6: BOOL;
21     ///Output: Output Text
22     o_xZone9 AT %Q*: BOOL;
23     ///Output: no description
24     o_xZone10: BOOL;
25     ///Output: no description
26     o_xZone11: BOOL;
27     ///Output: no description
28     o_xZone_12RepeaterIteration1: BOOL;
29     ///Output: no description
30     o_xZone_14RepeaterIteration2: BOOL;
31 END_VAR
```

100

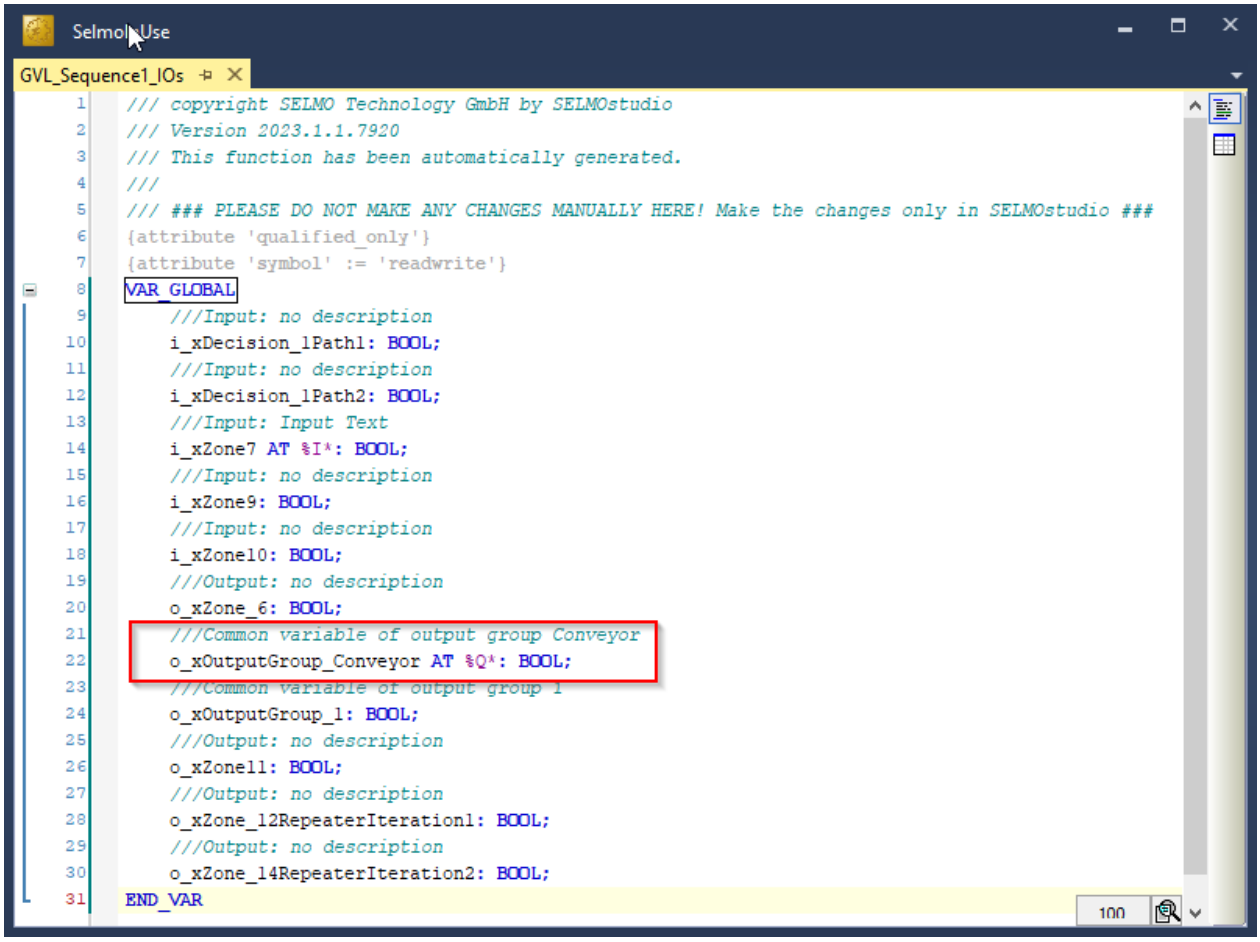
PLC

Output Group

The output group function enables the connection of several zones of the same type. Here, only one common output is used for all zones within the group. This means that if you combine several zones into one group, all these zones will control one output.



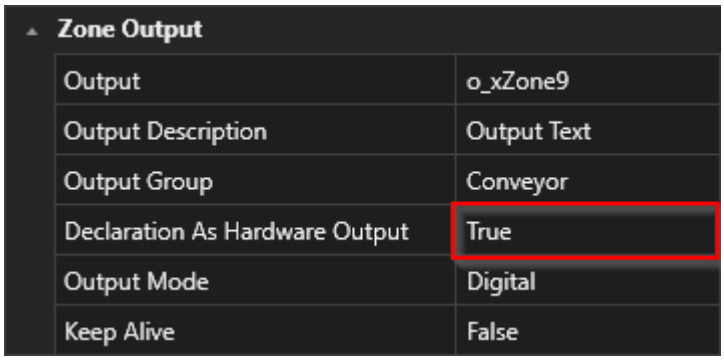
Studio



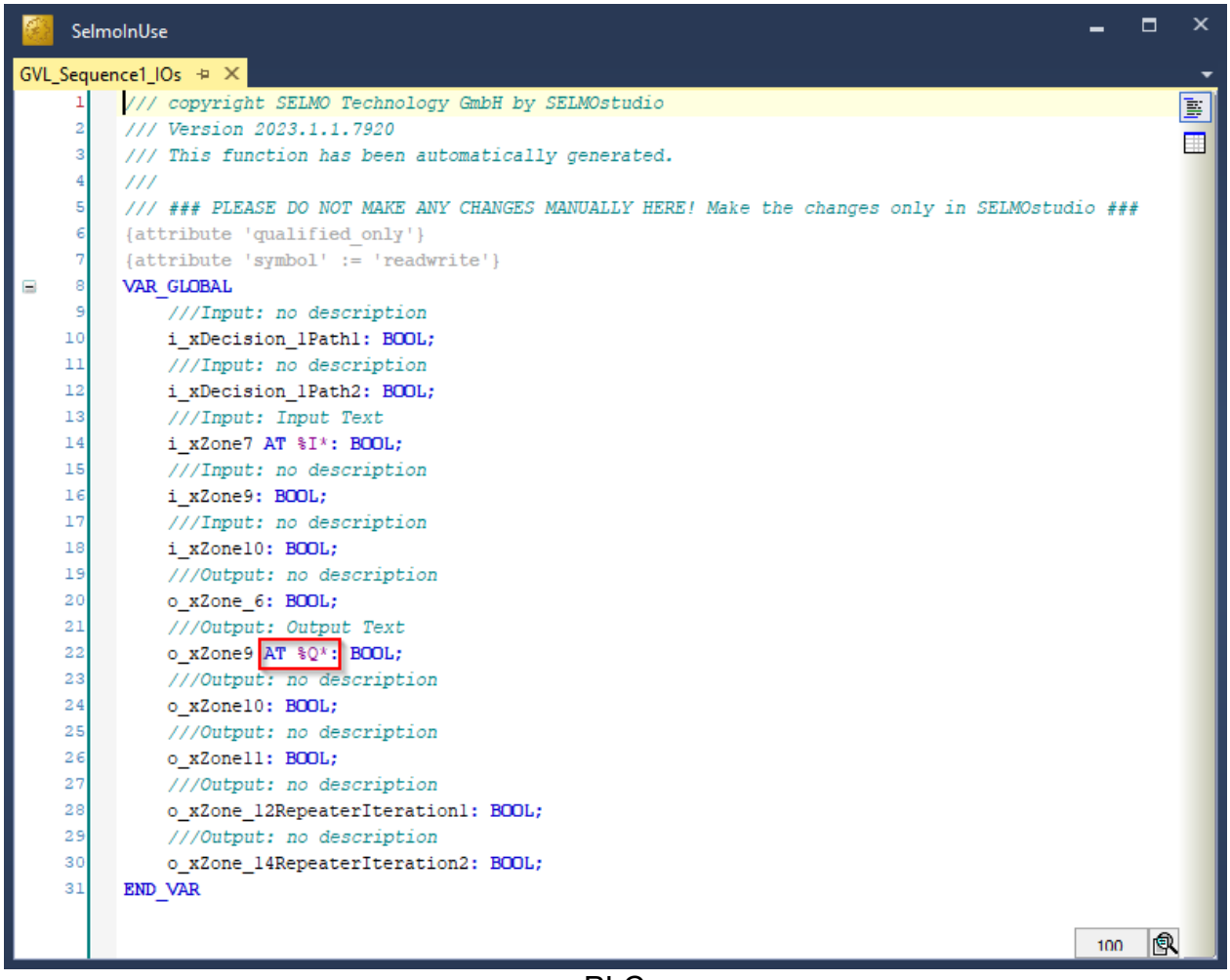
PLC

Declaration as Hardware Output

If you declare the output as "true", it will be declared as hardware output and included with the "AT %Q*" attribute in the programming logic. This means that the variable sends a signal or value on a physical output of the system, such as from a valve or an inverter.

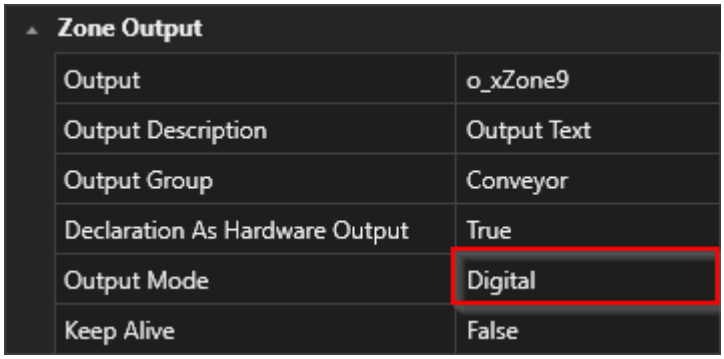


Studio



PLC

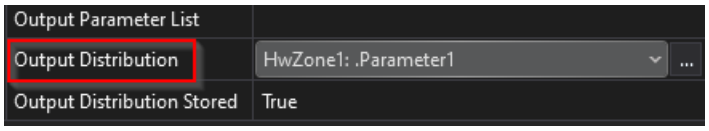
Output Mode



Studio

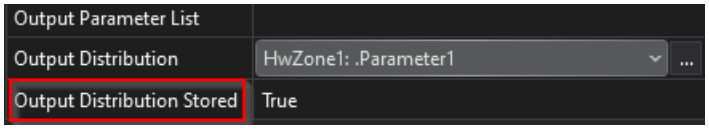
The type of signal is determined by the mode of the output. This can be either a digital or analog signal or a parameter. The mode of the output thus specifies what type of signal is expected and how this signal should be interpreted. For example, if the output mode is set to "digital", the system will send a signal consisting of a discrete value, while an analog output mode will send a continuous signal. A parameter mode, on the other hand, sends a value that represents a specific parameter. Overall, the type of signal that a system receives, and processes depends largely on the output mode that is set.

Output Distribution



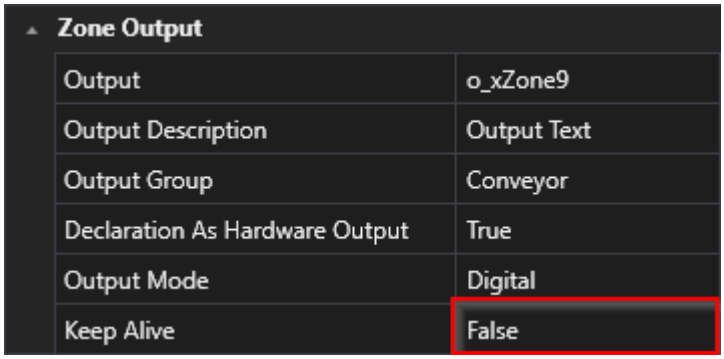
This function copies a value to another parameter. The output value from Zone is assigned to the parameter. The output value can be used as required in the program.

Output Distribution Stored



Only valid if the output has been set. After the sequence check of the zone, the output distribution value remains the same.

Keep Alive



Studio

Controlling the output offers two different approaches, which are determined by the "Keep-Alive" property:

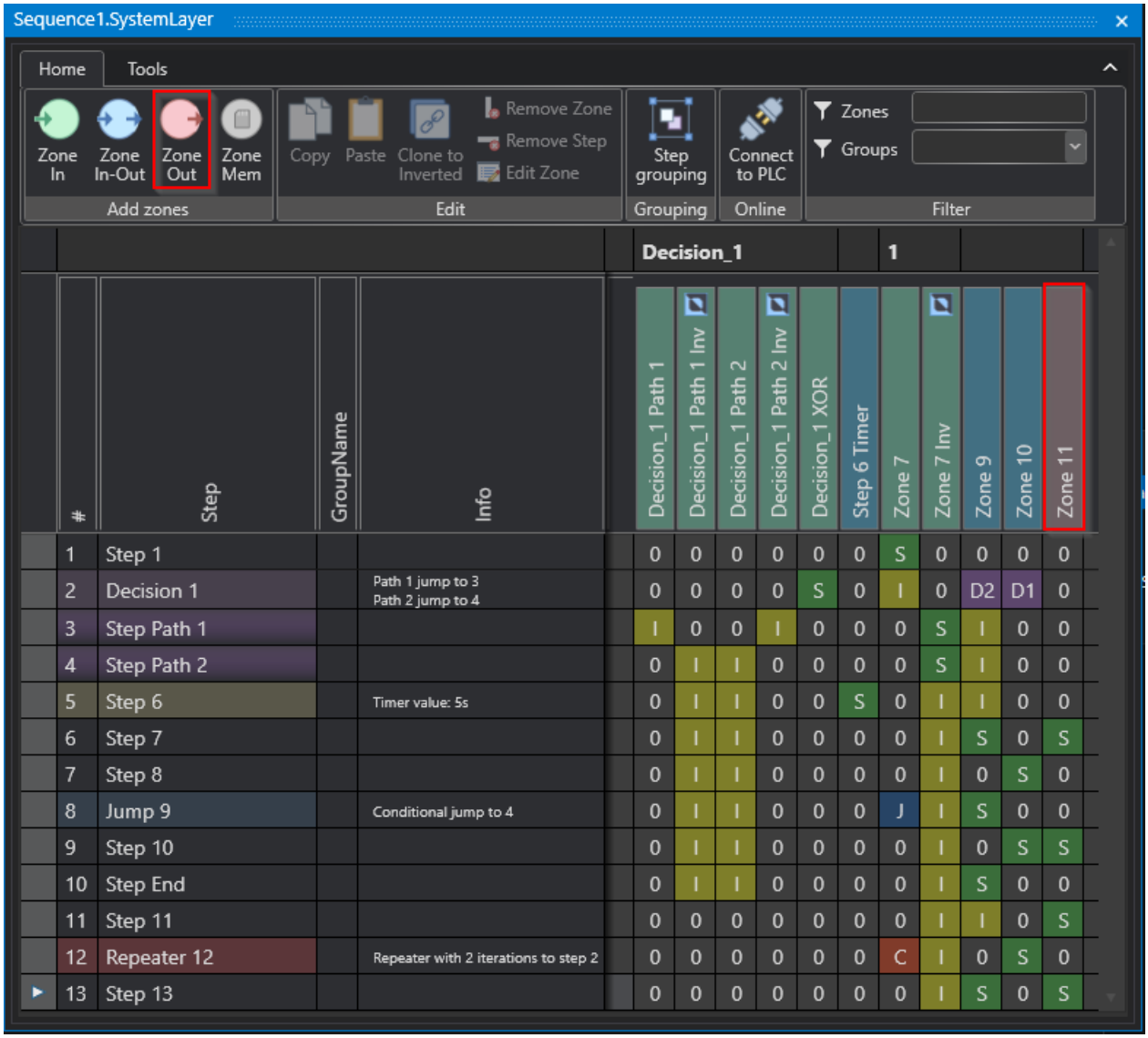
- If the "Keep-Alive" property is set to the value "False", the default approach is to control the output. In this case, the output is automatically disabled once the feedback(input) is reached.
- If, on the other hand, the "Keep Alive" property is set to the "True" value, the output is controlled independently of the feedback. This means that the control of the output does not depend on the feedback, and it remains active regardless of the feedback.

1.3.1.3 Zone Out

The **Zone-Out** is used for controlling actuators without feedback, e.g., lamps, etc. To set the output zone, it is necessary to enter the operand 'S'.

Create Zone-Out

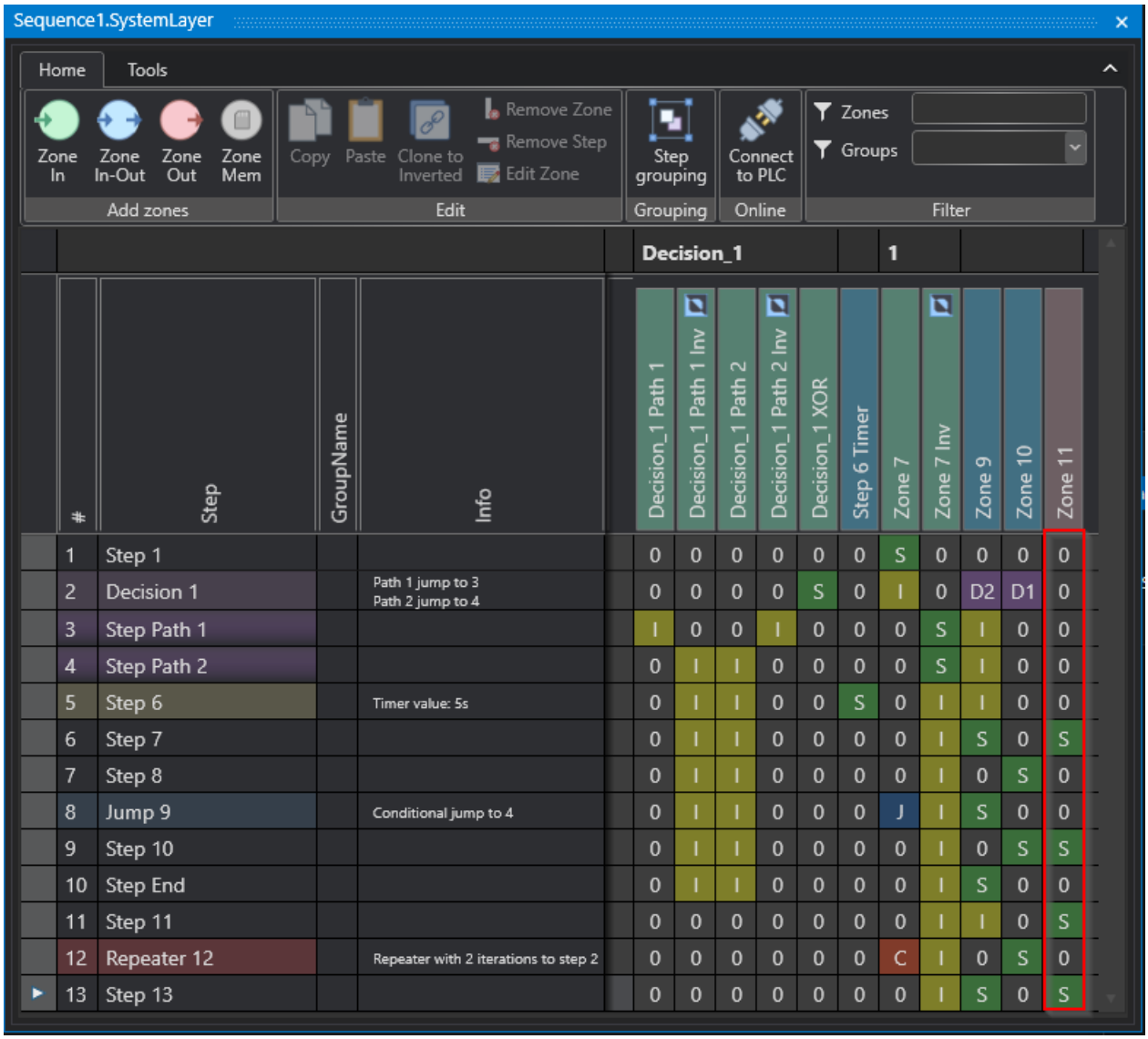
To create a Zone-Out, click on the corresponding icon.



Studio

Operands of the Zone-Out

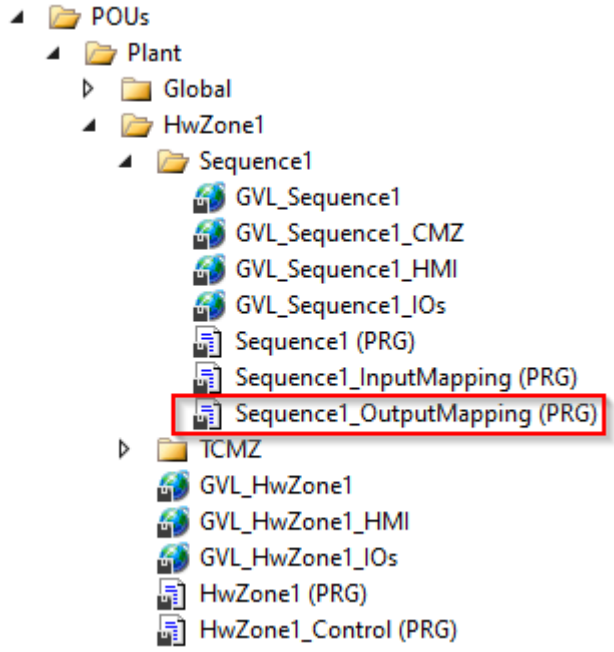
The Zone-Out switches actuators and does not expect any feedback, i.e., it operates pure switching logic. This zone type is mainly used for lights and signals, but also for actuators that have no feedback. However, it should be technically avoided, if possible, because this leads to uncertainties in the process. The operand "S" controls the output.



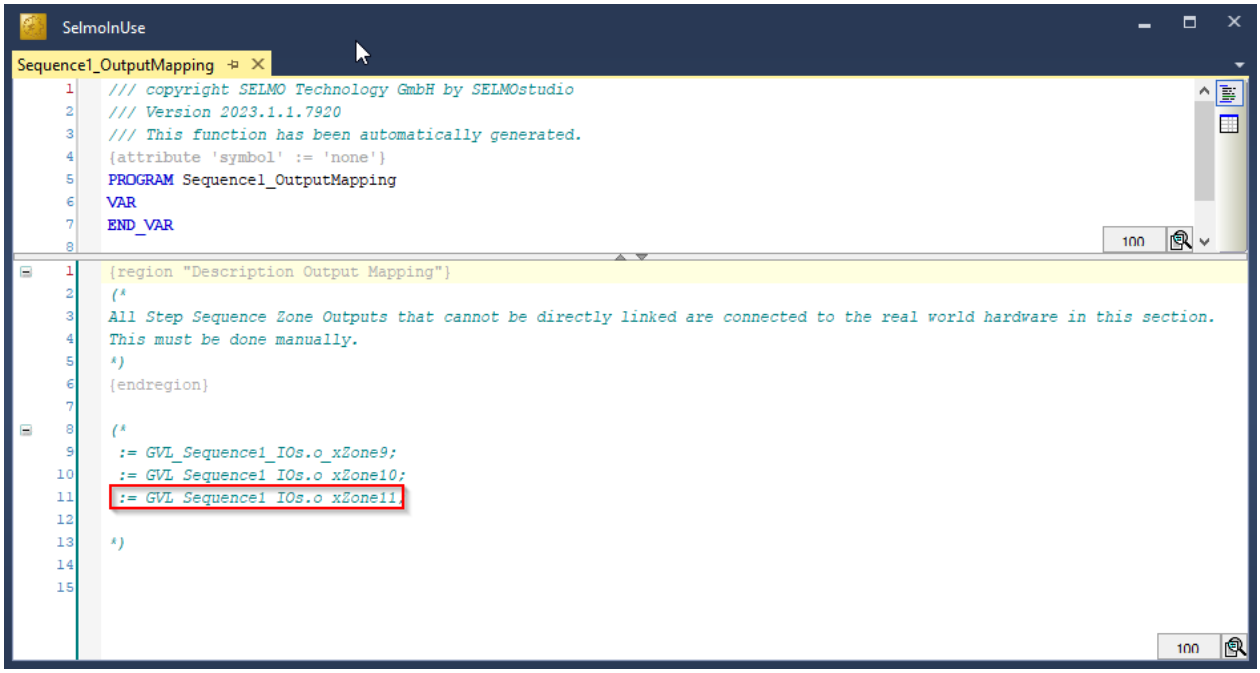
Studio

Assignment of the signals to the Zone-Out

Further, the output of the Zone-Out is entered in the code and can be linked to a signal, see the following code:

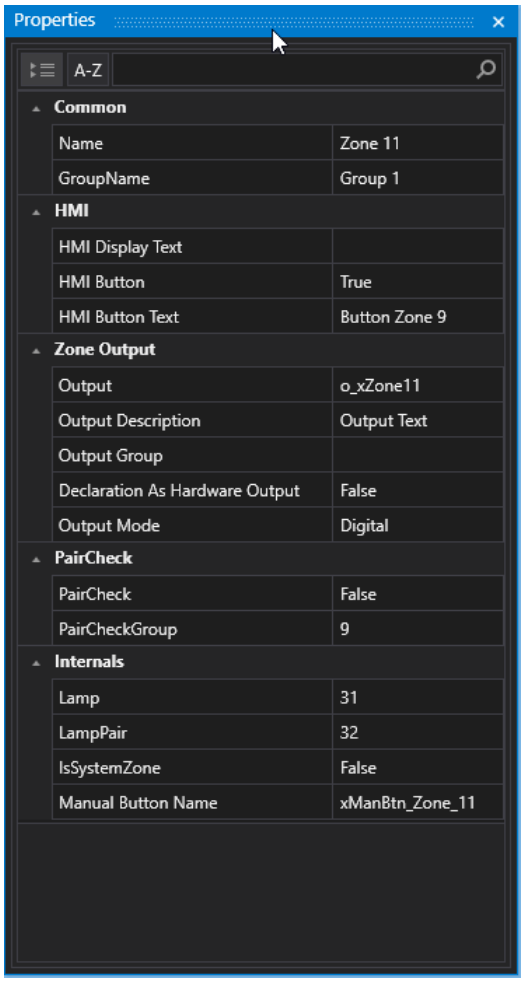


PLC



PLC

Properties

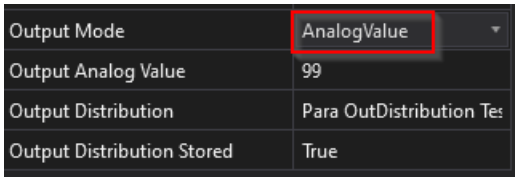


Studio

Output Mode

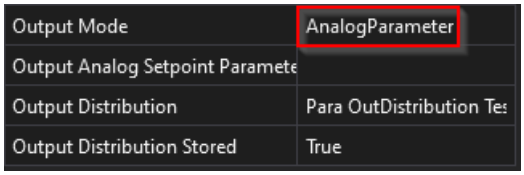
The type of signal is determined by the mode of the output. This can be either a digital or analog signal or a parameter. The mode of the output therefore specifies what type of signal is expected and how this signal should be interpreted. For example, if the output mode is set to "digital", the system will send a signal consisting of a discrete value, whereas an analog output mode will send a continuous signal. A parameter mode, on the other hand, sends a value that represents a specific parameter. Overall, the type of signal that a system receives and processes depends largely on the set output mode.

AnalogValue



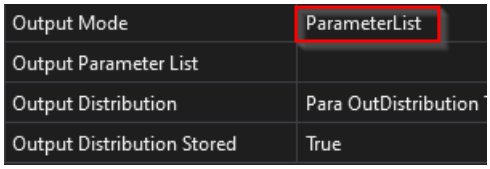
If the sequence check takes place, the value entered for Output Analog Value is transferred to the analog output, otherwise the value is 0. This function can also be found in the PLC code under the corresponding zone.

AnalogParameter



When the sequence check takes place, the selected parameter is transferred to the analog output. This function can also be found in the PLC code under the corresponding zone.

ParameterList



The 'ParameterList' is assigned a value to the output signal. Several values can be assigned to the output per step. To be able to use this function, a parameter list must already exist or be created in order to be able to select it in the desired zone. The selected parameter list is then in 'Output Parameter List'.

- **Name, Group Name, HMI Display Text, HMI Button, HMI Button Text, Input, Input Description, Input Inverted, Input Delay, Declaration as Hardware Input, Ghost Mode, Keep Alive, Output, Output Description, Output Groub, Declaration as Hardware Output, Output Distribution, Output Distribution Stored, PairCheck, PairCheckGroup, Lamp, Lamp Pair, Is System Zone, Manual Button Name**

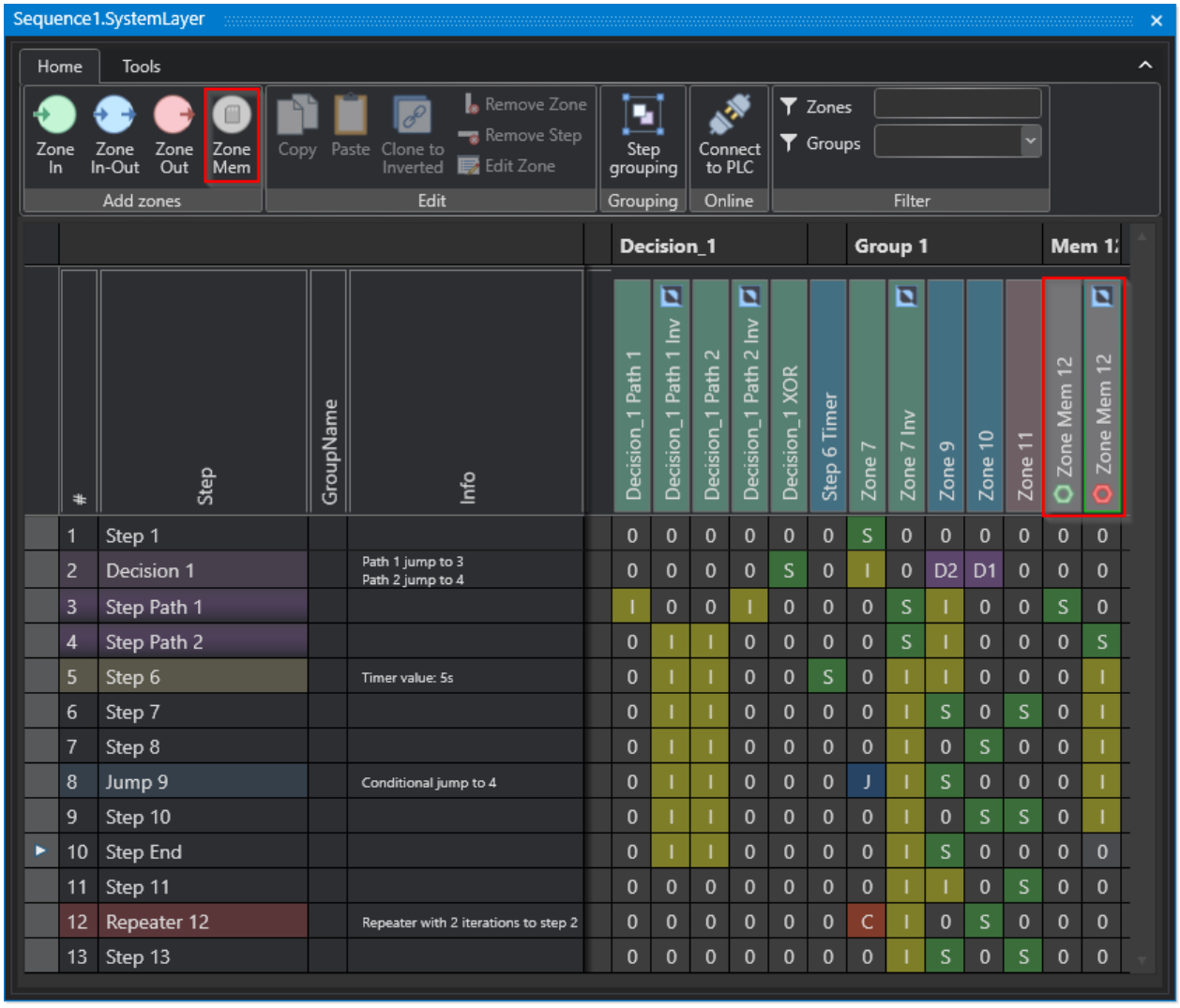
See Zone-InOut

1.3.1.4 Zone Mem

Automatically 2 zones are inserted to realize a memory that can be set and reset. The setting and resetting take place in each case with the operands 'S'. With the operands 'I' the memory can be queried. This is possible for the normal zone (query on logical 1) as well as for the inverted zone (query on logical 0).

Create Zone-Mem

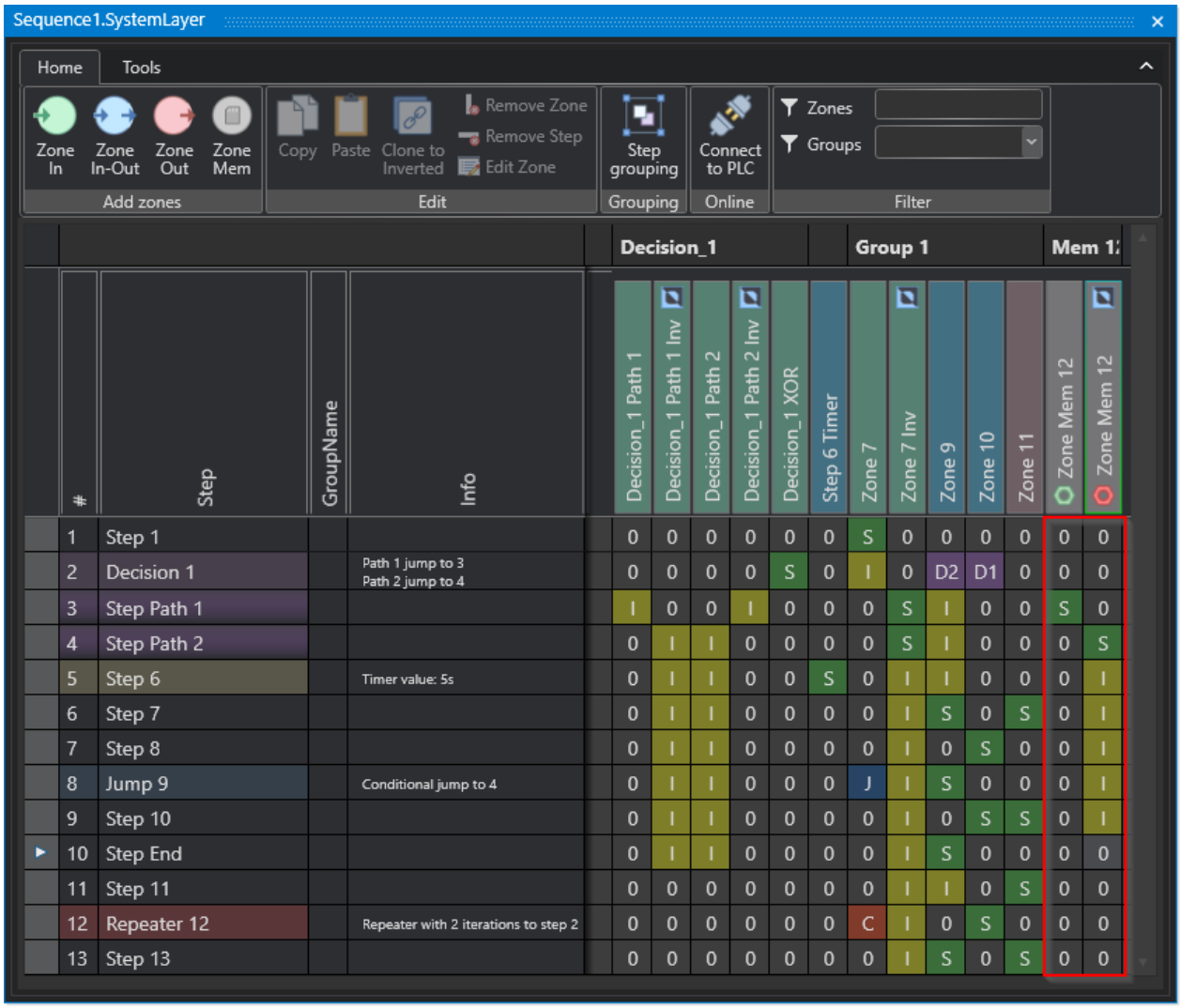
To create a zone mem, click on the corresponding icon.



Studio

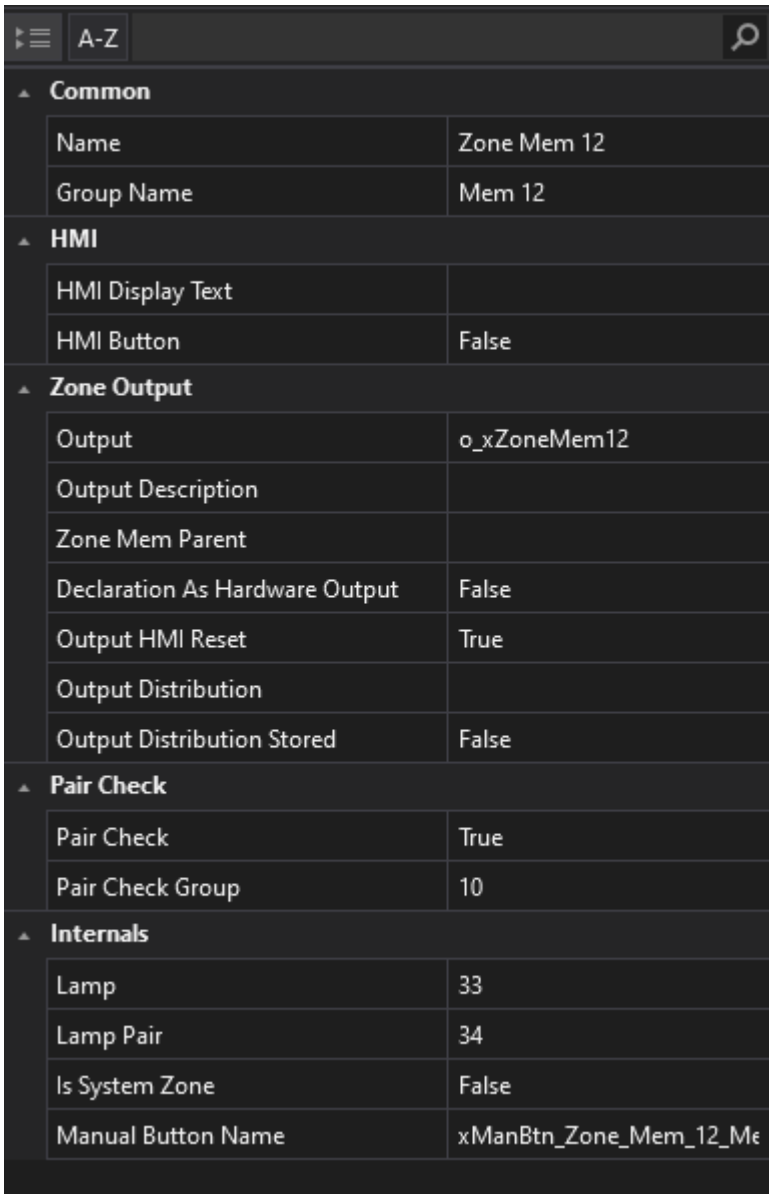
Operands of the zone mem

If a zone mem is to be monitored in the step, it is necessary to enter the operand 'I'. The operands 'S' are used to wait for a change of state from logical 0 to logical 1 or to set or reset the memory. The Zone-Mem is used to remember process states.



Studio

Properties



Studio

- **Name, Group Name, HMI Display Text, HMI Button, HMI Button, TextOutput, PairCheck, PairCheckGroup, Lamp, Lamp Pair, Is System Zone, Manual Button Name**

See Zone-InOut

Output HMI Reset

This function affects the reset button on the HMI. If this function is set to True, the memory zone is reset.

1.3.2 Bit Controlled

In the System Layer, the logical step sequence from the Logic Layer is connected with zones. The operands of the zones control and monitor the process flow. A separate system layer can be defined in each sequence. Thus, signals can be linked with zones from different Sequences.

Operands

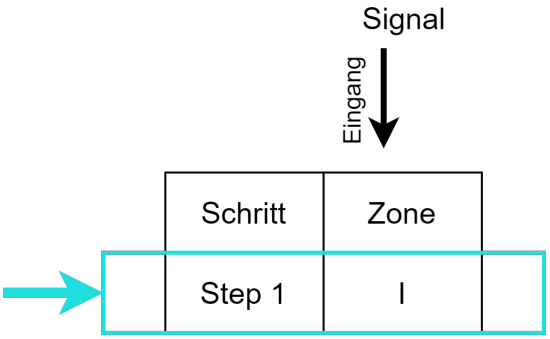
'0' don't care

This means that the zone is not monitored.

If it is an output zone, the output is switched off.

'I' Interlock check

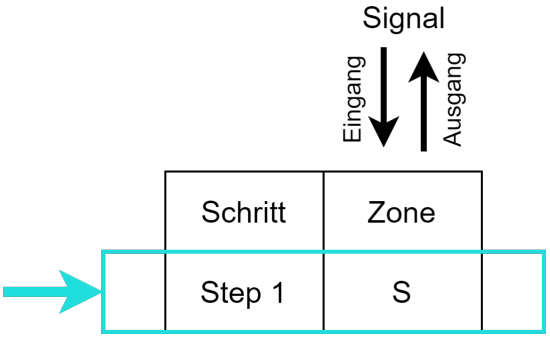
The interlock check is a function that monitors the state of the input signal linked to the zone. If the signal of the zone is not in the required state, the automatic operation of the sequence is interrupted.



With the interlock check (operand I) the value of the signal linked to the zone is monitored. The value of the signal is true.

'S' Sequence Check

Sequence Check is a function that can be set in a step of a zone using the S operand. The Boolean value of the signal associated with the zone (input of the zone) is checked for a change of state. The signal can take the values 0 (false) or 1 (true). A change of state of the signal is called a transition and is the transition from 0 to 1, or false to true. As long as the transition is not executed, the condition for a step change remains false and the step remains active. When the transition is completed, the step increment condition becomes true, and the step counter is incremented. For Zone-InOut it is valid that with the Sequence Check the output is switched until the feedback is fulfilled. For Zone-Out, only the output is set and remains active until another zone's transition satisfies the step advance circuit. Zone-Out does not stop stepping in any case.



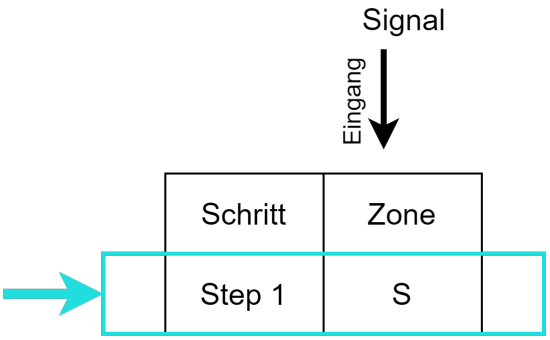
Sequence Check (Operand S) controls and monitors the transition of the signal associated with the zone. The value of the signal goes from 0 to 1, or false to true.

Inverted signal

Transitions from 1 to 0 of a signal can be checked by zones which use an inverted signal as input.

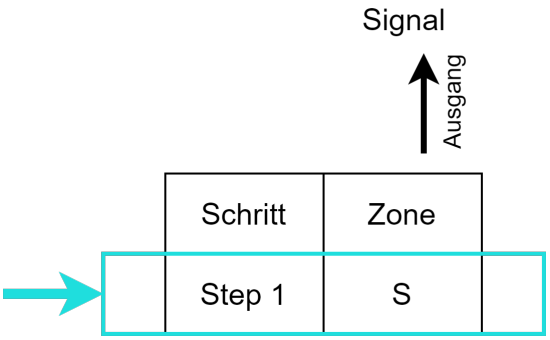
Input zone

Waits for a change of state from logical 0 to 1.



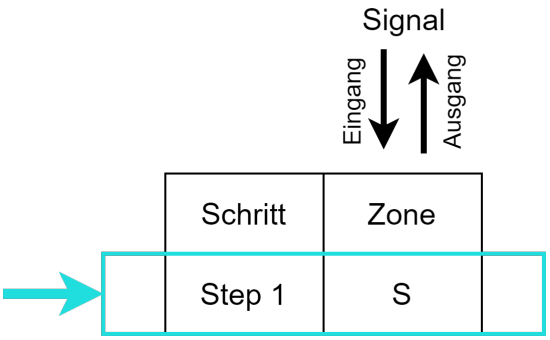
Output zone

Used to set the output to logic 1.



In-Out-Zone

An additional evaluation of the input zone takes place, the output remains active until the input zone becomes active.



'M' Monitoring check

Same function as 'Interlock check', but no interlock error is triggered and the automatic remains active.

The interlock is logged in the alarm history.

'D1' Decision Path 1

Provides the possibility to define one or more input zones, as condition for Path 1, in the Decision.

'D2' Decision Path 2

Provides the ability to define one or more input zones, as a condition for Path 2, in the Decision.

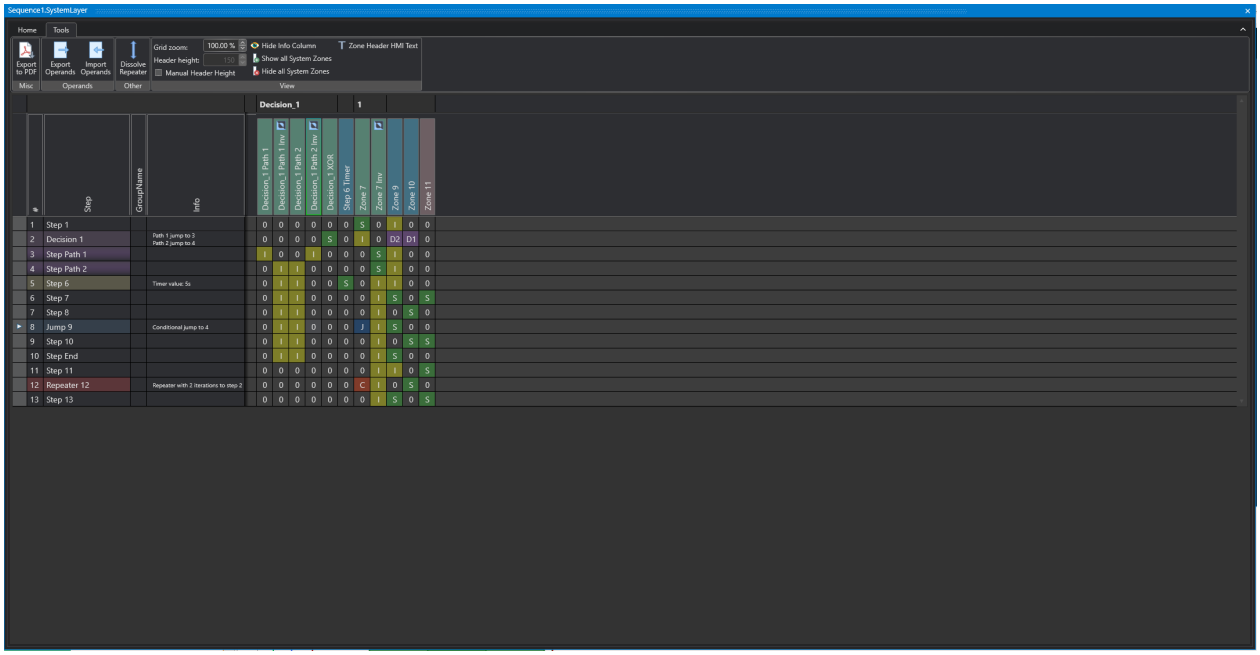
'J' Jump

Provides the ability to define one or more input zones, as a conditional jump variable, in the Jump step.

'C' Cancel

Used to define an input zone as a cancel variable for the repeater.

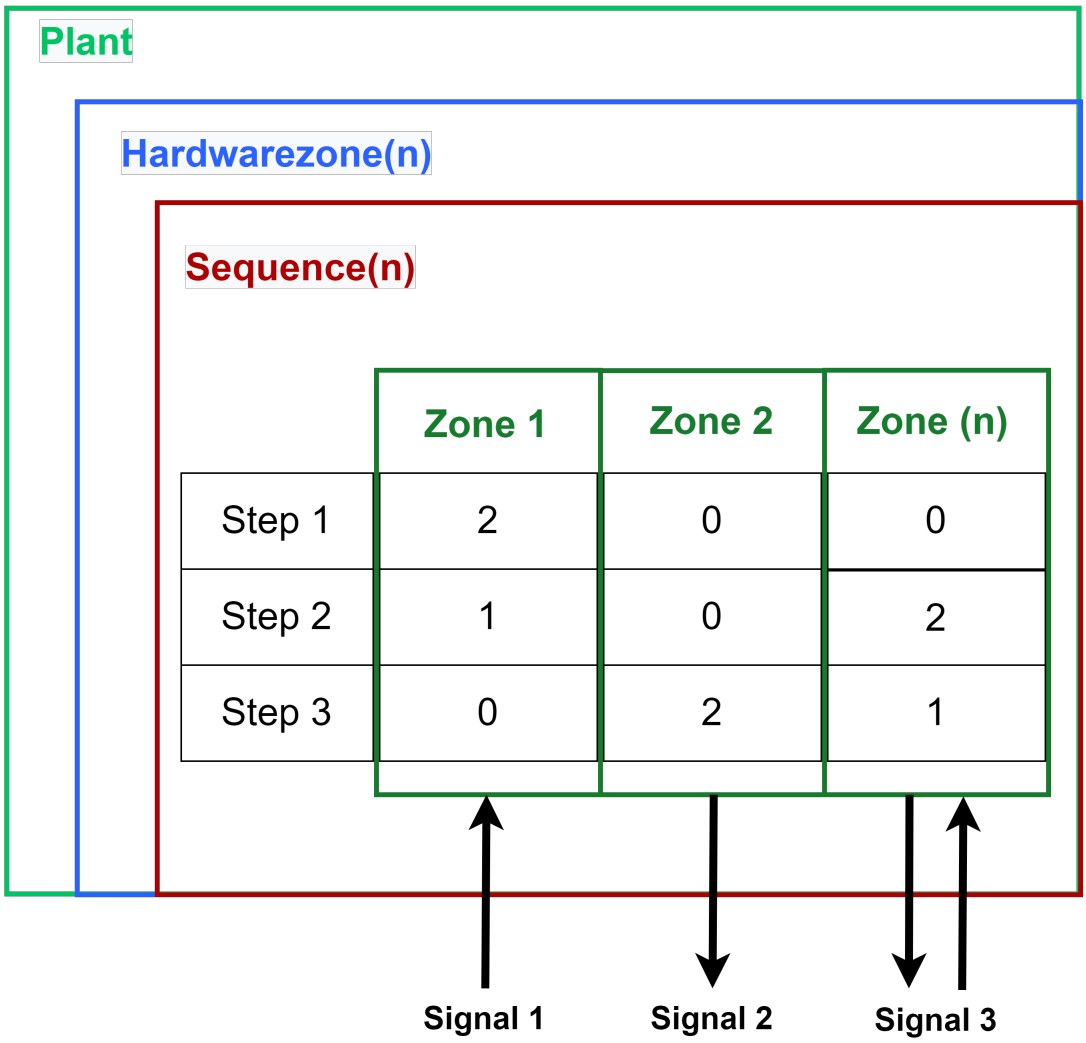
With the help of the operands, it is now defined in which step the system must have which state or which state the system must assume. By means of the Zone-InOut and Zone-Out is defined which outputs are switched in which step. This means that the defined states can also be the only valid ones. The system displays all other states of the system as errors and diagnoses the deviation from the target state.



Studio

Target/actual comparison

The bit-precise definition of the input and output signals enables a constant target-actual comparison of the system with the defined bit pattern for each process step; any deviation is detected and the deviation from the target state is displayed.



1.3.3 CMZ

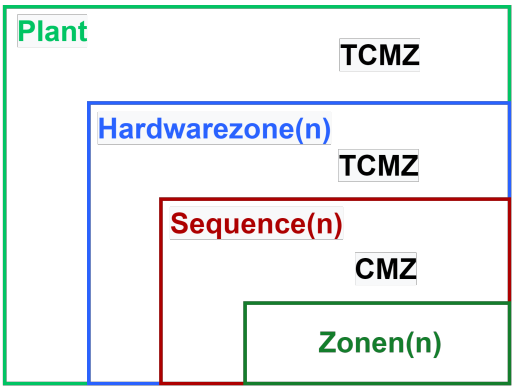
A "Constantly Monitored Zone" CMZ in machines refers to an area or zone in a machine that is constantly monitored by sensors or other monitoring mechanisms. These zones can include, for example, critical components or processes of a machine where the occurrence of a fault or malfunction can have serious consequences. These zones can be monitored by various types of sensors, such as temperature or pressure sensors, which continuously measure the values in the zone and transmit them to the machine's control system. The control system can then take appropriate action to prevent or minimize potential problems if there are deviations from normal values.

Examples of Constantly Monitored Zones in machines include:

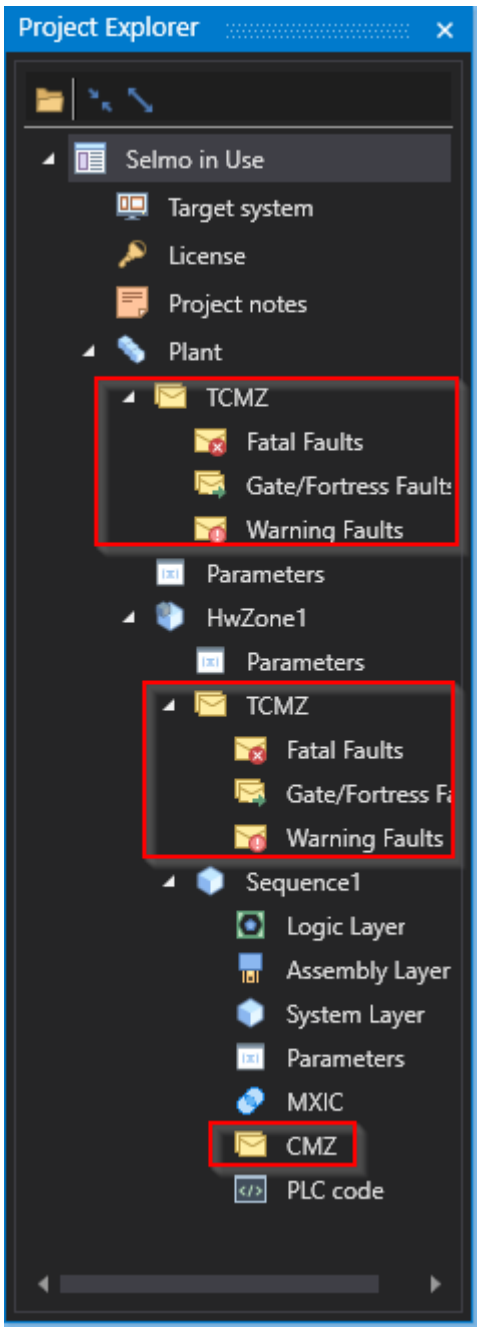
- The zone around the machining area in a CNC machine, where the temperature of the cutting tool and material being cut is constantly monitored to prevent overheating and damage.
- The zone around the burner in an industrial furnace, where the temperature and gas supply are constantly monitored to ensure firing quality and safety.
- The zone around the welding in a robotic welding machine, where the temperature and pressure of the welding equipment must be monitored to ensure consistent weld quality.

Overall, Constantly Monitored Zones on machines are an important part of safety and quality control to ensure smooth operation and protection of people and machines.

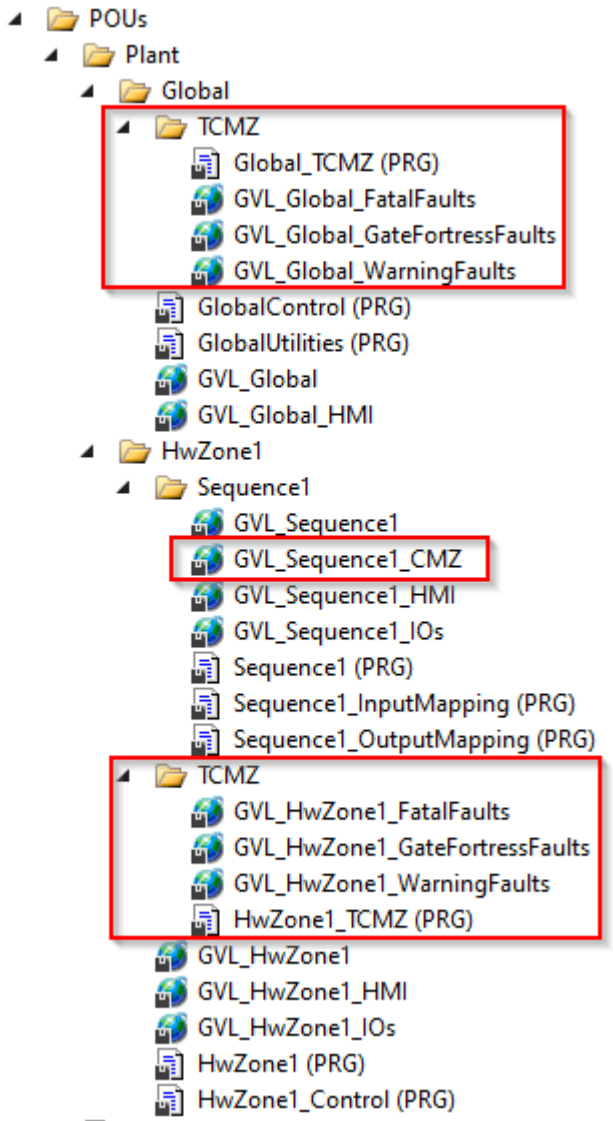
CMZs are found at the Plant, Hardware Zone and Sequence levels. The division into these levels determines which level is shut down when a fault occurs. If a fault occurs at the plant level, such as loss of compressed air, the automation of the entire plant is stopped. Errors on the hardware zone level lead to the stop of the respective hardware zone and errors in Sequences stop the respective Sequence. The errors are logged in the alarm history.



Depending on the level in which the CMZ was declared, a corresponding entry is created in the PLC code. This can be either at the Plant, Hardware Zone or Sequence level.

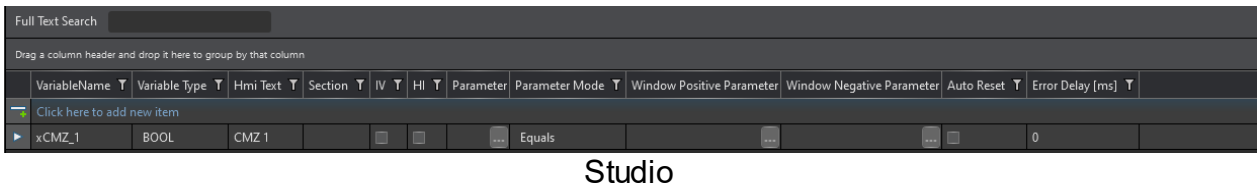


Studio



PLC

The CMZs are edited and created with the help of an editor. Thereby different options are provided, which are explained in the following:



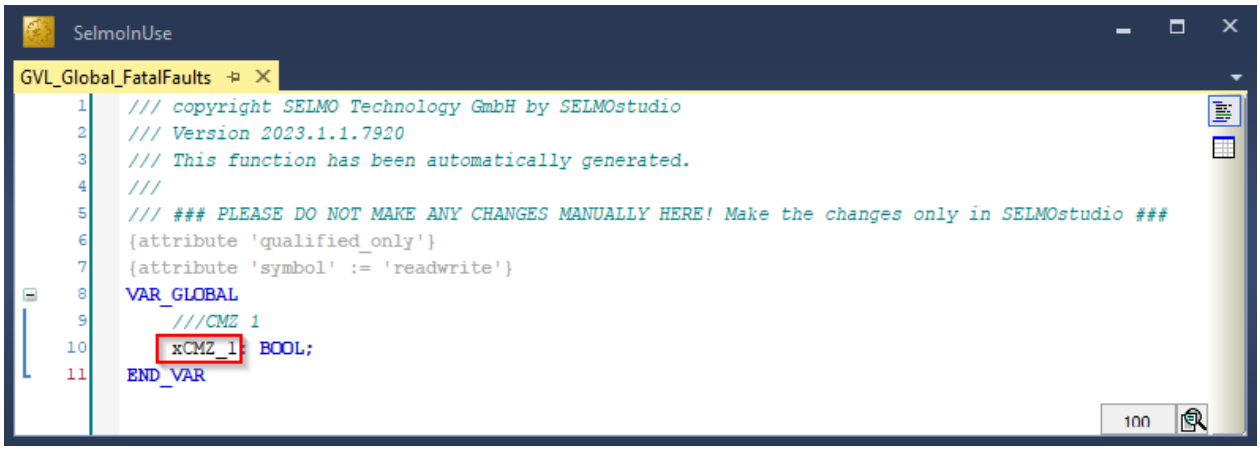
Studio

Variable Name

The name of a variable is important for the readability and comprehensibility of the code. A good variable should be unique, meaningful, and functional. That is, it should have only one meaning, describe the content or purpose of the variable, and match the data type and logic of the program.

Notice:

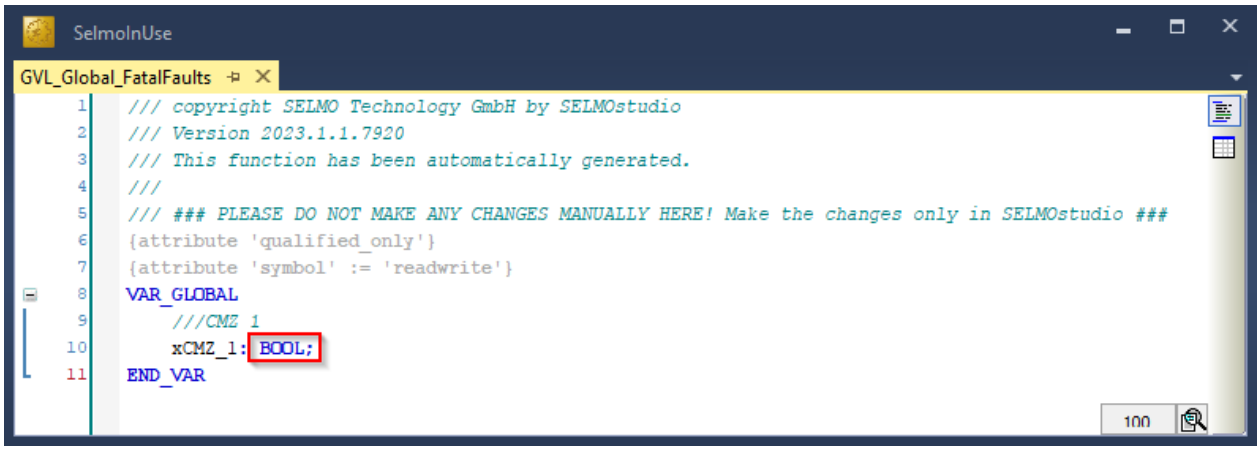
Selmo uses the PLCopen Coding Guidelines. This is an internationally recognized standard for programming automation systems, especially programmable logic controllers (PLCs). The PLCopen Coding Guidelines establish rules and best practices to help developers ensure that their PLC programs are readable, reusable and robust. The guidelines cover a variety of topics, including variable naming, code commenting, program structure, and error handling. The naming of variables in the PLCopen Coding Guidelines follows certain rules. For example, variable names should be meaningful and reflect the purpose and type of the variable. The name should be written in English and follow certain conventions, such as the use of CamelCase. Furthermore, rules for naming input and output variables, temporary variables and constants are specified. Overall, the PLCopen Coding Guideline aims to improve the readability, maintainability and robustness of PLC programs and thus contribute to higher efficiency and productivity in the development of automation systems.



PLC

Variable Type

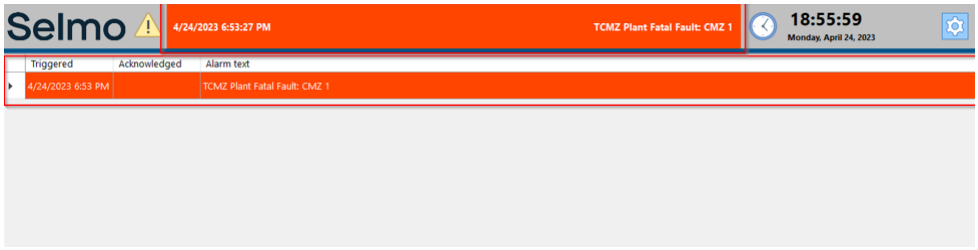
By default, the variable type is declared as a Boolean data type. However, the drop-down menu can be used to select the appropriate data type. The available data types correspond to the list of standard data types.



PLC

HMI Display Text

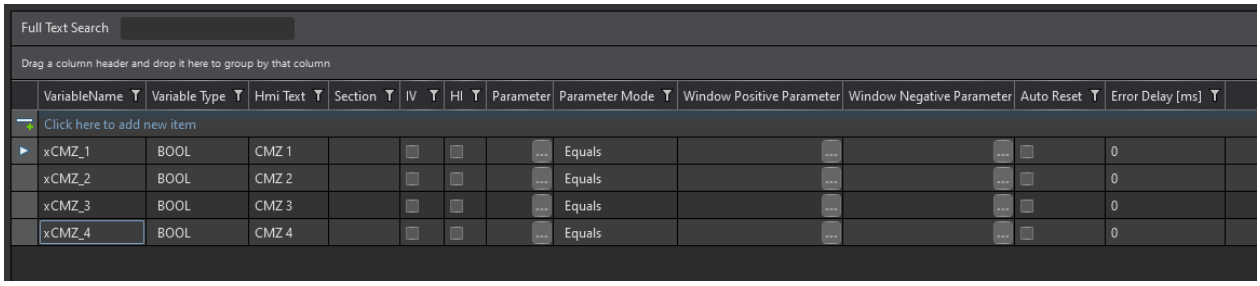
This text appears on the HMI when an error or warning occurs. It informs the user about the type and location of the problem. The text is displayed both in the Alarm Bar and on the Alarm Page of the HMI and serves as a comment of the variable in the PLC code.



HMI

Section

This is used to group and improve the clarity of the CMZ.

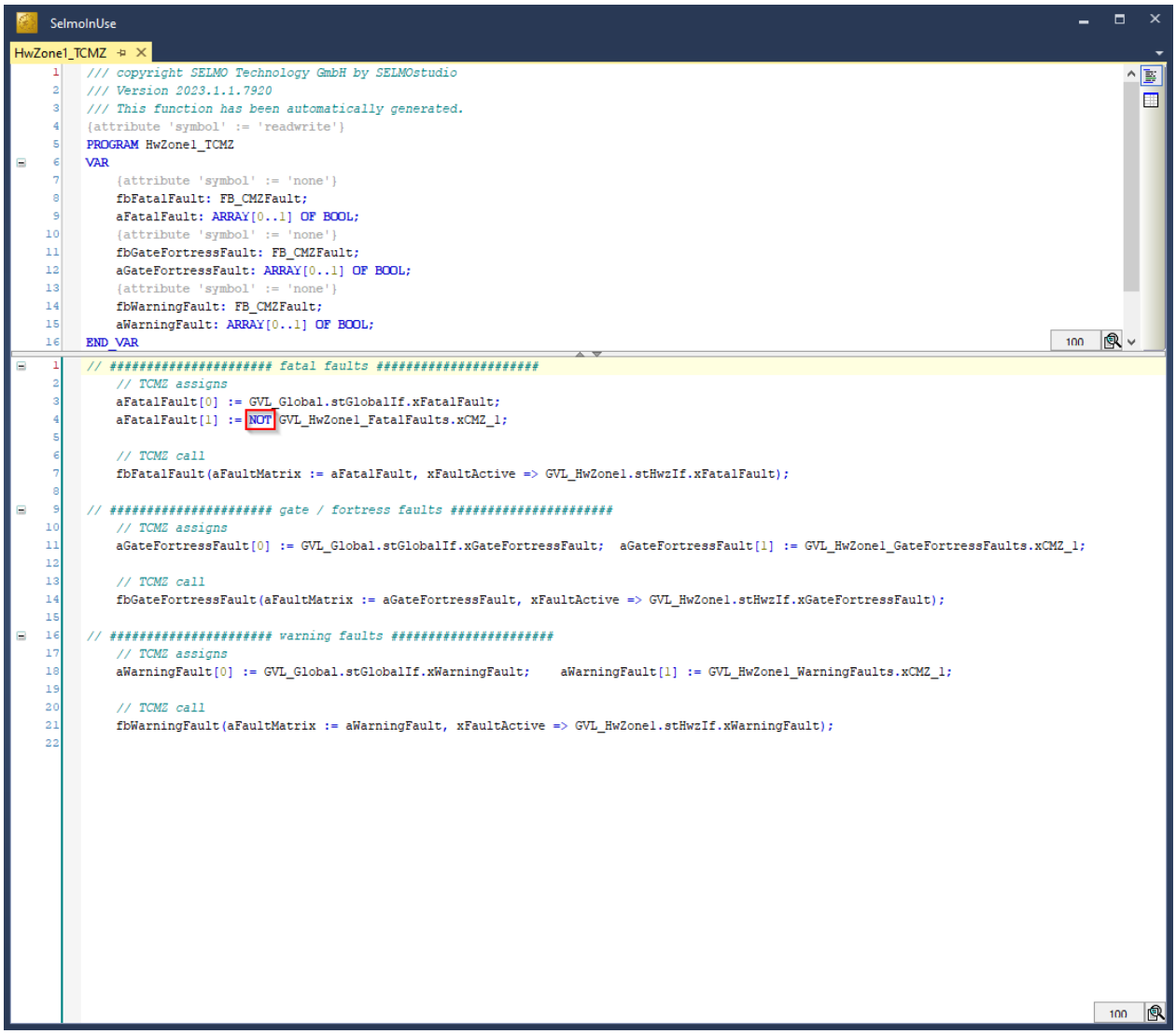


Studio

Inverted

If the field "Inverted" is activated, the value of the linked variable is inverted in the PLC code of the corresponding level.

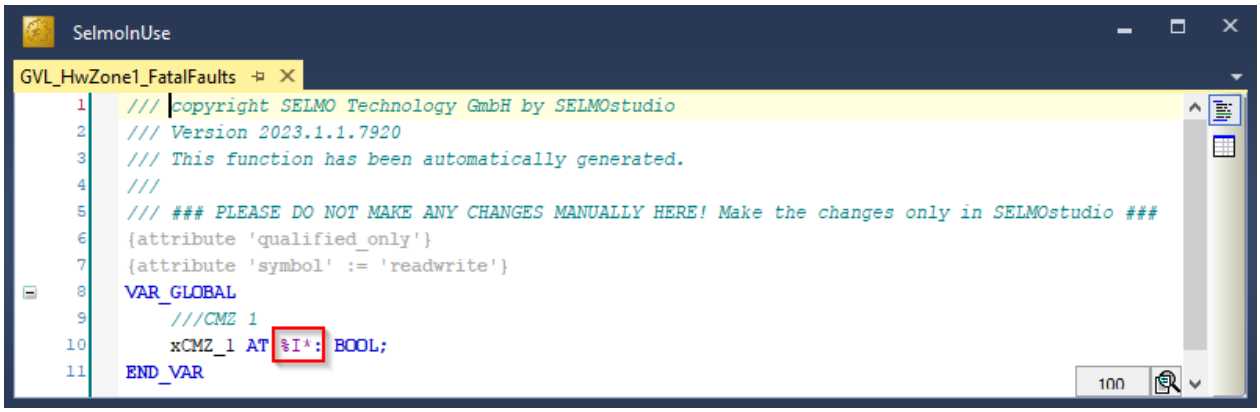
This is only possible with Boolean variable types.



PLC

Declaration as Input

If you declare the CMZ as "True", it will be declared as hardware input and included with the "AT %I*" attribute in the programming logic. This means that the variable receives a signal or value from a physical input of the system, such as a sensor or a switch.



PLC

HardwareInput

Prefix is added to the variable declaration in the generated PLC code.

Parameter

variables with a function (e.g. temperature maximum).

Parameter Mode

Selection between None, Greater Than, Less Than, Equals, GreaterEquals, LessEquals, NotEquals and Window.

Window Positive Parameter und Window Negative Parameter

A number is used to initialize this parameter.
To be able to select the window mode, the variable type must be something other than BOOL. Window is a comparison function and acts like a tolerance window. The window function can always be adjusted in the HMI. The Window Positive parameter is the maximum value and the Window Negative parameter is the minimum value.

Auto Reset (option only available in the sequence CMZ)

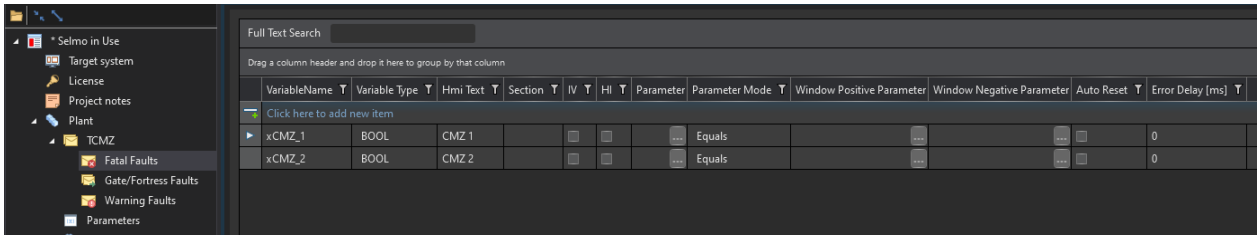
An Auto Reset in case of error refers to a function where an error is automatically reset without manual intervention. When an error occurs, it is automatically detected, and the system state is reset to the normal operating state. This feature is often used in automated systems to ensure that operation can continue without the need for an operator to manually intervene to correct the error. Error Auto Reset is particularly useful in critical applications where immediate intervention is required to minimize downtime or damage to equipment.

Error Delay [ms]

Used when the error should be triggered with a time delay. In summary, "Error Delay" is used as a term when an error occurs in signal processing and time delayed triggering is desired. The phenomenon where a signal briefly generates several fast pulses instead of a single pulse is referred to as "signal bounce" or also as "contact bounce".

1.3.3.1 Fatal Faults

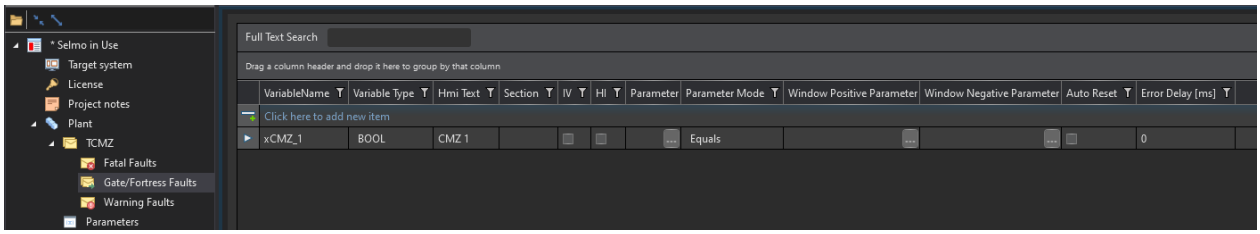
Fatal faults are serious errors of the system that deactivate the automatic mode when the condition of a linked variable occurs. For example, the input signal of a circuit breaker is linked to the variable xCMZ_1. This variable of data type bool is false in operation until the input signal of the circuit breaker assumes the value true. As a result, the automatic mode is deactivated immediately. Such a serious error must be corrected before manual enabling of the system is possible again. Serious errors cannot be bypassed.



Studio

1.3.3.2 Gate/Fortress Faults

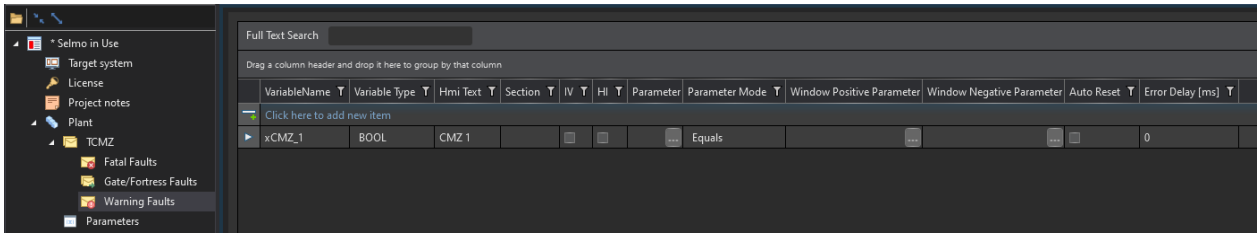
An access violation is called a Gate/Fortress error. Variables can be defined to monitor signals from safety gates, light curtains or sensors. Protected areas can thus be defined. If there is a violation of such an area during operation, the automatic mode of the system is deactivated and the corresponding access violation is displayed on the HMI.



Studio

1.3.3.3 Warning Faults

Warning faults are warnings of a system, which are valid as information for the user but do not endanger the automatic mode of the system. If these faults occur during operation, they are displayed on the HMI, but the automatic mode of the plant is maintained. Warning faults can be used to indicate an approaching hazard such as material shortages, decreasing levels, etc. to warn the user.

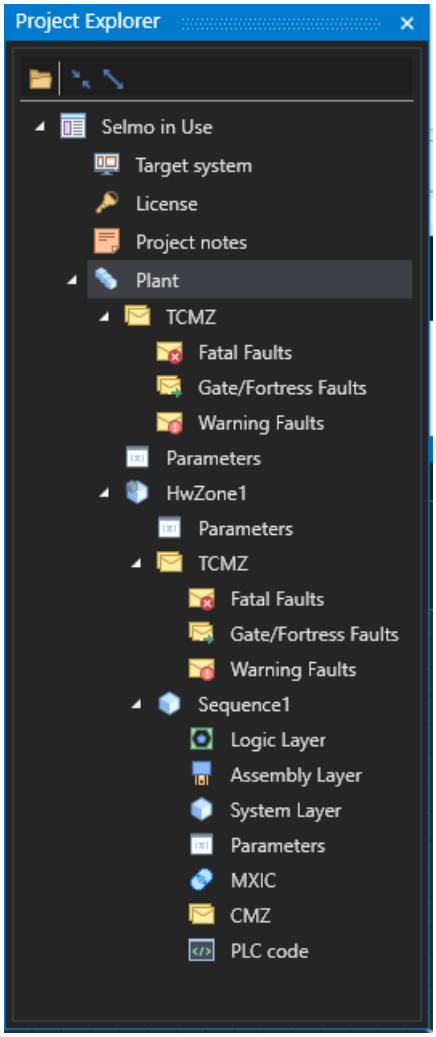


Studio

1.4 Result after step 1 - 3

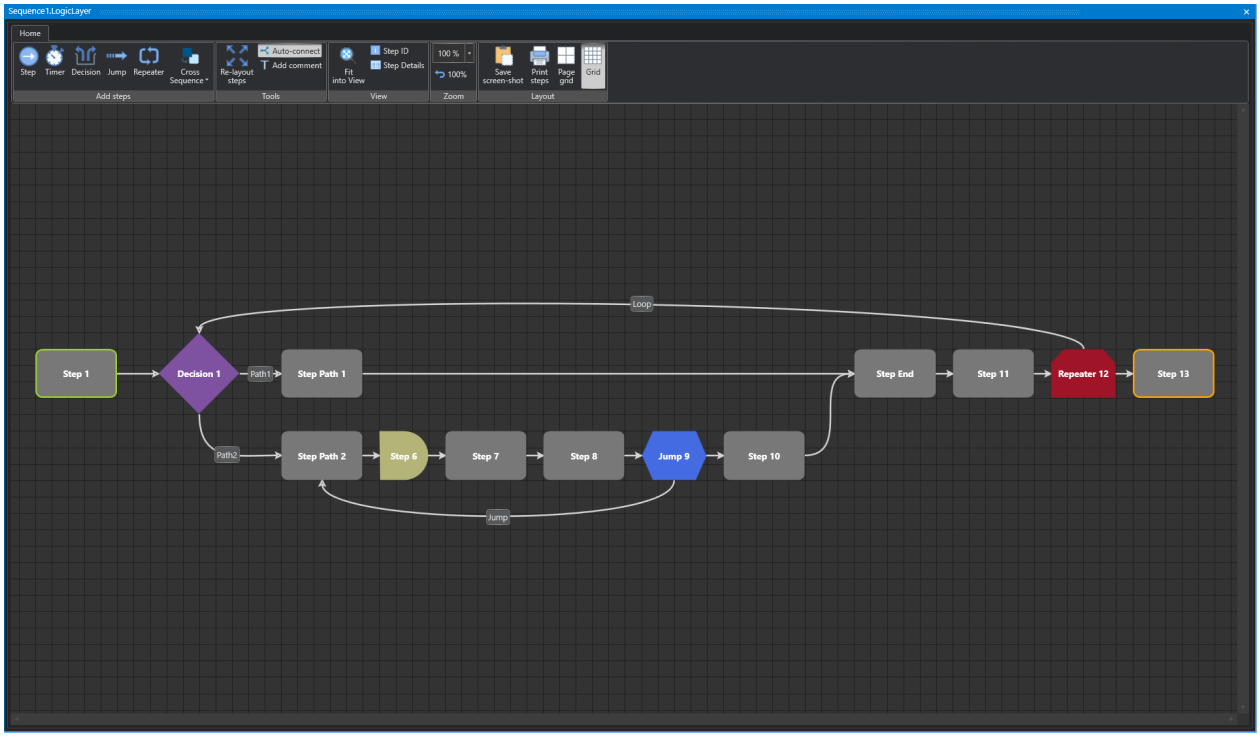
What is already available after step 1 to 3

After the first three steps, a basic structure of the Plant with a Hardware Zone and a Sequence is already in place.



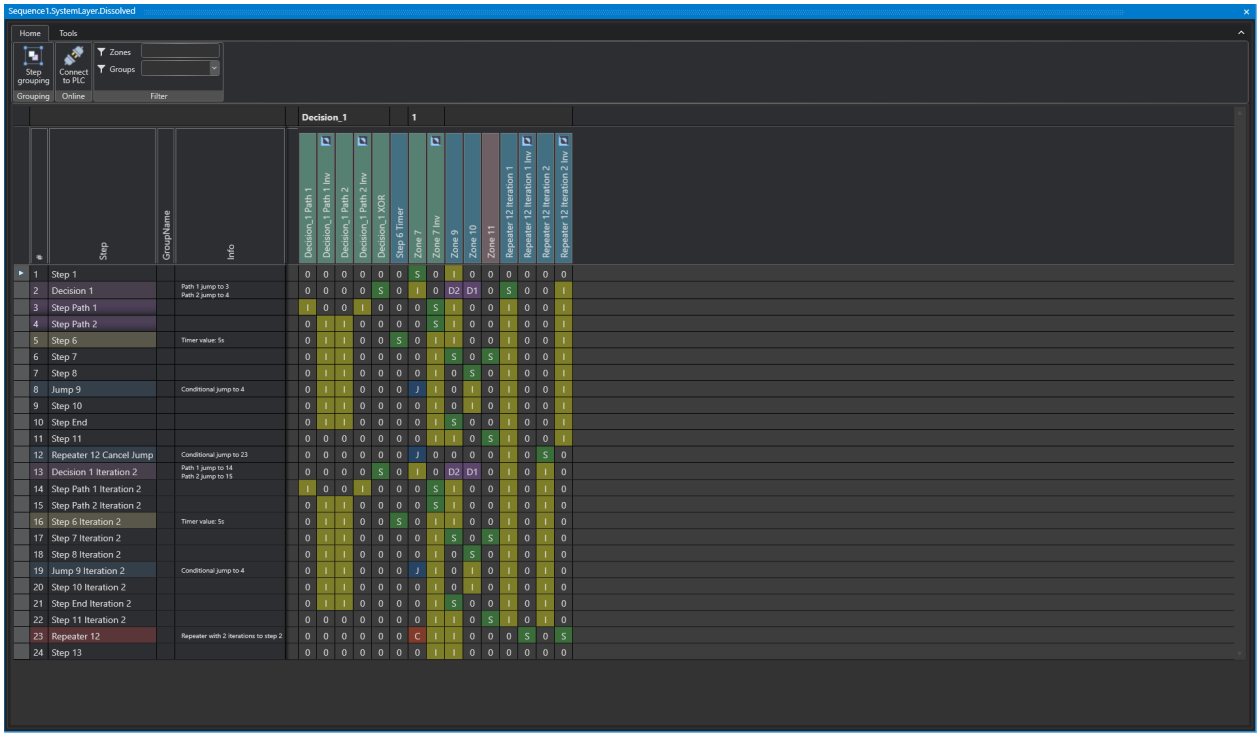
Studio

In addition, the logical flow of the process was shown graphically.



Studio

The signals were defined with the corresponding zones, states, operands and monitoring.



Studio

This is the basis for further programming and testing.

What is still missing

With this step we have laid the foundation for further development and testing of our system. Now we need to define the details that will ensure the quality and safety of our solution. These include the functions of the manual control (Manual Cross Interlock Check), which prevent the system from damaging itself in case of incorrect operation, as well as the flexibility provided by the parameter layer, which allows us to adapt the settings to different requirements.

1.5 Manual functions

Manual operation on machines refers to a function in which the machine is operated manually and without automatic control. In manual mode, the operator of a machine can perform certain functions and movements manually, for example, by operating levers, switches, or buttons on the machine. This allows the operator to precisely control the machine and perform certain functions that may not be automatically performed by the machine.

Manual operation can be helpful in a variety of situations, such as servicing the machine, setting up the machine for a specific task, or troubleshooting when a problem occurs. However, when an operator is working in manual mode, he or she must be especially careful because he or she is completely responsible for controlling the machine and must minimize safety hazards. In some cases, it may also be necessary to have special training or certification in manual operation of machines.

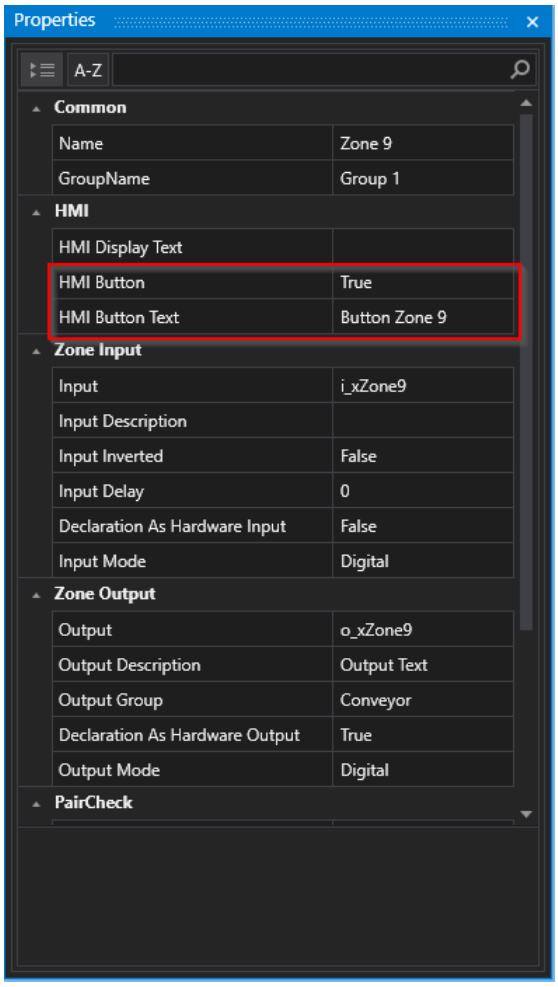
Manual operation is a user interface that functions only in manual mode and is used to control the output. Unlike the automatic mode, where the system operates automatically, manual mode allows the user to make manual interventions.

An HMI button controls the output until a feedback signal (input of the zone) becomes active. The feedback signal provides feedback to the system on whether the desired state has been achieved or changes need to be made. When the MXIC cross-lock is active, the output cannot be controlled, and the HMI displays information about the zone in which the interlock is taking place.

MXIC cross-locking is a safety function that prevents the zone from being activated unless certain conditions are met (zones x y are in a defined state). This is especially important in critical applications where the simultaneous occurrence of multiple events can lead to dangerous situations. Using the HMI button in combination with MXIC cross-locking ensures that only the desired output is activated and that potential hazards are avoided.

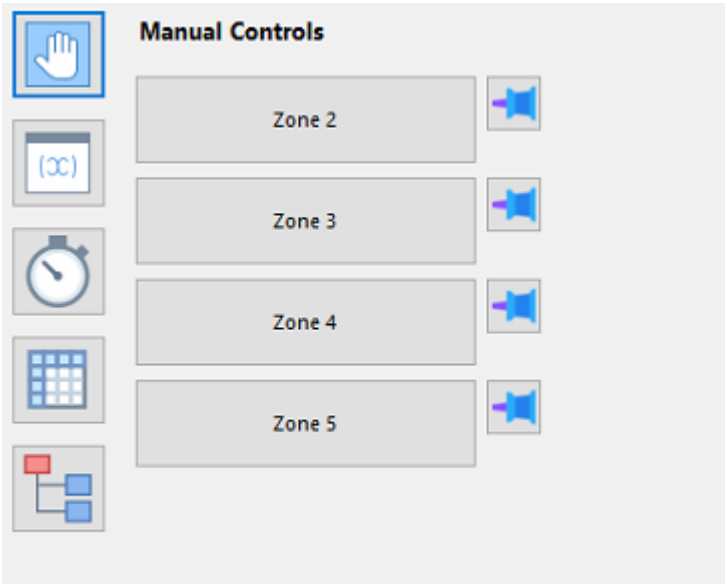
HMI Buttons

To enable HMI buttons for a specific zone (valid only for InOut and Out zones), the "HMI Button" option must be enabled in the zone properties. The HMI button label text can be customized using the "HMI Button Text" property, e.g., "Cylinder forward", "Axis 1 position 1" or "Send data to ERP".



Studio

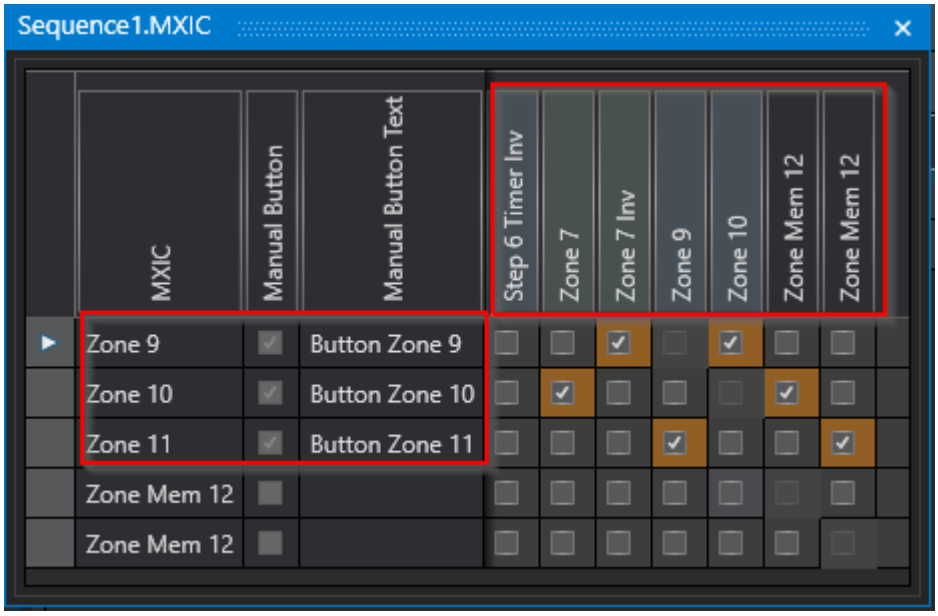
The buttons can be found in the HMI in the Sequence under Manual Controls.



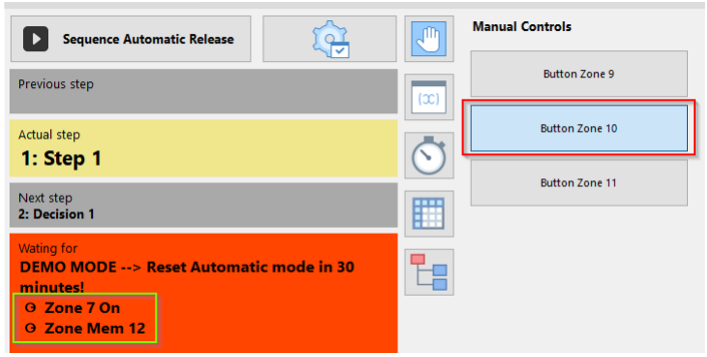
HMI

MXIC

The MXIC function (Manual Cross Interlock Check) prevents operating errors in manual mode. In the left column, all manual functions are listed and compared to the corresponding zones. For each manual function, it is determined which zones (with feedback) must be active for the manual movement to be executed. This information is also used for diagnostics and operator guidance. If the operator wants to run a blocked manual movement, it will be blocked and at the same time the HMI will show why the movement cannot be executed.



Studio



HMI

1.6 Flexibilization via parameters

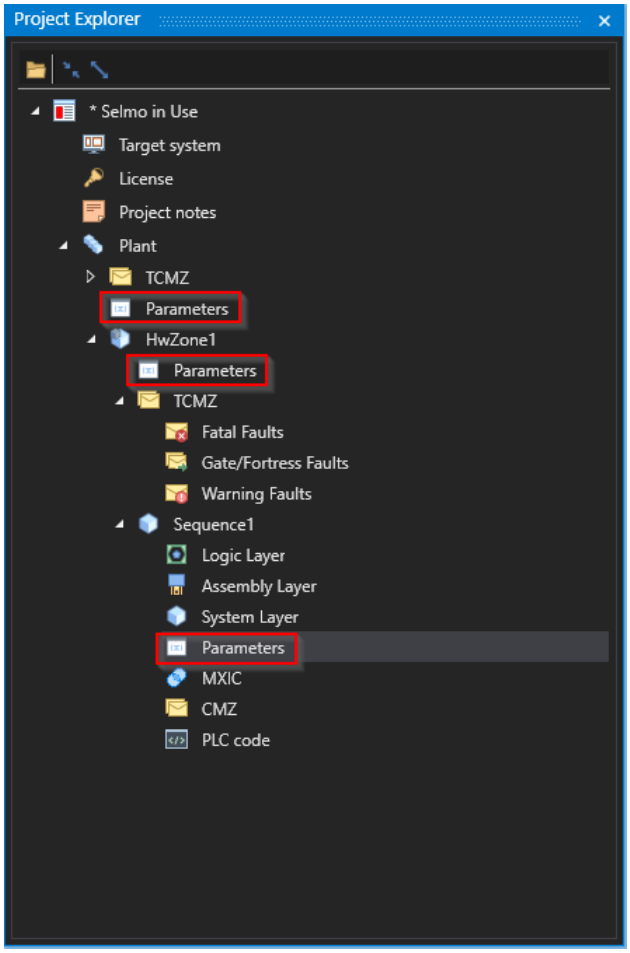
Flexibilization via parameters refers to the possibility of adapting a system or method so that it can be adapted to different conditions or requirements by changing parameters. Flexibilization via parameters allows processes or systems to be adapted quickly and efficiently to changing requirements without the need for extensive revisions or changes. An example of this would be the adaptation of production processes in industry to respond to changing customer needs or market conditions. By changing certain parameters such as production speed, product quality, or material usage, production can be quickly adapted without having to redesign the entire production system.

1.6.1 Parameter levels

The parameter levels of Plant, Hardware Zone and Sequence are frequently used in automation technology to enable a hierarchical structuring of parameters. The Plant level includes all parameters that are relevant for an entire plant or system. For example, parameters such as the operating mode or the overall performance of the plant are defined here. The hardware zone level refers to specific areas or components of the plant, such as a particular conveyor section or machine area. Here, parameters are defined that are only applicable to this area or component. The Sequence level refers to specific sequences or processes within the plant, such as the opening or closing of a valve. Here, parameters are defined that only apply to these particular sequences or processes. This hierarchical structuring of parameters enables clear and orderly management of parameters, which in turn facilitates maintenance and troubleshooting.

Parameter levels are one way to reduce the complexity of automated plants. They allow the plant to be divided into different areas, each with its own parameters. For example, a plant may consist of several hardware zones that have different sensors and actuators. Each hardware zone, in turn, can consist of multiple Sequences that control the plant's operations. For example, a Sequence can turn a sub-process on or off, control a temperature, or control part of a station. By using parameter levels, you can make the system clearer and easier to customize.

Parameter levels are divided as follows.



Studio

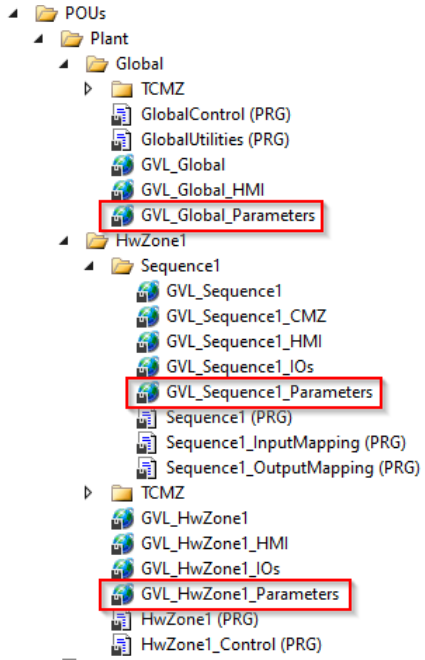
Plant level: The Plant level defines basic parameters that are relevant for the entire plant. These are settings and defaults that apply regardless of specific process steps or machines. Typical parameters defined at this level include, for example, temperature warnings that ensure that the plant operates within a certain temperature range. In addition, piece counters are also defined here, for example to monitor the number of products produced. Another important parameter at the plant level is recipes, which define the exact ingredients and quantity ratios manufacturing a product. By defining these basic parameters at the plant level, efficient and reliable production can be ensured.

Hardware Zone Level: At the Hardware Zone Level, parameters are defined that are specific to a particular machine area. Settings and defaults are defined here that apply only to this specific hardware zone and not to other parts of the plant. Typical parameters created at this level include display parameters that indicate the operation and status of the machine zone, such as temperatures, pressures, or flow rates. Another example of a parameter that affects the hardware zone is the recipes of the machine area. These are specific instructions that tell the machine what materials to use and in what quantities to produce a particular product. These recipes can be created and stored in the hardware zone to ensure the machine is consistent and efficient in producing products. Other parameters that could affect the hardware zone include configuring sensors, calibrating gauges, or setting safety parameters such as emergency stop switches or limits for temperatures and pressures. Overall, the creation of parameters in the hardware zone refers to the fine-tuning and configuration of machine areas to ensure optimal performance and efficiency while ensuring the safety of employees and equipment.

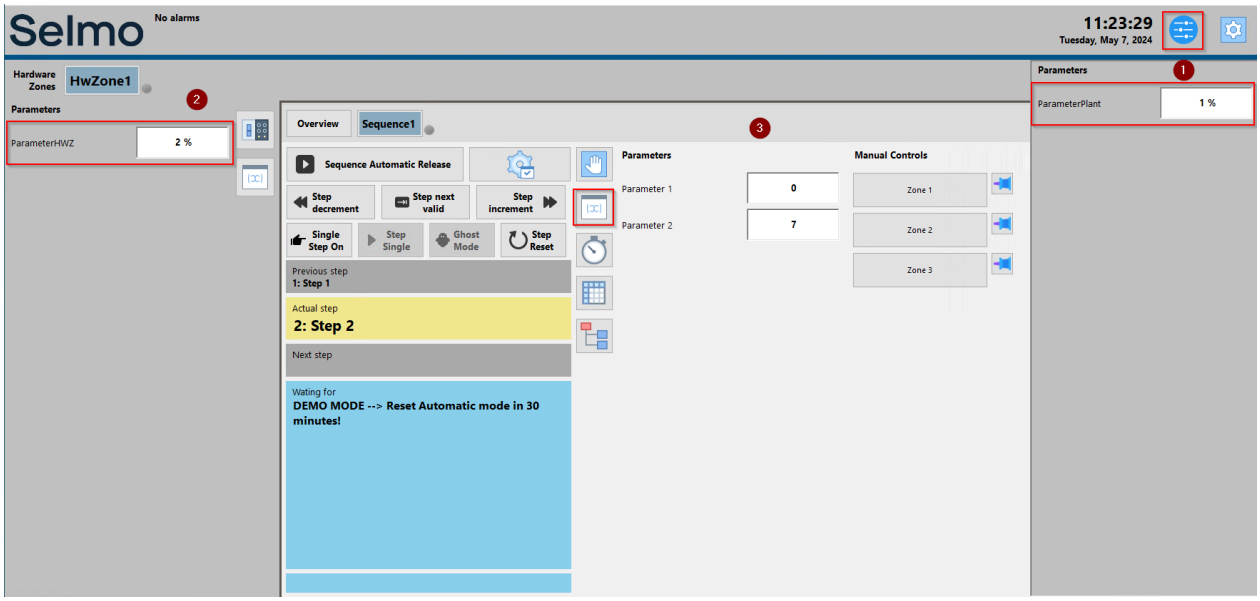
Sequence Level: The Sequence Level refers to the level of control elements that control the sequence of operations in a machine process. In this context, creating parameters relates to the settings that affect only the sequences within the Sequence. An example of a parameter at the Sequence level is the axis position. This means that the position of each axis of the machine system must be precisely defined so that the machine can perform the required movements and processes. These axis positions can be stored as parameters in the control level. Another example of a parameter at the sequence level is correction values. If irregularities occur during machine operation, correction values must be defined to compensate for them and optimize machine performance. For example, correction values can be set for temperature, pressure, or speed to ensure that the machine always operates within certain tolerances. The speed of the machine is another important parameter at the sequence level. The speed can be adjusted according to the requirements of the process to ensure that the machine produces the required output without being overloaded or having undesirable effects on the final product. In summary, parameters are set at the Sequence level to control and optimize specific functions of the machine system. These include axis positions, correction values, and speeds that help ensure that the machine process runs smoothly and accurately, and that the system's output is optimized.

Parameters set in a system or program can be used at all levels below. For example, if you think of a tree structure, the parameters set at the top level can be used at all levels below. However, if you try to use these parameters at a level higher than the original level, it will not work. A parameter is a specific setting made in a system or program. These settings can have various properties that can be edited. For example, these properties can be data type, value range, or default value. In summary, parameters are set in a system or program to define settings that can be used by different levels in the system or program. Each parameter has different properties that can be edited to change its functionality.

Depending on the level at which the parameter was declared, a corresponding entry is created in the PLC code and in the HMI. This can be done either at the Plant, Hardware Zone or Sequence level.



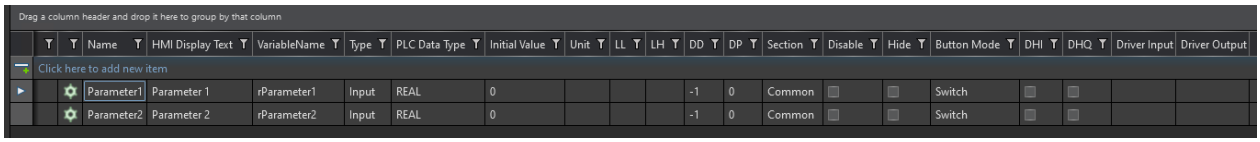
PLC



HMI

1. Plant level
2. Hardware zone level
3. Sequence level

The parameters are edited and created using an editor. Several options are provided, which are explained below:

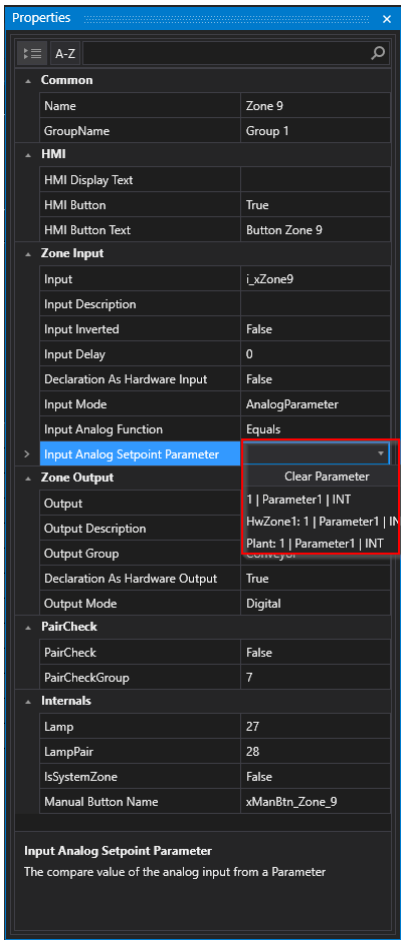


Studio

Name

The name of the parameter is used as the basis for the variable name. The notation for the name of the parameter is freely selectable, but it is recommended to use a unique and descriptive name.

If the parameter is used as an input parameter for a zone, it is selected based on the layer and the parameter name.

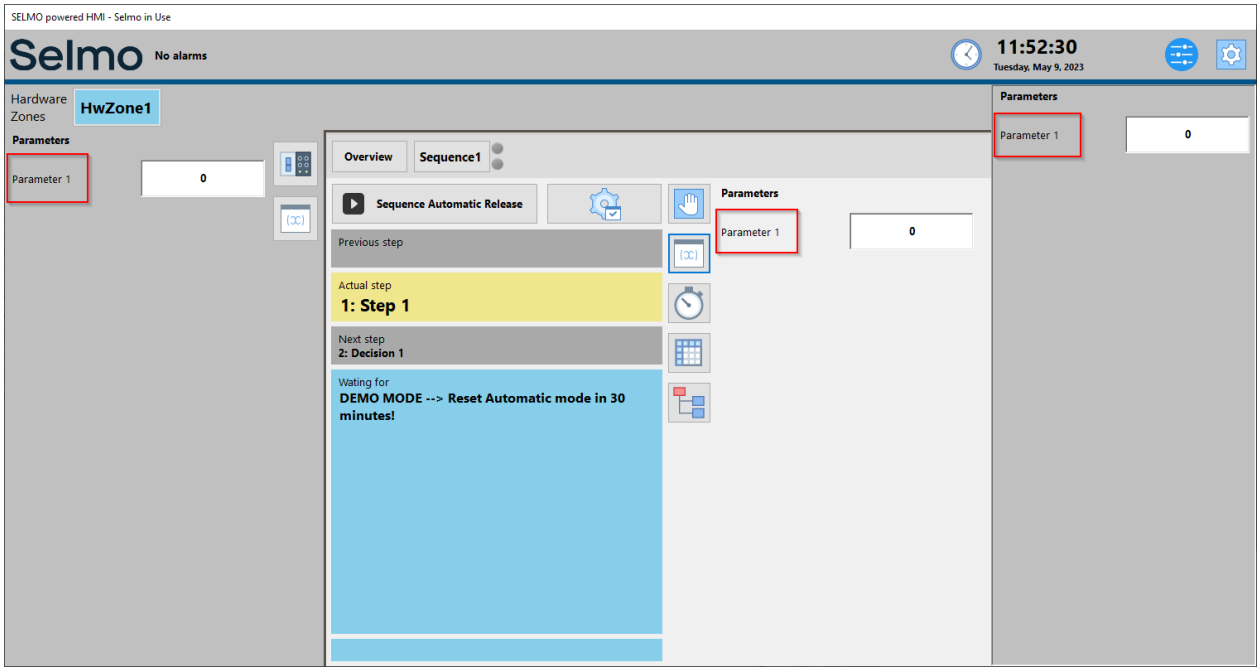


Studio

HMI Display Text

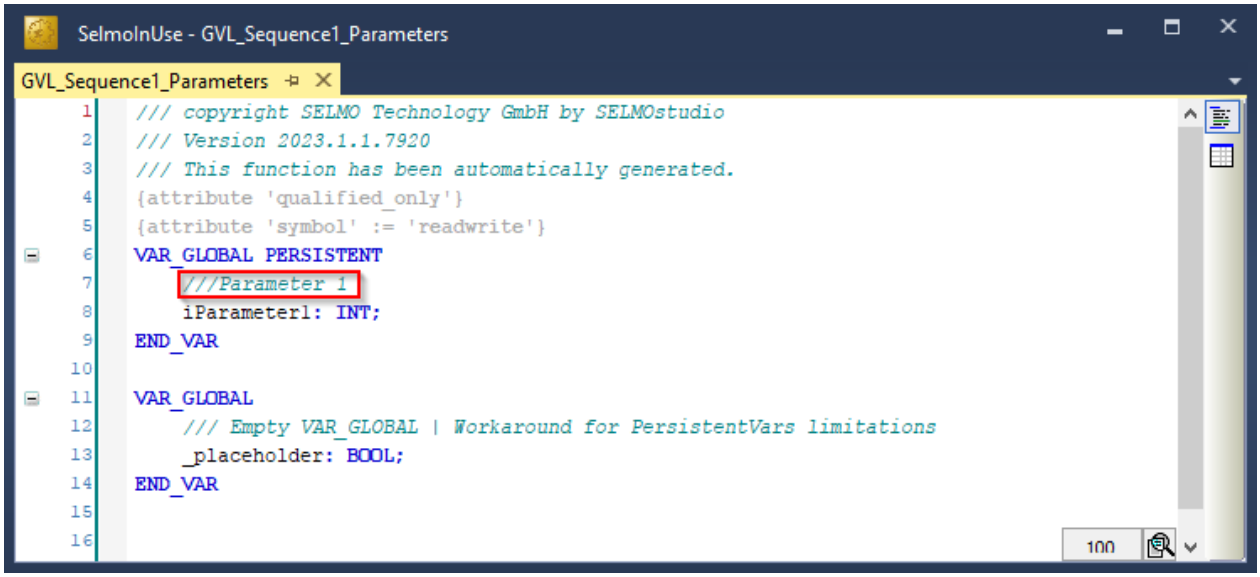
The display name of the parameter in the HMI refers to the name of the parameter shown on the HMI display of the system or machine. There is no special notation for the display name, but it is important that the parameter name is understandable and easy to read for the user.

However, for readability reasons, the amount of HMI display text should be kept to a minimum. This means that the display name of the parameter should be as short as possible, but still sufficiently meaningful to be quickly recognized and understood by the user.



HMI

In addition, the display name of the parameter in the HMI also serves as a comment for the parameter's associated variable in the PLC code.



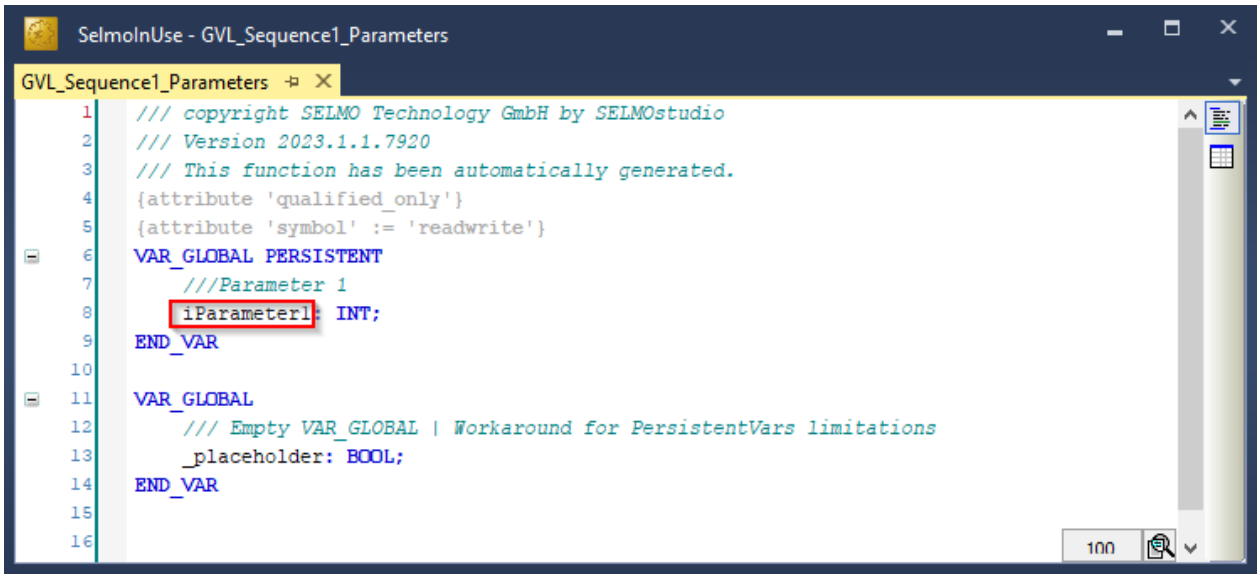
PLC

Variable Name

The variable name is automatically derived from the name of the parameter and is not editable. The name of a variable is important for the readability and comprehensibility of the code. A good variable should be unique, meaningful and functional. That is, it should have only one meaning, describe the content or purpose of the variable, and match the data type and logic of the program.

Tip:

Selmo uses the PLCopen Coding Guidelines. This is an internationally recognized standard for programming automation systems, especially programmable logic controllers (PLCs). The PLCopen Coding Guidelines establish rules and best practices to help developers ensure that their PLC programs are readable, reusable and robust. The guidelines cover a variety of topics, including variable naming, code commenting, program structure, and error handling. The naming of variables in the PLCopen Coding Guidelines follows certain rules. For example, variable names should be meaningful and reflect the purpose and type of the variable. The name should be written in English and follow certain conventions, such as the use of CamelCase. Furthermore, rules for naming input and output variables, temporary variables and constants are specified. Overall, the PLCopen Coding Guideline aims to improve the readability, maintainability and robustness of PLC programs and thus contribute to higher efficiency and productivity in the development of automation systems.



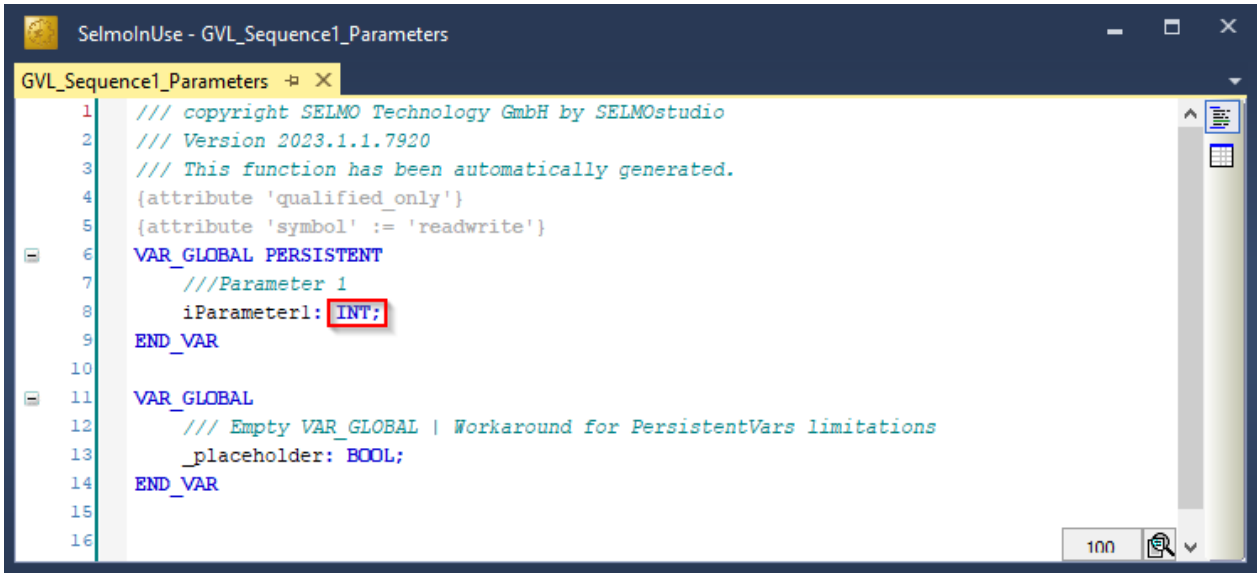
PLC

Type

The parameter type indicates whether the parameter is an input parameter or an output parameter. An input parameter is used to enter values into the system or make certain settings, while an output parameter is used to output values from the system or display certain information. An example of an input parameter could be a parameter that defines the desired position of a motor or a specific mode of operation. The user enters the value for this parameter, which is then used by the system. An example of an output parameter could be a parameter that indicates the actual position of a position encoder or provides a warning message when a certain condition is reached. In summary, the parameter type indicates whether the parameter is used to input or output values to control and monitor the system or machine.

PLC Data Type

The declaration of the variable type is done as integer data type by default. However, the drop-down menu can be used to select the appropriate data type. The available data types correspond to the list of standard data types.



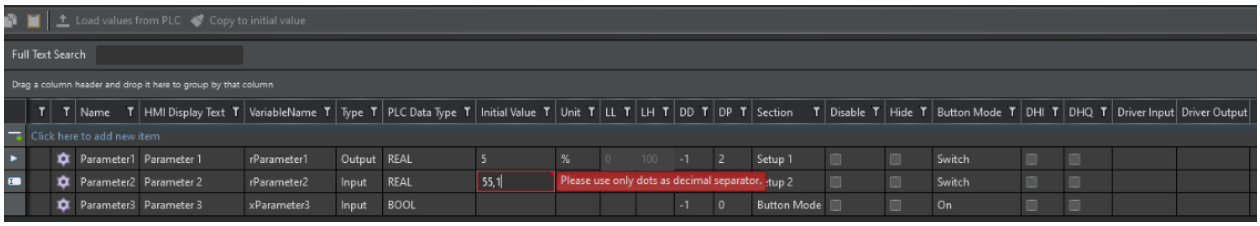
PLC

Initial Value

The initial value for a variable specifies the initial value that is assigned to the variable when the program is started or the parameter is reinitialized. Depending on the PLC data type, only certain input formats are permitted for the initial value. This means that the initial value for a variable must be entered in a specific format that matches the variable's data type.

Hint:

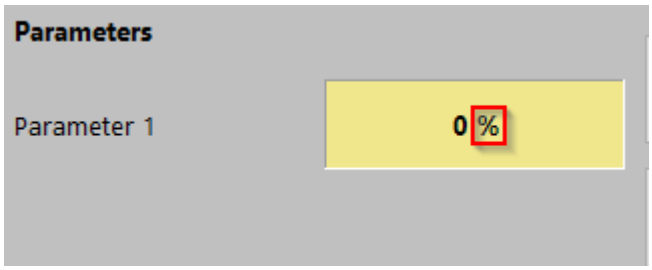
If the initial value is entered in an incorrect format, this can lead to malfunctions or errors in the program. It is therefore important to ensure that the initial value for a variable is entered in the correct format to ensure smooth functioning of the program. This is already checked in advance and displayed accordingly.



Studio

Unit

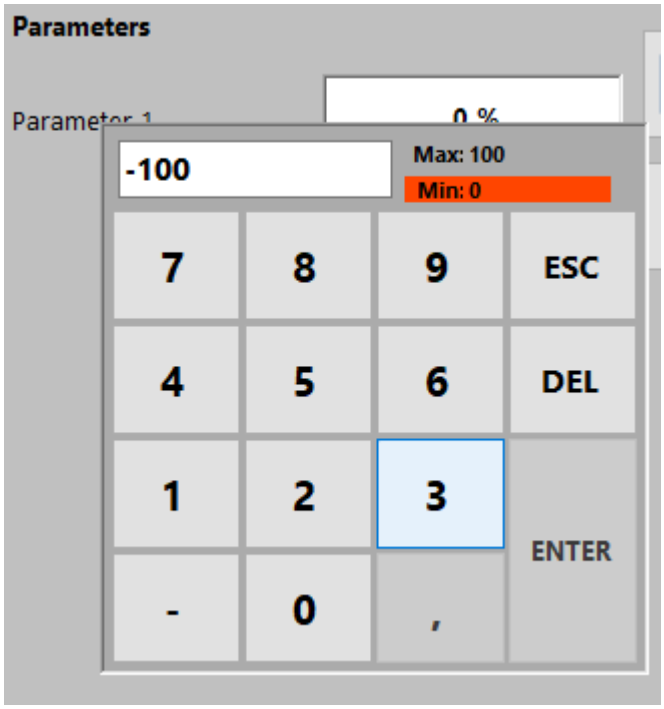
The display unit of the parameter in the HMI refers to the unit in which the parameter is shown on the HMI display of the system or machine. The unit may vary depending on the PLC data type, but all entries are valid and are not checked for correctness or consistency with other parameters or the system. However, it is important that the unit is understandable to the user and helps them to interpret the parameter correctly. A correct and unambiguous display unit can help to avoid misunderstandings and facilitate the operation of the system. Therefore, the parameter's display unit should be carefully chosen and properly documented to ensure correct understanding and error-free use of the system or machine.



HMI

LL (Limit Low)

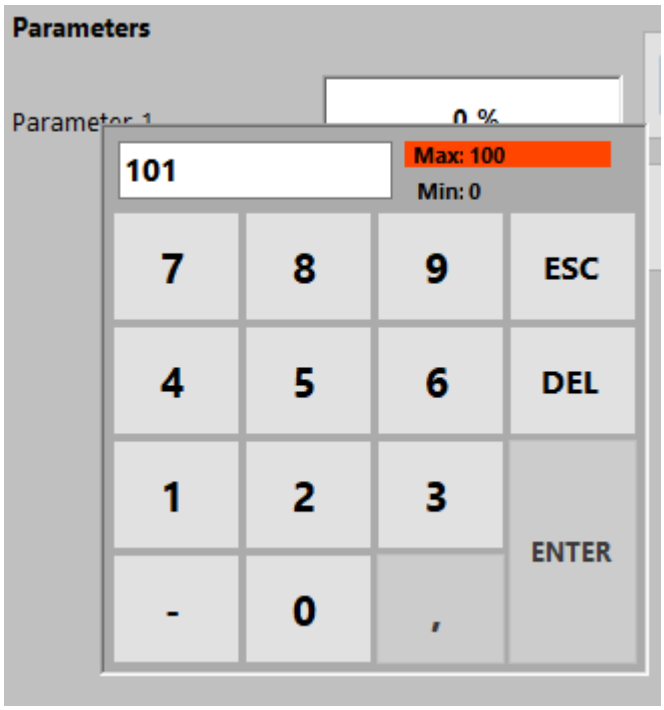
The minimum input value for an input parameter is the smallest value that is acceptable for that parameter. This value can vary depending on the PLC data type, and only certain input formats that match the parameter's data type are allowed. It is important to ensure that the minimum input value is entered in the correct format. Otherwise malfunctions or errors may occur in the program. For example, an incorrect input can lead to an overload of the system or an unexpected reaction. Therefore, the minimum input value should be carefully documented and entered to ensure correct operation of the system or machine. Suppose a value is entered for a parameter that is below the defined lower limit (LL), this limit is usually highlighted in red to indicate that the input is not accepted. The system or machine refuses to accept the value input to prevent the system from becoming faulty or unstable. The red marking alert the user to the error and helps them correct the input value accordingly.



HMI

LH (Limit High)

The maximum input value for a parameter is the highest value that is acceptable for that parameter. If a value is entered for a parameter that is above the defined upper limit (LH), this limit is usually highlighted in red to indicate that the input will not be accepted. Similar to an input value that is too low, the system refuses to accept the input of the value to prevent the system from becoming faulty or unstable. The upper limit value is important to ensure that the machine or system operates within safe and efficient operating parameters. Correct entry of the parameter is important to ensure system performance and safety. Therefore, the upper limit should be carefully documented and maintained to prevent the system from becoming faulty or unstable.

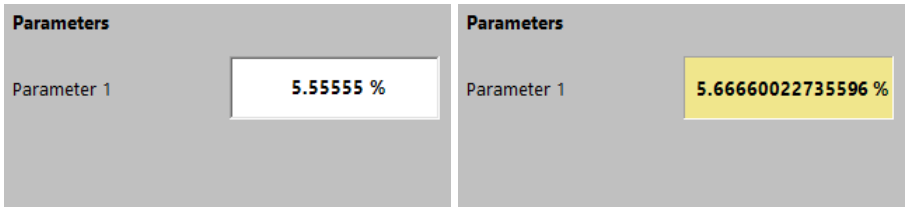


HMI

DD (Decimal Digits)

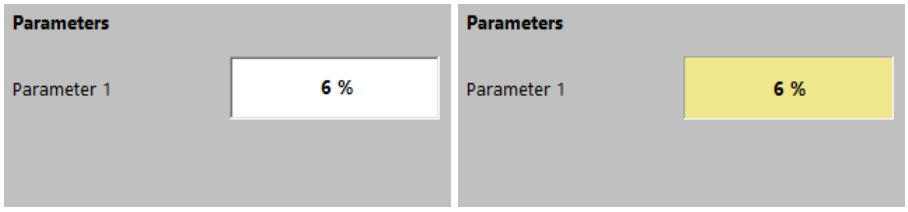
DD (Decimal Digits) stands for the number of decimal places to be included when displaying the parameter value. This number is usually defined in the parameter setup and may vary depending on the application. If the value of DD is set to 0, it means that no decimal places should be displayed, and the value is displayed as an integer. If DD is set to -1, no decimal places are considered, if DD is set to 1, one decimal place is considered and so on. The number of decimal places is important to ensure correct and accurate display of the parameter value. If the number of decimal places is insufficient, important information may be lost or displayed inaccurately. If the number of decimal places is too high, it may affect the readability of the value. Therefore, it is important to carefully define and monitor the number of decimal places to ensure accurate and readable representation of the parameter value.

DD -1 (Input/Output)



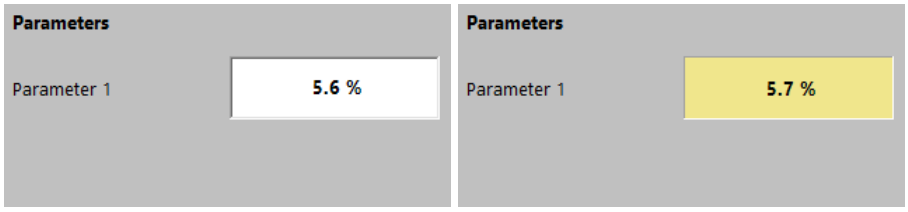
HMI HMI

DD 0 (Input/Output)



HMI HMI

DD 1 (Input/Output)



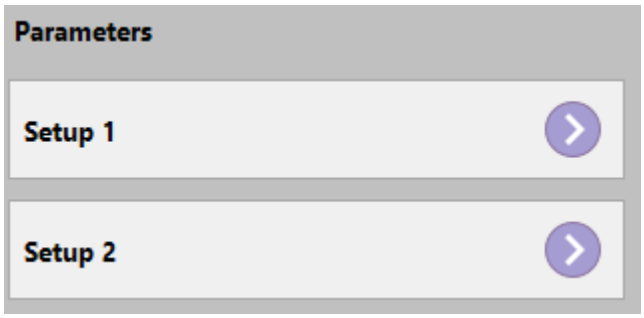
HMI HMI

DP (Decimal Place)

DP (Decimal Place) refers to the position of the decimal point when displaying the parameter value. This value is usually defined in the parameter setup and indicates where the parameter value should be divided to indicate the decimal place. For example, setting the value of DP to 2 means that the parameter value will be divided at the second digit from the right to indicate the decimal place. For example, the value 12345 would be represented as "123.45". The position of the decimal point is important to ensure that the parameter value is displayed correctly and understandably. If the decimal point is placed in the wrong location, it can cause confusion or inaccurate information. Therefore, it is essential to carefully define and monitor the position of the decimal point to ensure clear and accurate representation of the parameter value.

Section

The Section property can be used to group parameters and display them in a common menu item in the HMI. When parameters are marked with the same Section property, they are automatically grouped together in the same group. Grouping parameters into sections makes the menu clearer and more user-friendly, as related parameters are grouped together, and the user can find and change them more easily. The Section property can be named anything to describe the group of parameters appropriately. An example of a useful naming of a Section property is "Communication Settings" for parameters related to the communication of the system. Using Sections can make parameter handling easier and more efficient, resulting in an overall improved user experience.



HMI

Disable Input in Automatic

The Disable Input in Automatic function makes it possible to disable the changing of input variables while the automatic mode is active. This means that users cannot manually change the value of an input variable while the system is operating in Automatic mode. This feature is particularly useful for ensuring the security and integrity of the system, as it prevents users from inadvertently interfering with the operation of the system while it is operating in an automated mode. When the automated mode is disabled, users can manually change the value of the input variables again. This feature is particularly useful in industrial applications where it is important that the system operates safely and reliably, even when operated by different users.

Hide

If the "Hide" field is set for a parameter, this parameter is not displayed in the HMI. This means that users cannot see or access the parameter even if it is in the system. This feature can be useful when a parameter is used only for internal purposes or to configure the system and users do not need to know that it exists. It can also help to make the user interface clearer by hiding unimportant or rarely used parameters. However, it is important to note that hiding a parameter does not mean that it is no longer present in the system. The parameter can still be used by other system components, and changes to its value can still affect the system.

Button mode

Boolean parameters can be created as buttons. Different modes are available:

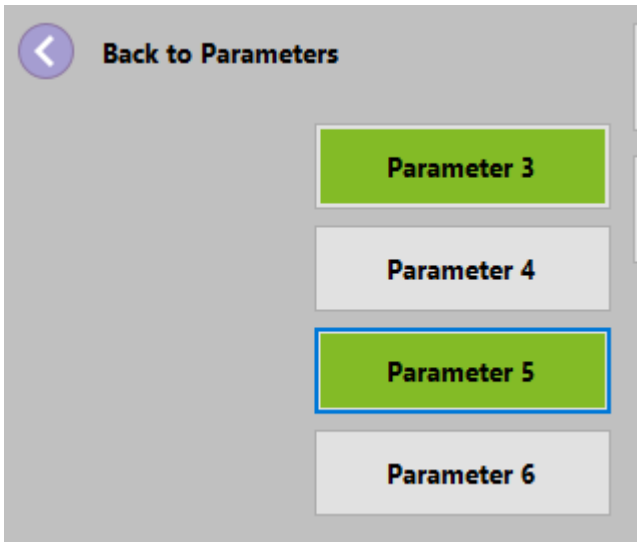
"On": switches the parameter variable to true when the HMI button is pressed.

"Off": Switches the parameter variable to "false" when the HMI button is pressed.

"Switch": Switches the parameter variable to "true" when the HMI button is pressed and to "false" when the HMI button is released.

"Toggle": Switches the state of the parameter variable between "true" and "false" each time the button is pressed.

If the HMI button is in the green state, this corresponds to the 'true' value of the parameter variable. If the button is in the gray state, this corresponds to the 'false' value.



HMI

DHI (Declaration as Hardware Input)

Boolean value which is only used for Bekhoff target system/Twincat target system so that it can be linked directly to the IOs. The address declaration shows %I* if you check it.

DHQ (Declaration as Hardware Output)

Boolean value which is only used for Bekhoff target system/Twincat target system so that it can be linked directly to the IOs. The address declaration shows %O* if you check it.

Driver Input

Information about which parameters are linked to the driver. The connection with analog inputs or outputs is important. In some drivers, at the assembly level, if an analog input is present, it is possible to link this input directly to a parameter. The parameter that has been selected can then be read under the 'Driver Input' field.

Driver Output

Information about which parameters are linked to the driver. The connection with analog inputs or outputs is important. In some drivers, at assembly level, if an analog output is

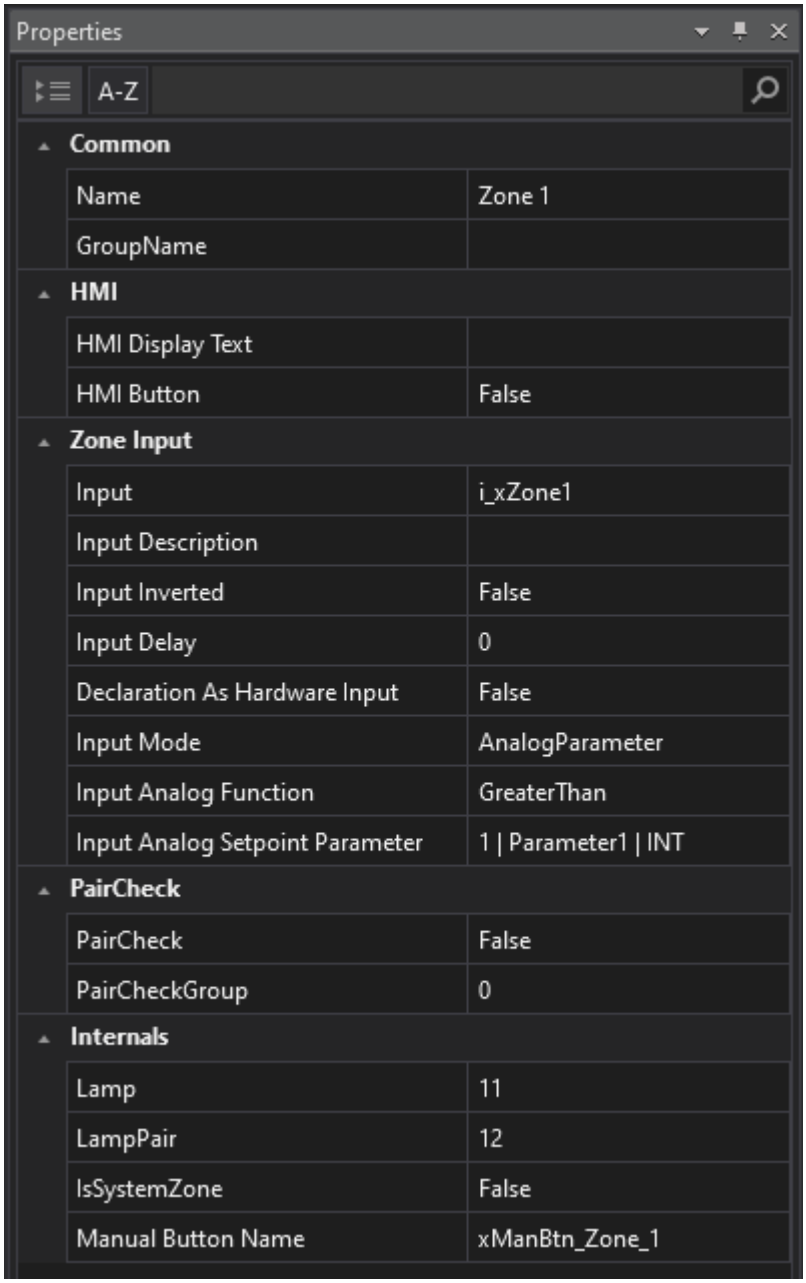
available, it is possible to link this output directly to a parameter. The parameter that was selected can then be read under the 'Driver output' field.

1.6.2 Scenarios where parameters can be included

The created parameters can be used to make the modeled process more flexible. In the following the usage points are explained and described.

Zone-In and Zone-InOut Inputs

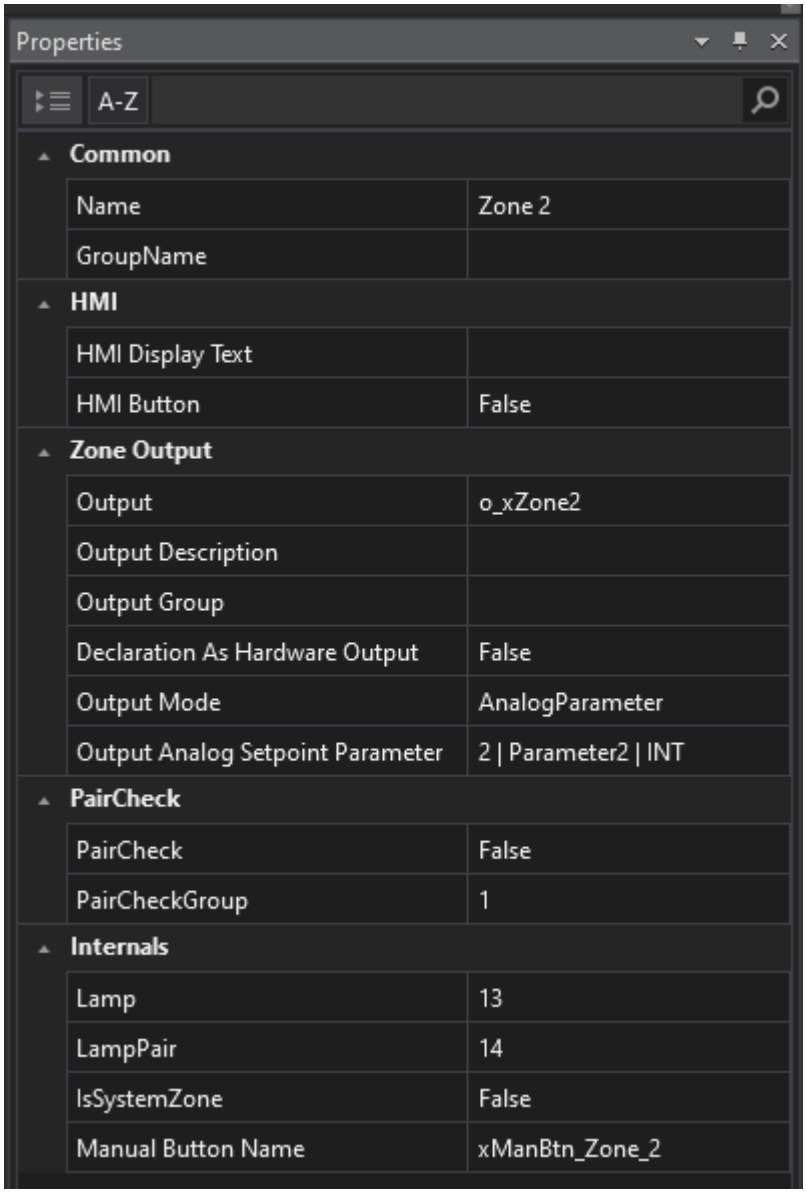
For inputs of Zone-In and Zone-InOut the Input Mode Analog Parameter can be activated, after that the created parameter on which should be compared is selected. The analog input is now compared with the parameter via the selected Input Analog function.



Studio

Zone-InOut and Zone-Out Outputs

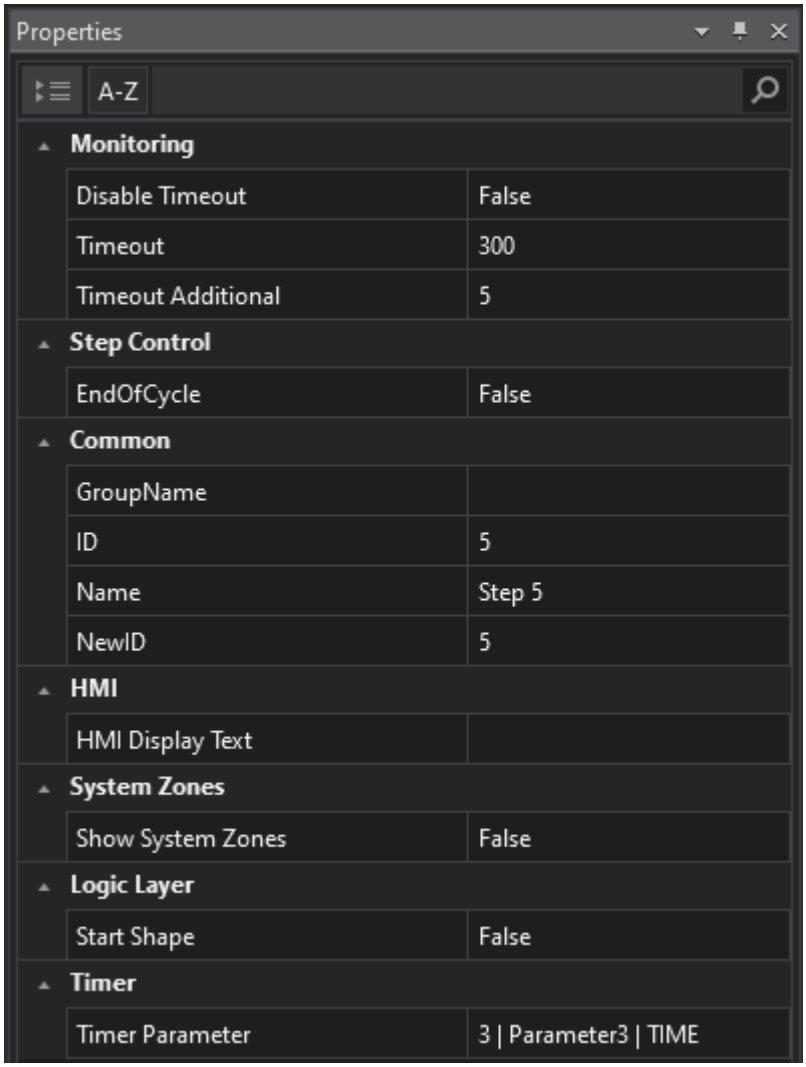
For outputs of Zone-InOut and Zone-Out the Output Mode: AnalogParameter can be activated, then the applied parameter to which the analog output should be connected is selected. If the output of the zone is switched out of the sequence, it takes the value of the parameter.



Studio

Time Step

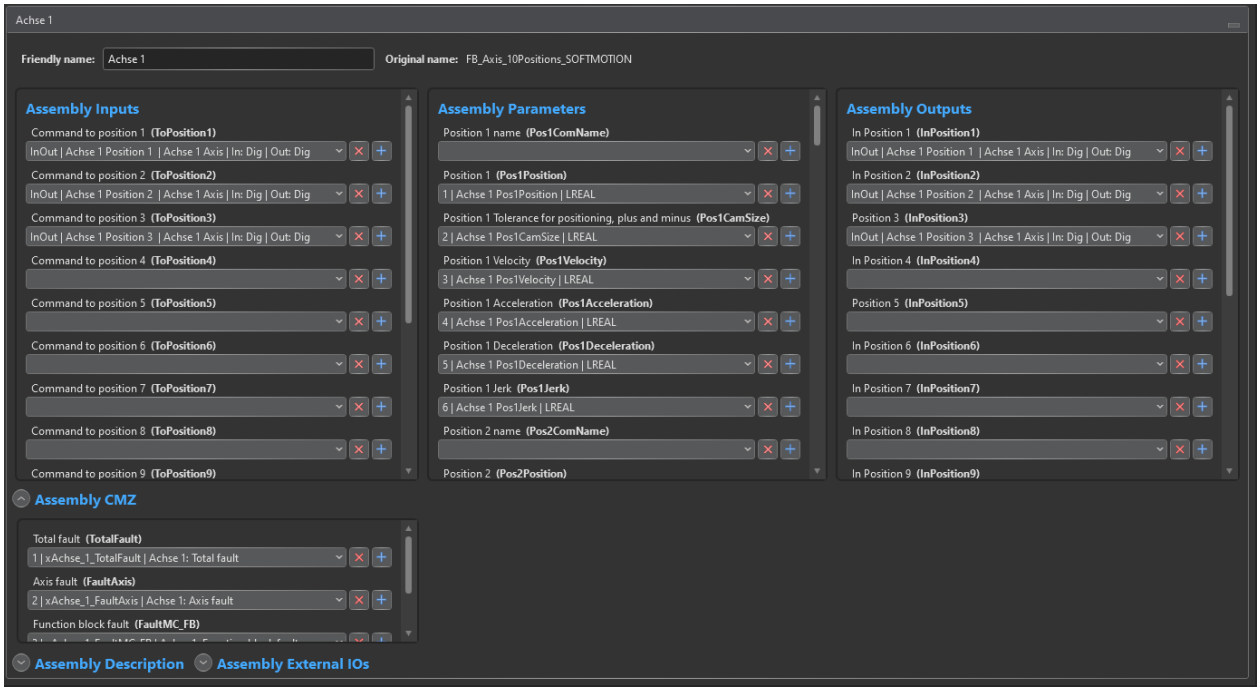
In Time Steps for the definition of the step time a parameter with data type Time must be created and finally selected in Time Step. There is the possibility to let the operator change the parameter or also the possibility to define the parameter as a fixed step time, for this the parameter is hidden in the HMI.



Studio

Driver

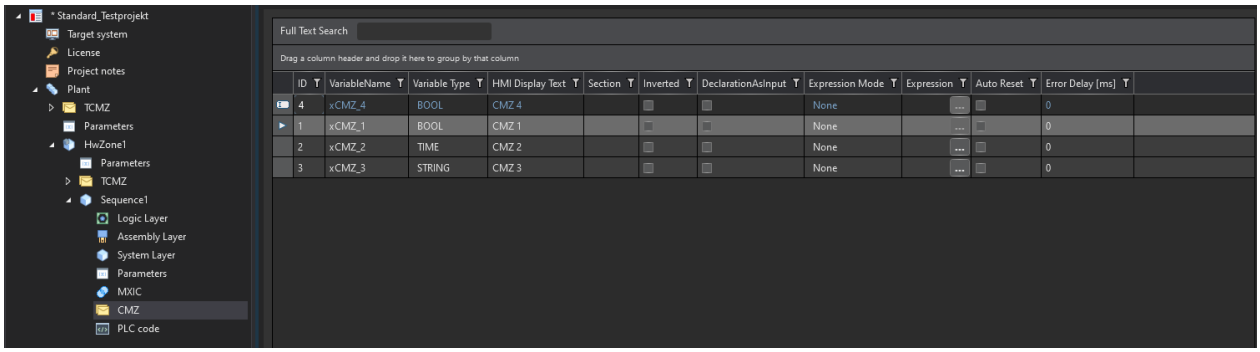
Parameters can also be linked to driver modules or are created directly by drivers. The following is an example of an axis driver. A wide variety of drivers can be operated with different parameters, e.g.: Control driver, printer driver, FU driver



Studio

CMZ

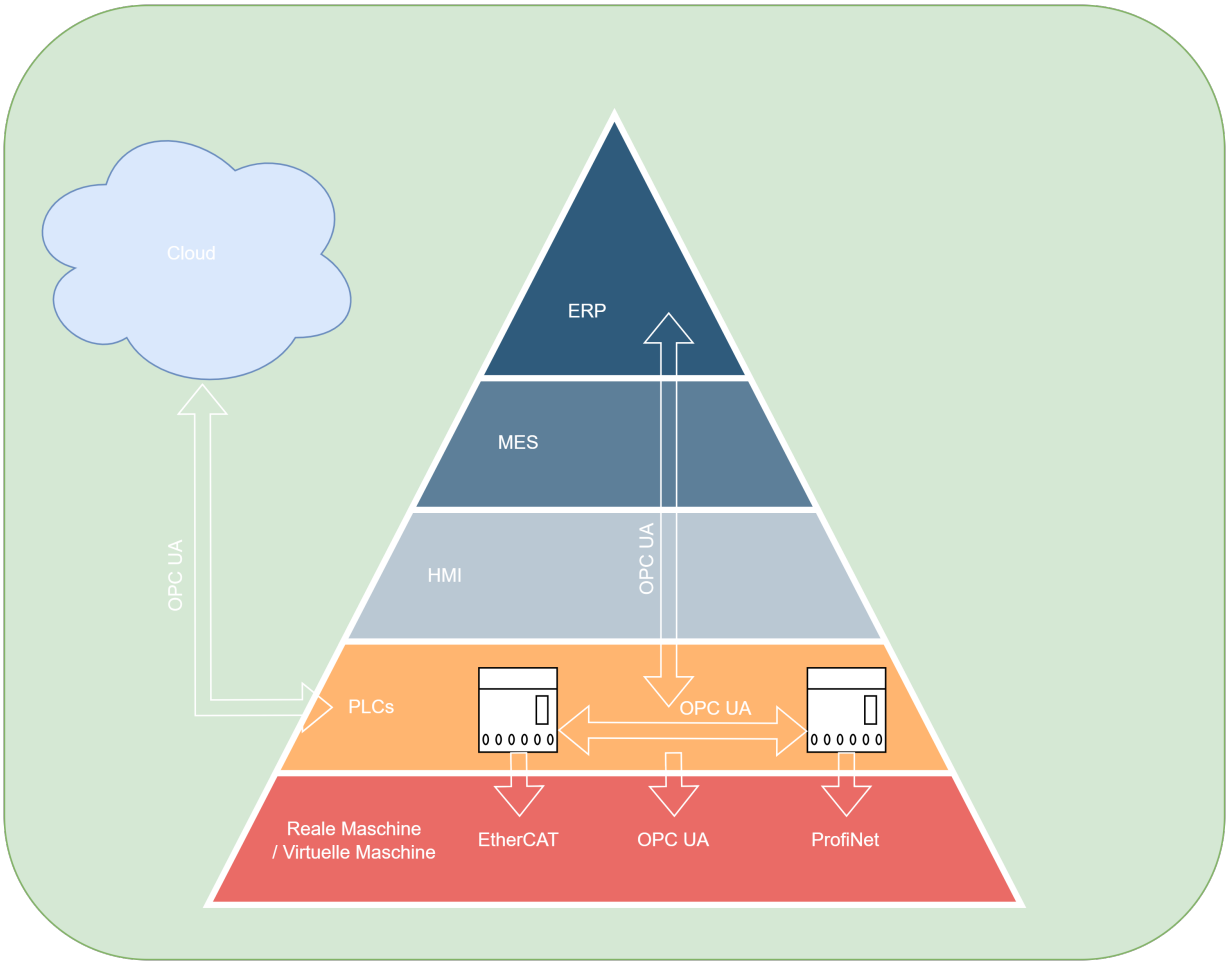
Alarm levels can be entered variably via the Expression Mode of the CMZ by means of parameters.



Studio

1.6.3 OPC UA

OPC UA (Open Platform Communications Unified Architecture) is an open communication standard for industrial communication that enables data and information to be exchanged between devices, machines and services. OPC UA supports digitalization and the requirements of Industrie 4.0 by providing a service-oriented architecture and semantic information modeling.



Connecting parameters to OPC UA means that the system's parameters can be represented as OPC UA variables or methods. These can then be read or written by other OPC UA clients to control or monitor the configuration or operation of the system. The Selmo standard supports OPC UA connectivity to higher-level or lower-level systems, such as ERP systems or sub-devices

1.7 Finalization

Finalization of the Selmo Solution has included the following steps:

- Zones Manual operation has been defined: This allows easy and safe operation of the machine in any condition.
- With MXIC (Manual Cross Interlock Check) the zones were interlocked: This prevents unwanted state transitions and increases the reliability of the software.
- Flexibility created by parameter layer: This allows quick and easy adaptation of the machine to different requirements and conditions.

End result:

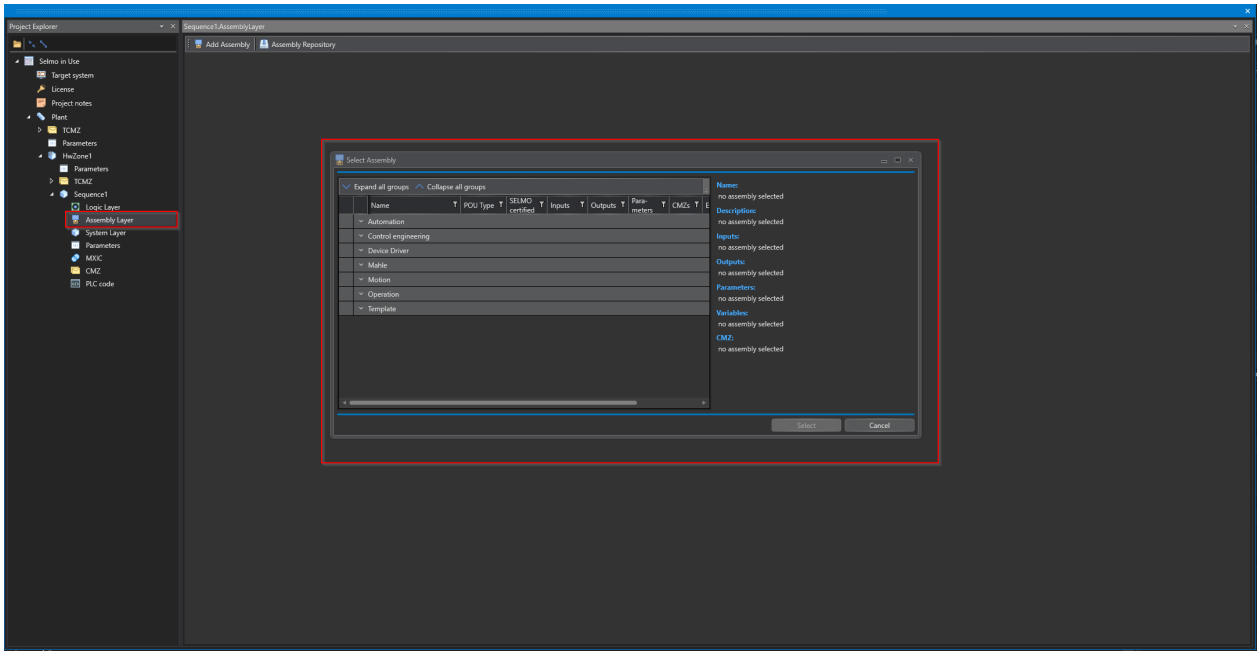
- High quality and stability of the software through standardization and through low code engineering: the software was generated automatically from the digital model of the process, without manual programming effort or sources of error.

For software that never lets you down!

The software is digitally verifiable at any time and can be reused or modified as needed. Communication between the machine user, designer and PLC programmer is transparent and understandable.

1.8 Assembly/driver layer

An assembly layer is a layer in software development that contains pre-built functional modules or components that developers can use to create software applications. It is a type of library that speeds up the development process by providing pre-existing solutions to common problems. The assembly layer is usually an abstract layer that sits between the actual application logic and the underlying hardware. It provides an interface between the application and the hardware and allows developers to abstract the underlying hardware by using pre-built building blocks. Using an assembly layer can help developers spend less time and energy developing basic functionality and instead focus on the specific requirements of the application.

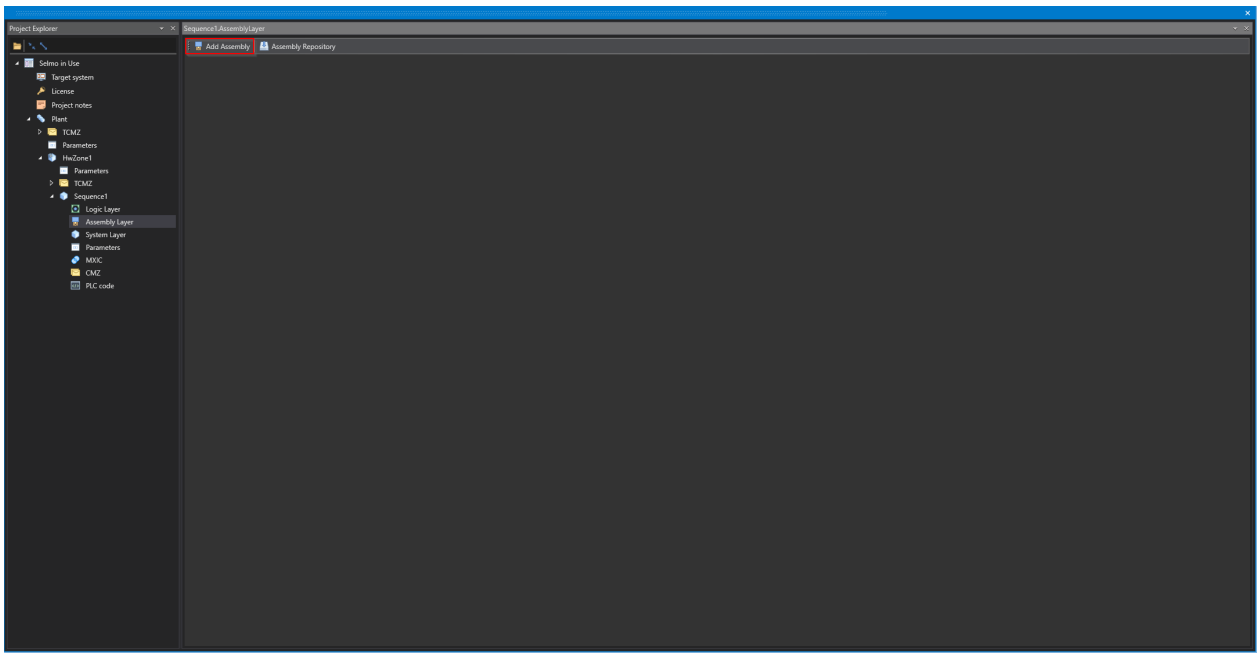


Studio

Adding drivers

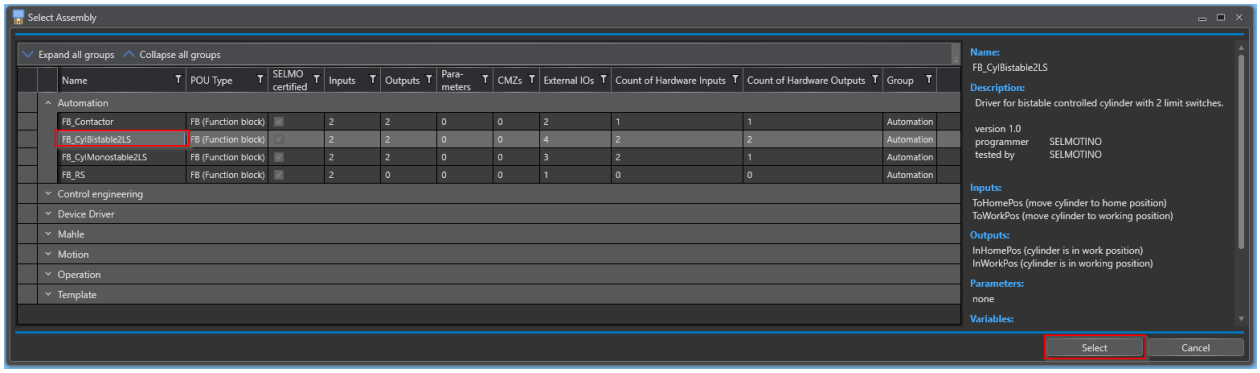
A driver is a software component that allows specific devices or functions to be controlled or managed. Adding additional drivers may be necessary to integrate more devices or functions into the application or to extend existing functionality.

By clicking the 'Add Assembly' button, additional drivers can be added in this sequence.



Studio

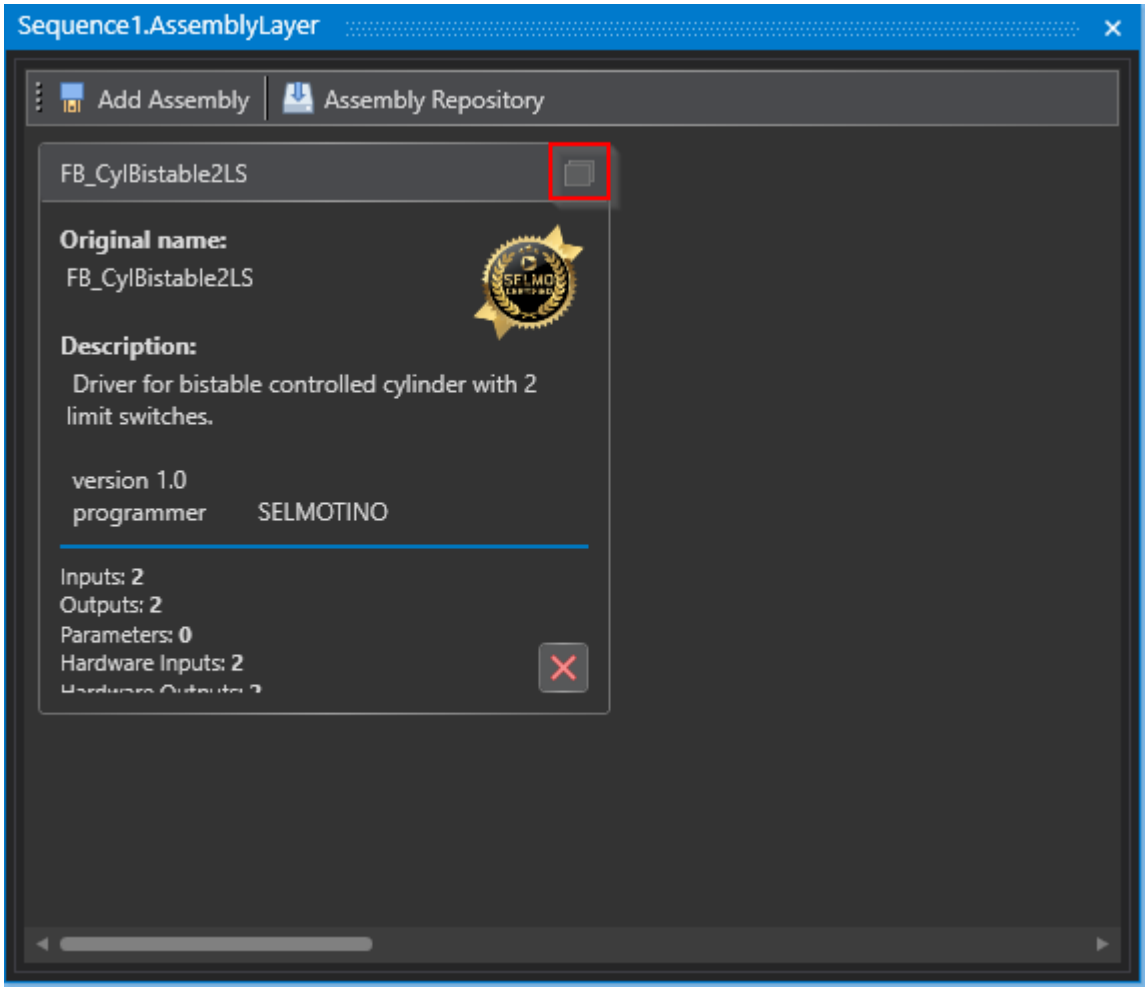
A dialog box will open allowing the user to select from a list of available drivers. Once a driver is selected and added, the user can configure the settings for that driver to match the application's requirements.



Studio

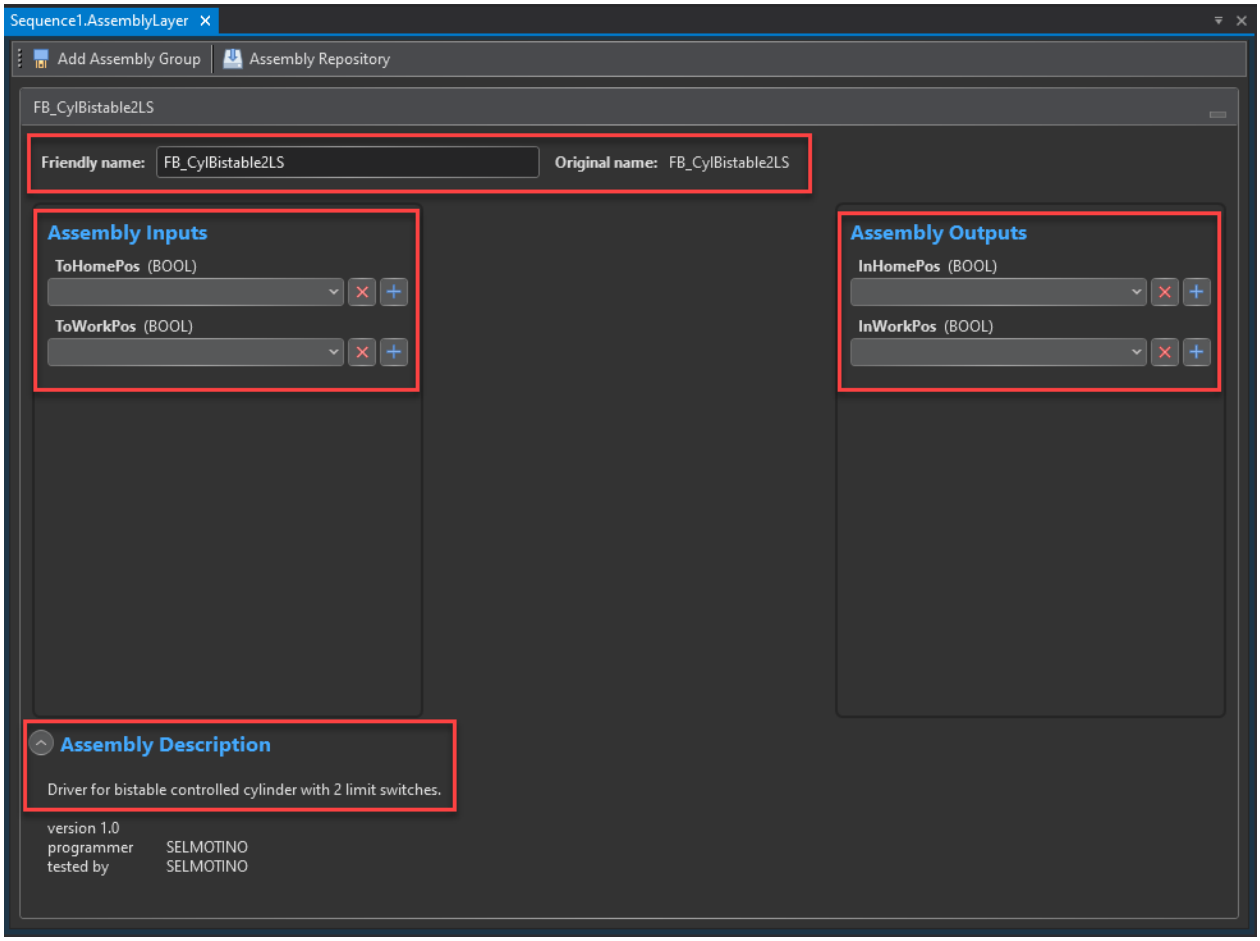
The assemblies, or drivers, can be found divided into groups, by selecting a driver and clicking Select, it can be added to the sequence.

Link Zones



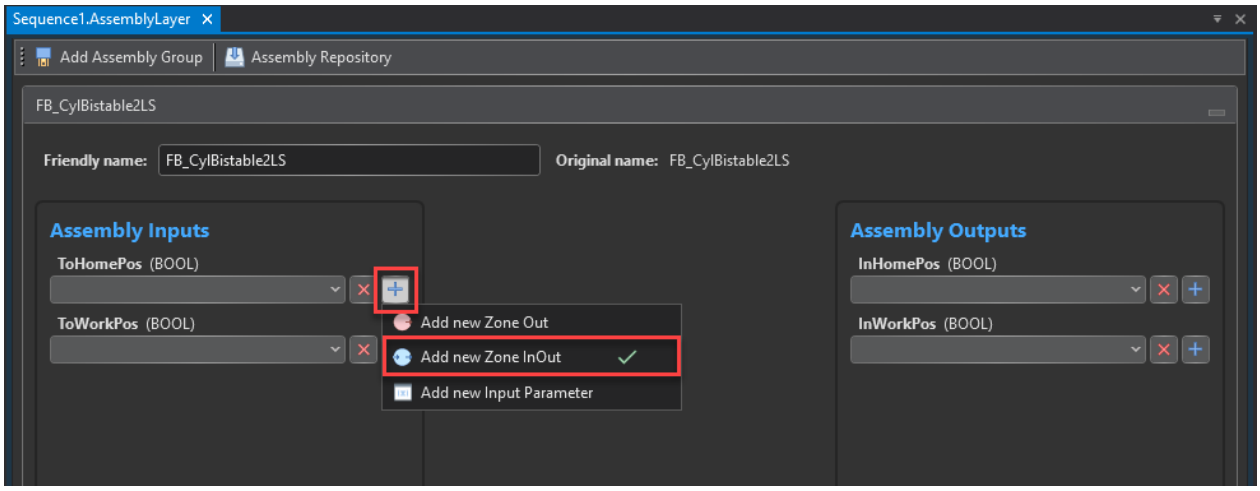
Studio

If you click on the "Maximize" button in the upper right corner of the driver, the options for this driver will open. There you can choose a name for the assembly and create or link the inputs and outputs. At the bottom of the window you will also find a short description of the driver, which can help you better understand its functions and features. By opening the options, you can customize the configuration of the driver to ensure that it integrates properly with your application and provides the desired functionality. The ability to create or link inputs and outputs is essential, as this allows you to receive data from or send data to other components in your application. Overall, the Maximize button and the options below provide a user-friendly way to integrate and customize drivers in your application.



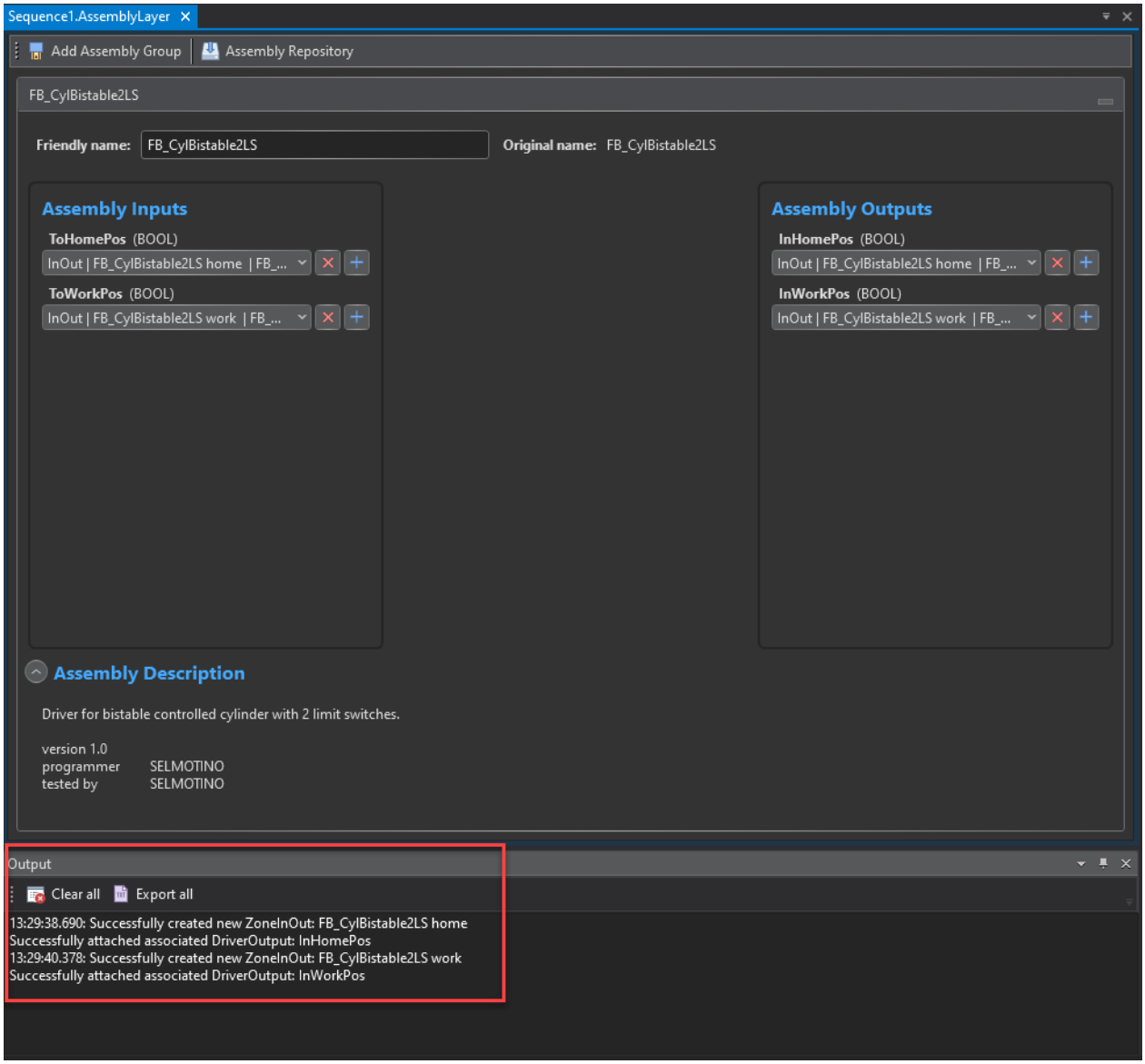
Studio

It suggests which zones to insert for inputs and outputs.



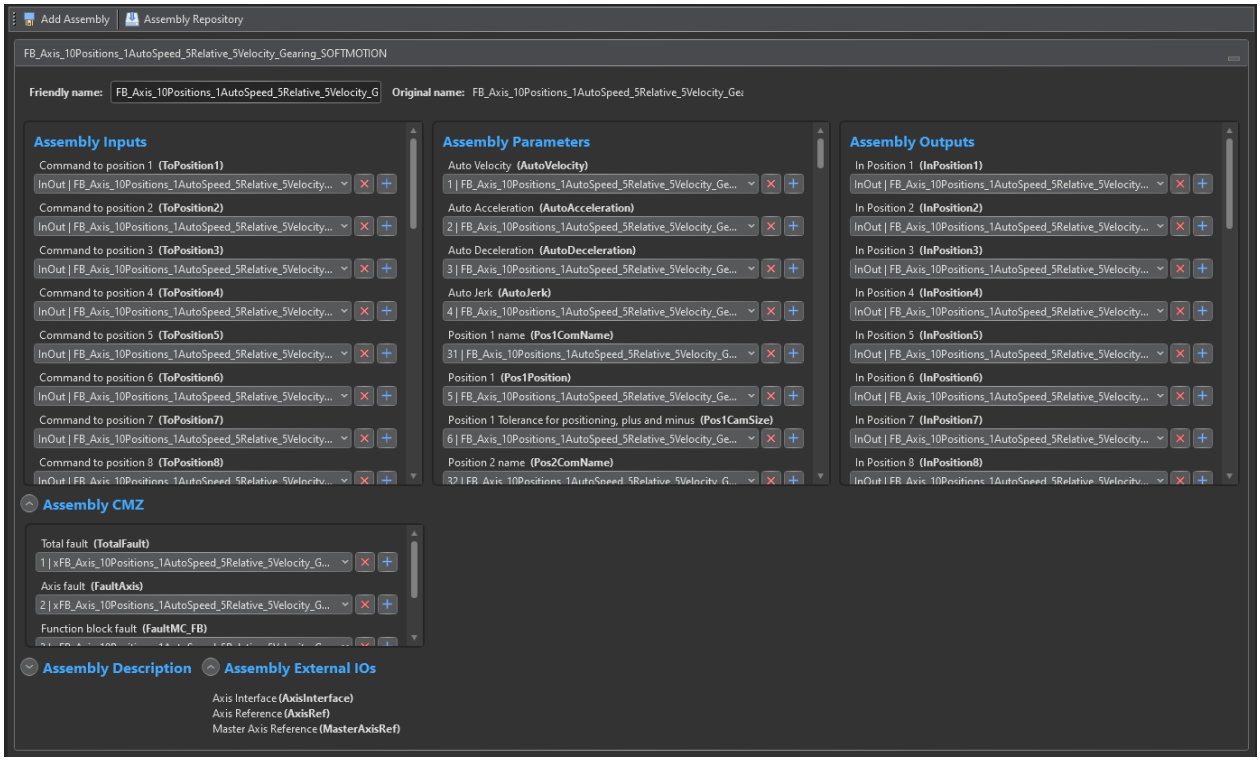
Studio

The output window shows that the zones have been successfully created.



Studio

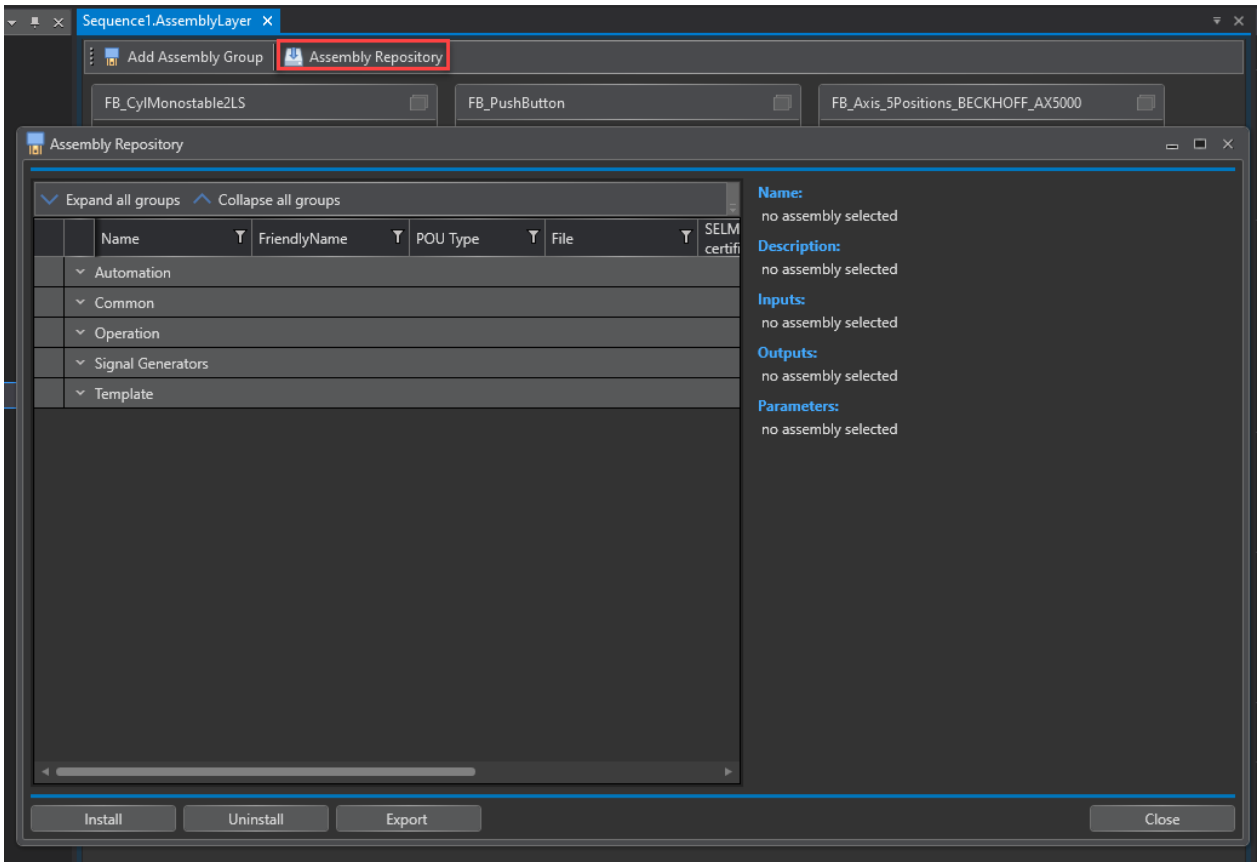
The zones are automatically integrated into the system layer, and the basic properties are already pre-filled, which saves a considerable amount of work. Only the process-defined properties need to be added. If the driver supports it, it is also possible to automatically create corresponding parameters and CMZ's.



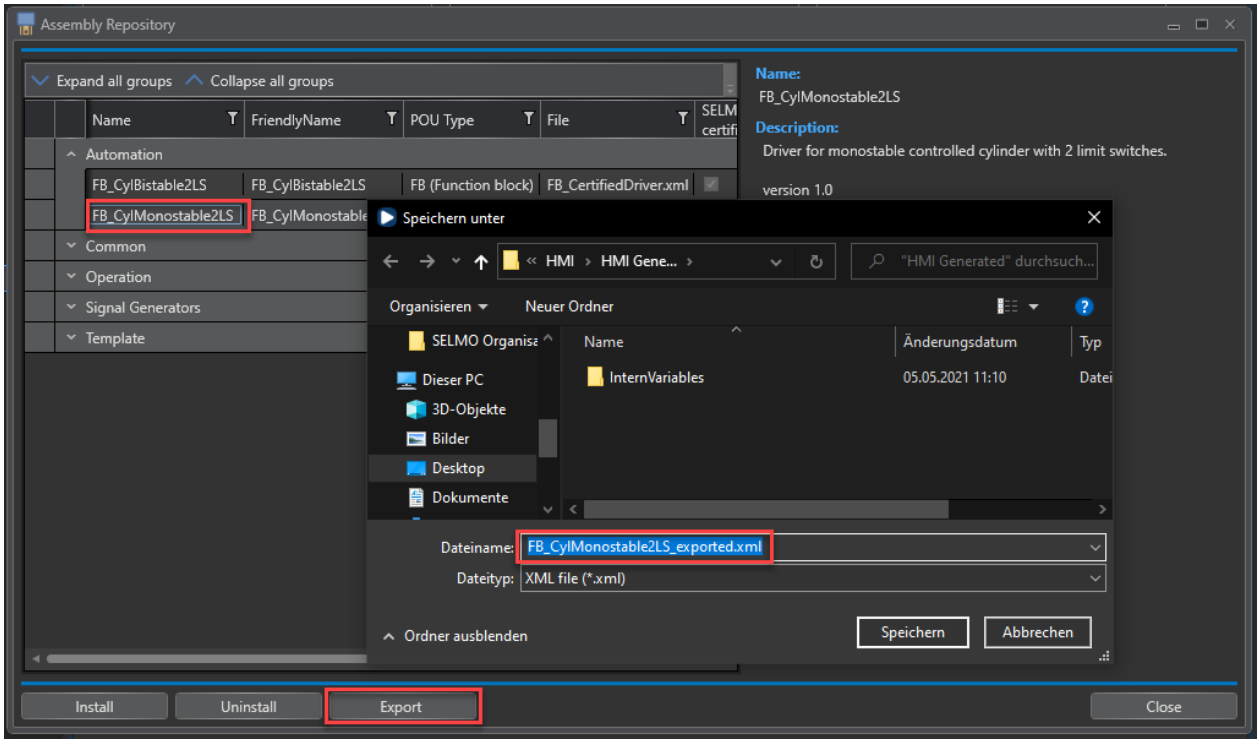
Studio

Assembly Repository

In the Assembly Repository you have an overview of all available drivers for your project. You have the possibility to import, export or remove from the collection new drivers created according to the Selmo structure.

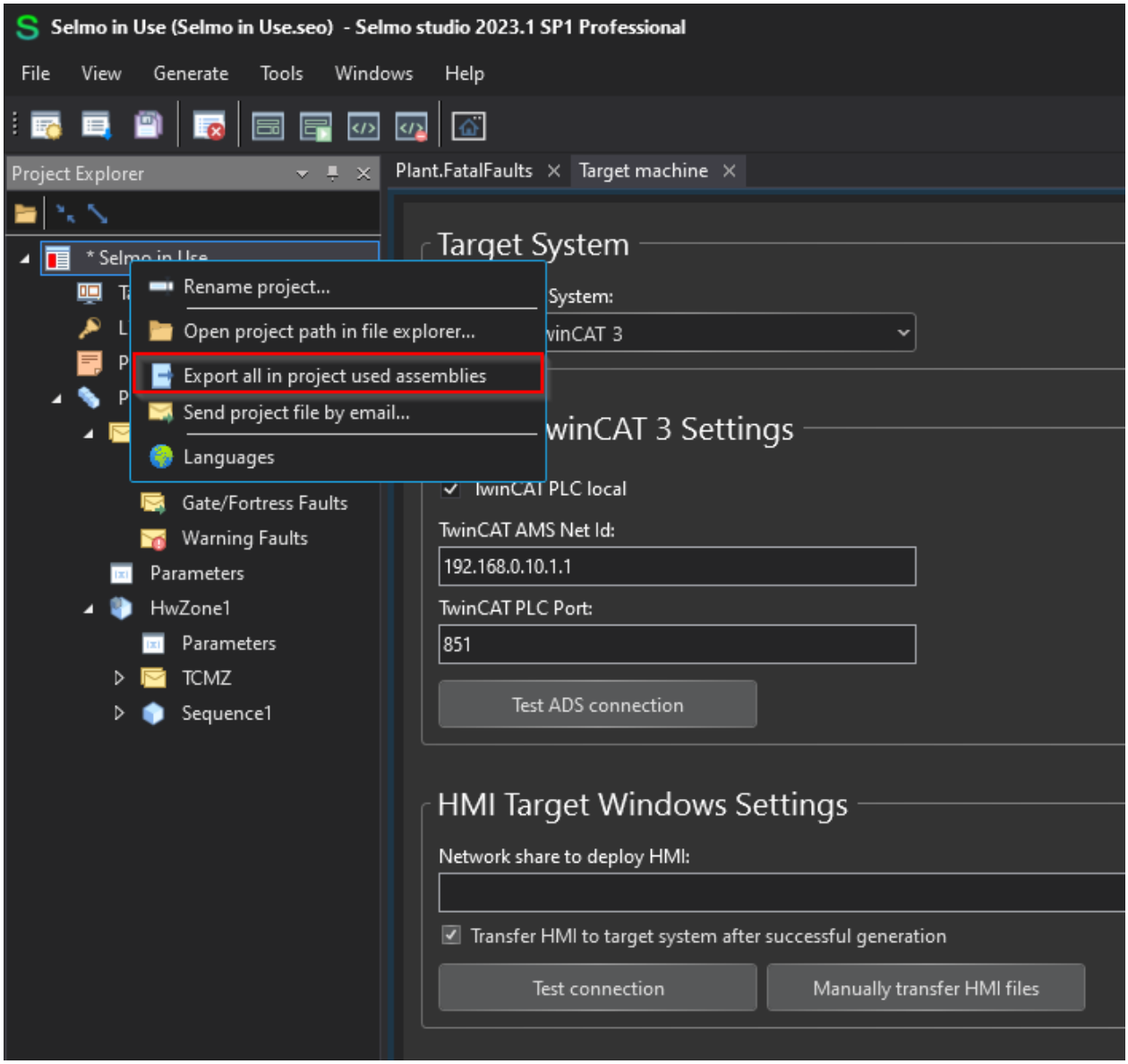


Studio



Studio

After the project has been completed or during project development, all used assemblies can be exported.



Studio

1.9 HMI

HMI stands for Human-Machine Interface and refers to the interface between a human operator and a machine, system or process. The HMI can be a graphical user interface that allows the operator to monitor data, change settings or perform actions.

At Selmo, HMI development is done in an efficient and automated way. Selmo has developed an advanced system that enables the generation of the HMI directly from the process model. Traditionally, creating an HMI, the user interface between man and machine, required a time-consuming process. Developers had to manually design and program graphics, controls, and interactions. This was not only labor-intensive but also error-prone and often showed a mismatch between the process model and the actual HMI.

Selmo recognized this problem and developed an innovative solution to streamline the HMI development process. By linking the process model to the HMI generation, the HMI elements are automatically generated from the process model data. The process model serves as the basic structure for the HMI. It contains information about the flow of the process, the different states and the relevant parameters. Based on this information, Selmo's system automatically generates the corresponding user interfaces. The advantages of this automated HMI generation are many.

First, it saves developers a significant amount of time and effort. Instead of creating each graphical component manually, they can focus on creating the process model while the HMI is generated automatically.

Second, automated generation improves the consistency and accuracy of the HMI. Because it is derived directly from the process model, the HMI is always up to date and accurately reflects the actual process. This avoids potential errors and inconsistencies that could occur with manual development.

In addition, automated HMI generation enables better collaboration between different team members. Developers, engineers and operators can work together on the process model and make changes that are automatically updated in the generated HMI.

The Selmo approach to automated HMI generation has the potential to significantly improve the efficiency and quality of HMI development in industrial automation. By integrating the process model and HMI, development is accelerated, errors are reduced and collaboration is facilitated.

Selmo is thus a pioneer in the industry and is driving progress in human-machine interface technology.

1.9.1 Sequence Control

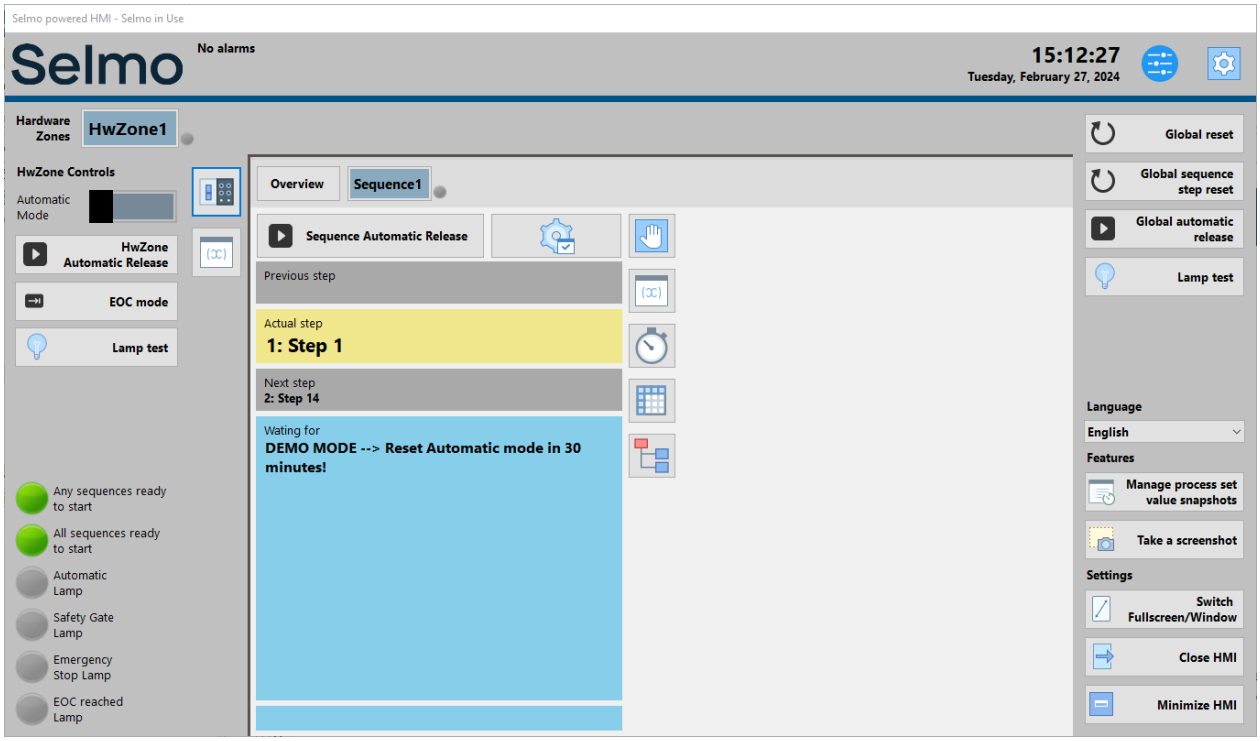
At Selmo, the HMI (Human-Machine Interface) is equipped with a powerful Sequence Control function that allows creating an individual Sequence Control for each sequence in the process. This function enables precise control, monitoring and diagnosis of sequences within the production process. The Sequence Control provides a clear overview of the steps and zones required for a particular sequence to run smoothly.

A particularly useful feature of Sequence Control is the Single Step function. This feature allows the operator to manually execute a step to test or verify the sequence. This allows the operator to step through the process and ensure that each step is working correctly before moving on to the next step in the process.

In addition, the Sequence Control system provides a step selection function. This feature allows the operator to choose between different steps in a Sequence based on current requirements or conditions. This is especially useful when certain steps need to be skipped or repeated to meet process requirements.

Another convenient feature of Sequence Control is the automatic search for a valid step. This means that the system automatically searches for the next valid step to execute based on current conditions or predefined rules. This saves time and ensures that the process runs smoothly without the operator having to manually search for the next step.

Overall, Sequence Control enables precise control and diagnosis of sequences. It provides a clear overview of the steps and zones required, facilitates step selection, and enables easy searching for valid steps. These features help improve the efficiency of the production process, minimize human error, and ensure reliable execution of sequences.



HMI

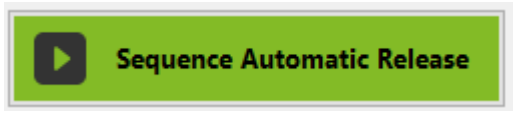
Start Sequence

To start the automatic mode of the sequence, the Automatic Mode switch must be activated.

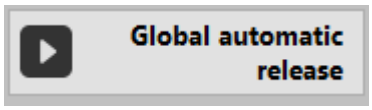
If no error or CMZ is active and thus the 'Waiting for' is colored blue, the automatic mode can be started via the button,

can be started. This releases all sequences in this HwZone.

If you only want to release this sequence, a special button is available for this purpose.

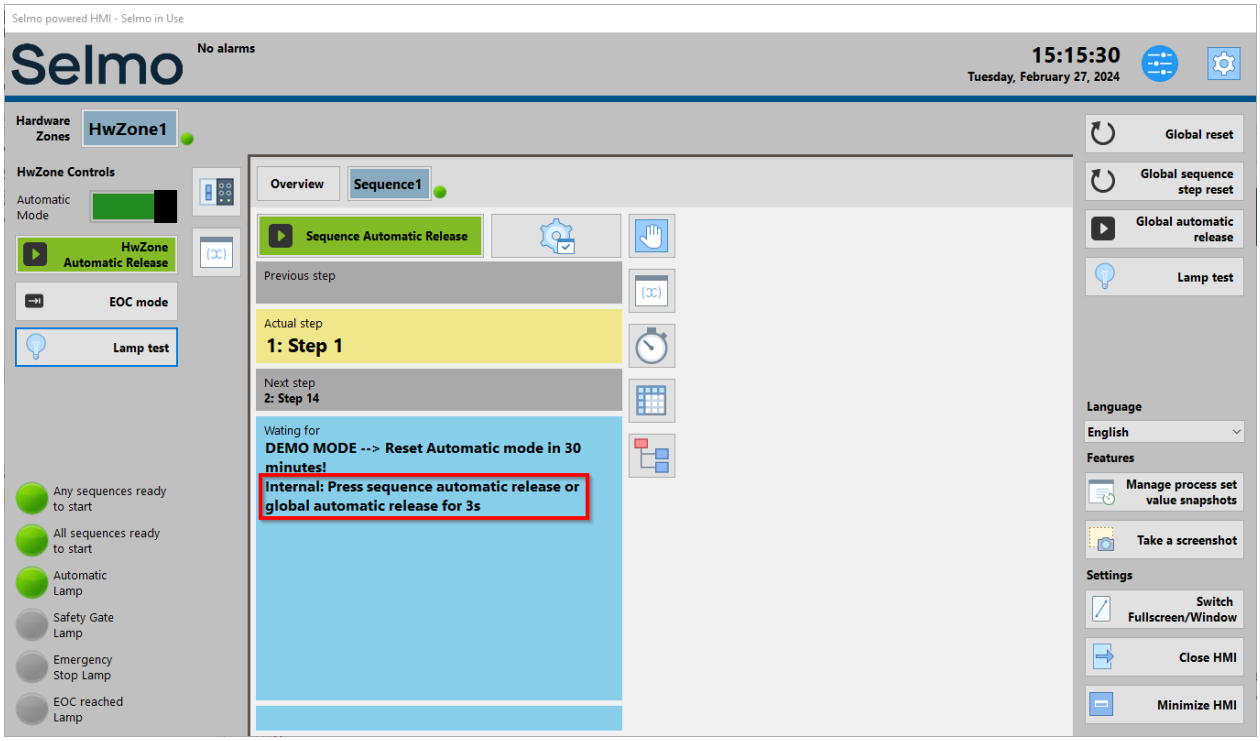


With this button it is possible to start all sequences in all hardware zones in which the automatic mode switch is set to 'Automatic'.



As shown in the 'Waiting for', it is necessary to keep the button pressed for the set time of the Startup-Delay in the Selmo-Studio,

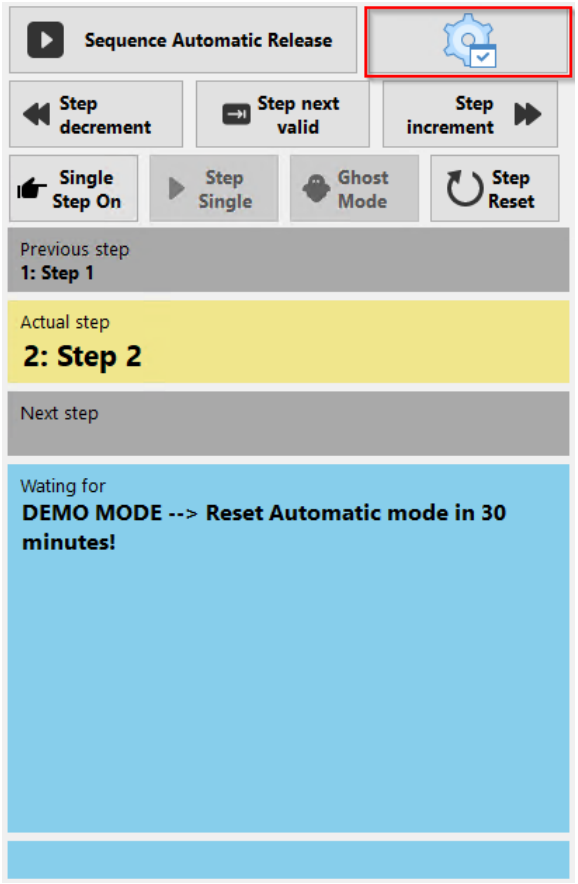
for the set time of the startup delay in the Selmo Studio. The default value is 3s.



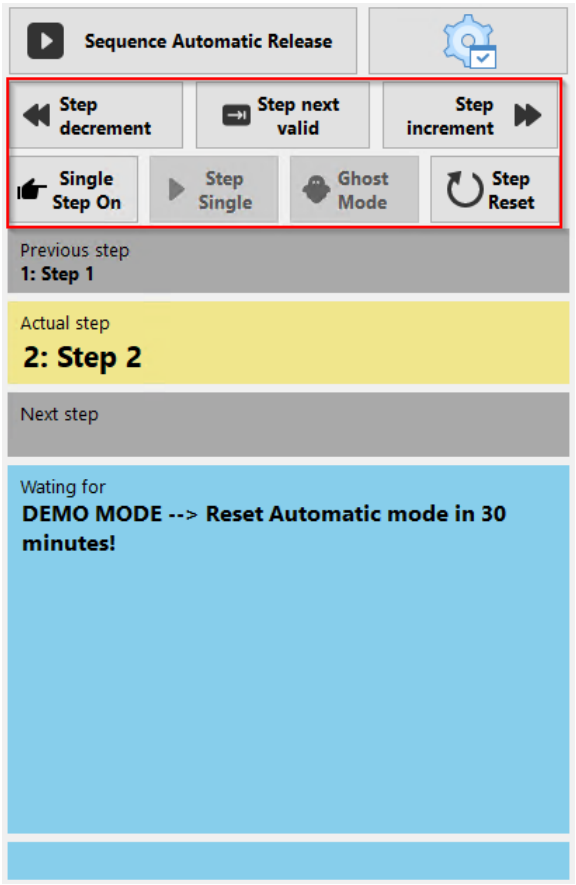
HMI

Sequence Control

Clicking on the gear icon opens the Sequence Control user interface.



HMI



HMI

Note

It is important to note that the step functions "Step decrement", "Step next valid", "Step increment" and "Step Reset" are only available in Manual Mode and give you the possibility to control the sequence step by step. Once the Automatic Mode is activated, the system takes over the automatic execution of the steps according to the predefined conditions and rules.

Step decrement

The "Step decrement" is a function that allows you to undo a step in the Sequence. By using this function, you can go back to previously executed steps in the sequence and thus restore the previous state. It is convenient for correcting mistakes or repeating certain actions that may have been skipped or performed incorrectly.

Step next valid

The "Step next valid" is a function that allows you to search for the next possible state that matches the current machine state. This function is especially useful when you are going through specific steps in a sequence and want to make sure that the next step is valid and matches the current state of the machine. By pressing the "Step next valid" button, the Sequence analyzes the current state of the machine and looks for the next valid step based on the conditions and rules set for the machine state. In this way, you can ensure that the steps in the Sequence are executed according to the current machine situation. This feature enables precise and automatic navigation through the Sequence by ensuring that only the steps compatible with the current machine condition are executed. This improves efficiency and avoids potential errors or inconsistencies.

Step increment

The "Step increment" is a function that allows you to take a step forward in the sequence. By pressing the "Step increment" button, you can jump to the next action in the sequence and advance the progress. This feature is especially useful if you want to skip specific steps in the Sequence, either because you are sure they have already been completed or for other reasons. Moving forward in the Sequence can save time and speed up the workflow by moving directly to the next relevant step. It is important to note that when using "Step increment", care should be taken to ensure that the skipped step does not contain significant actions or conditions that could affect the rest of the Sequence. A careful review of the sequence structure and the current state of the machine is advisable to ensure that skipping a step does not have unexpected consequences. Overall, the "Step increment" provides a convenient way to control the progress in a sequence and to target the relevant steps to be executed next.

Step reset

The "Step Reset" is a function that allows you to reset the step counter. Pressing the "Step Reset" button resets the counter to the initial value. This function is especially useful if you want to reset the progress in the Sequence and start over. By resetting the step counter, you have the option of replaying the Sequence without taking the previous progress into account. This can be useful if you want to troubleshoot or retest a Sequence without considering the previous runs. It is important to note that other variables or states related to step progress may also be reset when resetting the step counter. This depends on the specific implementation of the Sequence. The "Step Reset" provides a simple and effective way to reset the step progress in the Sequence and start over if needed.

Note

The Single-Step-On Mode can be turned on or off regardless of the operating mode. However, the confirmation button, like the "Step Single", only works when the automatic mode is enabled.

Single Step On

Switching on the "Single Step On" activates the Single Step mode. In Single Step mode, the Sequence is configured to stop after each step and wait for your confirmation before moving on to the next step. This mode allows you to step through the Sequence and review or test each action individually. After executing a step, you must press the "Step Single" button to resume execution and move to the next step. This gives you complete control over the process and allows you to monitor each step carefully. The Single Step mode is especially useful when troubleshooting, testing, or reviewing a Sequence step-by-step. You can check the machine's state or other relevant parameters after each step and make sure everything is working as expected before moving on to the next step.

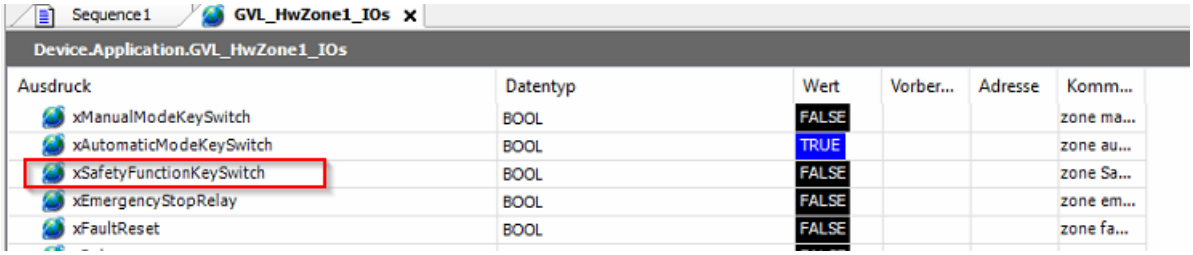
Step Single

The "Step Single" is a confirmation button used in Single Step mode to advance the steps step by step. In Single Step mode, the Sequence stops after each step and waits for your confirmation before moving to the next step. When ready to move to the next step, press the "Step Single" button to resume execution and trigger the next step. Pressing the "Step Single" button gives you control over how fast the Sequence progresses. You can control the progress step by step and review each action individually before moving to the next step. This allows for detailed monitoring and review of the sequence and is especially useful when troubleshooting, testing, or reviewing the sequence step-by-step. The "Step Single" button allows you to effectively use the single step mode and advance the sequence at your own pace.

Ghost Mode

Ghost mode can be used for commissioning purposes, for example. For example, if a sensor is missing in a zone, ghost mode can be activated for this zone in order to test the entire model anyway. With the zone in ghost mode, a delay starts until the sequence check is completed and then continues.

The ghost mode button is grayed out in the HMI, which means that ghost mode must be activated separately. Ghost mode can only be activated via the safety function key, which acts as a key switch and is only accessible to authorized persons. The safety function key exists per hardware zone and must be set to True in Codesys under the IOs of the hardware zone in order to be able to start the button in the HMI.



Step Information

HMI

Preview step

This step shows the previous step or state.

Actual step

This step shows the current state or progress.

Next step

This step shows the next step or the next required action.

Zones Information

HMI

Waiting for

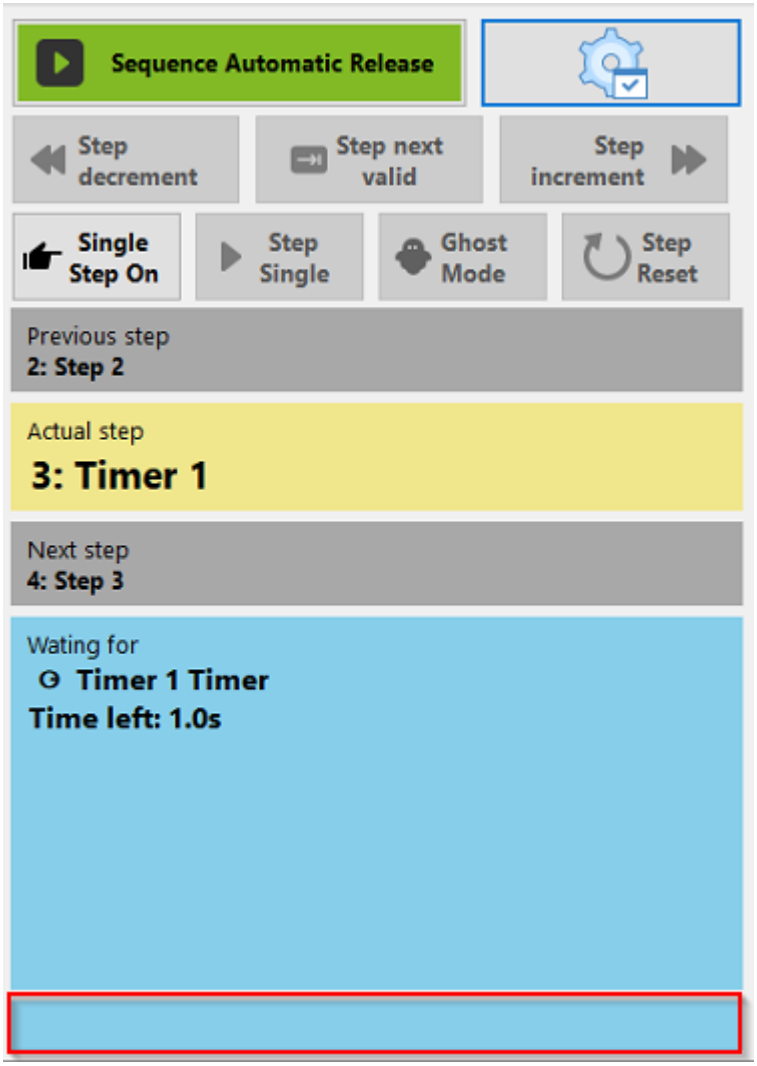
The defined conditions or zones for this step are displayed in order to reach the next step. This allows the operator to immediately see what signal is missing and act accordingly.

e.g.: Sequence-Check Zone 1

Zone 1

Monitoring-Check Zone 2

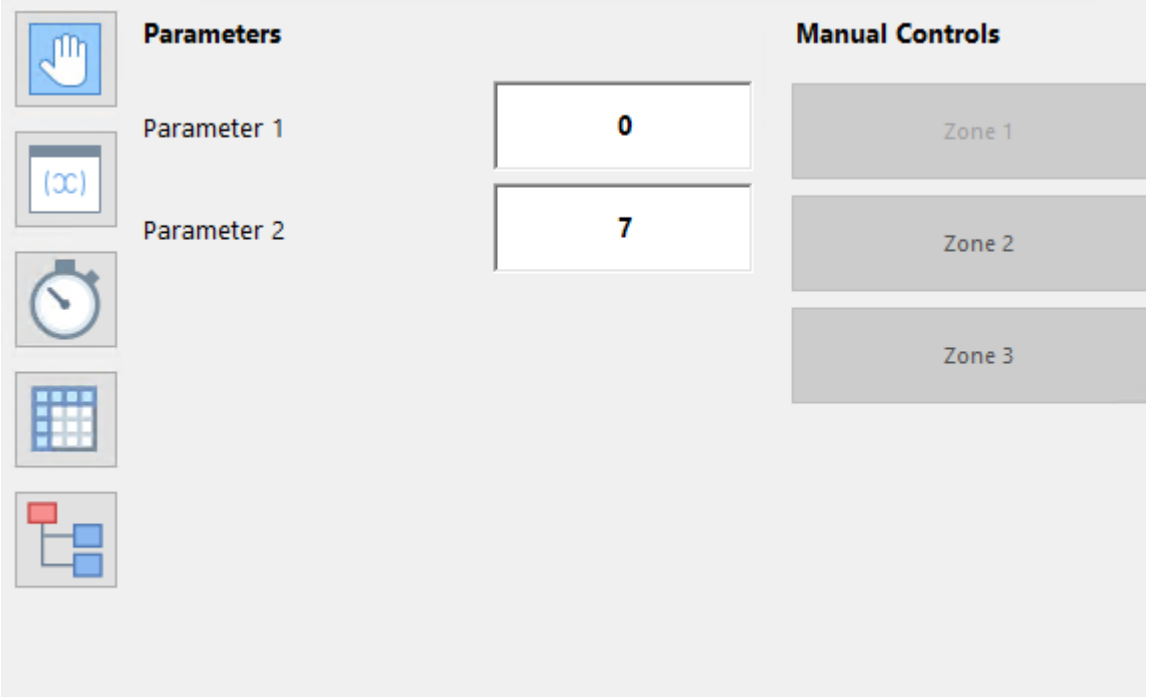
Zone 2



HMI

When the automation is stopped or interrupted by an interlock, the zone information (interlock or sequence check) is stored. This serves an improved analysis. These entries can be deleted by pressing the Global Reset button.

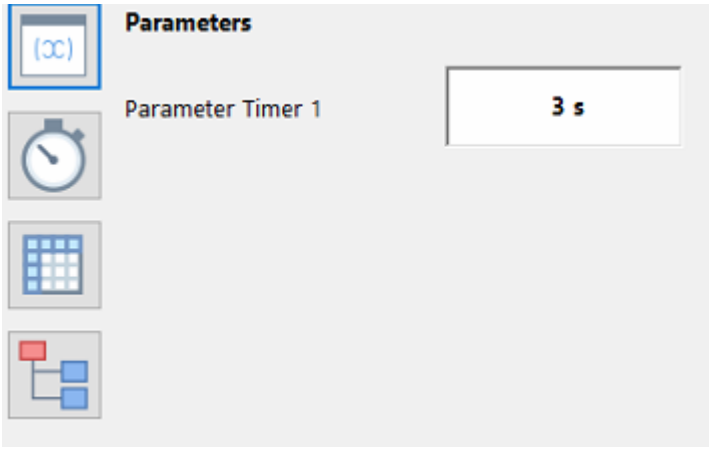
Manual mode



HMI

The zones defined in the Selmo Studio with the "HMI Button" allow to control them from the HMI as long as the Automatic mode is not active. It is important to note that manual operation is only possible when no fault (TCMZ, CMZ) is active.

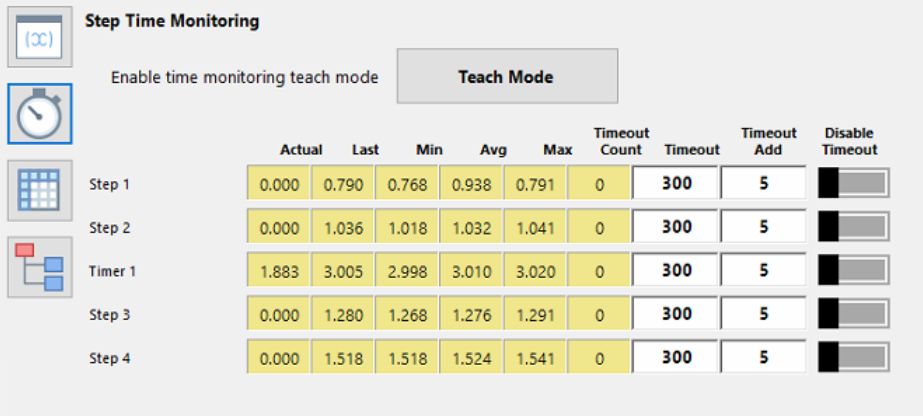
Parameter



HMI

In this section you can find the parameters 135 for this sequence created in Selmo Studio.

Step Time Monitoring



HMI

When evaluating the time, a step has been taken, various metrics can be used to get a comprehensive picture of performance. Here are some common metrics used in conjunction with Step Time Monitoring:

Actual: This is the actual measured time that a particular step took to complete. It is the exact value recorded based on the measured data.

Last (Last): This refers to the time that the step took the last time it was executed. It shows the last measured value and can be used to track changes in performance over time.

Min (Minimum): This is the smallest measured value for the time the step took. It indicates how fast the step can be executed under the best conditions.

Avg (Average): This metric gives the average value of the times the step took during a given period or number of executions. The average value can provide information about the typical or average step performance.

Max (Maximum): This is the largest measured value for the time the step took. It shows the longest time the step took to execute and may indicate bottlenecks or unusual delays.

Timeout Count: Refers to the number of occurrences when a step or operation did not complete within a defined time limit or timeout value. It counts the number of instances when the execution time of a step exceeded the maximum allowable time frame.

By combining these evaluations - Actual, Last, Min, Avg, Max and Timeout Count - a comprehensive analysis of the performance and execution times of individual steps can be enabled. This can identify potential bottlenecks, inefficient areas or issues that need to be optimized to improve overall performance.

Timeout

By setting a timeout value, the system can monitor the execution time of a step and verify that it completes within the defined time limit. If the execution time exceeds the timeout value, the step is considered failed or not completed in time.

Timeout Add

Timeout additional refers to an additional value that is added to the existing timeout value. This extends the timeout value for a given step. For example, suppose you have already set a timeout value of 10 seconds and want to add an additional value of 5 seconds. In this case the new timeout value would be 15 seconds.

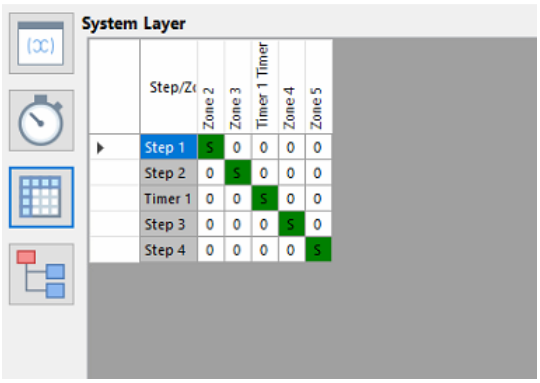
Disable Timeout

Disabling timeout, or disabling step time monitoring, refers to when a function or process has no time limits or limits on how long it can execute.

Teach Mode

The "Teach Mode" allows to teach the timeout times by measuring 5 min and deriving the timeout times from it.

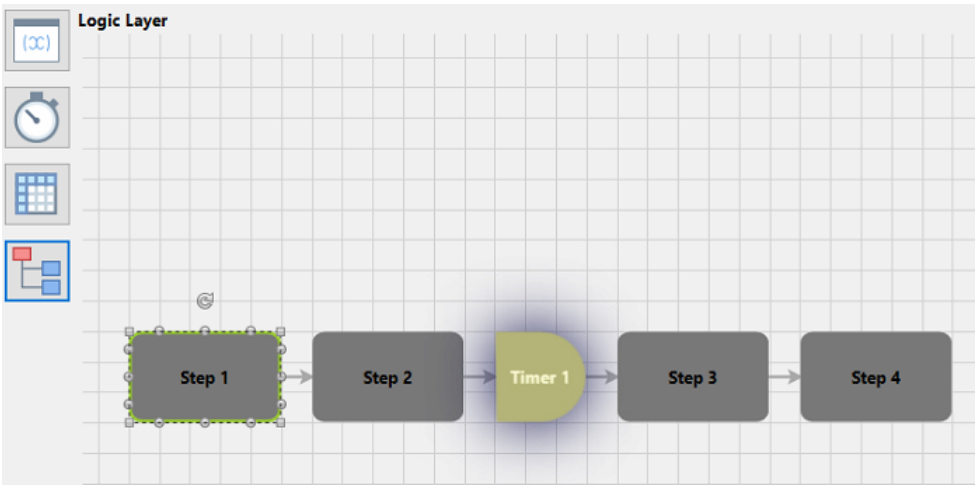
System Layer



HMI

View of the System Layer as defined in Selmo Studio.

Logic Layer



View of the Logic Layer as defined in Selmo Studio, with highlighting of the active step.

1.9.2 Alarm Handling

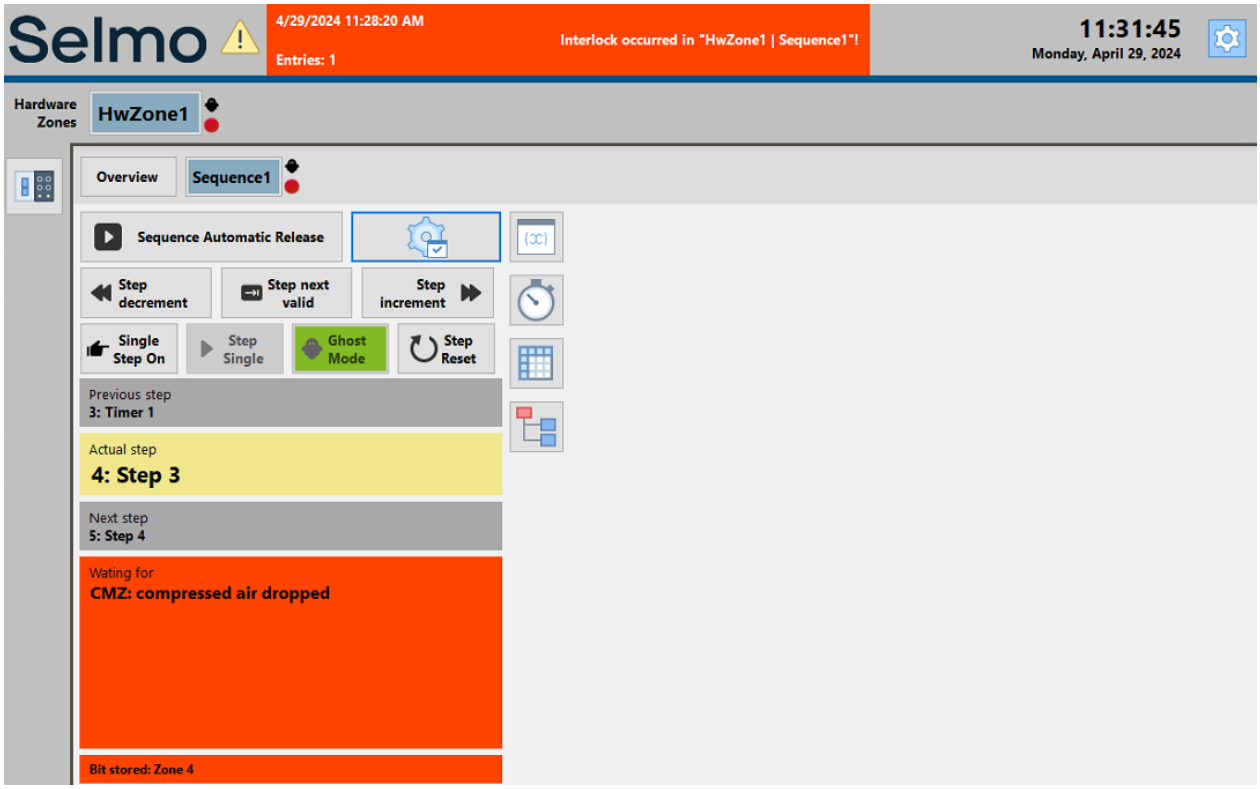
Alarm handling refers to the process of handling alarms or alerts that occur in a system. It includes the identification, capture, prioritization, processing, and resolution of alarm conditions.

Alarm handling includes the following steps:

1. Alarm detection: the system detects a deviation from normal operating conditions and generates an alarm.
2. Alarm display: The alarm is displayed on a user interface or monitoring system. This can be in the form of visual cues, audible signals, or messages.
3. Alarm Prioritization: Alarms are prioritized by their urgency or severity to determine which alarms need to be addressed first.
4. Alarm notification: The alarm management system notifies relevant individuals or teams when an alarm occurs. This can be done through email notifications, SMS messages, or other means of communication.
5. Alarm analysis: the cause of the alarm is investigated to determine what caused the problem. This may require a manual review of systems, logs, or sensor data.
6. Alarm Response: based on the analysis, an appropriate response is initiated to resolve the problem or minimize the impact. This may include taking corrective action, initiating maintenance, or calling in skilled personnel.
7. Alarm tracking and documentation: the entire alarm handling process is documented, including the timing of alarms, actions taken, and results.

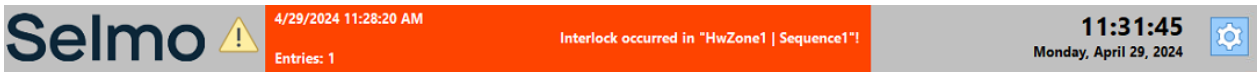
Effective alarm handling enables early detection of potential problems, rapid response, and minimization of operational disruptions to ensure smooth system operation.

When an alarm occurs in the form of a CMZ (Constantly Monitored Zone) or TCMZ (Total Constantly Monitored Zone), this is indicated in the alarm bar and in the "Waiting for" box of the Sequence. At the same time, the background color of the "Waiting for" indicator changes to red, and a red light is activated in the Sequence Overview to signal the fault. In any case, faults must be confirmed with a "Reset" before automatic operation can be resumed.



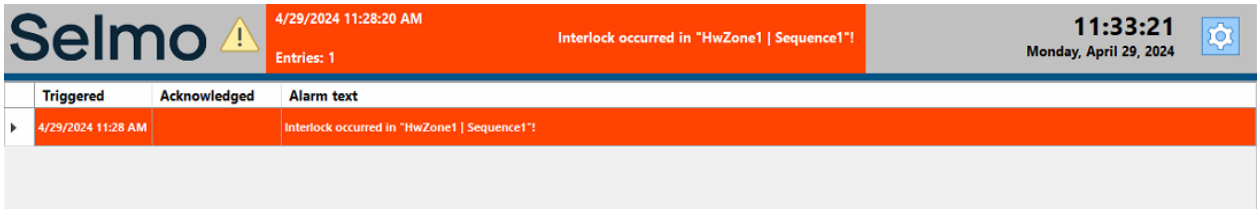
HMI

Warnings, on the other hand, are only displayed in the alarm bar, with an orange background. These warnings have no influence on automatic operation.



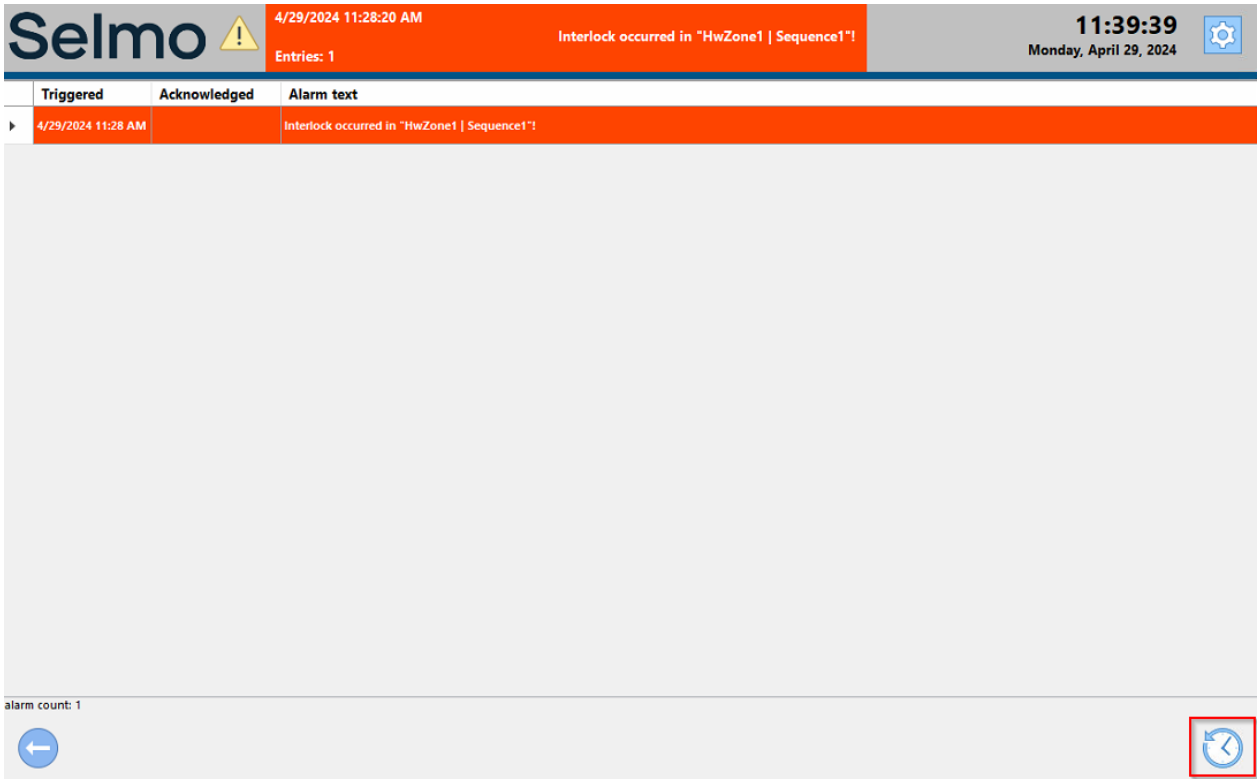
HMI

Double-clicking on the alarm bar opens the alarm overview on the HMI. In the alarm list, both faults and warnings are displayed as long as they are active. All relevant information on the alarm signals, such as type of fault or warning, time of occurrence and any assigned information, is listed here. The alarm list provides a clear display and tracking of all current alarm conditions, so that operating personnel are quickly informed of the status and can take appropriate action if necessary.



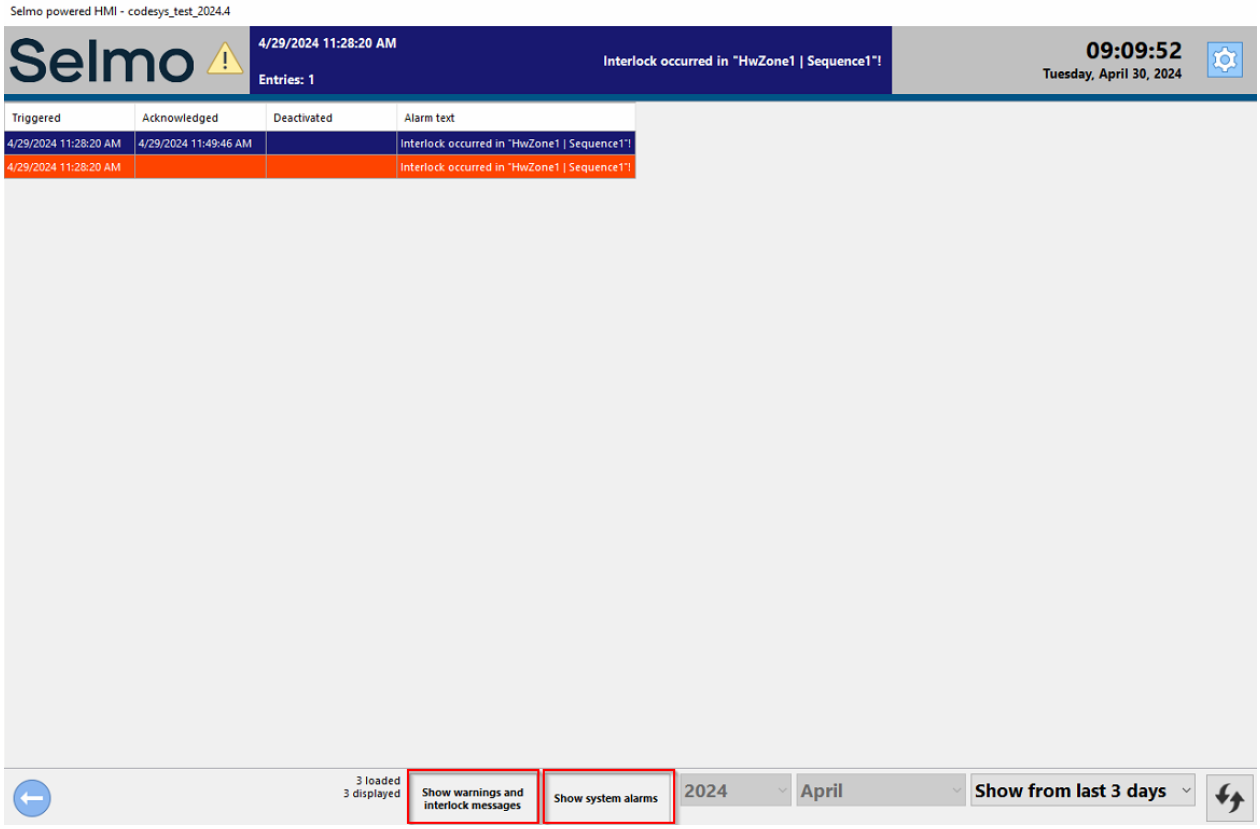
HMI

In the alarm history, the alarms that have occurred are logged and stored with a time stamp. All past alarm events are recorded and sorted chronologically. In this way, the alarm history enables subsequent analysis and tracing of past alarm conditions. Storing alarms with time stamps helps identify trends, detect recurring problems, and conduct follow-up investigations. Alarm history thus serves as a valuable tool for monitoring and evaluating system performance and assists in the continuous improvement of plant or machine efficiency.



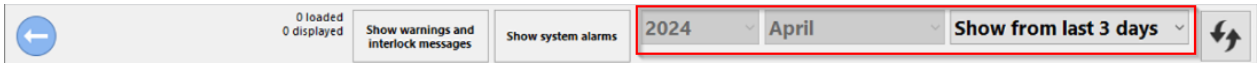
HMI

In order to display warnings and interlocks or system faults in addition to the TCMZ and CMZ, it is necessary to enable the "Show warning and interlock messages" and or "Show system alarms" option. By enabling this option, all warnings, and interlocks and or system errors will be displayed in the user interface.



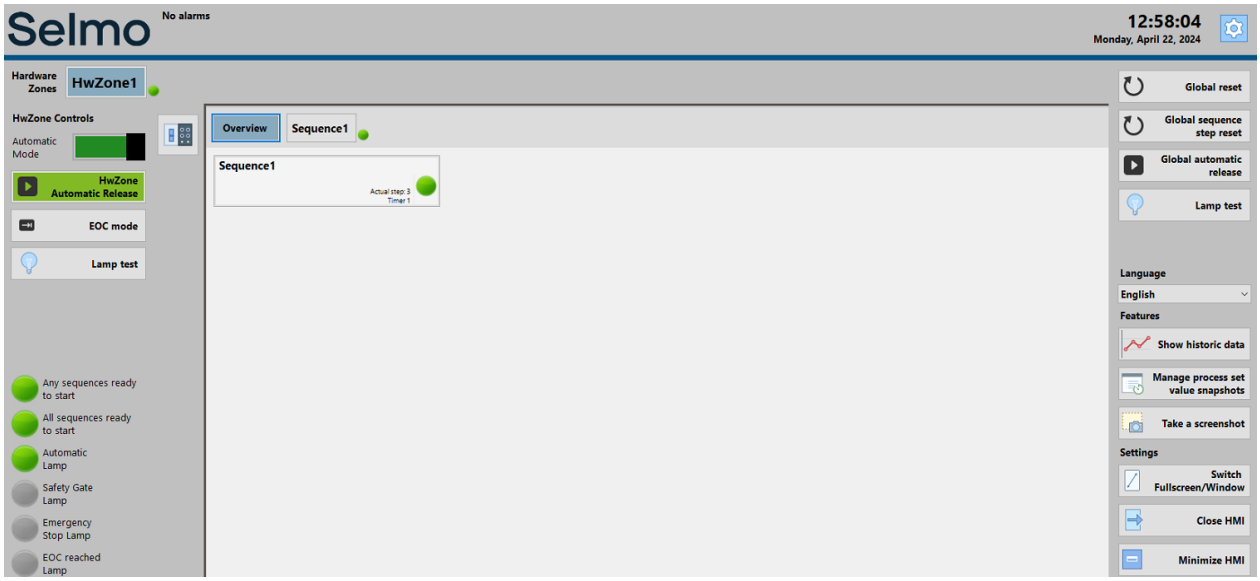
HMI

It is possible to retrieve alarms by year, month or for the last 3, 7 or 30 days. This allows operators to target past alarm events and select specific time periods to analyze alarm patterns, trends, or special events. The flexible retrieval of alarms by different time intervals allows detailed examination of alarm history and facilitates fault diagnosis, performance analysis and continuous system improvement.



HMI

1.9.3 Plant



HMI

Global Reset

A global acknowledgement of pending errors.

Global sequence step reset

Resets all sequences to step 1.

Global automatic release

Release of all sequences which are "Ready to start".

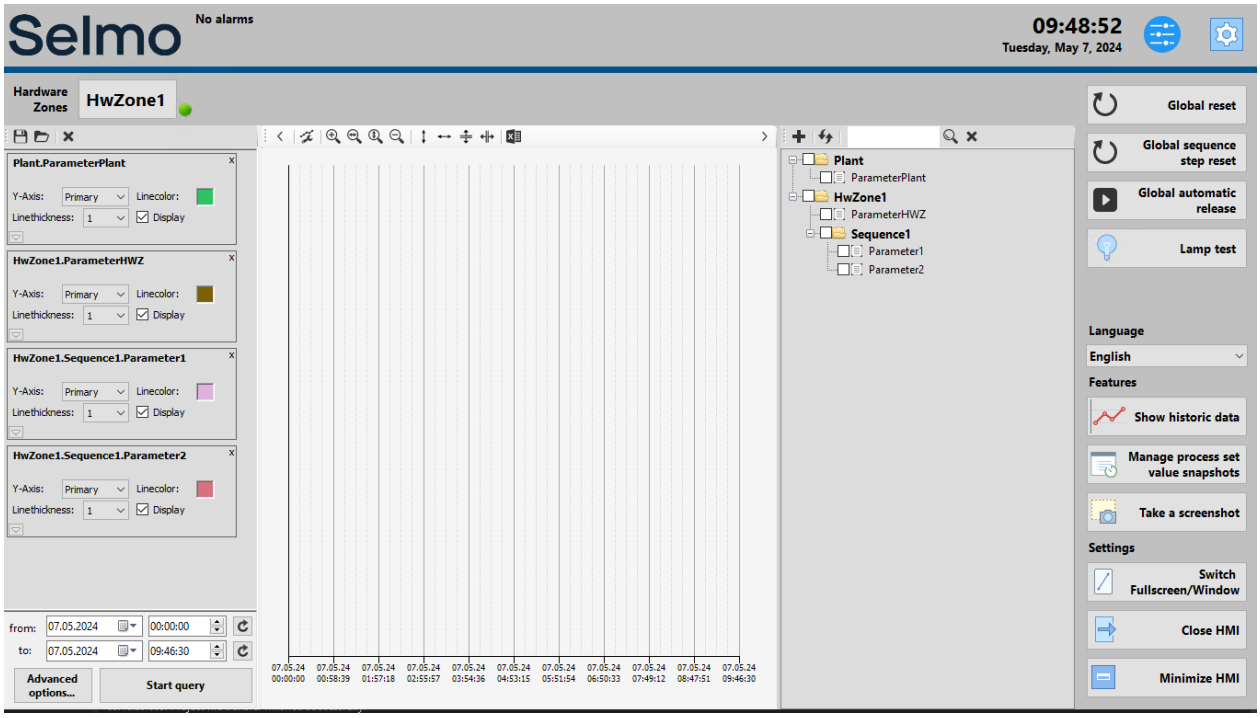
Must be pressed for the defined time of the startup delay, default 3s.

Language

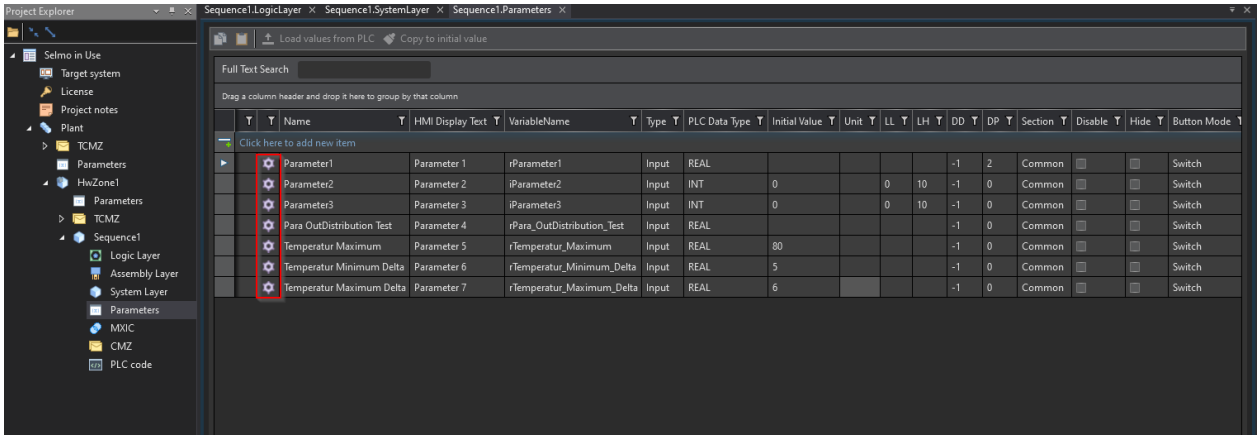
Language switching, with the languages defined in the Selmo Studio, from the dropdown menu.

Show historic data

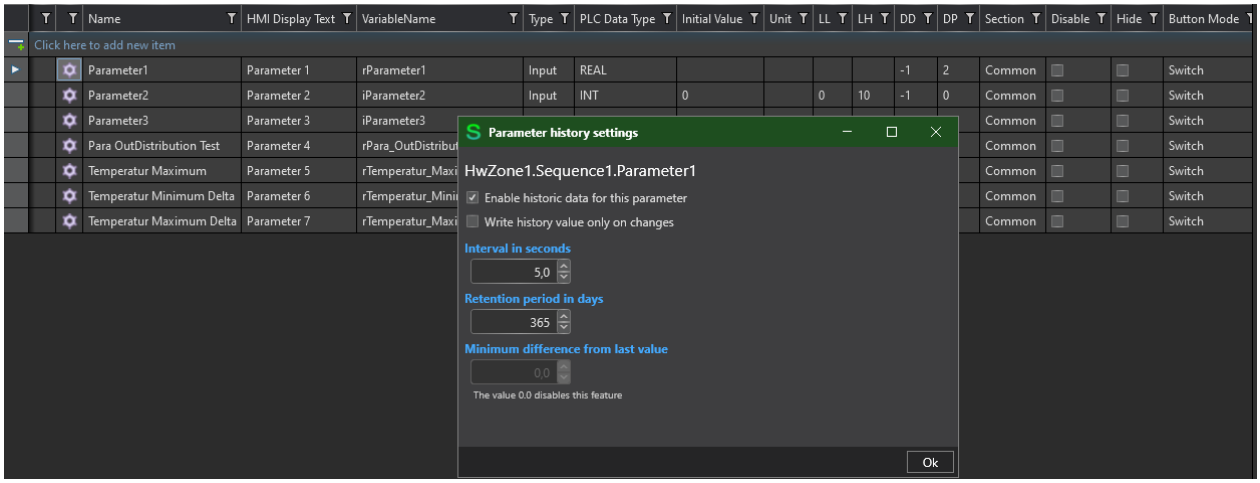
This function makes it possible to display historical data. This is only possible if you create parameters at plant, hardware zone and sequence level. The parameters to be displayed must be dragged from the right-hand side to the left-hand side.



To use this function, double-click on the cogwheel symbol under 'Parameters' in Selmo Studio.



The cogwheel symbol opens a separate window. Here it is important that 'Enable historical data for this parameter' is ticked so that the parameter can be used and is visible in 'Show historical data' in the HMI. In the window you also have the option of selecting the 'Interval in seconds' time and the dwell time can also be selected. The minimum difference to the last value can also be selected.



Once the settings have been set, the cogwheel symbol is colored green, indicating that it can be used for 'Show historical data' in the HMI.

Manage process set value snapshots

Provides the possibility to save and restore all parameters in the HMI.

Take a screenshot

Take a screenshot of the current view of the HMI.

Switch Fullscreen/Window

Switch between fullscreen and window mode of the HMI.

Close HMI

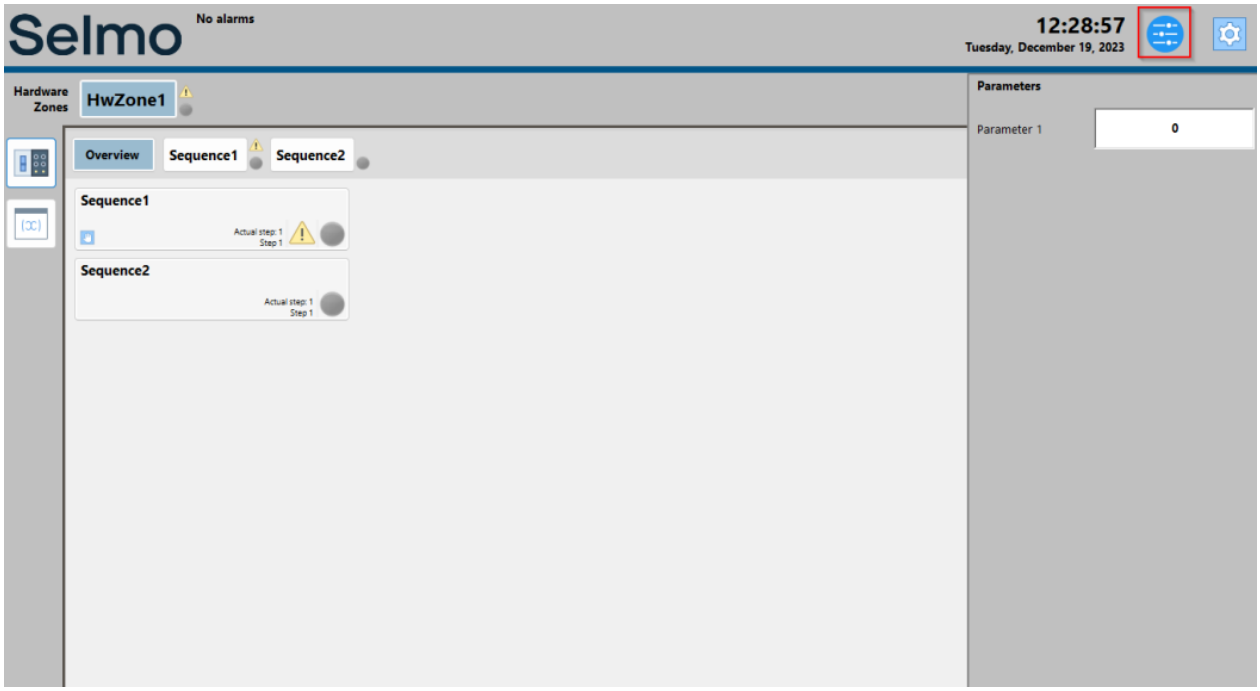
Close HMI application.

Minimize HMI

Minimize the HMI to the taskbar.

Plant Parameters

All parameters defined in the Plant are displayed here.

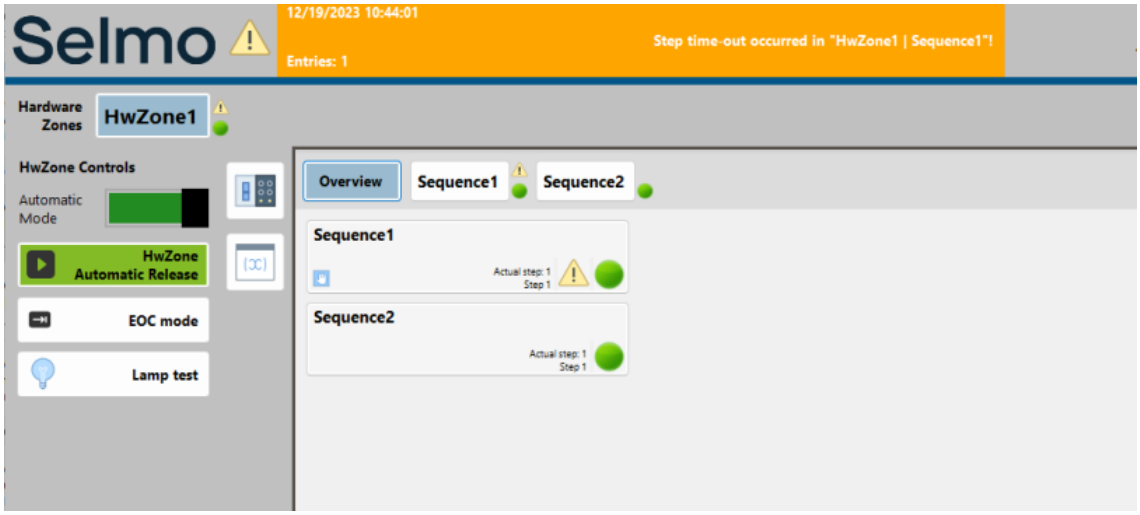


HMI

1.9.4 Hardware Zone

In the overview all sequences of the corresponding hardware zone are displayed. Each sequence has its own information window, in which the current step, the status (operating mode, alarm) and the timeout are displayed.

The warning sign  symbolizes a timeout, a single step and the EOC mode, it indicates that manual intervention is required.



HMI

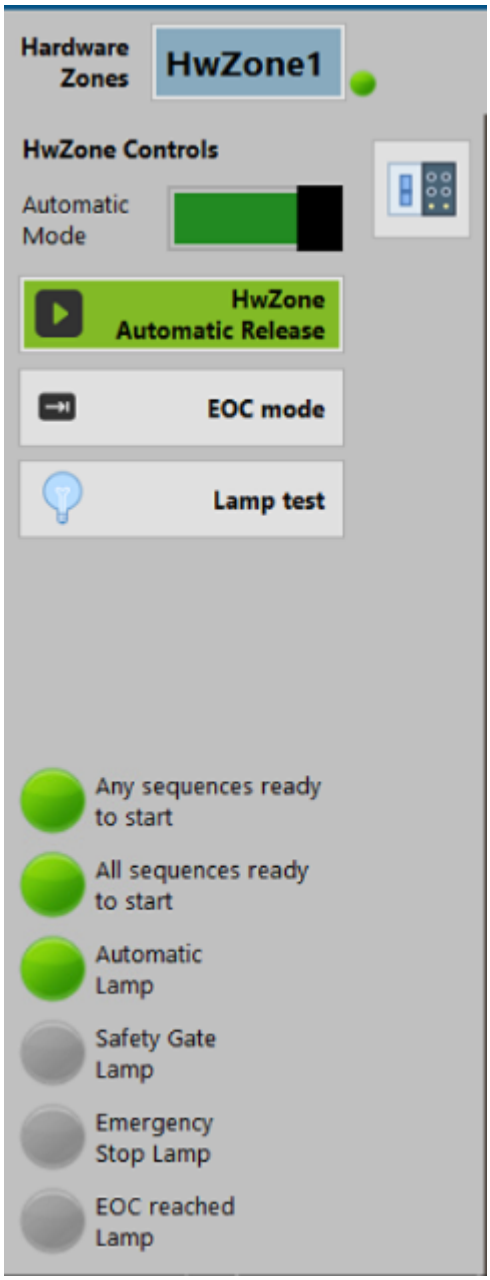
Sequence status lamps

Green: The sequence is running in automatic mode.

Red: The sequence has a fault.

Gray: Manual mode is active .

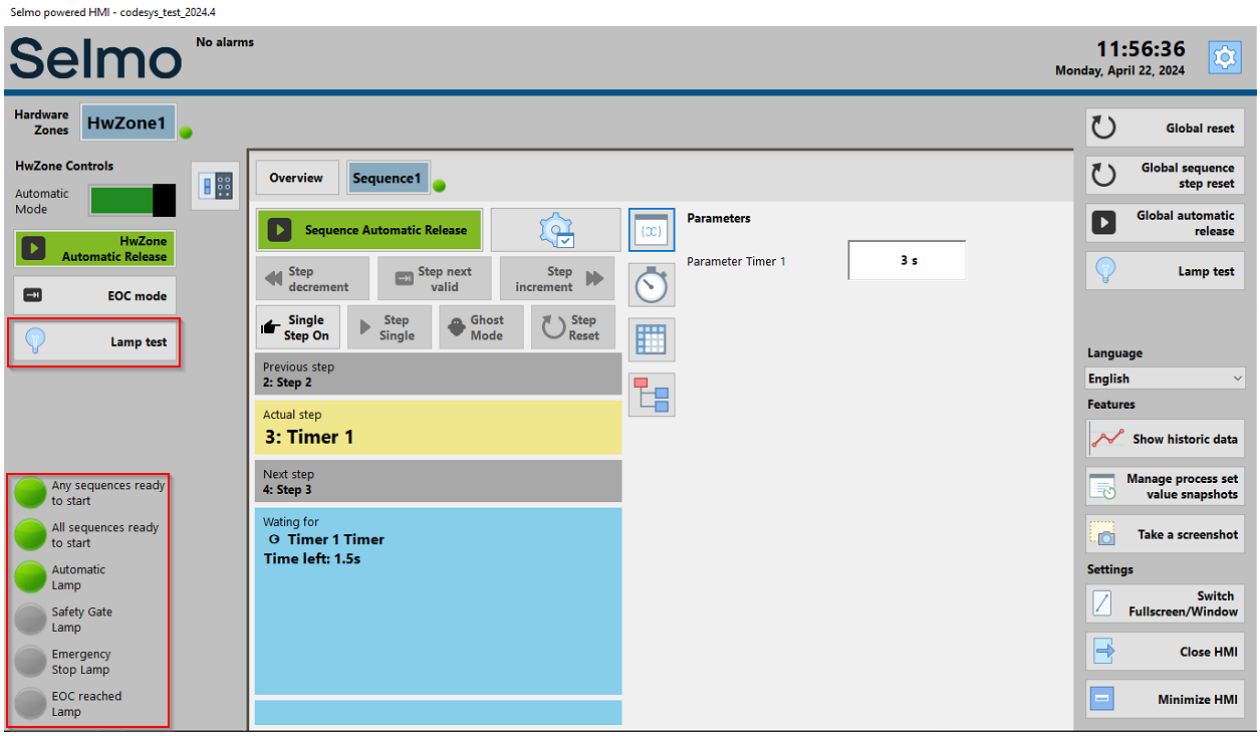
Hardware zone control



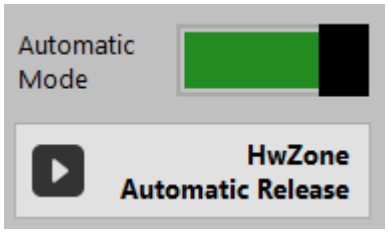
Lamp test

The lamp test is a standard function and the function of the lamps is checked when the lamp test is carried out. After the button has been clicked, all status lamps used in the sequences (Automatic Lamp, Saftey Gate Lamp, Emergency Stop Lamp and EOC reached Lamp) should light up.

The lamp test can be carried out at the Hardware Zone level and the Plant level. At the Plant level, all lamps light up.



Operating mode Selection



Automatic Mode

Automatic mode refers to an operating state in which a system, machine, or device operates automatically without requiring human intervention or control. In Automatic Mode, the system executes preprogrammed sequences, processes, or sequences independently, based on preset parameters, algorithms, or rules. This automates tasks and completes them efficiently without the need for continuous human supervision or intervention.

The green bar indicates that automatic mode has been activated. In this mode, the system operates independently according to preset parameters or rules.

The gray bar indicates that the manual mode has been selected. In this mode, the machine or function can be controlled manually by human intervention.

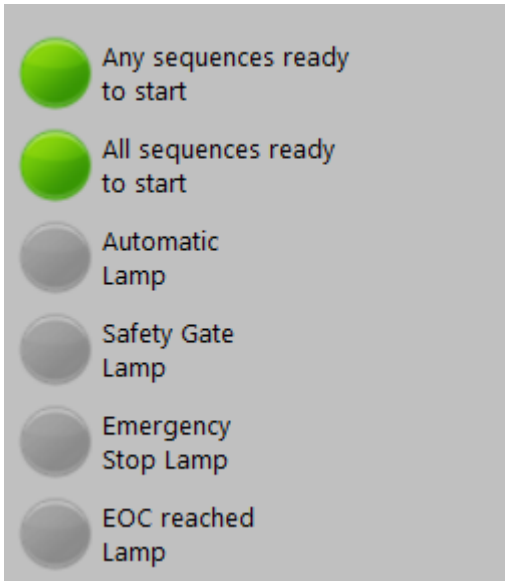
Automatic Release

The "Automatic Release" button enables the release of the "Automatic" mode. When this button is pressed for at least 3 seconds (Adjustable time), the system is set to Automatic mode, which enables it to operate automatically and independently. In this mode, the system executes predefined sequences, sequences or functions without the need for continuous user intervention or instructions.

EOC mode

By activating the "End of Cycle" mode, the sequences are executed until the EOC step and then the automatic operation is stopped.

Hardware zone status lamps



Any sequences ready to start

Indicates whether a sequence in the hardware zone is ready to start automatic operation.

All sequences ready to start

Indicates whether all sequences in the hardware zone are ready to start automatic operation.

Automatic Lamp

Flashing indicates that at least one sequence is ready to start.

Continuous light indicates that the automatic mode is active.

Safety Gate Lamp

The Safety Gate Lamp indicates whether the Safety Gate switch is activated.

Emergency Stop Lamp

The Emergency Stop Lamp signals whether an emergency stop is activated.

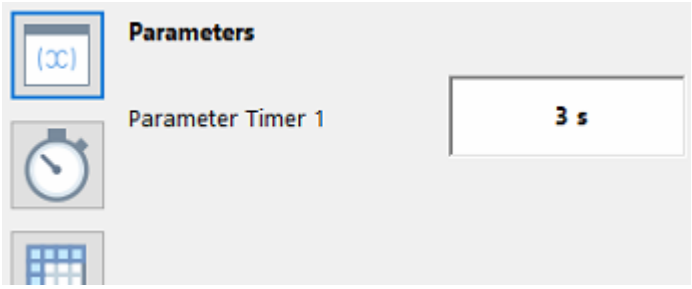
EOC reached Lamp

The EOC-reached lamp flashes when the EOC mode has been activated.

If the EOC-reached lamp is permanently on, it means that all sequences in the hardware zone have reached the EOC step.

Hardware zone parameters

All parameters defined for the hardware zone are displayed here



- H -