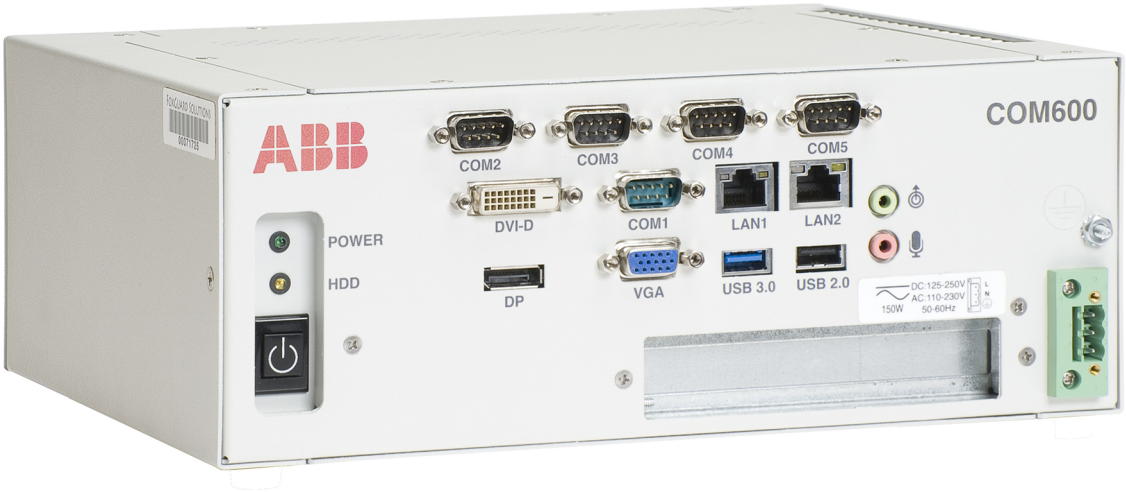
# COM600 series 5.1

## Sequence Control Configuration Manual



**Contents:**

# 1. About this manual

## 1.1. Copyright



### Warranty

Please inquire about the terms of warranty from your nearest ABB representative.

http://www.abb.com/substationautomation

## 1.2. Disclaimer

The data, examples and diagrams in this manual are included solely for the concept or product description and are not to be deemed as a statement of guaranteed properties. All persons responsible for applying the equipment addressed in this manual must satisfy themselves that each intended application is suitable and acceptable, including that any applicable safety or other operational requirements are complied with. In particular, any risks in applications where a system failure and/ or product failure would create a risk for harm to property or persons (including but not limited to personal injuries or death) shall be the sole responsibility of the person or entity applying the equipment, and those so responsible are hereby requested to ensure that all measures are taken to exclude or mitigate such risks.

This product is designed to be connected and to communicate information and data via a network interface, which should be connected to a secure network. It is sole responsibility of person or entity responsible for network administration to ensure a secure connection to the network and to establish and maintain any appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of anti virus programs, etc) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB is not liable for damages and/or losses related to such security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information.

This document has been carefully checked by ABB but deviations cannot be completely ruled out. In case any errors are detected, the reader is kindly requested to notify the manufacturer. Other than under explicit contractual commitments, in no event shall ABB

be responsible or liable for any loss or damage resulting from the use of this manual or the application of the equipment.

## 1.3. Conformity

This product complies with the directive of the Council of the European Communities on the approximation of the laws of the Member States relating to electromagnetic compatibility (EMC Directive 2004/108/EC) and concerning electrical equipment for use within specified voltage limits (Low-voltage directive 2006/95/EC). This conformity is the result of tests conducted by ABB in accordance with the product standards EN 50263 and EN 60255-26 for the EMC directive, and with the product standards EN 60255-1 and EN 60255-27 for the low voltage directive. The product is designed in accordance with the international standards of the IEC 60255 series.

## 1.4. Trademarks

ABB is a registered trademark of ABB Group. All other brand or product names mentioned in this document may be trademarks or registered trademarks of their respective holders.

## 1.5. General information

This user's manual provides thorough information on the sequence control feature for COM600.

Information in this user's manual is intended for application engineers who configure the sequence control views. As a prerequisite, you should have basic knowledge of logic programming and IEC 61131-3 standard.

## 1.6. Document conventions

The following conventions are used for the presentation of material:

- The words in names of screen elements (for example, the title in the title bar of a window, the label for a field of a dialog box) are initially capitalized.
- Capital letters are used for the name of a keyboard key if it is labeled on the keyboard. For example, press the ENTER key.
- Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the space bar, comma key, and so on.
- Press CTRL+C indicates that you must hold down the CTRL key while pressing the C key (to copy a selected object in this case).
- Press ESC E C indicates that you press and release each key in sequence (to copy a selected object in this case).
- The names of push and toggle buttons are boldfaced. For example, click **OK**.

- The names of menus and menu items are boldfaced. For example, the **File** menu.
  - The following convention is used for menu operations: **MenuName** > **MenuItem** > **CascadedMenuItem**. For example: select **File** > **New** > **Type**.
  - The **Start** menu name always refers to the **Start** menu on the Windows taskbar.
- System prompts/messages and user responses/input are shown in the Courier font. For example, if you enter a value out of range, the following message is displayed:

```
Entered value is not valid. The value must be 0 - 30  .
```

- You can be asked to enter the string MIF349 in a field. The string is shown as follows in the procedure:

  MIF349
- Variables are shown using lowercase letters:

  sequence name

## 1.7. Use of symbols

This publication includes warning, caution, and information icons that point out safety-related conditions or other important information. It also includes tip icons to point out useful information to the reader. The corresponding icons should be interpreted as follows.

The electrical warning icon indicates the presence of a hazard which could result in electrical shock.

The warning icon indicates the presence of a hazard which could result in personal injury.

The caution icon indicates important information or warning related to the concept discussed in the text. It may indicate the presence of a hazard which could result in corruption of software or damage to equipment or property.

The information icon alerts the reader to relevant facts and conditions.

The tip icon indicates advice on, for example, how to design your project or how to use a certain function.

## 1.8. Terminology

| Term | Description |
|---|---|
| Alarm | An abnormal state of a condition. |
| Alarms and Events; AE | An OPC service for providing information about alarms and events to OPC clients. |
| COM600 Series; COM600 | COM600 as a generic name for COM600S IEC and COM600F ANSI products |
| Data Access; DA | An OPC service for providing information about process data to OPC clients. |
| Data Object; DO | Part of a logical node object representing specific information, for example, status, or measurement. From an object-oriented point of view, a data object is an instance of a class data object. DOs are normally used as transaction objects; that is, they are data structures. |
| Data Set | The data set is the content basis for reporting and logging. The data set contains references to the data and data attribute values. |
| Device | A physical device that behaves as its own communication node in the network, for example, protection relay. |
| Event | Change of process data or an OPC internal value. Normally, an event consists of value, quality, and timestamp. |
| Intelligent Electronic Device | A physical IEC 61850 device that behaves as its own communication node in the IEC 61850 protocol. |
| Logical Device; LD | Representation of a group of functions. Each function is defined as a logical node. A physical device consists of one or several LDs. |
| Logical Node; LN | The smallest part of a function that exchanges data. An LN is an object defined by its data and methods. |
| OPC | Series of standards specifications aiming at open connectivity in industrial automation and the enterprise systems that support industry. |
| OPC item | Representation of a connection to the data source within the OPC server. An OPC item is identified by a string <object path>:<property name>. Associated with each OPC item are Value, Quality, and Time Stamp. |
| Property | Named data item. |
| Report Control Block | The report control block controls the reporting processes for event data as they occur. The reporting process continues as long as the communication is available. |

## 1.9. Abbreviations

| Abbreviation | Description |
|---|---|
| AE | Alarms and Events |
| DA | Data Access |
| DO | Data Object |
| GW | Gateway, component connecting two communication networks together |
| WebHMI | Web Human Machine Interface |
| IEC | International Electrotechnical Commission |
| IED | Intelligent Electronic Device |
| INS | Integer Status |
| LAN | Local Area Network |
| LD | Logical Device |
| LN | Logical Node |
| MSB | Multiple State Button |
| NCC | Network Control Center |
| OLE | Object Linking and Embedding |
| OPC | OLE for Process Control |
| P&C | Protection & Control |
| PLC | Programmable Logic Controller |
| POU | Program Organization Unit |
| RTS | Request To Send |
| SA | Substation Automation |
| SCD | Substation Configuration Description |
| SCL | Substation Configuration Language |
| SFC | Sequential Function Chart |
| SLD | Single Line Diagram |
| SPC | Single Point Control |
| XML | eXtended Markup Language |

## 1.10. Related documents

| Name of the manual | MRS number |
|---|---|
| COM600 Logic Processor User's Manual | 1MRS756738 |
| COM600 User's Manual | 1MRS756125 |

## 1.11. Document revisions

| Document version/date | Product revision | History |
|---|---|---|
| A/31.5.2012 | 4.0 | Document created |
| B/13.3.2015 | 4.1 | Document revised |
| C/24.5.2017 | 5.0 | Document revised |
| D/22.3.2018 | 5.1 | Document revised |

# 2. Introduction

## 2.1. General information about the COM600 series

The COM600 product series are versatile Substation Management Units that help realize smart substation and grid automation solutions in industrial and utility distribution networks.

They get deployed together with protection and control IEDs, substation devices such as RTUs, meters and PLCs in dedicated cabinets and switchgear.

The COM600 product is an all-in-one unit that functions as:
- Communication gateway
- Web Human Machine Interface (WebHMI)
- Automation controller
- Real-time and historical data management unit

The COM600 product series use process information and device data, acquired over Ethernet or serial communication protocol interfaces to execute specific substation functions and applications. Thus, they are critical building blocks to realize substation secondary system solutions and in the process solving diverse customer needs.

## 2.2. COM600 product series variants and rationale

To facilitate substation and grid automation solutions in IEC and ANSI market areas, a variant-based system similar to Relion® 615 and 620 series is being followed from COM600 5.0 release.

The main reasons for such an approach are the following:

- To ensure all COM600 product series features are advantageously used in end-customer projects in the medium voltage substation automation domain.
- To ensure an optimum feature set to be bundled together to realize specific applications required in IEC and ANSI market areas.
- To ensure a future-proof product approach.

This release then comprises of two variants, based on the primary intent or application are defined as follows:
- COM600S IEC – COM600 for substation automation, analysis and data management (for IEC markets)
  - COM600S IEC is a substation automation, analyzer and data management unit that integrates devices, facilitates operations, manages communication and runs analysis applications pertinent to equipment or operations in utility or industrial distribution substations.
- COM600F ANSI – COM600 as distribution automation controller (for ANSI markets)

- COM600F is a dedicated distribution automation controller unit that runs distributed grid and feeder applications for ANSI power networks and inherits all core features of the COM600 series.

## 2.3. Overview of sequence control

The sequence control feature allows you to create sequences in the Logic Processor environment using the sequence control library and control breakers through them. The created sequences can be controlled using COM600 WebHMI.

# 3. Sequence Control configuration

## 3.1. Overview of configuration

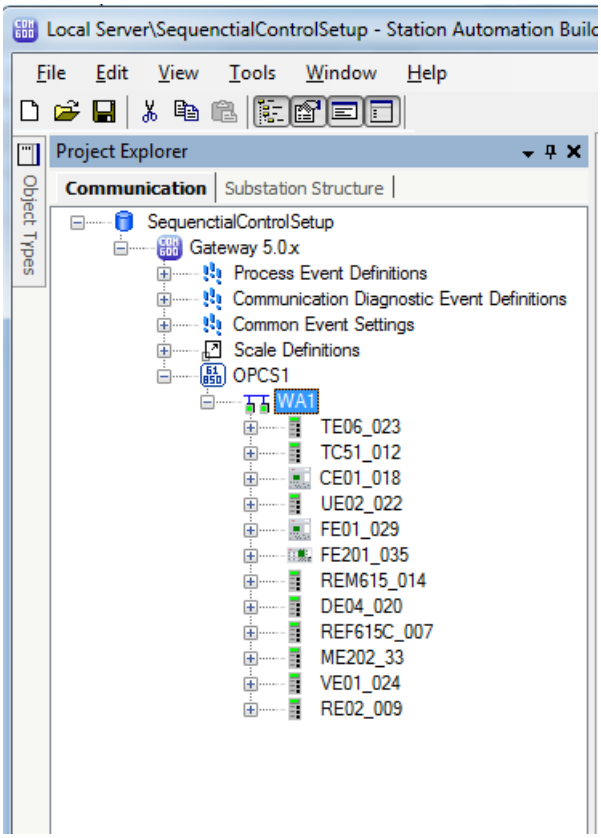
This section describes the steps involved in configuring a sequence. The same procedure can be repeated to configure multiple sequences.

Before configuring a sequence, the communication information to multiple IEDs must be configured. For more information on configuring the communication structure for COM600, see COM600 User's Manual.

Sequence control views allow you to run configured sequences on COM600. Configuring sequence control involves the following steps:

- defining a sequence logical device and associated logical nodes for the logic processor IED
- defining a sequence (SFC program) in the logic processor
- cross-referencing sequence data objects and breaker data objects to corresponding sequence object and switch object members in the logic processor
- defining a sequence SLD for using sequence control through COM600 WebHMI configuration.

Figure 3.1-1 shows an example of a communication structure in the SAB600 tool including four IEDs that communicate with COM600 using IEC 61850 protocol.



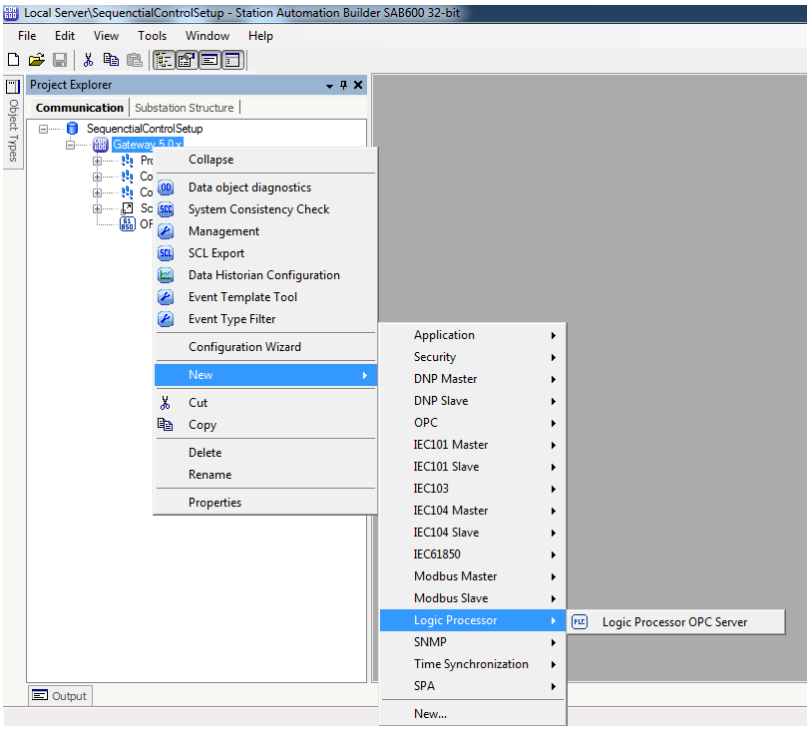

SAB600_SC_Communication_View.png

*Figure 3.1-1 An example of a communication structure in SAB600 with IEDs*

## 3.2. Adding a Logic Processor IED

To add a Logic Processor IED to the communication structure:

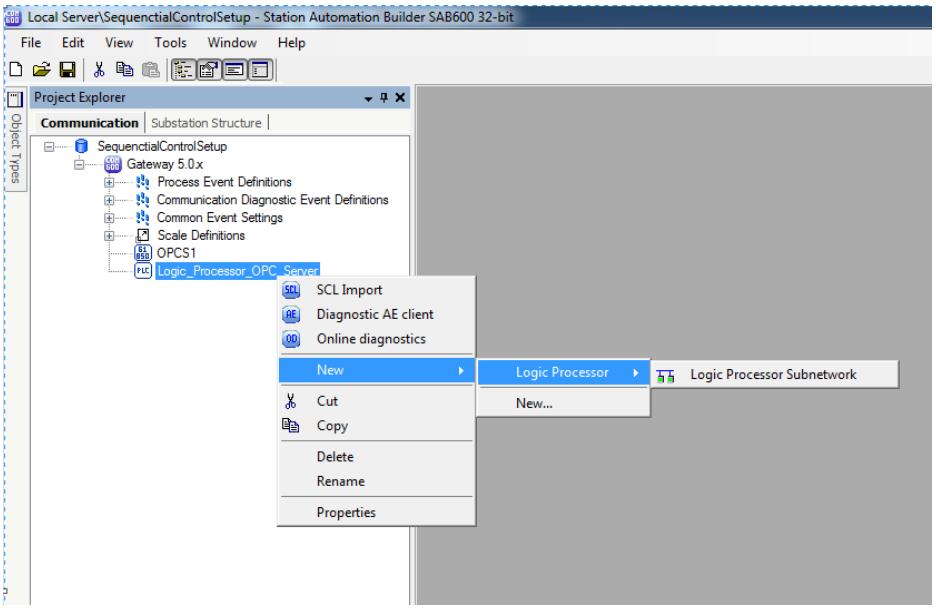
1. Right-click the Gateway object and select **New > Logic Processor OPC Server**.




SAB600_SC_PLC_Server.png

*Figure 3.2-1 Adding Logic Processor OPC Server object*

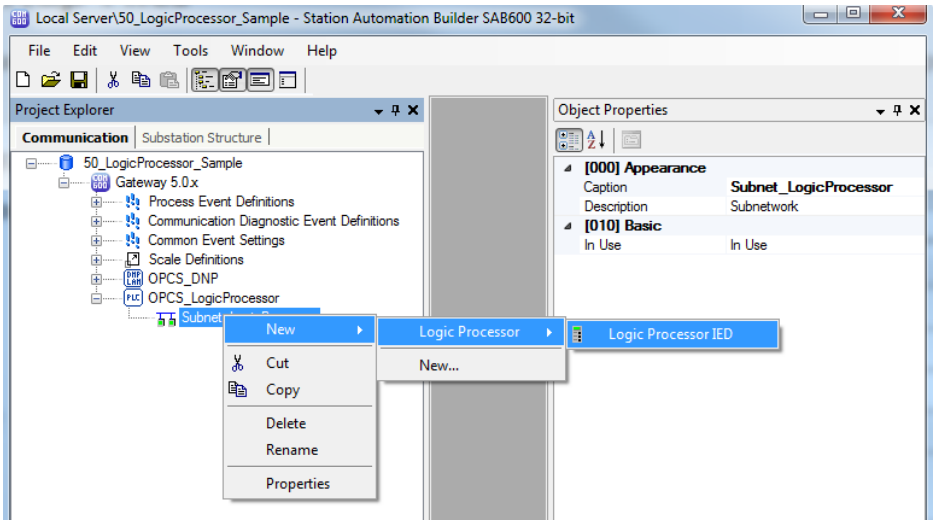
2. Right-click the Logic Processor OPC Server object and add **Logic Processor Subnetwork**.




SAB600_SC_PLC_Subnetwork.png

*Figure 3.2-2 Adding Logic Processor Subnetwork object*

3. Right-click the Logic Processor Subnetwork object and add **Logic Processor IED**.




SAB600_SC_PLC_IED.png

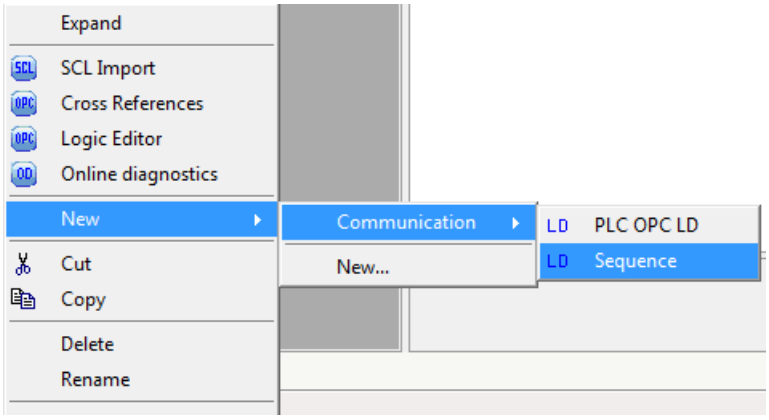
*Figure 3.2-3 Adding Logic Processor IED object*

## 3.3. Adding Sequence Logical Device

A corresponding Sequence Logical Device must be defined for each sequence in the Logic Processor IED.

To add a Sequence Logical Device:

1. Right-click the Logic Processor IED and select **New > Communication > Sequence** (see Figure 3.3-1).




SAB600_SC_PLC_LD.png

*Figure 3.3-1 Adding a Sequence Logical Device object to the Logic Processor IED*

2. Rename the Sequence Logical Device with a suitable sequence name by editing the caption parameter in the object properties window.

## 3.4. Adding logical nodes for a sequence

At a minimum, each added sequence logical device should have the predefined SEQGGIO1 and STEPGGIO1 logical nodes with data objects. The SEQGGIO logical node has data objects related to the entire sequence functionality, whereas the STEPGGIO logical node has data objects for a step in the sequence. The predefined logical nodes and data attributes for a sequence logical device are described in Table 3.4-1.

The description parameter of the data objects in the logical devices is used in event reporting for sequence execution. To be able to identify the step number involved, edit the description parameter of the “St” data object for a STEPGGIO logical node.

Additional STEPGGIO logical nodes should be added for a sequence with multiple steps.

***Table 3.4-1 Predefined logical nodes and data objects for a sequence logical device***

| Logical node name | Data object name | Data object type | Data object description |
|---|---|---|---|
| SEQGGIO1 | St | INS | State |
| | Str | SPC | Control |
| | Auto | SPC | Execution mode |
| | StepExec | SPC | Manual mode acknowledgement |
| | OnStepErr | SPC | On step error |
| STEPGGIO1 | St | INS | Step (number) state |

To add additional logical node objects:

1. Right-click the sequence logical device and select **New** > **Communication** > **Sequence Step LN**.

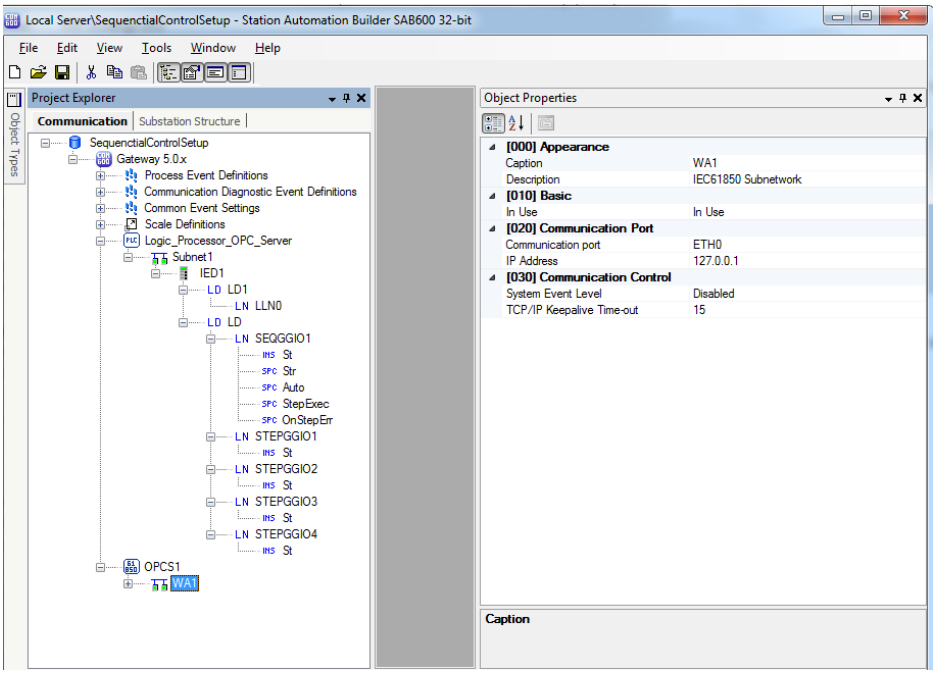

SAB600_SC_PLC_LN.png

*Figure 3.4-1 Adding a logical node for a sequence logical device*

2. Edit the description parameter for the "St" data object in the added STEPGGIO1 logical node for additional clarity.

3. Add additional STEPGGIO logical nodes depending on the number of steps intended for the sequence being configured.
4. Repeat the steps 1 - 4 to add all the needed logical nodes (SEQGGIO1/ STEPGGIO*) and associated data objects as defined in Table 3.4-1.

An example of a finished configuration for a four step sequence after all the logical nodes and associated data objects have been defined is shown in Figure 3.4-2




SAB600_SC_Four_Step_Sequence_Example.png

*Figure 3.4-2 Communication structure view for a four step sequence*

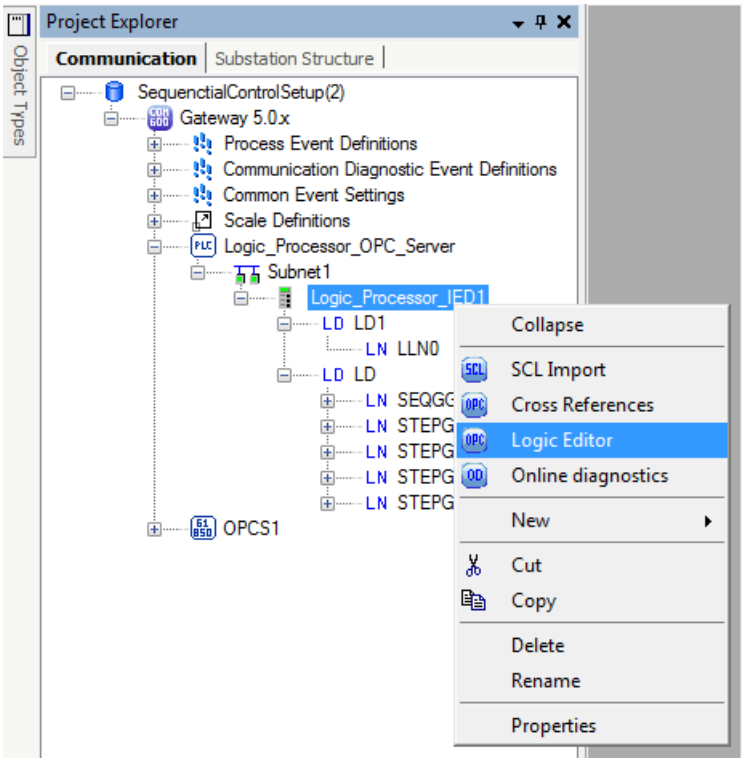
# 4. Sequence definition

## 4.1. Sequence definition using Logic Editor

This section describes the steps involved in defining a sequence in Logic Processor.

The sequence logic runs in the Logic Processor (CoDeSys) programming environment and it is implemented using IEC 61131- Structured Text programming language. The sequence should be implemented as a standard PLC Sequence Flow Chat (SFC) program within the logic processor environment.

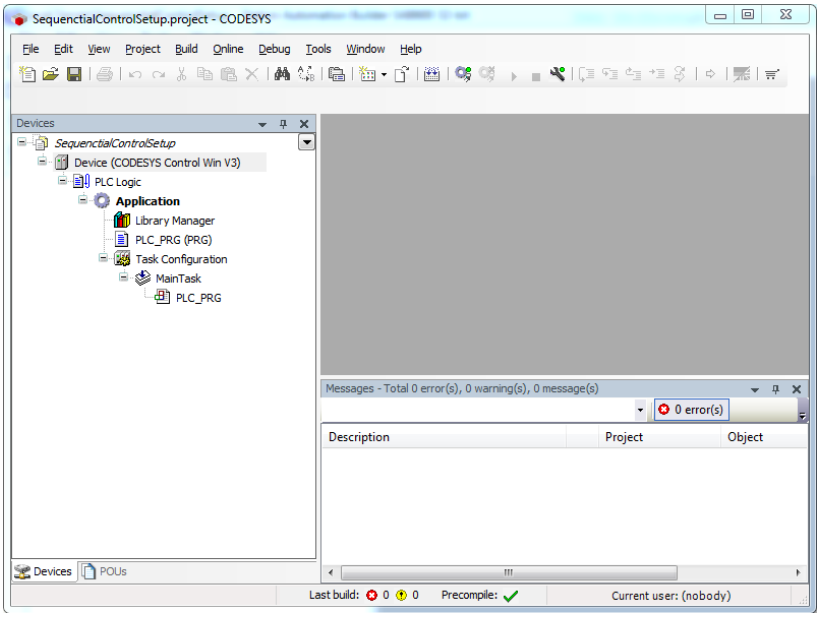
To launch the Logic Editor, right-click the Logic Processor IED and select **Logic Editor**.




SAB600_SC_Logic_Editor.png

*Figure 4.1-1 Launching Logic Editor*

The Logic Editor opens with a default application that has a predefined PLC_PRG, POU (Program Organization Unit).




Logic_Editor_View.png

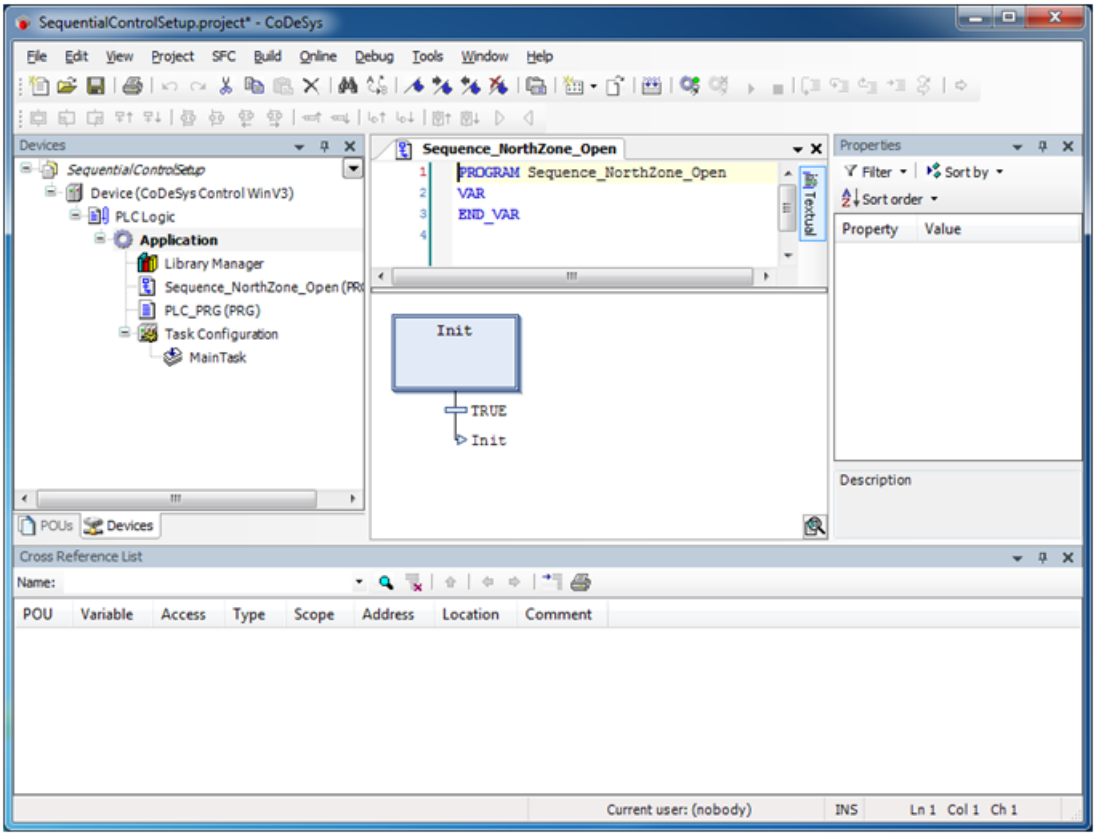
*Figure 4.1-2 Default Logic Editor view*

## 4.2. Adding a new Sequence POU object

To add a new sequence POU object:

1. Right-click the Application object and select **POU**.
2. In the Add POU dialog, define a name for the intended sequence and set **Sequential Function Chart (SFC)** as the implementation language from the drop-down menu.

After adding the POU object, the Logic Editor dialog shows a default Init step as shown in Figure 4.2-1.

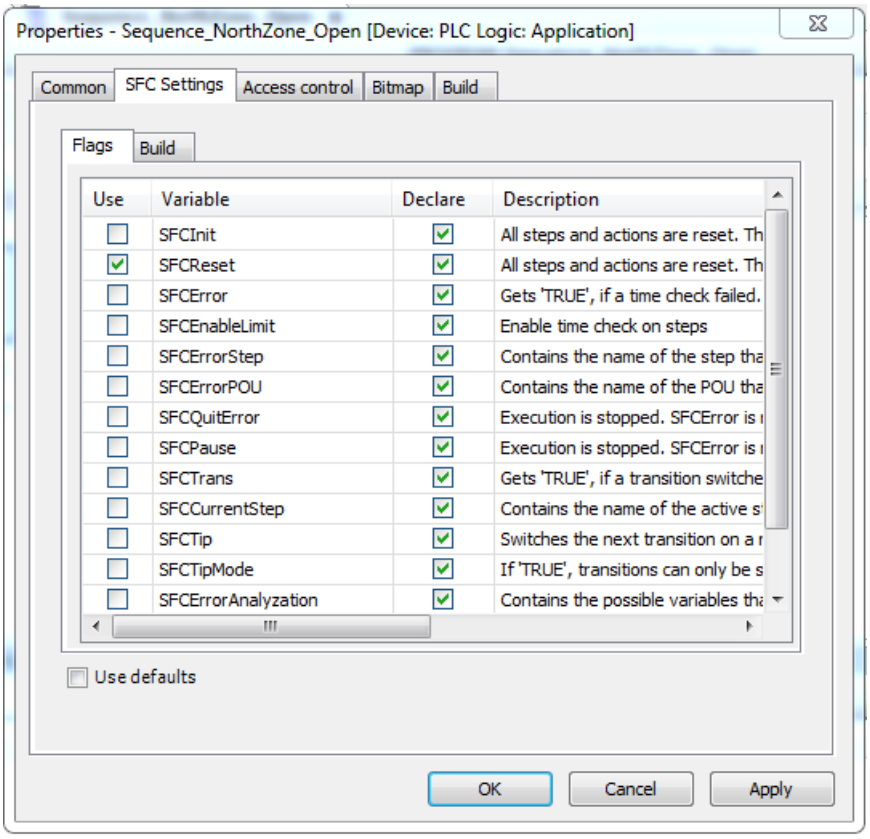



Logic_Editor_with_POU.png

*Figure 4.2-1 Logic Editor view after adding a Sequence POU*

To modify a sequence POU:

1. Right-click a sequence POU and select **Properties**.
2. In the Properties dialog, go to the SFC Settings tab and select the SFCReset variable.




Logic_Editor_SFC_Settings.png

*Figure 4.2-2 Modifying SFC default flag settings for a sequence POU*

3. Click **OK** or **Apply** to save the changes.

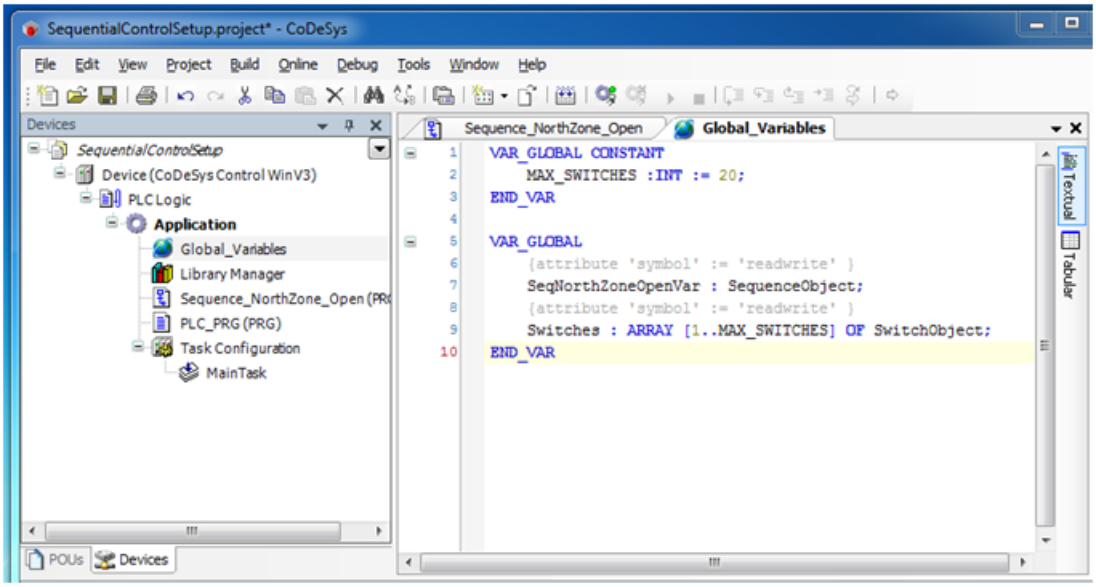
## 4.3. Creating a global variable list

Create a global variable list for the Application object in the CoDeSys project, if not already available.

To create a global variable list:

1. Right-click the Application object and select **Add Object > Global Variables List**.

2. Name the new global variables list.
3. Select the added global variables list from the Devices view. In the corresponding program editor, create a new global sequence object, see Figure 4.3-1.




global_variables.png

*Figure 4.3-1 Creating a global sequence object*

The variables defined in the global variable list are:

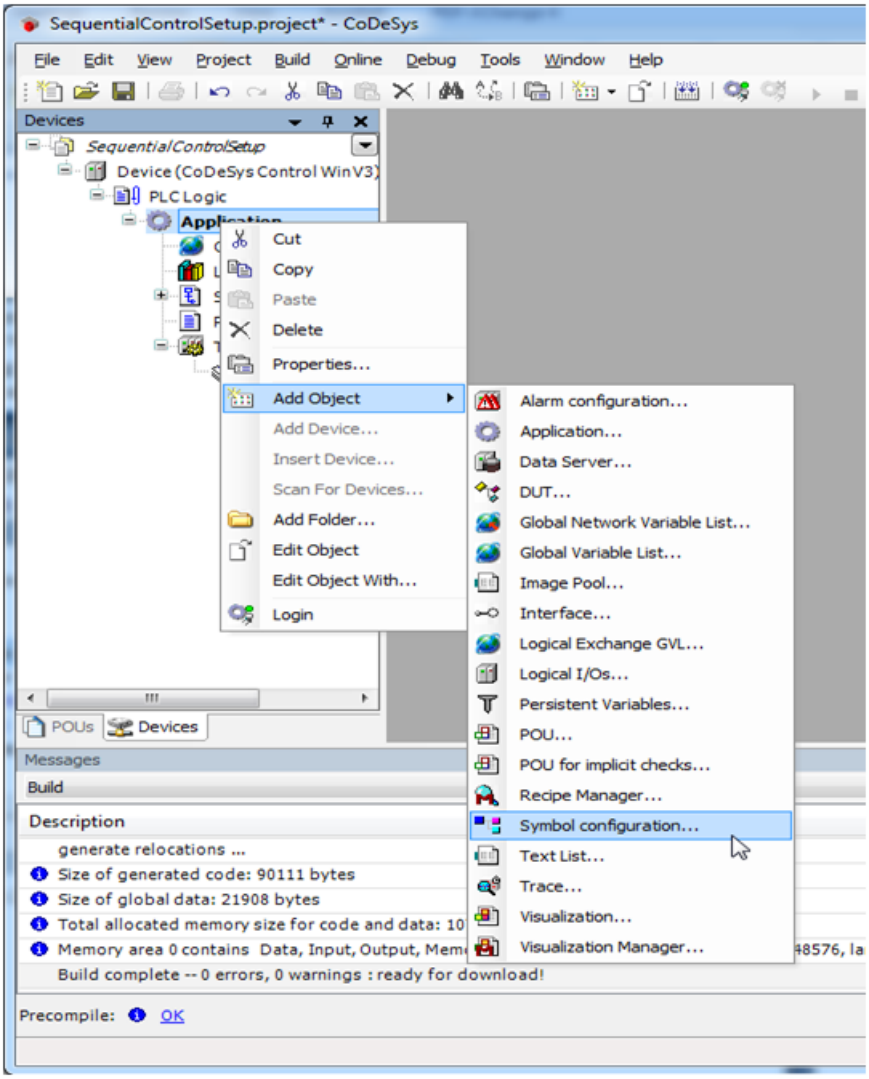
- **MAX_SWITCHES**: A constant variable indicating the number of switches available for a sequence. This constant indicates the number of switches available to the logic processor for sequence control purposes. The number is limited to 20 switches per sequence.
- **SequenceObject**: A variable corresponding to a sequence. Each sequence intended within the logic processor should have a corresponding sequence object defined.
- **Array of SwitchObjects**: Variables allowing the logic processor runtime to control the switches available in COM600 runtime. Multiple sequences defined within the logic processor use the same array of switch objects available for execution. Each step in a sequence uses the index in the array of switches to control a particular switch.

## 4.4. Adding symbol configuration

Generate a symbol list to select which variables from logic programming are cross-referenced with data in the COM600 communication structure.

To add a symbol configuration:

1. Right-click **Application** in the Devices tree.
2. Select **Add Object > Symbol configuration**.



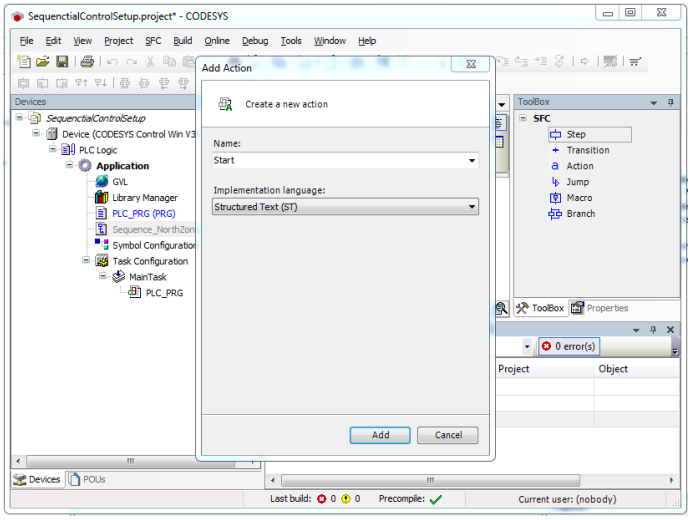

symbol_configuration.png

*Figure 4.4-1 Adding a symbol configuration object to the project*

## 4.5. Adding a new sequence start and end action

To add a new sequence start action:

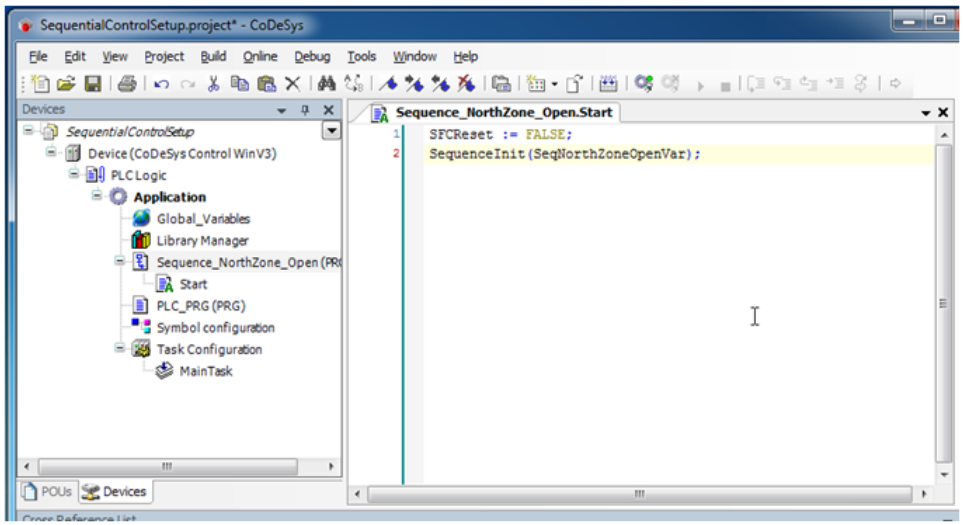
1. Right-click the added POU object in the Devices view and select **Add Object** > **Action**.
2. Name the action as **Start**. Set **Structured Text (ST)** as the implementation language from the drop-down menu, see Figure 4.5-1.




Logic_Editor_Add_Start_Action.png

*Figure 4.5-1 Adding sequence start action to a sequence PRG*

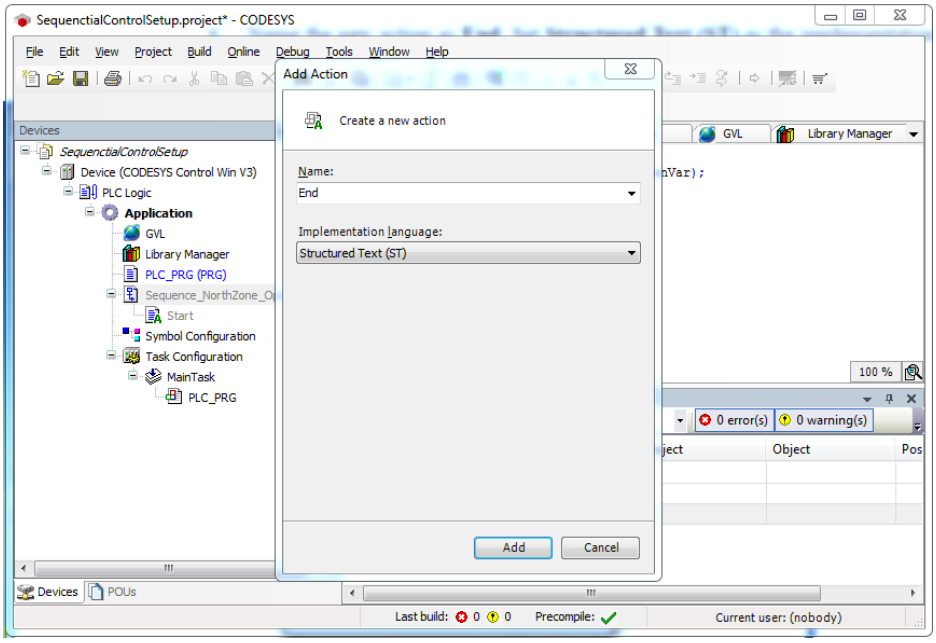
3. Click **Add**.
4. Select the added Start action from the Devices view. In the corresponding program editor, define the sequence start action, see Figure 4.5-2.




sequence_start.png

*Figure 4.5-2 Sequence start action definition*

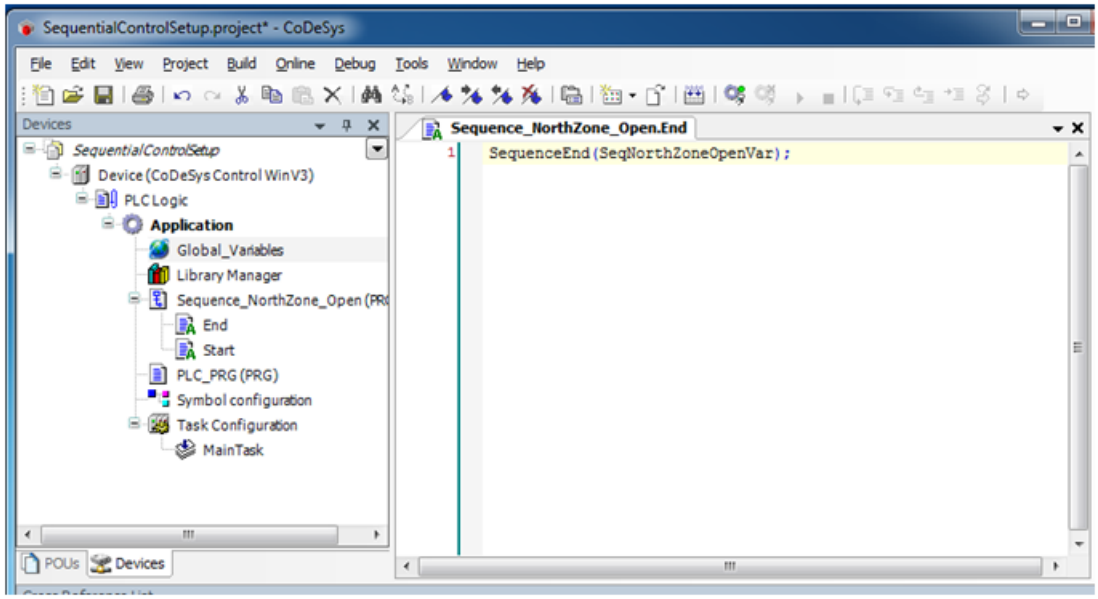
5. Right-click the added POU object in the Devices view and add a new action by selecting **Add Object > Action**.
6. Name the new action as **End**. Set **Structured Text (ST)** as the implementation language, see Figure 4.5-3.




Logic_Editor_Add_End_Action.png

*Figure 4.5-3 Adding sequence end action to sequence PRG*

7. Select the added End action from the Devices view. In the corresponding program editor, define the sequence end action, see Figure 4.5-4.



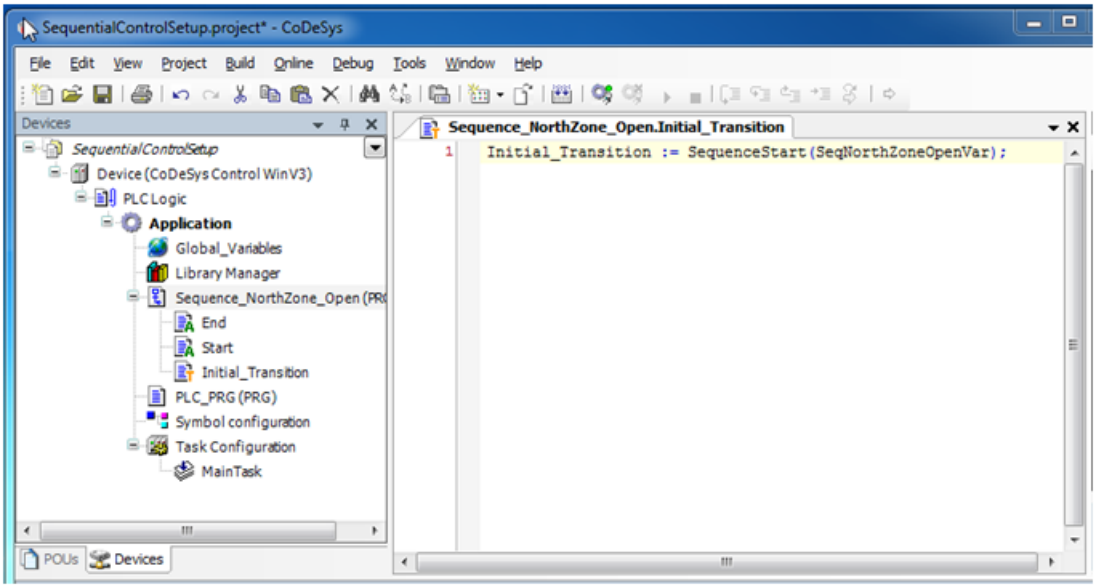

sequence_end.png

*Figure 4.5-4 Sequence end action definition*

## 4.6. Adding an initial sequence transition

To add a new transition object:

1. Right-click the added POU object in the Devices view and add a new Transition object by selecting **Add Object > Transition**.
2. Name the new transition object as Initial_Transition and set **Structured Text (ST)** as the implementation language from the drop-down menu.
3. Select the added Initial_Transition object from the Devices view. In the corresponding program editor, add the sequence Initial_Transition definition, see Figure 4.6-1.



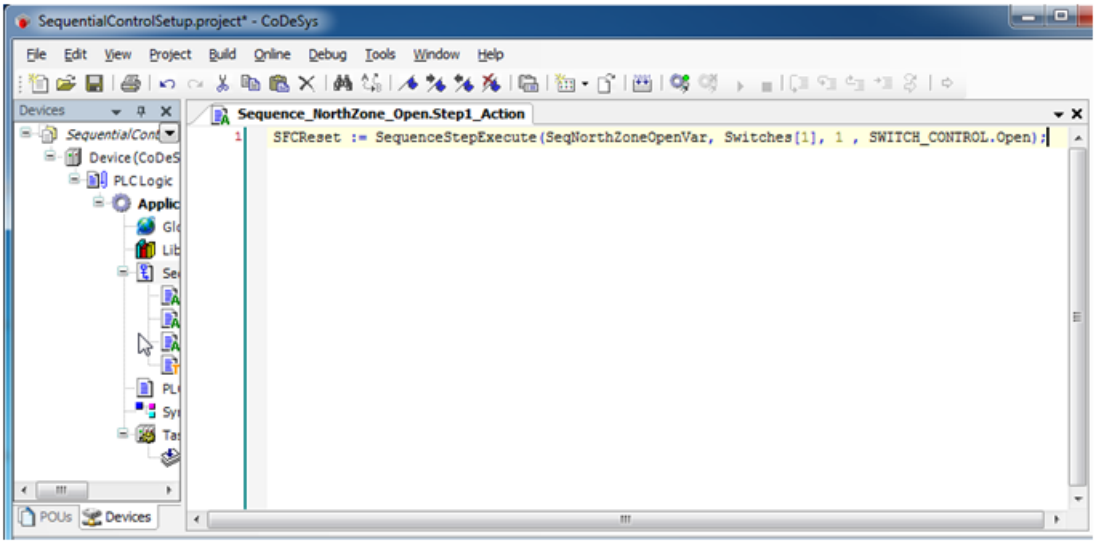

adding_initial_transition.png

*Figure 4.6-1 Defining initial transition for a sequence*

## 4.7. Adding a new action object

To add a new action object:

1. Right-click the added POU object in the Devices view and add a new action by selecting **Add Object > Action**.
2. Name the new transition object as Step1_Action and set **Structured Text (ST)** as the implementation language from the drop-down menu.
3. Select the added Step1_Action object from the Devices view. In the corresponding program editor, add the sequence Step1_Action object definition, see Figure 4.7-1.




step1_action.png

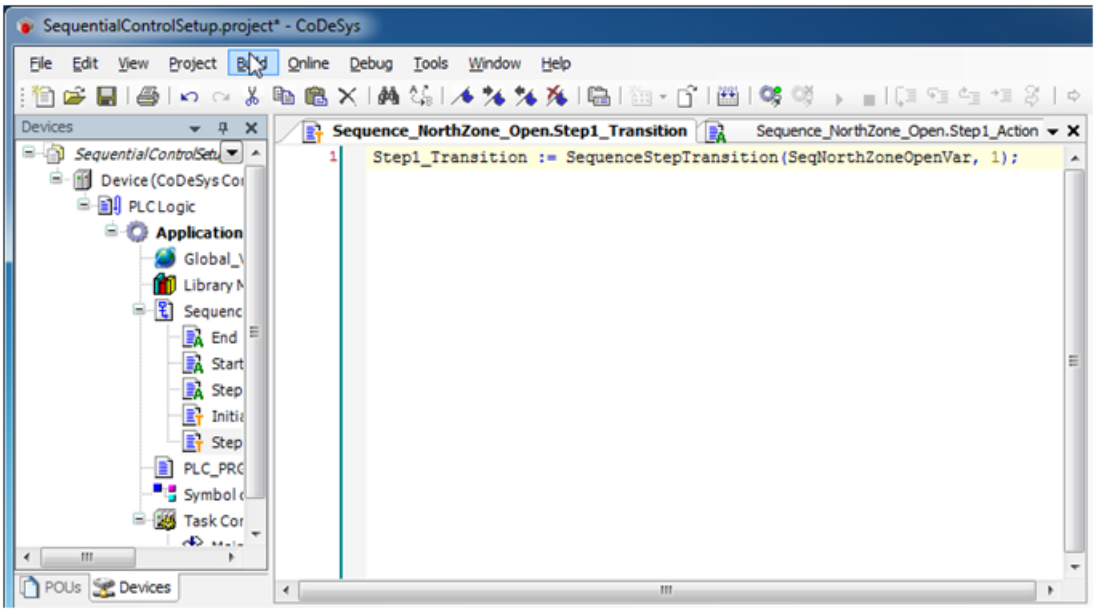
*Figure 4.7-1 Defining step 1 for a sequence*

## 4.8. Adding a new transition object

To add a new transition object:

1. Right-click the added POU object in the Devices view and add a new transition by selecting **Add Object > Transition**.

2. Name the new transition object as Step1_Transition and set **Structured Text (ST)** as the implementation language from the drop-down menu
3. Select the added Step1_Transition object from the Devices view. In the corresponding program editor, add the sequence Step1_Transition object definition, see Figure 4.8-1.




step1_transition.png

*Figure 4.8-1 Defining step 1 transition for a sequence*

Depending on the number of steps intended for a sequence, add additional actions and transition objects for the sequence POU as described in 4.7, Adding a new action object and in this section, with the following exceptions:

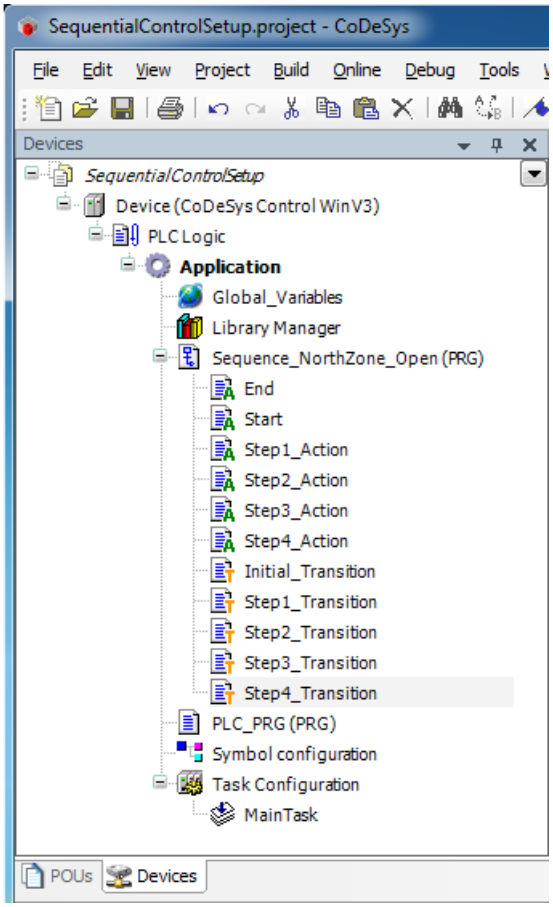
- When adding an action object, make sure that the program statement in the program editor corresponds to the action step number. For example, for a Step2_Action object, the program statement in the program editor should be:

  ```
  SFCReset := SequenceStepExecute(SeqNorthZoneOpenVar, Switches[2], 2,
  SWITCH_CONTROL.Open);
  ```

- Similarly, when adding a transition object, make sure that the program statement in the program editor corresponds to the transition step number. For example, for a Step2_Transition object, the program statement in the program editor should be:

  ```
  Step2_Transition := SequenceStepTransition(SeqNorthZoneVar, 2);
  ```

An example of the Device view for a four step sequence in the logic editor is shown in Figure 4.8-2.




four_step_sequence.png

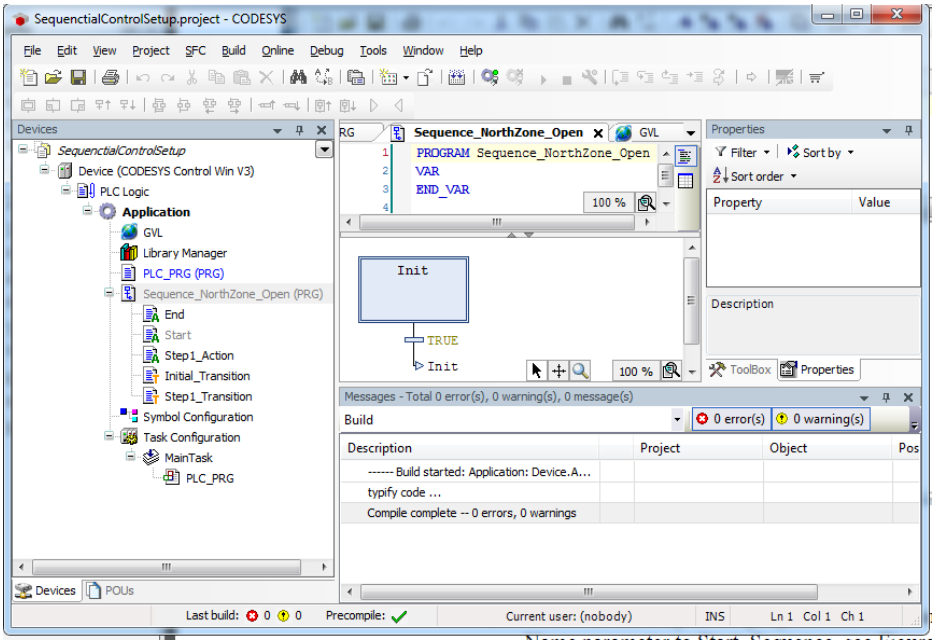
*Figure 4.8-2 Device view for a four step sequence showing action and transition objects*

# 4.9. Assigning actions to the added sequence steps

## 4.9.1. Editing start sequence

To edit the start sequence:

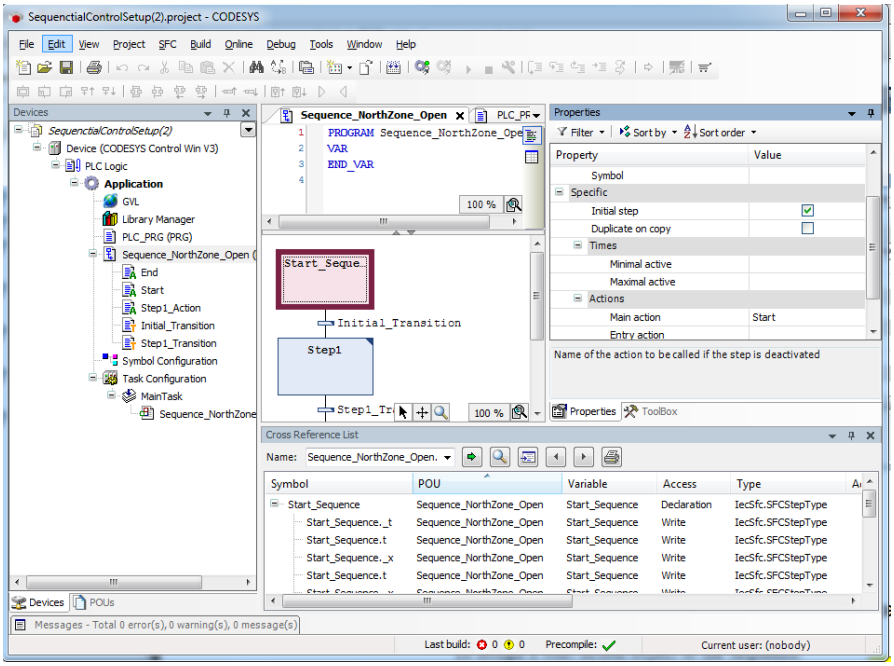
1. Double-click the added POU in the Device view to launch the program editor for the PLC Program. The default view for the Sequence POU in the program editor is shown in Figure 4.9.1-1.




Logic_Editor_Sequence_POU.png

*Figure 4.9.1-1 The default view for Sequence POU in the program editor*

2. Select the init step in the program editor and in the properties view. Rename the Name parameter to Start_Sequence, see Figure 4.9.1-2



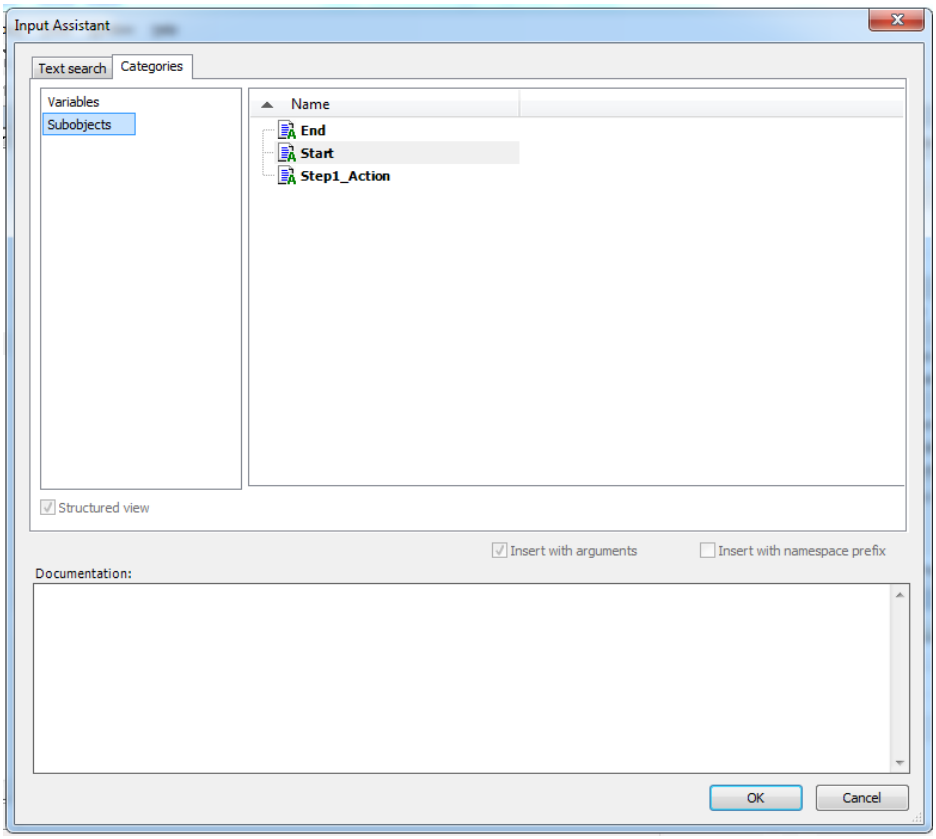

Logic_Editor_Add_Start_Sequence.png

*Figure 4.9.1-2 Adding Start_Sequence step in Sequence_NorthZone POU SFC*

## 4.9.2. Assigning a start action object to a sequence

To assign a start action object to the sequence:

1. Double-click the Value column for the Main active parameter available in the properties view.
2. Click the "…" button to launch the Input Assistant dialog.
3. In the Input Assistant dialog, click **Subobjects** in the Categories list.
4. Select the Start action object and click **OK**. This assigns the Start action object to the Start_Sequence step.



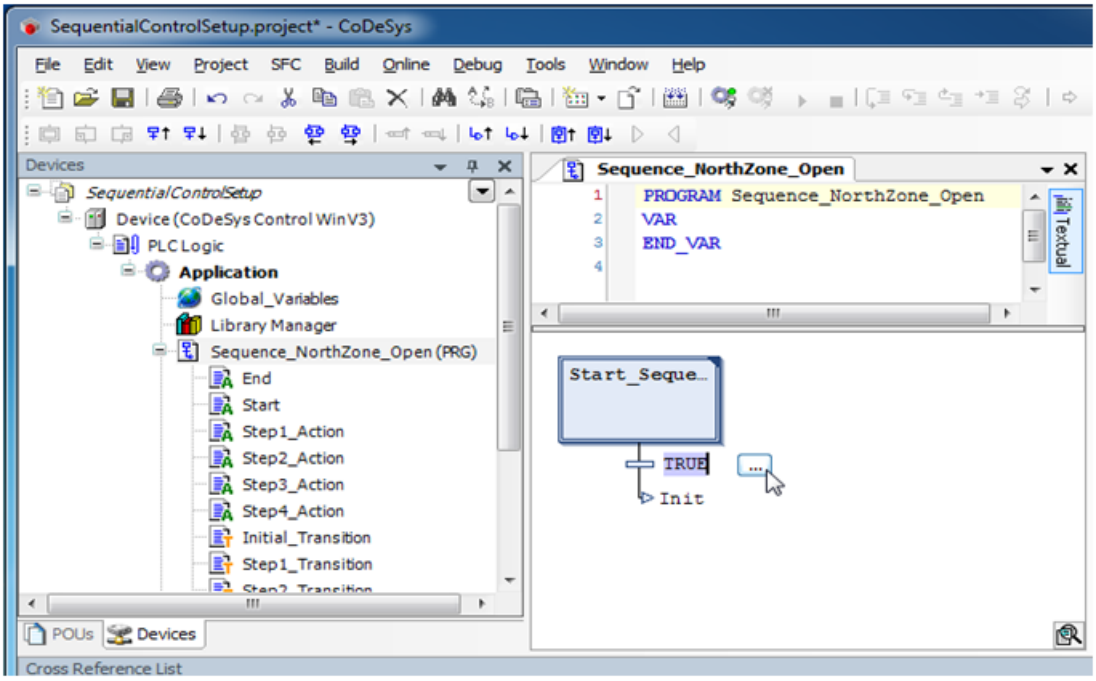

Logic_Editor_Input_Assistant.png

*Figure 4.9.2-1 Input assistant dialog for adding an action object*

## 4.9.3. Assigning an initial step action to a sequence

To assign initial step transition object for the POU object:

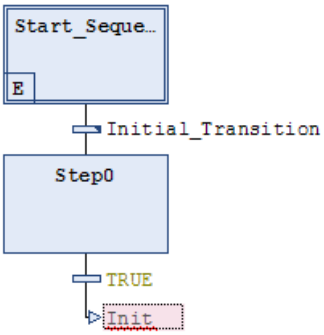
1. Launch the Input Assistant dialog by clicking the "…" button next to the start sequence object, see Figure 4.9.3-1.




launching_input_assistant.png

*Figure 4.9.3-1 Launching the input assistant dialog to select Initial_Transition object*

2. Assign the Initial_Transition object as the transition condition for the Start_Sequence step.
3. Insert a new step transition for the POU object by right-clicking it and selecting **Insert step-transition**. The added step transition is shown in the program editor view, see Figure 4.9.3-2




Logic_Editor_Step_Transition.png

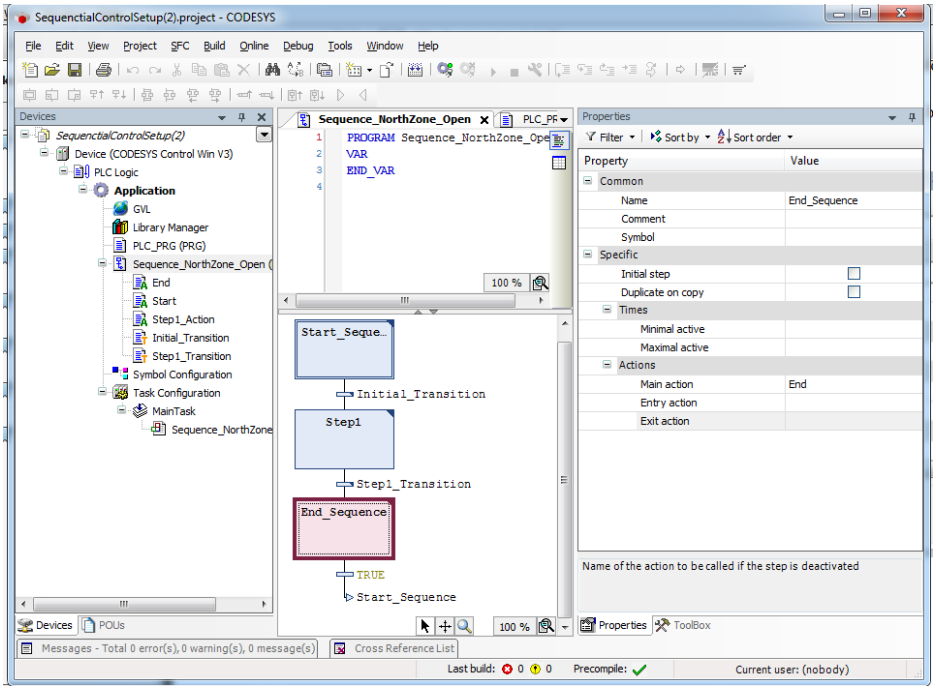
*Figure 4.9.3-2 Step-transition in the program editor view*

4. Rename the added Trans0 object to **TRUE** by double-clicking the object and editing the name.

## 4.9.4. Assigning an end action to a sequence

To assign an end action object to the sequence:

1. Select **Step0** and change the Name parameter to End_Sequence in the properties view, see Figure 4.9.4-1.
2. Assign the End action object to the End_Sequence step by selecting it and editing the Main entry parameter in the properties view.



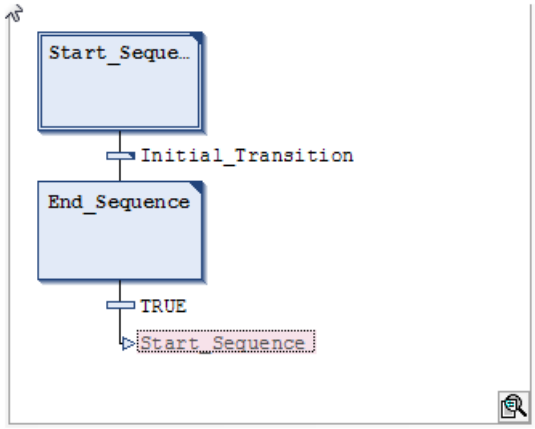

Logic_Editor_End_Sequence.png

*Figure 4.9.4-1 Adding an End_Sequence step to Sequence SFC*

## 4.9.5. Assigning a step transition to a sequence

To assign step transition to the sequence:

1. Assign the Init jump to the Start_Sequence step in SFC. Figure 4.9.5-1 shows the program editor after the Start_Sequence and End_Sequence steps with the Initial_Transition objects have been set.

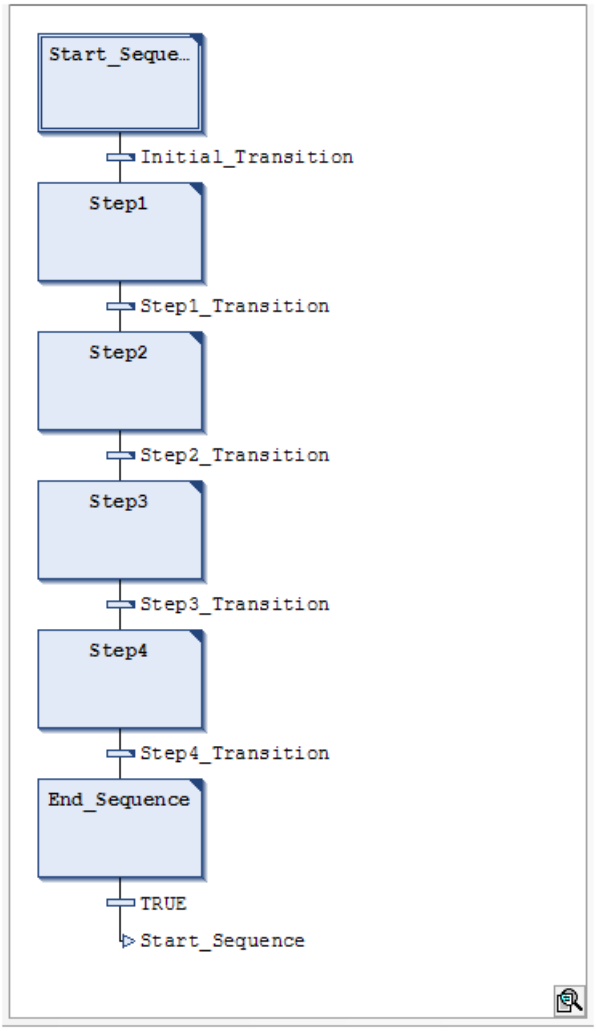



sequence_SFC.png

*Figure 4.9.5-1 Sequence SFC with start and end sequence steps*

2. Right-click the **Initial_Transition** object and select **Insert step-transition after** to add step 1 in Sequence SFC.
3. Rename the added step to **Step 1** by editing the Name parameter in the properties view. Also assign a Step1_Action action object for the added step by editing the Main Entry property.
4. Edit the transition for the added step and assign it to the Step1_transition object.

Depending on the number of steps intended for the sequence, continue to assign suitable action and transition objects for each step. An example of a completed four-step sequence SFC is shown in Figure 4.9.5-2.




four_step_example.png

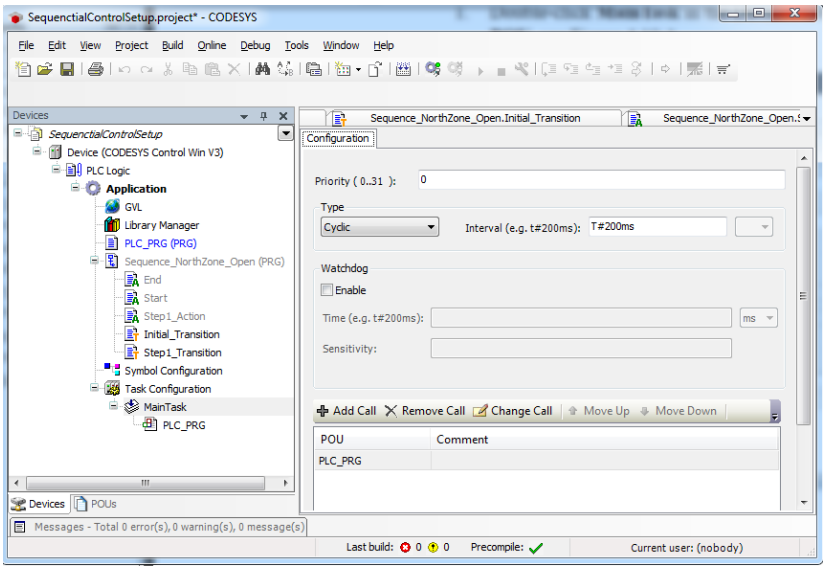
*Figure 4.9.5-2 A completed four step sequence SFC*

## 4.10. Adding a sequence POU to PLC MainTask configuration

After a successful build, add the sequence POU to PLC MainTask configuration.

To add the sequence POU to PLC MainTask configuration:

1. Double-click **MainTask** in the Devices view. In the MainTask editor, click **Add Call**, see Figure 4.10-1.




Logic_Editor_Add_Sequence_POU.png

*Figure 4.10-1 Adding Sequence POU to PLC MainTask*

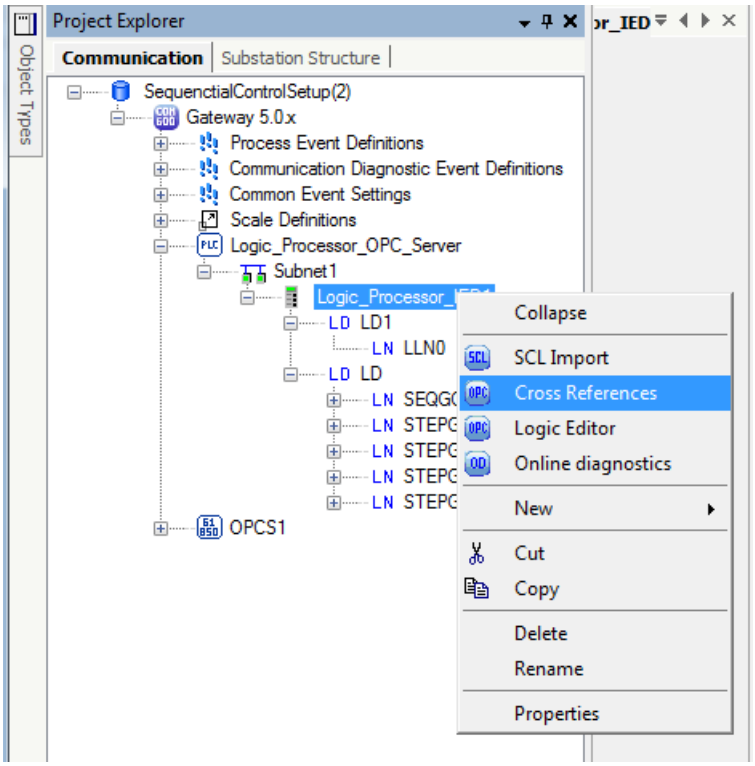
2. In the Input Assistant dialog, select Categories tab and Select program object.
3. Continue to build the program by selecting **Build > Rebuild** in the menu bar.
4. Update the build to COM600 Runtime. For more information, see COM600 Logic Processor User's manual.
5. In case of excessive build errors, make sure that the sequence control library is referred to in the Logic Editor project. To check this, click **Library Manager** in the Devices view and verify that **Sequence Control Library** is listed as one of the libraries. If not, click the **Add library** link and select **Sequence Control Library** under **Miscellaneous** in the **Add Library** dialog.
6. Exit Logic Editor and open SAB600 before proceeding to the next configuration steps.

## 4.11. Configuring sequence cross-references

This section describes the steps involved in configuring cross-references needed for the sequence control functionality. For generic information on the concept of cross-references between COM600 communication structure and Logic Processor variables, see COM600 Logic Processor User's Manual.

To configure cross-references for sequence control:

1. In SAB600 project, right-click the Logic_Processor IED and select **Cross References** to launch the Cross References tool, see Figure 4.11-1.
   The Cross References tool shows all the member variables associated with the declared global variable (see 4.3, Creating a global variable list).




SAB600_SC_Cross_References.png

*Figure 4.11-1 Launching the Cross References tool*

If the variables defined from Logic Processor are not shown, click **Import** and select the “ProjectName”.Device.Application.xml file (for example, SequentialControl-Setup.Device.Application.xml).

2. Assign the breaker (*CSWI*) data attributes to a corresponding SwitchObject variable. The array indices available with the Switches variable from the logic processor indicate the breaker object indentification. For example, Switches[1].Status refers to status indication for the breaker mapped to SwitchObject in Switches[1].
3. To map a data attribute from COM600 communication structure to a variable from the Logic Processor, select the data object from the communication structure view and drag and drop it to the corresponding sequence variable to be mapped.
4. Edit the data object manually to map the right data attribute by selecting the cell in the OPC Server Path column and editing the attribute. For any sequence object from the logic processor, the breaker data attributes needed to be mapped are shown in Table 4.11-1.
5. Finish mapping the CSWI data attributes to corresponding switch select/operate controls for all the indices for the Switches array.
6. Click **Apply** and close the Cross References tool.

***Table 4.11-1 Breaker data attributes mapping to Logic Processor Sequence Object***

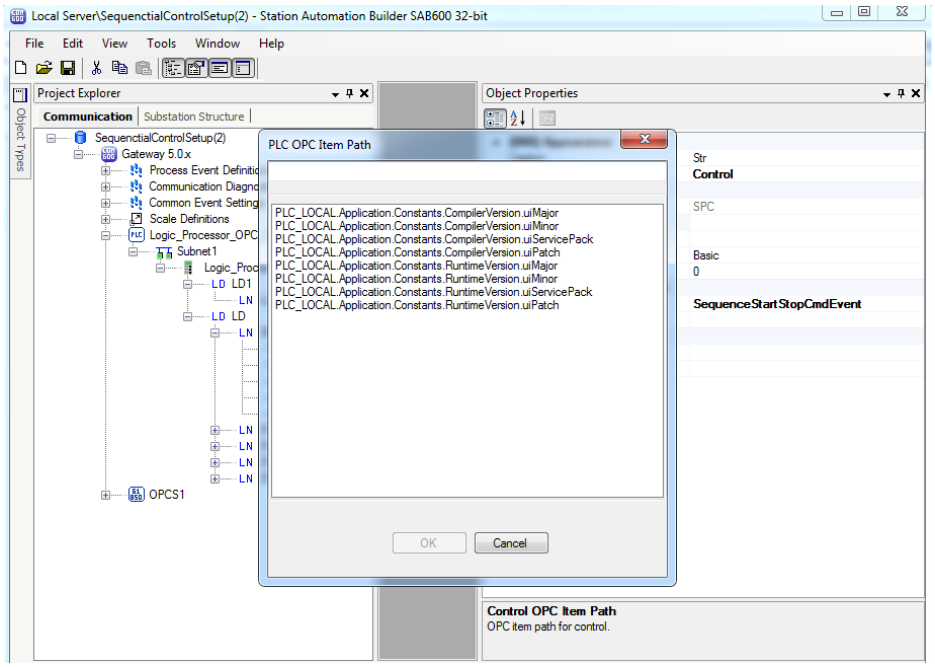
| Breaker Data Attributes | Data Variable of a Sequence Object | Direction |
|---|---|---|
| *CSWI*/Pos/stVal | **.SwitchArray[Index].SwStatus[1] | -> |

| Breaker Data Attributes | Data Variable of a Sequence Object | Direction |
|---|---|---|
| *CSWI*/Pos/stSeld | **.SwitchArray[Index].SwSelectedStatus[1] | -> |
| *CSWI*/Pos/ctlSelOn | **.SwitchArray[Index].SwSelOn[1] | <- |
| *CSWI*/Pos/ctlOperOn | **.SwitchArray[Index].SwOperOn[1] | <- |
| *CSWI*/Pos/ctlSelOff | **.SwitchArray[Index].SwSelOff[1] | <- |
| *CSWI*/Pos/ctlOperOff | **.SwitchArray[Index].SwOperOff[1] | <- |
| *CSWI*/Mod/stVal | **.SwitchArray[Index].SwitchMod[1] | -> |
| *CSWI*/Health/stVal | **.SwitchArray[Index].SwitchHealth[1] | -> |

## 4.12. Assigning item paths to data objects

To assign item paths to data objects:

1. Select **Logic Processor_IED > Sequence logical device > SEQGGIO1 > Str data object** and assign the needed logic processor variables to it.
   Select the Control OPC Item Path parameter in the Object Properties view and click the "…" button to launch the PLC OPC Item Path dialog, see Figure 4.12-1.




SAB600_SC_PLC_OPC_Item_Path.png

*Figure 4.12-1 Assigning OPC Item Path to SEQGGIO1.Str data object*

   Set the **.(SequenceObject).SeqStartStopControl logic processor variable as the control item path. Similarly, assign ****.(SequenceObject).SeqStartStopControl** as the OPC item for State OPC item path.

2. Select **Logic Processor_IED > Sequence logical device > SEQGGIO1 > St data object** and assign the needed logic processor variables to it.

Select the State OPC Item Path parameter in the Object Properties view and assign it to the **.(SequenceObject).SeqState variable through the PLC OPC Item Path.

3. Select **Logic Processor_IED > Sequence logical device > SEQGGIO1 > Auto data object** and assign the needed logic processor variable to it.
   Select the Control OPC Item Path parameter in the Object Properties view and assign it to **,(SeqnenceObject).SeqExecutionType variable through the PLC OPC Item Path dialog. Similarly, assign **.(SequenceObject).SeqExecutionType variable for the State OPC Item Path parameter.
4. Select **Logic Processor_IED > Sequence logical device > SEQGGIO1 > StepExec data object** and assign the needed logic processor variable to it.
   Select the Control OPC Item Path parameter in the Object Properties view and assign it to **(SequenceObject).SeqProceedFlag variable through the PLC OPC Item Path dialog. Similarly, assign **.(SequenceObject).SeqProceedFlag variable for the State OPC Item Path parameter.
5. Select **Logic Processor_IED > Sequence logical device > SEQGGIO1 > OnStepErr data object** and assign the needed logic processor variable to it.
   Select the Control OPC Item Path parameter in the Object Properties view and assign it to **.(SequenceObject).SeqExecutionOnStepError variable through the PLC OPC Item Path dialog. Similarly, assign **.(SequenceObject).SeqExecutionOnStepError variable for the State OPC Item Path parameter.
6. Select **Logic Processor_IED > Sequence logical device > STEPGGIO1 > St data object** and assign the needed logic processor variable to it.
   Select the State OPC Item Path parameter in the Object Properties view and assign it to **.(SequenceObject).StepOperationStatus[1] using the PLC OPC Item Path dialog. Here the index selected for the StepOperationStatus variable corresponds to the step number in the sequence.

Continue assigning the St data object of all the other STEPGGIO logical nodes available for the sequence and map it to the corresponding index in StepOperationStatus variable. For example, STEPGGIO2.Status should be assigned to **.(SequenceObject).StepOperationStatus[2], and STEPGGIO3.Status should be assigned to **.(SequenceObject).StepOperationStatus[3], and similarly for the rest of the STEPGGIO logical nodes available.

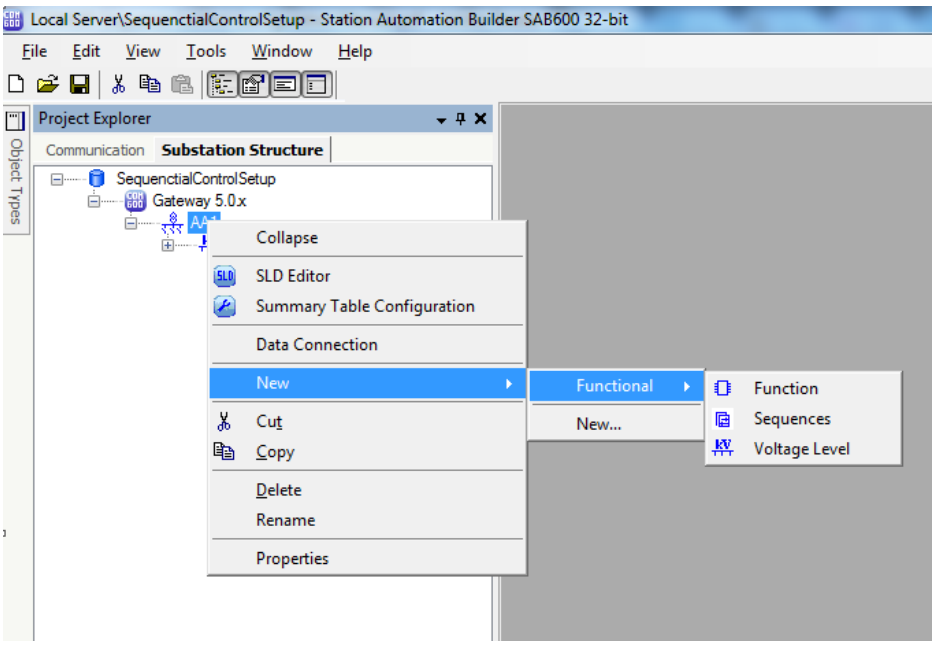
# 5. Configuring WebHMI

## 5.1. Configuring WebHMI for sequence control

This section describes the steps involved in configuring WebHMI for the sequence control functionality. Before starting the WebHMI configuration for sequence control, the required communication parameters and the associated data connections needed for the single line diagram functionality should already be configured in the SAB600 project. For more information on WebHMI configuration, see COM600 HMI Configuration Manual.

To configure WebHMI for sequence control:

1. In SAB600, right-click the Substation object in the Substation Structure and select **New** > **Functional** > **Sequences** to add a sequence voltage level to the Substation structure, see Figure 5.1-1.




SAB600_SC_Substation_Sequence.png

*Figure 5.1-1 Adding sequences to the substation structure*

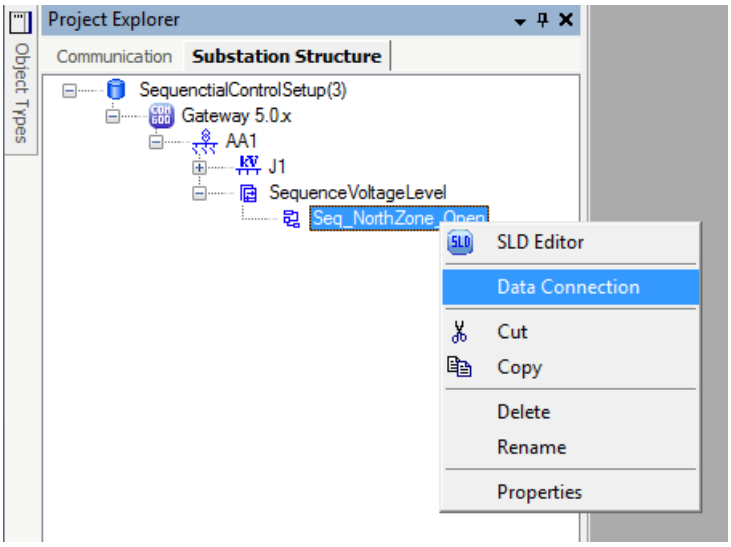
2. Assign a name to the added sequence voltage level by editing the Caption parameter in the object properties window.
3. Add a sequence bay object to the added sequence voltage level by right-clicking the sequence voltage level and selecting **New** > **Functional** > **Sequence**.
4. Rename the added sequence bay object to correspond the sequence name.

## 5.2. Data connection

Connect the sequence data objects defined in the communication structure to the sequence bay object.

To connect sequence data objects:

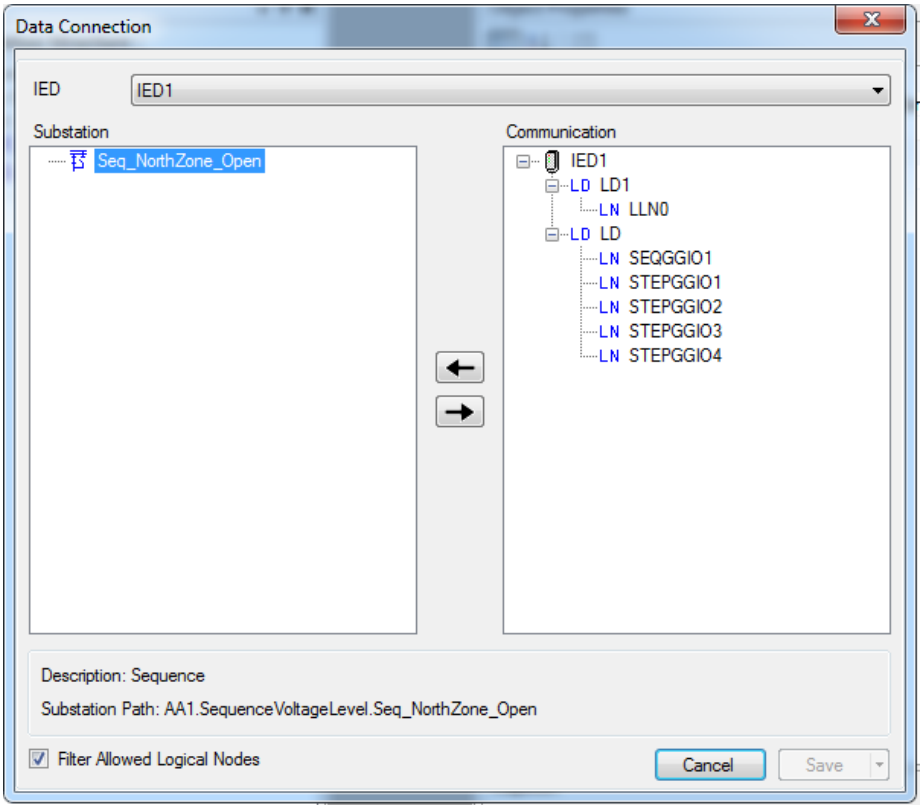
1. Right-click the sequence bay and select **Data Connection** to launch the Data Connection dialog.




SAB600_SC_Launch_Data_Connection.png

*Figure 5.2-1 Launching data connection dialog for the bay object*

2. In the Data Connection dialog, select **Logic_Processor_IED** from the IED drop-down menu.
3. Move all the logical nodes defined for the sequence logical device to the left column in the dialog using the arrow button, see Figure 5.2-2.




SAB600_SC_ Data_Connection.png

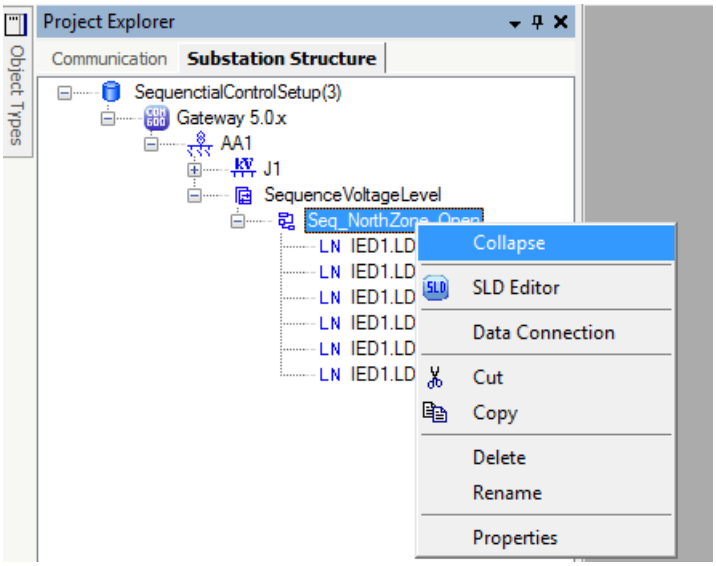
*Figure 5.2-2 Connecting sequence logical nodes to sequence bay*

4. Click **Save**.

## 5.3. Adding a sequence start/stop control

To create a sequence SLD:

1. Launch the SLD Editor for the sequence bay object by right-clicking it in the Substation structure and selecting **SLD Editor**.




SAB600_SC_ SLD_Editor.png

*Figure 5.3-1 Launching SLD Editor for voltage level*

2. Drag and drop a **2-State Indicator/button** and a **Multiple State Button** from the Generic section in the Symbols dialog to the SLD layout page. The **2-State Indicator/button** control is required for starting and stopping a sequence.

3. Right-click the **2-State Indicator/Button** and select **Configure 2-State Indicator/Button** to launch the 2-State Indicator/Button Configuration dialog (see Figure 5.3-2).

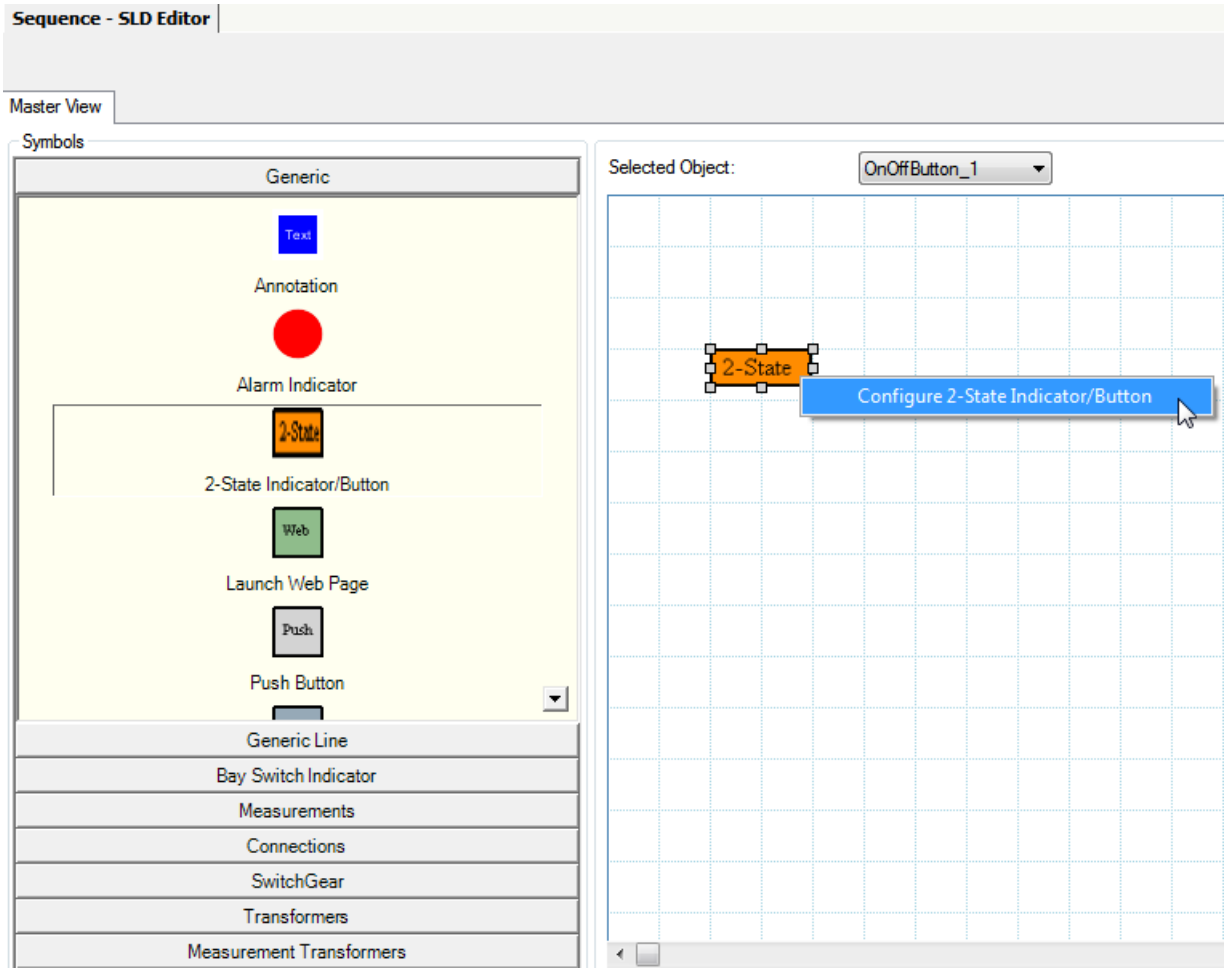
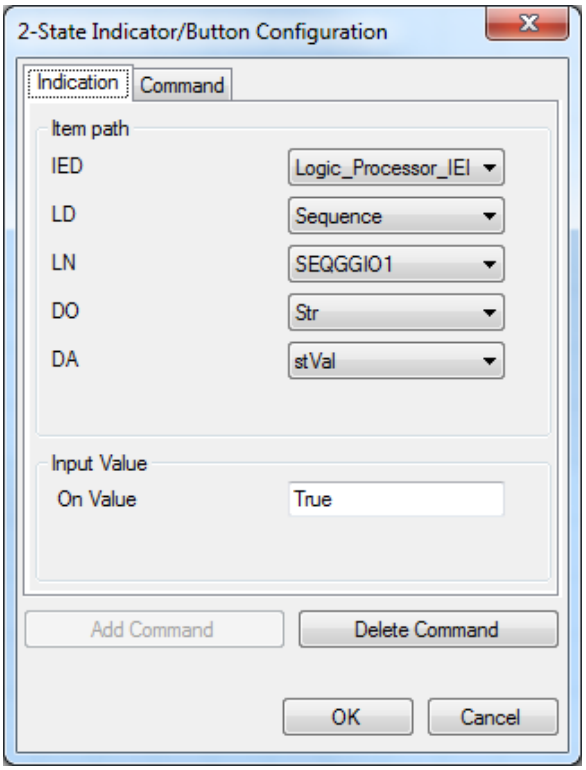



*Figure 5.3-2 Adding a two state indicator/button to sequence SLD for sequence open/close control*

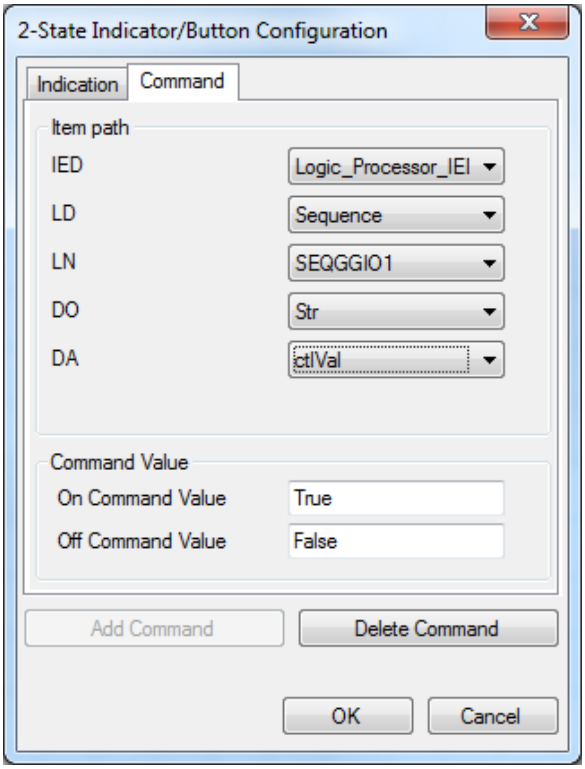
4. Configure the Indication properties as shown in Figure 5.3-3.




2_state_control_configuration.png

*Figure 5.3-3 Configuring a 2-State button with indication value*

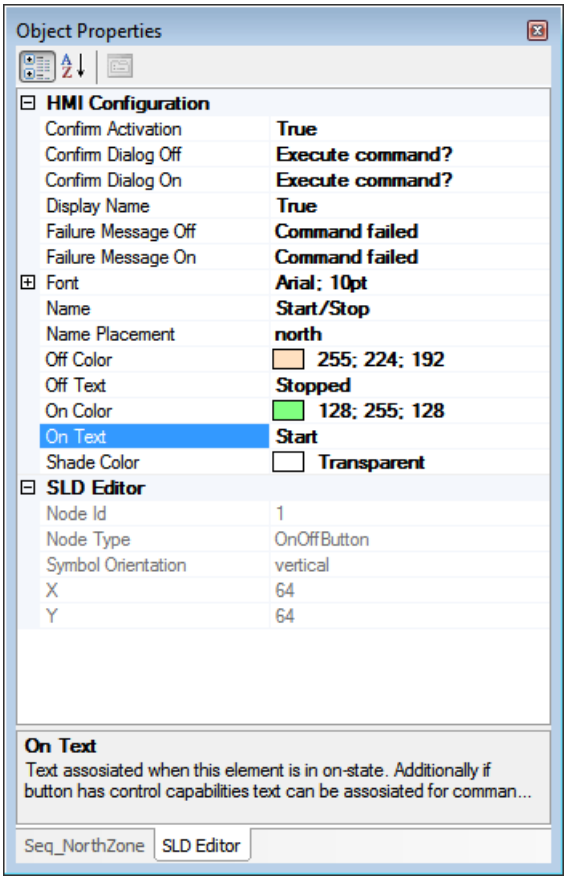
5. Click **Add Command** to add the Command tab. Configure the Command properties as shown in Figure 5.3-4.




2_state_control_configuration_command.png

*Figure 5.3-4 Configuring a 2-State button for control command*

6. In the Object Properties window, set the corresponding property values for the added start/stop control, see Figure 5.3-5.




2_state_object_properties.png

*Figure 5.3-5 Object properties for the 2-State button*

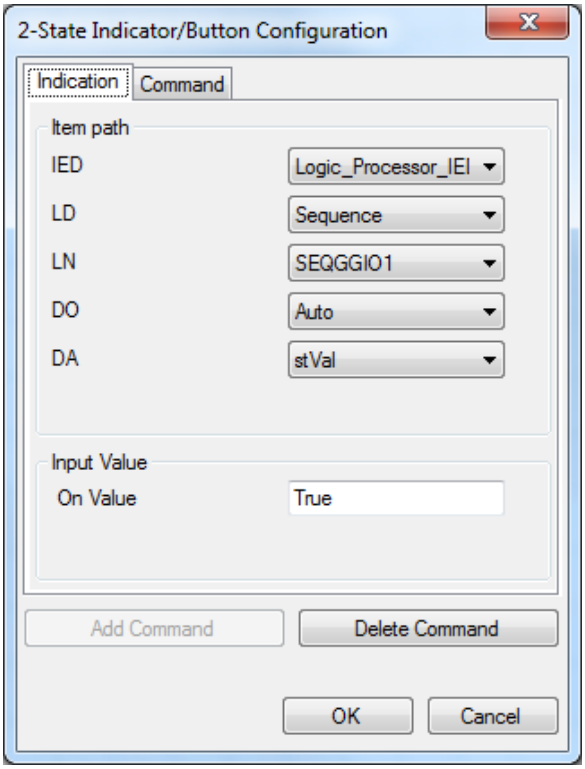
7. Repeat the steps 2 - 6 to add more 2-State Indicator/Button objects to the sequence SLD functionality.

## 5.4. Adding an execution mode control

A new 2-State Indicator/Button is needed to change the mode of execution (automatic/manual) for the sequence.

To add an execution mode control:

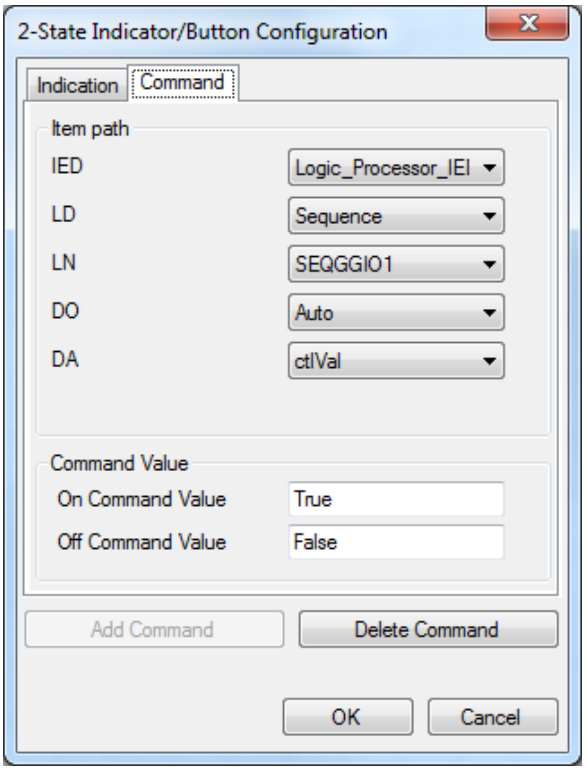
1. Drag and drop the **2-State Indicator/Button** control from the Generic section in the Symbols dialog to the SLD layout page.
2. Right-click the **2-State Indicator/Button** and select **Configure 2-State Indicator/Button** to launch the 2-State Indicator/Button Configuration dialog.
3. Configure the Indication properties as shown in Figure 5.3-3.




execution_mode_indication_properties.png

*Figure 5.4-1 Configuring the Indication tab properties for a sequence execution mode 2-State Indicator/Button*

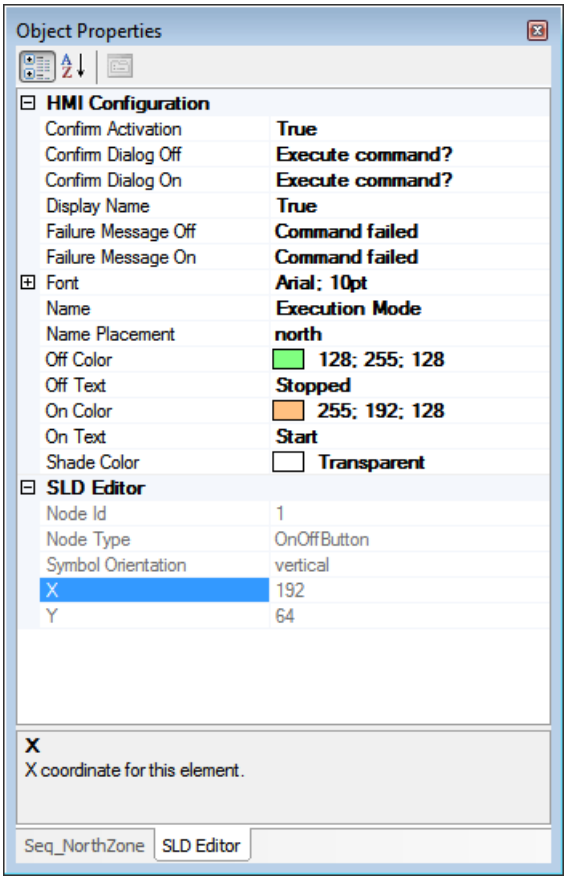
4. Click **Add Command** to add the Command tab. Configure the Command properties as shown in Figure 5.3-4.




execution_mode_command_properties.png

*Figure 5.4-2 Configuring the Command tab properties for a sequence execution mode 2-State Indicator/Button*

5. In the Object Properties window, set the corresponding property values for the added start/stop control, see Figure 5.3-5.

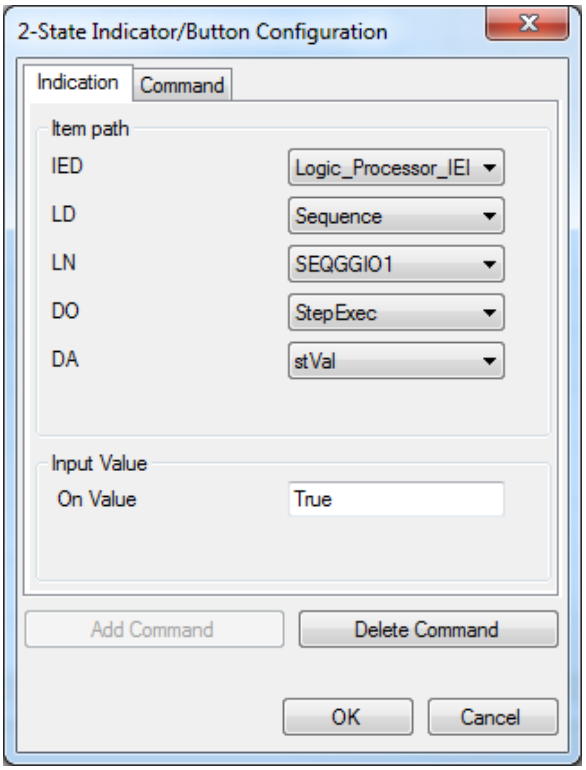



execute_command_object_properties.png

*Figure 5.4-3 Object properties for a sequence execution mode 2-State Indicator/Button*

Similarly, add a new 2-State Indicator/Button for acknowledging the sequence in manual mode:

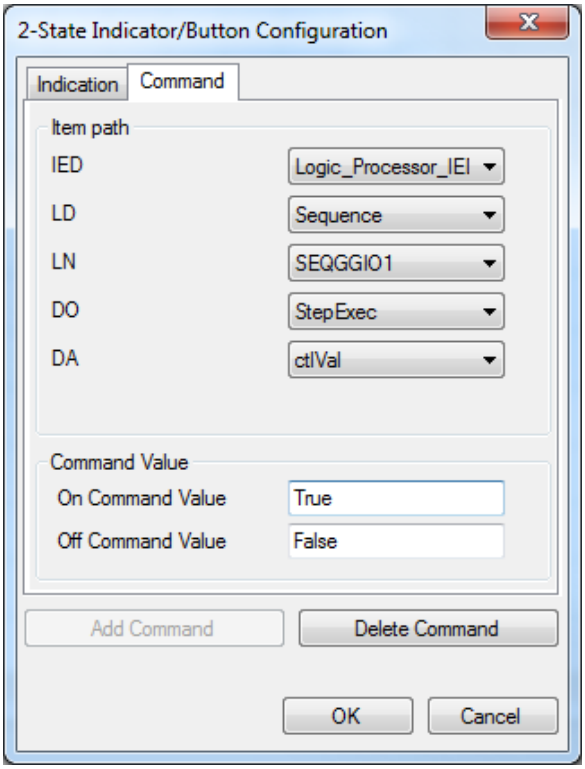
1. Configure the Indication properties as shown in Figure 5.4-4.




execution_manual_mode_indication.png

*Figure 5.4-4 Configuring the Indication tab properties for a sequence manual mode acknowledgement 2-State Indicator/Button*

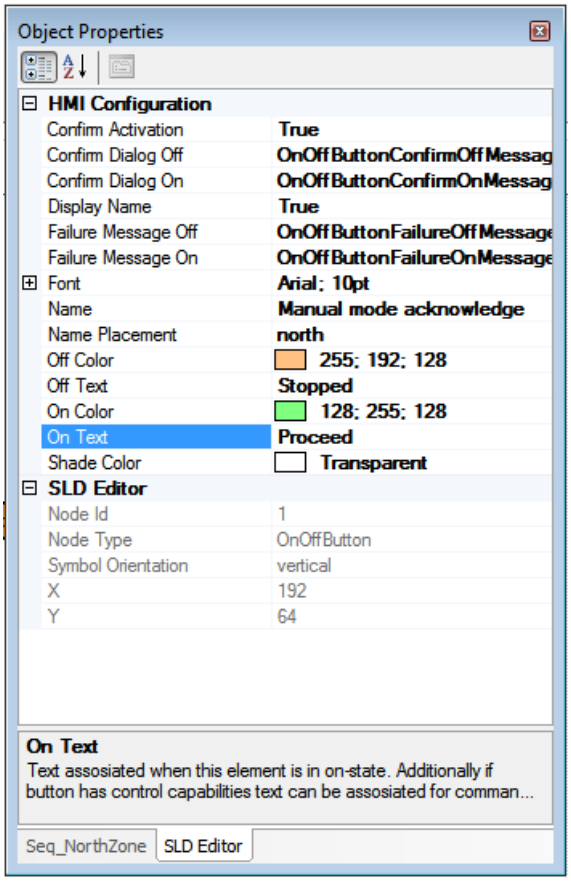
2. Click **Add Command** to add the Command tab. Configure the Command properties as shown in Figure 5.4-5.




execution_manual_mode_command.png

*Figure 5.4-5 Configuring the Command tab properties for a sequence manual mode acknowledgement 2-State Indicator/Button*

3. In the Object Properties window, set the corresponding property values for the added start/stop control, see Figure 5.4-6.




execute_manual_object_properties.png

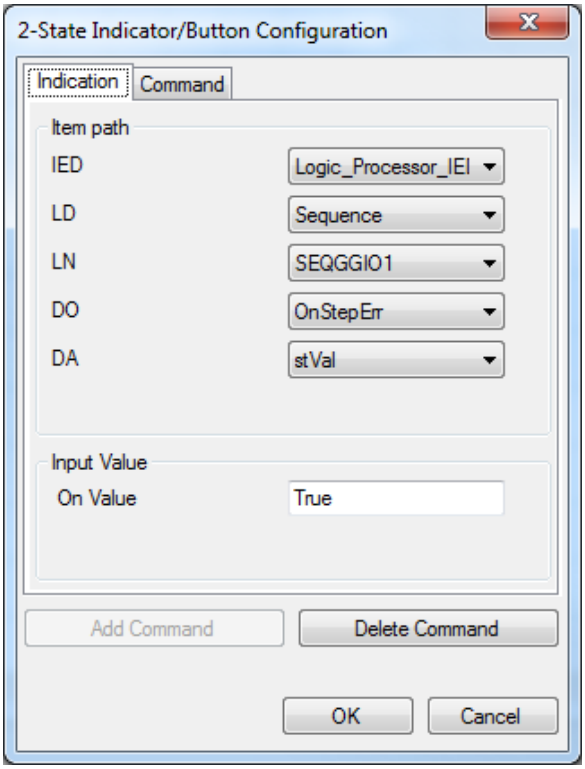
*Figure 5.4-6 Object properties for a a sequence manual mode acknowledgement 2-State Indicator/Button*

## 5.5. Adding a button for sequence execution on step error

Add a 2-State Indicator/Button for the purposes of sequence execution on step error setting.

To add a button for executing a sequence on step error:

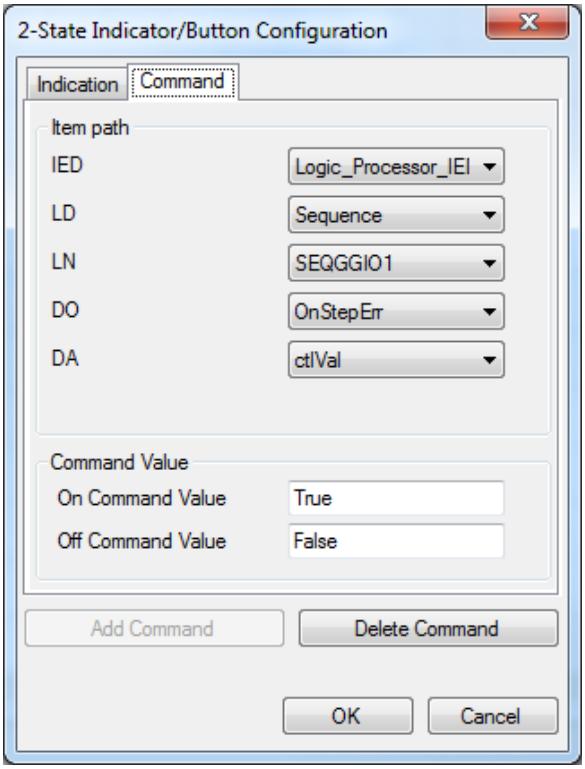
1. Drag and drop the **2-State Indicator/Button** control from the Generic section in the Symbols dialog to the SLD layout page.
2. Right-click the **2-State Indicator/Button** and select **Configure 2-State Indicator/Button** to launch the 2-State Indicator/Button Configuration dialog.
3. Configure the Indication properties as shown in Figure 5.3-3.




on_step_error_indication.png

*Figure 5.5-1 Configuring the Indication tab properties for a sequence execution on step error 2-State Indicator/Button*

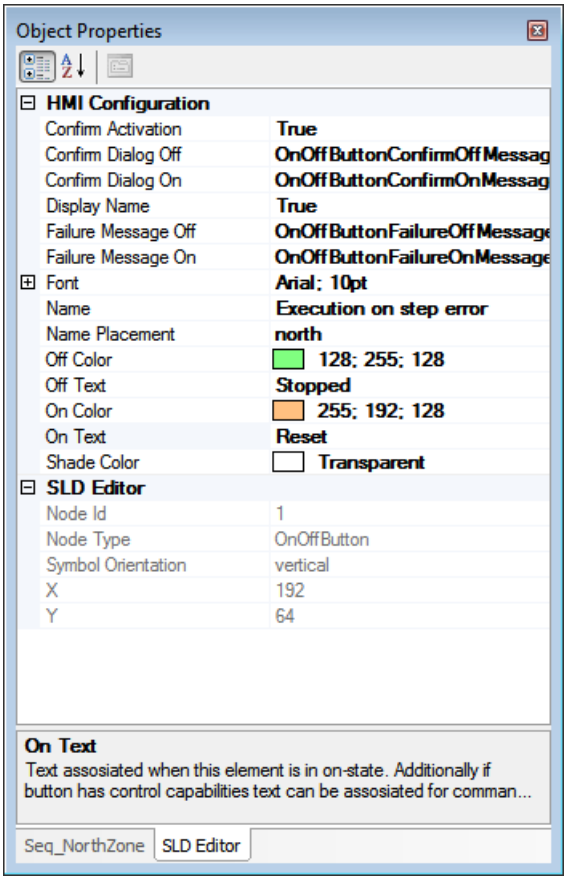
4. Click **Add Command** to add the Command tab. Configure the Command properties as shown in Figure 5.3-4.




on_step_error_command.png

*Figure 5.5-2 Configuring the Command tab properties for a sequence execution on step error 2-State Indicator/Button*

5. In the Object Properties window, set the corresponding property values for the added start/stop control, see Figure 5.3-5.




on_step_error_properties.png

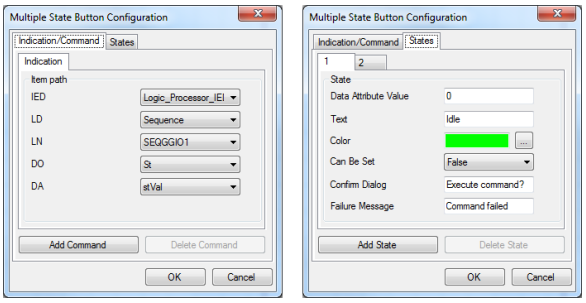
*Figure 5.5-3 Object properties for a sequence execution on step error 2-State Indicator/Button*

## 5.6. Adding a button for viewing sequence status

Add Multiple State Button controls to view the status of the sequence execution. These controls can show the status of the sequence as a whole and status of each step in the sequence.

To add a button for viewing the status of the whole sequence:

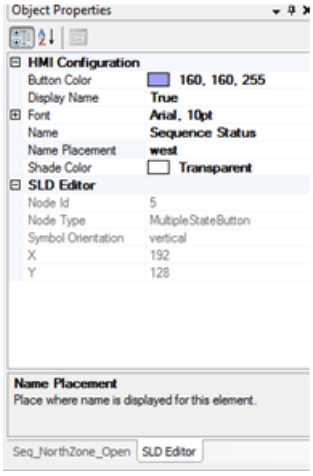
1. Drag and drop the **Multiple State Button** control from the Generic section in the Symbols dialog to the SLD layout page.
2. Right-click the **Multiple State Button** and select **Configure Multiple State Button** to launch the configuration dialog.
3. Configure the Indication properties as shown in Figure 5.3-3. Click **Add State** to add more states for the sequence (idle, running, failed, passed).
   The status values showing the sequence status as a whole and the status of each step in the sequence can be one of four possible states, see Table 5.6-1. Include all the states in the button configuration showing sequence status.




multiple_state_button_configuration.png

*Figure 5.6-1 Configuring the indication tab properties for a Multiple State Button showing sequence idle status*

4. In the Object Properties window, set the corresponding property values for the sequence status control, see Figure 5.3-5.




on_step_error_properties.png

*Figure 5.6-2 Multi State Button object properties for a sequence status*

5. Repeat the steps to add the status for individual steps in the sequence. Select the appropriate data attribute path when configuring the indication properties for the added Multiple State Button.
6. After completing the configuration, click **Apply** in the SLD Editor to save the changes.

***Table 5.6-1 Possible states for a sequence/sequence step***

| Data attribute value | Text |
|---|---|
| 0 | Idle |
| 1 | Running |
| 2 | Failed |
| 3 | Passed |

## 5.7. Updating COM600 runtime environment

Update the COM600 runtime environment with the sequence configuration. Before updating make sure that SAB600 is configured to update the correct COM600 runtime.

To update the COM600 runtime environment:
1. Right-click the Gateway object in the Communication view and select **Management**.
2. Click **Update and reload configuration** to update the COM600 runtime.

# 6. Executing a sequence

## 6.1. Executing a sequence with COM600 WebHMI

Once the sequence configuration is completed and the COM600 runtime is updated with the configuration, the COM600 WebHMI can be used to execute the sequence.

An example of a four-step sequence in the WebHMI is shown in Figure 4.8-2.

The buttons used in the execution are:

- **Start/Stop**: Use this button to start or stop the sequence.
- **Execution mode**: Use this button to change the execution mode of the sequence between automatic and manual.
- **Manual mode acknowledge**: Use this button to manually acknowledge each step when the sequence is run in manual mode.
- **Execution on step error**: Use this button to reset or continue the sequence execution on a step failure.

The indicators related to steps are status-only objects, showing the status of each step during the execution of the sequence.


*Figure 6.1-1 An example of a four-step sequence in COM600 WebHMI*

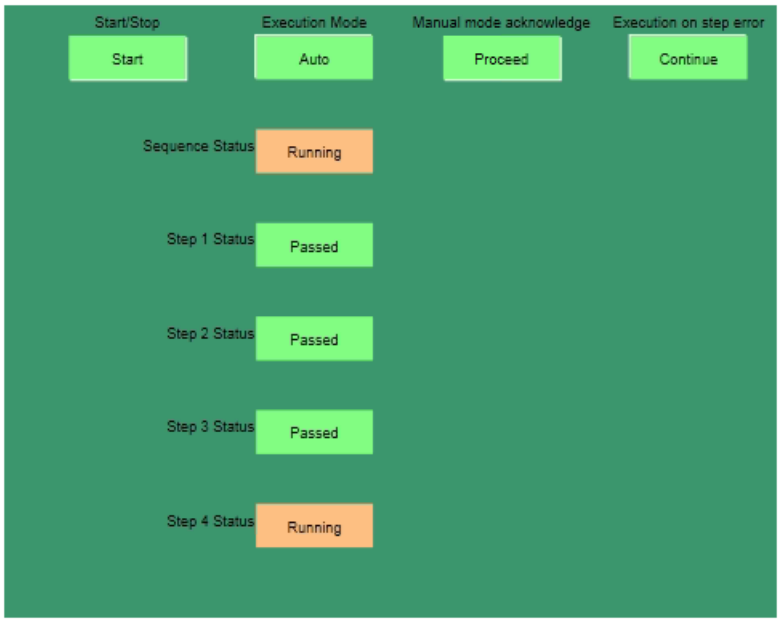
## 6.2. Automatic execution

In the automatic execution mode, the sequence execution simply runs through each step without any need of manual intervention.

To run the sequence in automatic mode:

1. Make sure that the execution mode button indicates **Auto**. If not, double-click the button and select **Auto**.
2. Double-click **Start/Stop** and in the following dialog, click **Start** to begin the sequence execution.
   If needed, stop the sequence execution in the same dialog by selecting **Stop**. As each step is executed, the corresponding breaker/switch mapped for the step is controlled to the desired state.

An example of a four-step sequence in automatic execution mode is shown in Figure 6.2-1.




automatic_execution.png

*Figure 6.2-1 A four-step sequence in automatic execution mode*

## 6.3. Manual execution

In the manual execution mode, the execution pauses and waits for user acknowledgement after each step in the sequence.

To run the sequence in manual mode:

1. Make sure that the execution mode button indicates **Manual**. If not, double-click the button and in the following dialog, select **Manual**.

2. Double-click the **Start/Stop** button and in the following dialog, click **Start** to begin the sequence execution.
3. Before the execution of each step, the sequence pauses for manual acknowledgement. Use the **Manual mode acknowledgement** control to acknowledge the sequence.
4. If needed, stop the sequence by double-clicking the **Start/Stop** object. In the following dialog, click **Stop**.
   Notice that as each step is executed, the corresponding breaker/switch mapped for the step is controlled to the desired state. An example of a four-step sequence in manual execution mode is shown in Figure 6.3-1.


manual_mode_execution.png

*Figure 6.3-1 An example of a four-step sequence in manual execution mode*

# Index

## A



## C



## D



## E



## G



## L

# S

**ABB Distribution Solutions**
**Distribution Automation**
P.O. Box 699
FI-65101 Vaasa, Finland
Phone: +358 10 22 11

**ABB Distribution Automation**
4300 Coral Ridge Drive
Coral Springs, Florida 33065
Phone: +1 954 752 6700

**www.abb.com/mediumvoltage**
**www.abb.com/substationautomation**