COM600 series 5.1
Logic Processor User's Manual
Contents:

1. About this manual .......................................................... 5
   1.1. Copyright ............................................................... 5
   1.2. Disclaimer ............................................................. 5
   1.3. Conformity .............................................................. 6
   1.4. Trademarks .............................................................. 6
   1.5. General information .................................................. 6
   1.6. Document conventions .............................................. 6
   1.7. Use of symbols ......................................................... 7
   1.8. Terminology ........................................................... 8
   1.9. Abbreviations .......................................................... 9
   1.10. Related documents .................................................. 10
   1.11. Document revisions ................................................ 10

2. Introduction ......................................................................... 11
   2.1. General information about the COM600 series .............. 11
   2.2. COM600 product series variants and rationale .............. 11
   2.3. Functional overview .................................................. 12
   2.4. Features ................................................................. 13

3. Configuration ........................................................................ 15
   3.1. Overview of configuring Logic Processor ...................... 15
   3.2. Building communication structure objects .................... 15
   3.3. Logic editor ............................................................ 16
       3.3.1. Creating the logic in the CoDeSys programming environment ............................................. 16
       3.3.2. Adding symbol configuration ................................................................. 17
       3.3.3. Add POU to Application MainTask ..................................................... 19
       3.3.4. Setting device path ............................................................... 20
       3.3.5. Downloading and starting PLC application ........................................... 22
       3.3.6. Logic Editor Online Help .................................................. 23
   3.4. Making cross-references ............................................. 24
   3.5. Creating virtual data objects in the Logic Processor IED ........ 25
   3.6. Downloading Logic Processor OPC Server configuration .... 27

4. Application example ............................................................ 28
   4.1. Logic requirement ..................................................... 28
   4.2. Building object tree in SAB600 of three REF615s ........... 29
   4.3. Adding Logic Processor IED ........................................ 30
   4.4. Creating logic configuration ........................................ 30
   4.5. Adding POU to Application MainTask ............................ 34
   4.6. Add Symbol Configuration .......................................... 35
   4.7. Setting device path ...................................................... 36
   4.8. Building and activating the application ......................... 37
4.9. Making cross-references .............................................................. 37

Index .............................................................................................................. 39
1. About this manual

1.1. Copyright

This document and parts thereof must not be reproduced or copied without written permission from ABB, and the contents thereof must not be imparted to a third party, nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

Trademarks

ABB and Relion are registered trademarks of the ABB Group. All other brand or product names mentioned in this document may be trademarks or registered trademarks of their respective holders.

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 Logic Processor OPC server configuration for COM600.

Information in this user’s manual is intended for application engineers who configure the Logic Processor function in COM600. 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**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>An abnormal state of a condition.</td>
</tr>
<tr>
<td>Alarms and Events; AE</td>
<td>An OPC service for providing information about alarms and events to OPC clients.</td>
</tr>
<tr>
<td>COM600 Series; COM600</td>
<td>COM600 as a generic name for COM600S IEC and COM600F ANSI products</td>
</tr>
<tr>
<td>Data Access; DA</td>
<td>An OPC service for providing information about process data to OPC clients.</td>
</tr>
<tr>
<td>Data Object; DO</td>
<td>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.</td>
</tr>
<tr>
<td>Data Set</td>
<td>The data set is the content basis for reporting and logging. The data set contains references to the data and data attribute values.</td>
</tr>
<tr>
<td>Device</td>
<td>A physical device that behaves as its own communication node in the network, for example, protection relay.</td>
</tr>
<tr>
<td>Event</td>
<td>Change of process data or an OPC internal value. Normally, an event consists of value, quality, and timestamp.</td>
</tr>
<tr>
<td>Intelligent Electronic Device</td>
<td>A physical IEC 61850 device that behaves as its own communication node in the IEC 61850 protocol.</td>
</tr>
<tr>
<td>Logical Device; LD</td>
<td>Representation of a group of functions. Each function is defined as a logical node. A physical device consists of one or several LDs.</td>
</tr>
<tr>
<td>Logical Node; LN</td>
<td>The smallest part of a function that exchanges data. An LN is an object defined by its data and methods.</td>
</tr>
<tr>
<td>OPC</td>
<td>Series of standards specifications aiming at open connectivity in industrial automation and the enterprise systems that support industry.</td>
</tr>
</tbody>
</table>
### Term

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC item</td>
<td>Representation of a connection to the data source within the OPC server. An OPC item is identified by a string <code>&lt;object path&gt;:&lt;property name&gt;</code>. Associated with each OPC item are Value, Quality, and Time Stamp.</td>
</tr>
<tr>
<td>Property</td>
<td>Named data item.</td>
</tr>
<tr>
<td>Report Control Block</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 1.9. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>Alarms and Events</td>
</tr>
<tr>
<td>DA</td>
<td>Data Access</td>
</tr>
<tr>
<td>DO</td>
<td>Data Object</td>
</tr>
<tr>
<td>GW</td>
<td>Gateway, component connecting two communication networks together</td>
</tr>
<tr>
<td>WebHMI</td>
<td>Web Human Machine Interface</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>IED</td>
<td>Intelligent Electronic Device</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LD</td>
<td>Logical Device</td>
</tr>
<tr>
<td>LN</td>
<td>Logical Node</td>
</tr>
<tr>
<td>NCC</td>
<td>Network Control Center</td>
</tr>
<tr>
<td>OLE</td>
<td>Object Linking and Embedding</td>
</tr>
<tr>
<td>OPC</td>
<td>OLE for Process Control</td>
</tr>
<tr>
<td>P&amp;C</td>
<td>Protection &amp; Control</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controller</td>
</tr>
<tr>
<td>POU</td>
<td>Program Organization Unit</td>
</tr>
<tr>
<td>RTS</td>
<td>Request To Send</td>
</tr>
<tr>
<td>SA</td>
<td>Substation Automation</td>
</tr>
<tr>
<td>SCD</td>
<td>Substation Configuration Description</td>
</tr>
<tr>
<td>SCL</td>
<td>Substation Configuration Language</td>
</tr>
<tr>
<td>SFC</td>
<td>Sequential Function Chart</td>
</tr>
<tr>
<td>SLD</td>
<td>Single Line Diagram</td>
</tr>
<tr>
<td>XML</td>
<td>eXtended Markup Language</td>
</tr>
</tbody>
</table>
1.10. Related documents

<table>
<thead>
<tr>
<th>Name of the manual</th>
<th>MRS number</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM600 User's Manual</td>
<td>1MRS756125</td>
</tr>
<tr>
<td>COM600 Operator's Manual</td>
<td>1MRS756705</td>
</tr>
<tr>
<td>COM600 HMI Configuration Manual</td>
<td>1MRS756740</td>
</tr>
<tr>
<td>COM600 Data Historian Operator's Manual</td>
<td>1MRS756739</td>
</tr>
<tr>
<td>COM600 Sequence Control Configuration Manual</td>
<td>1MRS755001</td>
</tr>
<tr>
<td>Master Protocols (Ethernet based) Configuration</td>
<td>1MRS758689</td>
</tr>
<tr>
<td>and Operation Manual</td>
<td></td>
</tr>
<tr>
<td>Master Protocols (Ethernet based) Technical</td>
<td>1MRS758690</td>
</tr>
<tr>
<td>Reference Manual</td>
<td></td>
</tr>
<tr>
<td>Slave Protocols (Ethernet based) Configuration</td>
<td>1MRS758691</td>
</tr>
<tr>
<td>and Operation Manual</td>
<td></td>
</tr>
<tr>
<td>Slave Protocols (Ethernet based) Technical</td>
<td>1MRS758692</td>
</tr>
<tr>
<td>Reference Manual</td>
<td></td>
</tr>
<tr>
<td>DNP 3.0 Serial Master (OPC) User's Manual</td>
<td>1MRS755657</td>
</tr>
<tr>
<td>DNP 3.0 Serial Slave (OPC) User's Manual</td>
<td>1MRS755495</td>
</tr>
<tr>
<td>IEC 60870-5-101 Slave (OPC) User's Manual</td>
<td>1MRS755382</td>
</tr>
<tr>
<td>IEC 60870-5-101 Master (OPC) User's Manual</td>
<td>1MRS756703</td>
</tr>
<tr>
<td>IEC 60870-5-103 Master (OPC) User's Manual</td>
<td>1MRS752278</td>
</tr>
<tr>
<td>COM600 Logic Processor User's Manual</td>
<td>1MRS756738</td>
</tr>
<tr>
<td>Modbus Serial Master (OPC) User's Manual</td>
<td>1MRS756126</td>
</tr>
<tr>
<td>Modbus Serial Slave (OPC) User's Manual</td>
<td>1MRS756913</td>
</tr>
</tbody>
</table>

1.11. Document revisions

<table>
<thead>
<tr>
<th>Document version/date</th>
<th>Product revision</th>
<th>History</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/13.2.2009</td>
<td>3.3</td>
<td>Document created</td>
</tr>
<tr>
<td>B/06.11.2009</td>
<td>3.4</td>
<td>Document revised</td>
</tr>
<tr>
<td>C/30.06.2011</td>
<td>3.5</td>
<td>Document revised</td>
</tr>
<tr>
<td>D/31.5.2012</td>
<td>4.0</td>
<td>Document revised</td>
</tr>
<tr>
<td>E/13.3.2015</td>
<td>4.1</td>
<td>Document revised</td>
</tr>
<tr>
<td>F/24.5.2017</td>
<td>5.0</td>
<td>Document revised</td>
</tr>
<tr>
<td>G/6.3.2018</td>
<td>5.1</td>
<td>Document revised</td>
</tr>
</tbody>
</table>
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. Functional overview

The Logic Processor function enables developing customer-specific automated applications for COM600. Applications are programmed using the IEC 61131-3 logic editor (CoDeSys). The COM600 unit has the logic engine (CoDeSys Control Win V3 Runtime), which can load and execute the created IEC 61131-3 applications. Information transfer between the logic engine and other COM600 components, such as OPC Servers, slave clients and WebHMI, is handled by the Logic Processor OPC Server. It enables the logic variables to be connected to the process signals available via different communication protocols in COM600.

The shortest possible data transfer cycle between process signals and logic variables is 50 ms.

The logic program might not detect progress data changes that are faster than or close to the transfer cycle time.

The default task interval of the logic program is 200 ms. The theoretical response time of the logic program to process signal change with a process control command is 300 ms (=50 ms + 200 ms + 50 ms). The task interval of the logic program can be set from the Task Configuration dialog (see 3.3.3, Add POU to Application MainTask).

For information about the actual logic programming, see CoDeSys documentation.
CoDeSys Control Win V3 runtime system is manufactured by 3S-Smart Software Solutions GmbH (www.3s-software.com).

2.4. Features

Logic editor (CoDeSys V3 programming system) supports all five standard programming languages defined by the IEC 61131-3 standard:

- Ladder diagram (LD)
- Sequential Function Charts (SFC)
- Function Block Diagram (FBD)
Logic editor also supports online debugging.

CoDeSys Control Win V3 Runtime system includes the following features:

- Loading and execution of the IEC 61131-3 applications
- Debug monitoring for IEC applications

The Logic Processor OPC server supports the following features:

- Updating of logic variables based on process indication/measurement values
- Controlling process control signals from logic variables
- Logic variable presented as 61850 data object for other COM600 component (WebHMI, slave clients)
- Controls from WebHMI or NCC (via slave client) to logic variable

The following techniques are used:

- OPC Data Access Server v. 1.0/2.0
- OPC Alarms and Events Server v. 1.10
- OPC Data Access Client v.2.0
- IEC 61850 data modeling

For example measurements, indications, and controls can be exchanged between the logic runtime and other COM600 components.
3. Configuration

3.1. Overview of configuring Logic Processor

The prerequisite is to have knowledge about logic programming and IEC 61131-3 standard.

Also, CoDeSys programming environment must be installed from the SAB600 DVD to be able to use the Logic Editor in SAB600. The version of the programming environment is dependent on the used COM600 version. The installed versions can be selected during the setup. For information about CoDeSys programming environment installation, see COM600 User's Manual.

Before starting to configure the Logic Processor, it is necessary that the process communication has been configured.

The Logic Processor configuration in COM600 can be divided into the following tasks:

2. Creating the logic in the CoDeSys programming environment.
3. Building a cross-reference between process data and logic variables data by using the Cross-References tool.
4. Creating virtual data objects in the Logic Processor OPC server and connecting the data objects to logic variables with the object properties.

3.2. Building communication structure objects

To build the communication structure:

1. Add the Logic Processor OPC Server object in the Communication structure by selecting the Gateway object.
2. Right-click the Gateway object and select New > Logic Processor OPC Server.
3. Right-click the Logic Processor OPC Server object and add Logic Processor Sub-network.
4. Right-click the Logic Processor Subnetwork object and add Logic Processor IED.
3.3. Logic editor

3.3.1. Creating the logic in the CoDeSys programming environment

Logic Processor IED must be added to the communication structure before starting the Logic Processor CoDeSys programming environment.

To launch the CoDeSys programming environment:
1. Right-click the Logic Processor IED object and select **Logic Editor**.
   The CoDeSys programming environment will start as a new application instance.
2. Select **CoDeSys** from the taskbar.
   If this is the first time to launch CoDeSys, an empty project with the same name as the SAB600 project will be opened. If a project has already been saved, the saved project will be opened. In the new default project, device CoDeSys Control Win V3 has already been added, and it should not be changed. A default POU (Program Organization Unit) PLC_PRG is already available in the structure tree, under Application.
You can modify the existing POU and add more POUs under the device. CoDeSys programming help is available from the Help menu.

### 3.3.2. Adding symbol configuration

After the POUs in the PLC Application have been defined and created, you can perform additional symbol configuration on the variables associated with them. These variables will be available for external communication through OPC standard DA access. The variables that are selected in symbol configuration are made available in SAB600 Logic Processor Cross References tool to set up data connection with other OPC Protocol (DNP/Modbus/IEC 61850) servers/clients in the SAB600 communication structure.

All variables must be defined for the PLC Application before any Symbol Configuration can be done. Double-click the POU to add variables and logic.
To add a symbol configuration:
1. Select **Application** in the **Device tree**.
2. Right-click **Application**.
3. Select **Add Object**.
4. Select **Symbol configuration**. A new **Symbol Configuration** object is added under **Application** and the symbol configuration editor opens. In the **Symbol Configuration** editor click the **View drop-down** button and select **Unconfigured** from **Project** button. This should show all variables available from all POUs in the open project.
5. Click **Build** to view available symbols. All latest variables available from all POUs after compilation are displayed.
6. Select each of the variables that are intended to be available for external entities through OPC DA Access. Assign necessary access rights (**Read**, **Write**, **Read/Write**) to each of the selected variables.

During the compilation of the project, a symbol list is created and it is exported to a file (XML) in the project directory, and also loaded to the device during the application download in the form of a child application.
3.3.3. Add POU to Application MainTask

A PLC Application comprises of one or more POUs (PLC Programs). After creating a POU to the application, add it to Application Maintask for execution.

To add a POU to Application’s maintask:
1. Select **Application** in the **Device Tree**.
2. Double-click the **MainTask** child object under **Task Configuration**. The configuration editor for MainTask opens.

![MainTask Configuration Editor](MainTask_Configuration_Editor.png)

Figure 3.3.3-1 MainTask Configuration Editor

3. To add any additional POUs, select **Add Call** button in **MainTask Configuration Editor**.
4. The **Input Assistant** window opens. Select the desired POU available under **Application**, and click **OK**.
5. In the **MainTask Configuration Editor**, set **Priority** to 0 and select **Cyclic** for the task type and “T#200ms” for 200-millisecond interval time.
3.3.4. Setting device path

Before downloading a PLC Application to Logic Processor (CoDeSys Control Win V3 Runtime) in COM600, an active Device path must be set between Logic Editor and Logic Processor.

To set active Device path between Logic Editor and Logic Processor:
1. Double-click the **Device** object available in **Logic Editor**. The configuration editor for Device opens.

![Device Configuration Editor](Device_Configuration_Editor.png)

*Figure 3.3.4-1 Device Configuration Editor*
2. In the Device configuration editor under **Communication Settings** tab, click the **Gateway** drop-down button and then click **Manage gateways** …

![Manage Gateways](Manage_Gateways.png)

**Figure 3.3.4-2 Manage Gateways**

3. In the **Manage gateways** window, click **Add** to add a Gateway using the IP Address of the COM600 being used. Click **OK** to close the window. Do this only if the desired Gateway is not available in this dialog already.

4. In the **Device** configuration editor under **Communication Settings** tab, click **Scan Network**… The **Select Device** window opens and shows the “Logic Processor” device available in COM600. The Logic Processor item displayed under Gateway is typically identified by COM600 Computer Name.

5. Select the node showing COM600 Computer Name and click **OK**. This sets the active path between Logic Editor and Logic Processor, allowing further PLC Application download possible.

   ![Information Symbol](information.png)
   
   If the Logic Processor does not show up under gateway, then there could be a possible firewall rule blocking the connection from Logic Editor to Logic Processor. Check the Inbound rules defined in COM600 Windows Firewall. The rules specific to Logic Processor can be further identified by looking for “Gateway Service” as the rule name in Windows Firewall configuration.
Once the active path is set, the Device Communication Settings page should show up as below. Further downloads can be accomplished by login to device.

![Select Device Active Path to COM600](Select_Device_Active_Path.png)

6. Once the active path is set, the Device Communication Settings page should show up as below. Further downloads can be accomplished by login to device.

![Device Communication Settings after setting active path](Device_Communication_Settings_Active_Path.png)

### 3.3.5. Downloading and starting PLC application

After setting the device active path between Logic Editor and Logic Processor, PLC Application can be built and downloaded to Logic Processor.

To download and start PLC Application:

1. Select **Build** from the menu bar in Logic Editor, and click **Build** to build and compile PLC Application in a project. All objects belonging to this application are syntactically checked. Any potential error messages or warnings are displayed in the message view.
2. Select **Online** from the menu bar in Logic Editor, and click **Login** to connect to Logic Processor in COM600. Then
   a. Further prompts should appear to automatically download the application. Click **Yes** and continue.
   b. If not, select **Online Menu** and click **Download** to download PLC Application.

![Login to Logic Processor](Login_to_Lo
3.4. **Making cross-references**

After logging out from the device, you can close the CoDeSys programming environment and launch the Cross References tool to build a cross-reference between the COM600 communication structure and Logic Processor variables.

Please note that changes to configuration needs to be uploaded to COM600. In another words configuration cannot be changed online.

To make cross-references:
1. In SAB600, right-click on Logic Processor IED and select **Cross References**.
2. The Cross References tool reads all the symbols defined in Logic Processor. The variable selected from the CoDeSys symbol configuration is shown in the **OPC Item** column.

![Importing Logic Processor Variables](Importing.Logic_Processor_Variables.png)

*Figure 3.4-1 Importing Logic Processor variables*

The **Import** button can be used for fetching a symbol file from another project.

3. From the communication structure, select a data object and drag and drop it to the proper row in the Cross References tool.
Click the desired Data attribute in the tree, and click Apply to confirm the change change in the table.

5. The Direction field is used for setting the direction of the value transfer, either from data object data attribute to logic variable (->) or from logic variable to data object data attribute (<-).

6. After completing the configuration of cross-reference, click Save to save the changes, and close the Cross References tool.

The following special attribute names can be used in addition to those available in the source data objects:

• With SPS and SPC: attribute EaCnt, that is incremented with 1 each time stVal is updated with the value True. The value is reset to 0 each time stVal is updated with the value False. This allows logic processor to catch several consecutive updates of the object with the value True.

• With SPC: new attribute EiCtlVal with the type Integer (VT_I4). It behaves as ctlVal, so that writing a value > 0 corresponds to writing True to ctlVal. Writing a value <= 0 corresponds to writing False to ctlVal. This allows logic to control multiple times to the same direction, for example, by writing values 1, 2, 3, and so on for True and 0, -1, -2, and so on for False.

3.5. Creating virtual data objects in the Logic Processor IED

Cross references are used to enable communication through OPC DA access, to a PLC Application executing in Logic Processor. Typical cross references scenarios include:

• Transferring status/monitoring data from an OPC communication server (for example DNP/IEC 61850) that is defined in SAB communication structure to a PLC Application to trigger a logic,
and/or,

- Transferring data from a PLC Application to an OPC communication server (for example DNP/IEC6 1850) to initiate a control mechanism to a desired outcome, based on the logic defined in the PLC Application.

In addition to this, the configured symbols (variables within PLC Application) can be exposed to entities outside of Logic Processor for purposes like,

- Configuring a WebHMI element in COM600 SLD to visually display the state of a logic in PLC Application for operator consumption,
- Or,
- Transferring the status of a PLC Application to entities outside of COM600 through slave communication using OPC clients (DNP Slave/ Modbus slave/ IEC 61850 Proxy server), for further SCADA purposes.

This is done using virtual data objects added to Logic Processor IED. These data objects can be configured to point to a symbol defined in PLC Application and exposing the data from PLC Application to outside entities, and vice versa.

To create virtual data objects under Logic Processor IED:
1. Add the desired Logical Device (for example LD1) under Logic Processor IED. Right click **Logic Processor IED**, select **New > Communication > PLC OPC LD**. Rename added object to desired Logical Device name.
2. Add the desired Logical Node (for example LLN0) under Logical Device added in step 1. Right click **Logic Processor IED**, select **New > Communication > PLC OPC LN**. Rename the added object to desired Logical Node name.
3. Add the desired Data object under Logical Node added in step 2. Right click **Logic Processor IED**, select **New > Data Objects > APC/SPC/SPS**. Each data object available has a specific set of attributes as defined in IEC 61850 standard.
4. Modify object properties for data object added in step 3. In the **OPC Item Path** property field, click the browse button with three dots to open the **PLC OPC Item Path** logic variable dialog.
5. Select the symbol available from the list shown and click **OK**.

These virtual data objects can be used with COM600 WebHMI and slave clients.
When communication configuration of IEDs, logic configuration of Logic Processor, and the cross-references between them are done, go to SAB600, select Management on Gateway, and download the configuration to COM600.

Please note that if configuration is updated after downloading configuration to COM600 it has to be downloaded again.

COM600 starts transferring information between the logic application variables and the process data of the connected IEDs.
4. Application example

4.1. Logic requirement

The following is an example of building PLC logic in COM600, and not intended to implement a complete breaker failure protection logic. An example project is provided on the SAB600 installation CD.

Figure 4.1-1 System diagram (Feeder #2 as the source)

In the system shown in Figure 4.1-1 with three breakers, the failure protection plan is the following:
1. Each REF615 has a breaker failure protection. Each REF615 uses DNP 3.0.
2. Assume there is a fault F1 on Feeder #1.
3. Normally, REF615 protecting Feeder #1 should detect the fault and send an OPEN/Trip command to Bkr-2.
4. Assume there is a problem in Bkr-2 (mechanical) and breaker 2 cannot open.
5. The breaker failure protective element issues a signal "Breaker 2 failed".
6. COM600 scans each relay and as soon as it notices a failure operated from breaker 2, it sends a TRIP/OPEN command to Breaker 1 (Bkr-1) and Breaker 3 (Bkr-3). It is assumed that sources are connected via Bkr-1 and Bkr-3, so both will feed a fault F1.
7. COM600 permanently sends a TRIP signal to Bkr-1 and Bkr-3 until it receives a Reset signal. Assume that an external Reset signal is connected to REF615 on Bkr-1 as a binary input. Energizing that binary input, COM600 resets output TRIP signals that are sent to Bkr-1 and Bkr-3.

4.2. Building object tree in SAB600 of three REF615s

To build the object tree:
1. In the communication structure under the Gateway object, add a DNP LAN OPC Server.
2. Under DNP LAN OPC Server, add three DNP LAN Channels.
3. Under channels, add DNP IEDs for the three REF615s.

4. In breaker 1 IED, add a logic node Brk1CSWI1.
   a. Add one DPC data object (Pos) to read and control the switch.
   b. Add one SPS data object (Reset) to read Reset signal.
5. In breaker 2 IED, add a logic node Brk2CSWI1.
a. Add one DPC data object (Pos) to read and control the switch.
b. Add one SPS data object (BreakFailure) to read break failure signal.

6. In breaker 3 IED, add a logic node Brk3CSWI1.
   a. Add one DPC data object (Pos) to read and control the switch.

4.3. Adding Logic Processor IED

To add a Logic Processor IED:
1. In SAB600, under Gateway, add Logic Processor OPC Server.
2. Under Logic Processor OPC Server, add Logic Processor Subnetwork.
3. Add Logic Processor IED.

4.4. Creating logic configuration

To build logic configuration:
1. Right-click on Logic Processor IED in SAB600 and select Logic Editor to launch the CoDeSys programming environment.
2. The logic editor starts with a new project.
3. Select Add object from the Project menu.
4. Select POU on the left side of the Add Object dialog.
5. Enter a name “PLC_BreakerFailureProtection” for the POU and select the Program radio button in the Type section.
6. Select Function Block Diagram (FBD) for the implementation language.
7. Click **Add** to create a new POU.
8. A further Function Block Diagram editor window opens for the new program. The Function Block diagram is a graphically-oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines, which represent either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.
9. To insert an element in the editor, select it in the ToolBox by a mouse-click and by drag and drop it to the editor window. Select **View >Toolbox** to open the ToolBox window, if it is not already open. The **Function Block Dialog ToolBox** is grouped by six catalogs: General, Boolean Operators, Math operators, Other operators, Function blocks, and POUs.
10. Insert a rising edge detector to catch the breaker failure signal. Select R_TRIG, a rising edge detection function block from the Function blocks catalog, and drag and drop it to the Function Block Diagram editing area.

11. When the function block is first inserted, characters '???' are shown in the input, output, and object tags. Replace the '???' with a new variable name and press the <Return> key. The Auto Declare dialog opens.

12. In the Auto Declare dialog, the variable name, and scope are filled in automatically.

13. Enter the desired type and initialization value, and the declaration code is displayed in the declaration part of the editor.

14. Click OK.

The new variable is added to the declaration part of the Function Block Diagram.
In the declaration part of the diagram, the br2Failure variable is defined as a Boolean variable and initialized to FALSE.

15. Add a second function block in the ToolBox by selecting the RS (Reset Set function block) from the Boolean Operators catalog and dragging and dropping it to the output point of the R TRIG object.

The logic can be built as follows.
In this Function Block Diagram, the first row is to catch the rising edge of breaker 2 failure and check Reset signals.

The second and third row are to generate an On/Off square wave (Tripping signal) from a break failure status by using Timer Pulse, Timer and XOR gate function blocks.

The last row is to send generated Tripping signal to open the breakers. A Timer Delay is added between the Select Off and Operator outputs to make sure it is a Select/Operate operation.

When it receives a Reset signal, the logic sets all signals to Off and stops sending trip signal to IEDs.

### Adding POU to Application MainTask

To add the PLC_Breaker Failure POU to MainTask:

1. Remove the default PLC_PRG from under Application tree by right-clicking it and selecting **Delete**.
2. Expand the Task Configuration under Application Tree. Double-click **MainTask** to open the MainTask Editor.
3. In the MainTask Editor, select **PLC_PRG POU** and click **Remove Call**. PLC_PRG POU is removed from MainTask.
4. Click **Add Call** and select **PLC_BreakerFailureProtection** POU to add it to Logic Processor Main Task list.
4.6. Add Symbol Configuration

To configure symbols from POU:

1. Right-click Application in the Device Tree. Select Add Object > Symbol Configuration.
2. A Symbol Configuration object is added to Application and Symbol Configuration Editor opens.
3. To get the currently available item pool, click Build. Resolve any errors.
4. Add variables from POU to Application Symbol list by simply clicking the checkbox next to each of the variables and assigning appropriate access rights. Select variable,
   a. br2Failure, and set access right to “read”
   b. br1SelectTrip, and set access right to “write”
   c. br1Trip, and set access right to “write”
   d. br2SelectTrip, and set access right to “write”
   e. br2Trip, and set access right to “write”
   f. br3SelectTrip, and set access right to “write”
   g. br3Trip, and set access right to “write”
   h. reset, and set access right to “read”
4.7.

Setting device path

To setup Logic Editor to communicate with Logic Processor in COM600:
1. Double-click Device in the Devices Window. The Device Editor opens showing Communication Settings tab.
2. Select Gateway drop-down button, and click Manage Gateways.
3. Add gateway pointing to IP address of COM600 if not available already.
4. Click Scan Network. The Select Device dialog opens. This dialog shows the COM600 available at the specified IP address.
5. Select item with the COM600 computer name and click OK. This should set the active path for Logic Editor to communication with Logic Processor in COM600.
6. Once the active path is set, the Device Communication Settings page should show up as below. Further downloads can be accomplished by login to device.
4.8. Building and activating the application

1. Select **Build > Build** to compile the project.
   All objects belonging to the application is syntactically checked. Any potential error messages or warnings are displayed within the message view.

2. If the project is built successfully without error, select **Online > Login** to connect to the currently active application to the target device PLC and thus changes into the online mode.
   If there is no application running on target device before Login, the current active application is downloaded to the device.

3. Select **Debug > Start** to start the program on the PLC.
   The program starts running.

4. Select **Online > Logout** to log out from the device and **File > Exit** to close the logic processor configuration tool.

4.9. Making cross-references

After logging out from the device, close the Logic Editor and launch the Cross References tool in SAB to build further cross-references between the COM600 communication structure and Logic Processor variables.

To make cross-references:

1. In SAB600, right click **Logic Processor IED** and select **Cross References**.
2. The Cross References tool reads all the symbols defined in Logic Processor. The variable selected from the Logic Editor Symbol Configuration is show in the OPC Item column.
3. From the communication structure, select the appropriate data objects and drag and drop to each of the symbols available. Once a data object is dragged and dropped on to a row for a specific symbol, the **Select Data Attribute** dialog opens, which can then be used to select appropriate data attribute available under a specific data object. This way the communication data object representing the symbol will be connected with each other for a correct data flow. The Direction column allows to configure if the data needs to be read from SAB Communication in to a Symbol, or vice versa.

4. After completing the configuration of cross-references, click **Save** to save the setting, and close the cross references tool.
Index

A
activating .................................................................................................................. 22

B
building
  communication structure .................................................................................. 15

C
CoDeSys ........................................................................................................... 16
CoDeSys SP runtime .......................................................................................... 13
cross-reference ............................................................................................. 24, 37

L
logic ................................................................................................................... 30
Logic Processor
  CoDeSys SP runtime .................................................................................. 12
logic programming
  logic editor .................................................................................................. 13

M
monitoring and debugging ................................................................................ 23

V
variable
  OPC symbol .................................................................................................. 17
virtual data object
  variable ....................................................................................................... 25