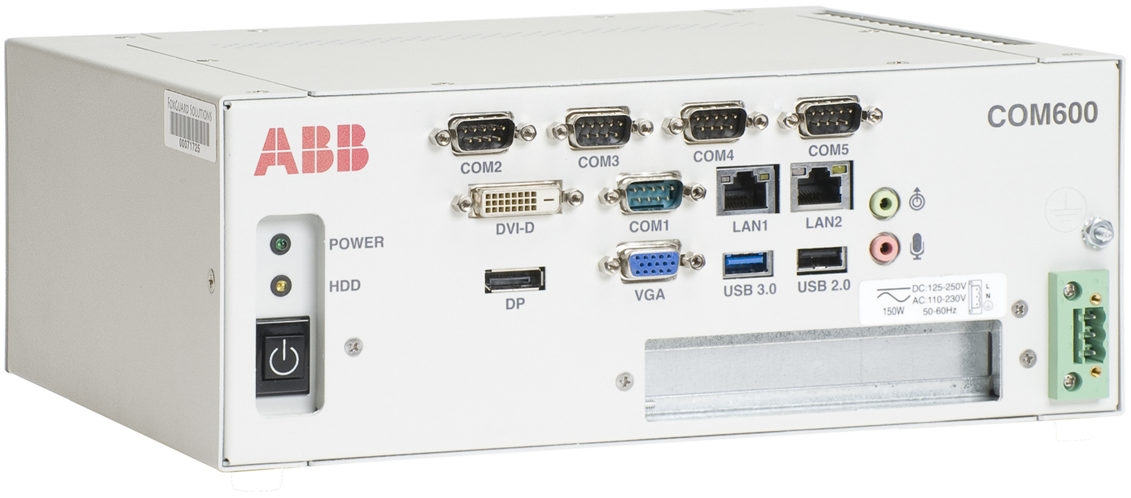
COM600 series 5.1

Logic Processor User's Manual



Contents:

1. About this manual

1.1. Copyright

Trademarks

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

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.

Term	**Description**
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
LAN	Local Area Network
LD	Logical Device
LN	Logical Node
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
XML	eXtended Markup Language

1.10. Related documents

Name of the manual	**MRS number**
COM600 User’s Manual	1MRS756125
COM600 Operator's Manual	1MRS756705
COM600 HMI Configuration Manual	1MRS756740
COM600 Data Historian Operator's Manual	1MRS756739
COM600 Sequence Control Configuration Manual	1MRS755001
Master Protocols (Ethernet based) Configuration and Operation Manual	1MRS758689
Master Protocols (Ethernet based) Technical Reference Manual	1MRS758690
Slave Protocols (Ethernet based) Configuration and Operation Manual	1MRS758691
Slave Protocols (Ethernet based) Technical Reference Manual	1MRS758692
DNP 3.0 Serial Master (OPC) User's Manual	1MRS756567
DNP 3.0 Serial Slave (OPC) User's Manual	1MRS755495
IEC 60870-5-101 Slave (OPC) User's Manual	1MRS755382
IEC 60870-5-101 Master (OPC) User's Manual	1MRS756703
IEC 60870-5-103 Master (OPC) User's Manual	1MRS752278
COM600 Logic Processor User's Manual	1MRS756738
Modbus Serial Master (OPC) User's Manual	1MRS756126
Modbus Serial Slave (OPC) User's Manual	1MRS756913

1.11. Document revisions

Document version/date	**Product revision**	**History**
A/13.2.2009	3.3	Document created
B/06.11.2009	3.4	Document revised
C/30.06.2011	3.5	Document revised
D/31.5.2012	4.0	Document revised
E/13.3.2015	4.1	Document revised
F/24.5.2017	5.0	Document revised
G/6.3.2018	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. Functional overview

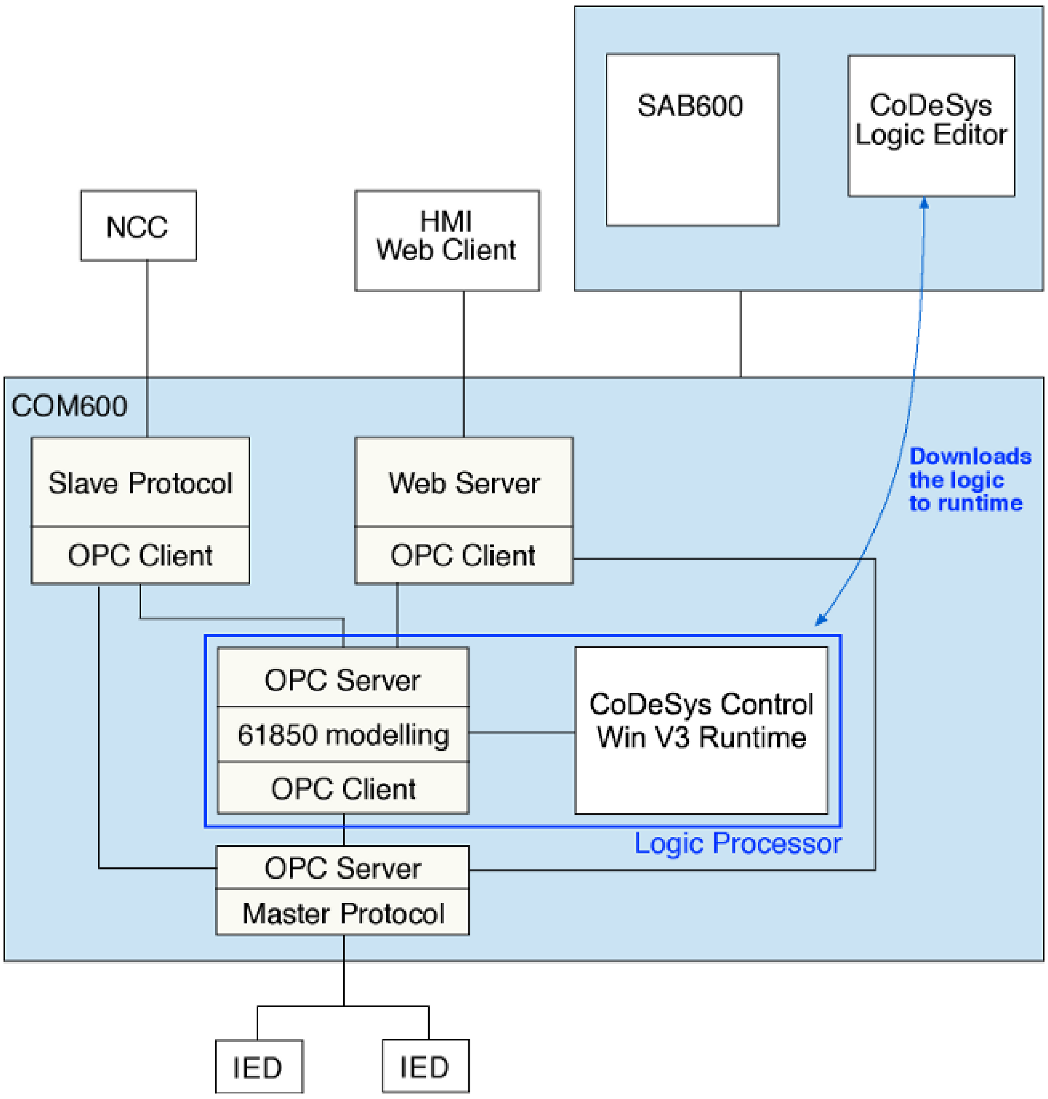
The Logic Processor function enables developing customer-specific automated applications for COM600. Applications are programmed using the IEC 61131-3 logic editor (CoDeSys). The COM600 unit has the logic engine (CoDeSys Control Win V3 Runtime), which can load and execute the created IEC 61131-3 applications. 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 inteval 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.



Logic Processor.bmp

Figure 2.3-1 Functional overview of Logic Processor

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)

- Structured Text (ST)
- Instruction List (IL)

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.

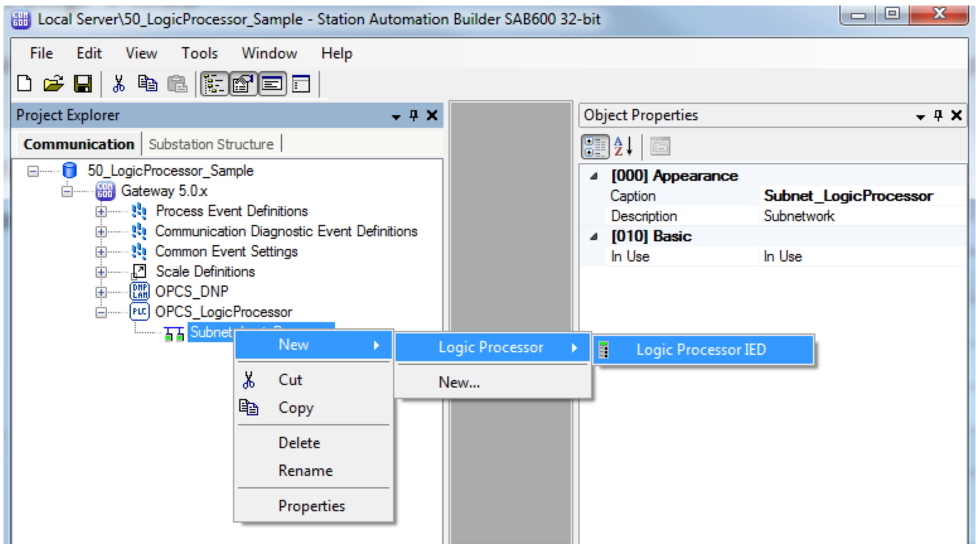
The Logic Processor configuration in COM600 can be divided into the following tasks:

1. Building Communication Structure objects for the Logic Processor.
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 Subnetwork**.
4. Right-click the Logic Processor Subnetwork object and add **Logic Processor IED**.



Add_Logic_Processor_IED.bmp

Figure 3.2-1 Adding a Logic Processor IED

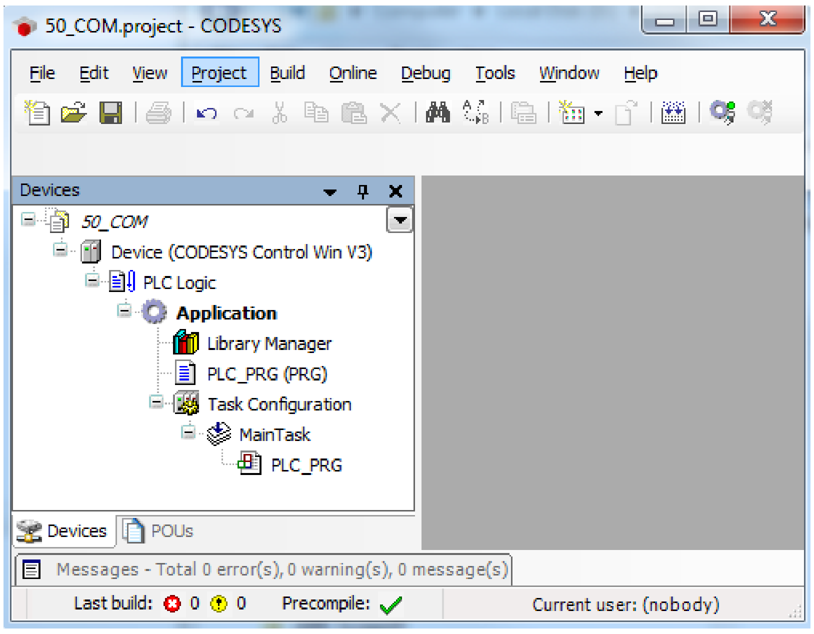
3.3. Logic editor

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.


50_COM.project - CODESYS
File Edit View Project Build Online Debug Tools Window Help
Devices
50_COM
Device (CODESYS Control Win V3)
PLC Logic
Application
Library Manager
PLC_PRG (PRG)
Task Configuration
MainTask
PLC_PRG
Devices
POUs
Messages - Total 0 error(s), 0 warning(s), 0 message(s)
Last build: 0 0 Precompile:
Current user: (nobody)

Default_POU.bmp

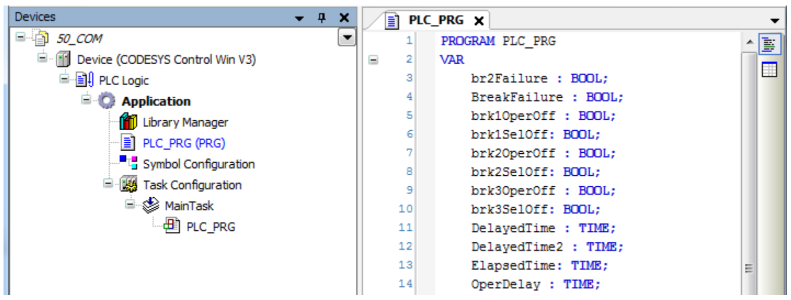
Figure 3.3.1-1 Default POU in the CoDeSys project

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.



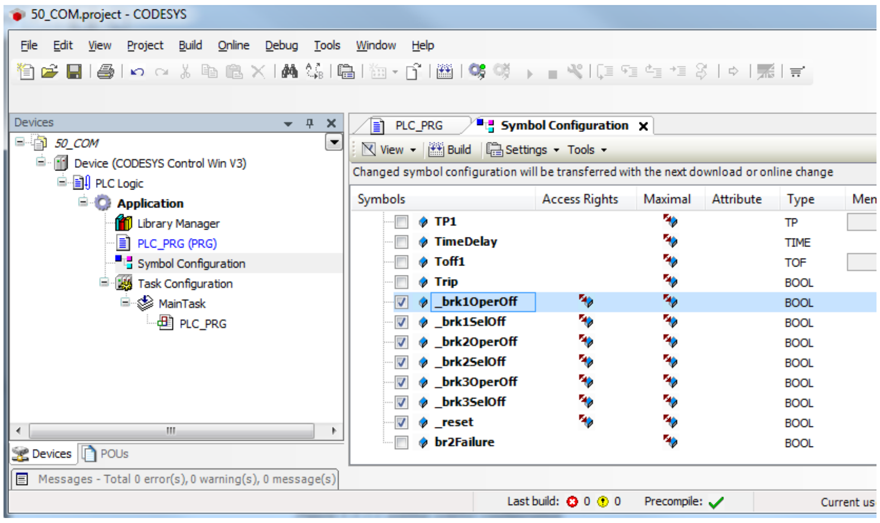
```
PROGRAM PLC_PRG
VAR
    br2Failure : BOOL;
    BreakFailure : BOOL;
    brk1OperOff : BOOL;
    brk1SelOff: BOOL;
    brk2OperOff : BOOL;
    brk2SelOff: BOOL;
    brk3OperOff : BOOL;
    brk3SelOff: BOOL;
    DelayedTime : TIME;
    DelayedTime2 : TIME;
    ElapsedTime: TIME;
    OperDelay : TIME;
```

Sample_PLC_PRG_POU.png

Figure 3.3.2-1 Sample PLC_PRG POU with variables

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.



Changed symbol configuration will be transferred with the next download or online change

Messages - Total 0 error(s), 0 warning(s), 0 message(s)

Symbols	Access Rights	Maximal	Attribute	Type	Men
☐ TP1		⇄		TP	
☐ TimeDelay		⇄		TIME	
☐ Toff1		⇄		TOF	
☐ Trip		⇄		BOOL	
✓ _brk1OperOff	⇄	⇄		BOOL	
✓ _brk1SelOff	⇄	⇄		BOOL	
✓ _brk2OperOff	⇄	⇄		BOOL	
✓ _brk2SelOff	⇄	⇄		BOOL	
✓ _brk3OperOff	⇄	⇄		BOOL	
✓ _brk3SelOff	⇄	⇄		BOOL	
✓ _reset	⇄	⇄		BOOL	
☐ br2Failure		⇄		BOOL	

Symbol_Configuration_PLC.png

Figure 3.3.2-2 Symbol Configuration

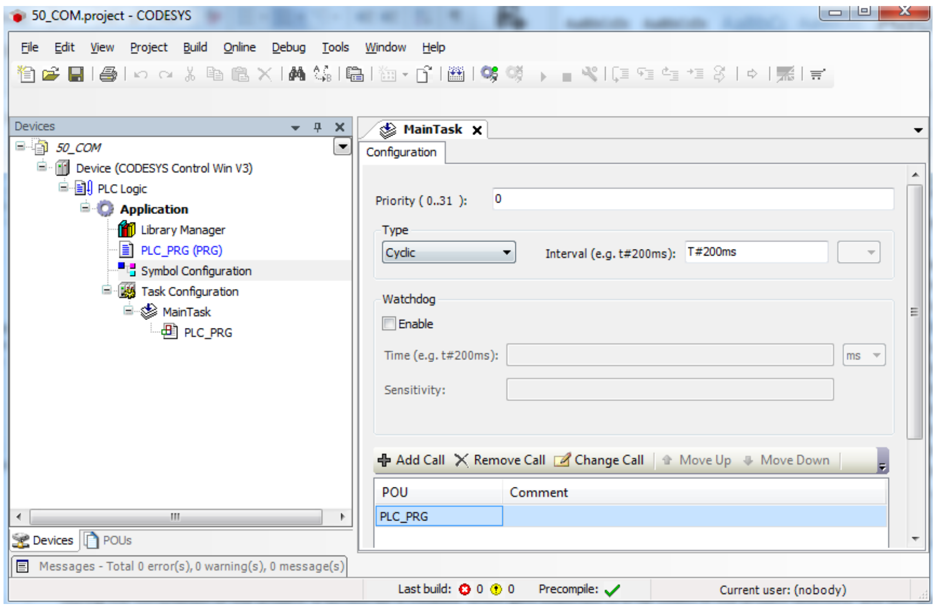
During the compilation of the project, a symbol list is created and it is exported to a file (XML) in the project directory, and also loaded to the device during the application download in the form of a child application.

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.



50_COM.project - CODESYS

Devices: 50_COM > Device (CODESYS Control Win V3) > PLC Logic > Application > Library Manager, PLC_PRG (PRG), Symbol Configuration, Task Configuration > MainTask > PLC_PRG

MainTask — Configuration

Priority (0..31): 0

Type: Cyclic — Interval (e.g. t#200ms): T#200ms

Watchdog: Enable — Time (e.g. t#200ms): — Sensitivity:

POU	Comment
PLC_PRG	

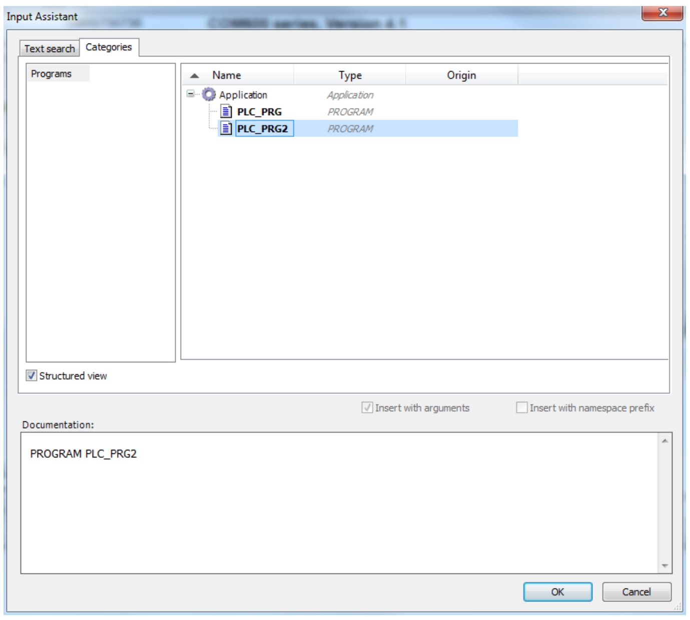
Messages - Total 0 error(s), 0 warning(s), 0 message(s)

Last build: 0 0 Precompile: ✓ Current user: (nobody)

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 Assitant** 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.



PROGRAM PLC_PRG2

Selecting_POU_MainTask.png

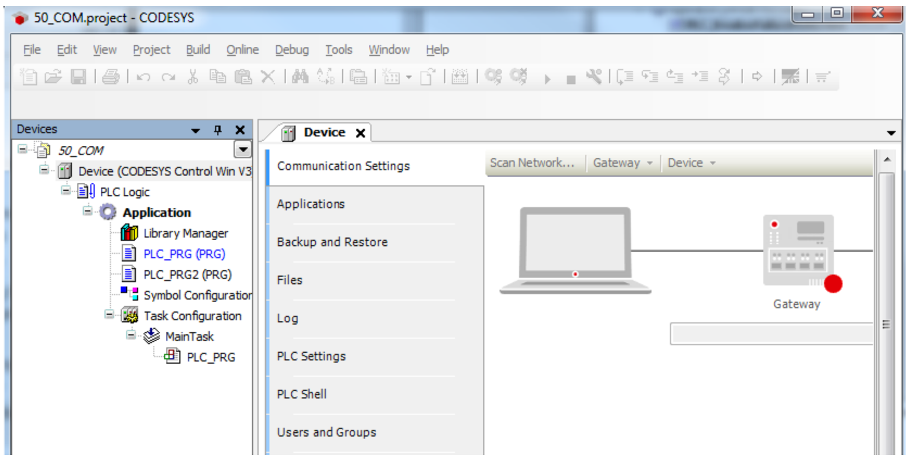
Figure 3.3.3-2 Selecting POU to add to main task

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:

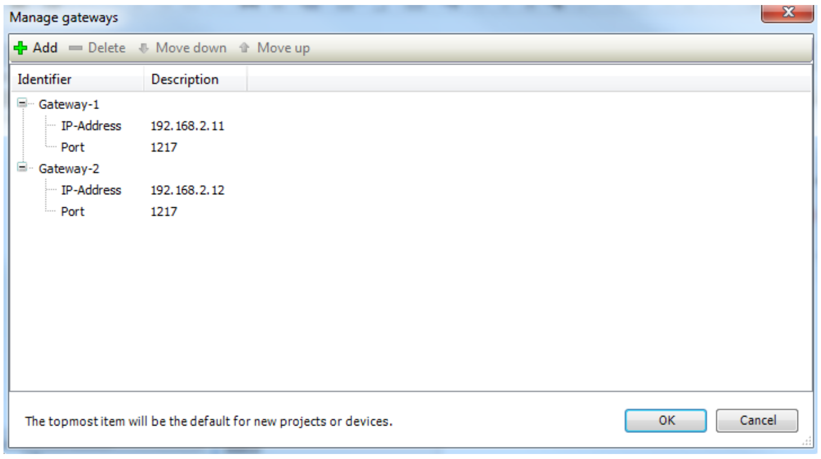
1. Double-click the **Device** object available in **Logic Editor**. The configuration editor for Device opens.



Device_Configuration_Editor.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 ….**



Identifier	Description
Gateway-1	
IP-Address	192.168.2.11
Port	1217
Gateway-2	
IP-Address	192.168.2.12
Port	1217

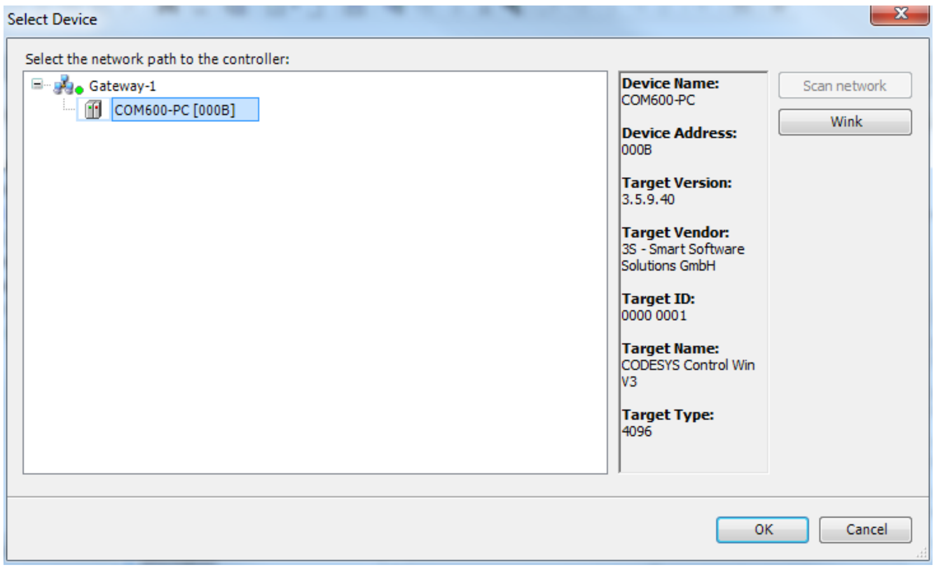
The topmost item will be the default for new projects or devices.

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.

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.



Select Device

Select the network path to the controller:

Gateway-1
COM600-PC [000B]

Device Name:
COM600-PC

Device Address:
000B

Target Version:
3.5.9.40

Target Vendor:
3S - Smart Software Solutions GmbH

Target ID:
0000 0001

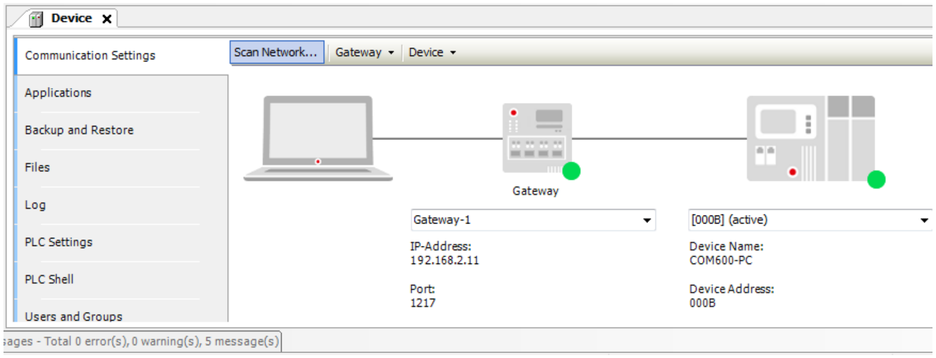
Target Name:
CODESYS Control Win V3

Target Type:
4096

Select_Device_Active_Path.png

Figure 3.3.4-3 Select Device Active Path to 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.



Gateway-1
IP-Address:
192.168.2.11
Port:
1217

[000B] (active)
Device Name:
COM600-PC
Device Address:
000B

sages - Total 0 error(s), 0 warning(s), 5 message(s)

Device_Communication_Settings_Active_Path.png

Figure 3.3.4-4 Device Communication Settings after setting active path

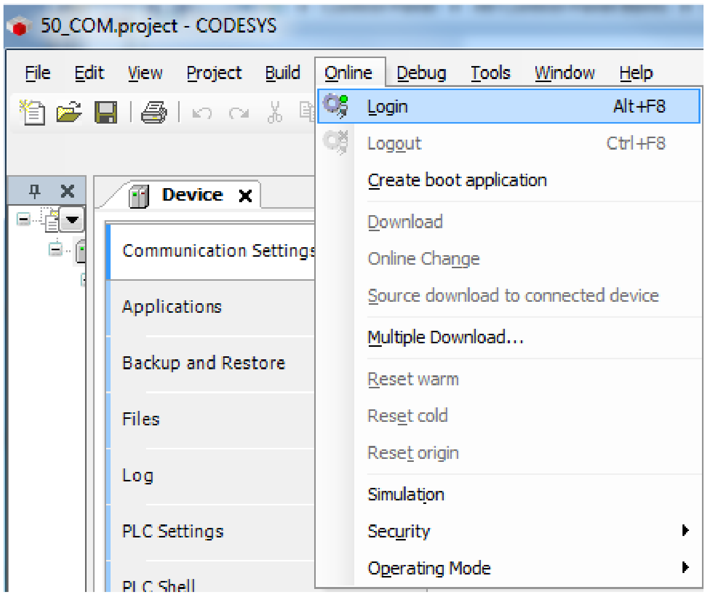
3.3.5. Downloading and starting PLC application

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.png

Figure 3.3.5-1 Login to Logic Processor

3. After successful download of PLC Application, start the application so that it gets into running/executing state. To start PLC Application, select **Debug** from the menu bar in Logic Editor, and click **Start**. After PLC Application has started, the status bar in Logic Editor shows RUN in green.

Once PLC Application has been successfully downloaded and is executing in Logic Processor, the connection from Logic Editor can be terminated by logging out of the Device. This should not affect the running state of the PLC Application in Logic Processor.

To terminate the Logic Editor session and close it:
1. Select **Online** from the menu bar in Logic Editor, and click **Logout** to disconnect Logic Editor from Logic Processor.
2. Close Logic Editor Application by clicking the window close button. Save Project on close if prompted.

3.3.6. Logic Editor Online Help

Describing fully the use of Logic Editor (CoDeSys Development System) and all of its functionality is outside the scope of this document. When in doubt, resort to the online help available in Logic Editor. To launch online help in Logic Editor, select **Help** from the menu bar in Logic Editor, and click **Contents**.

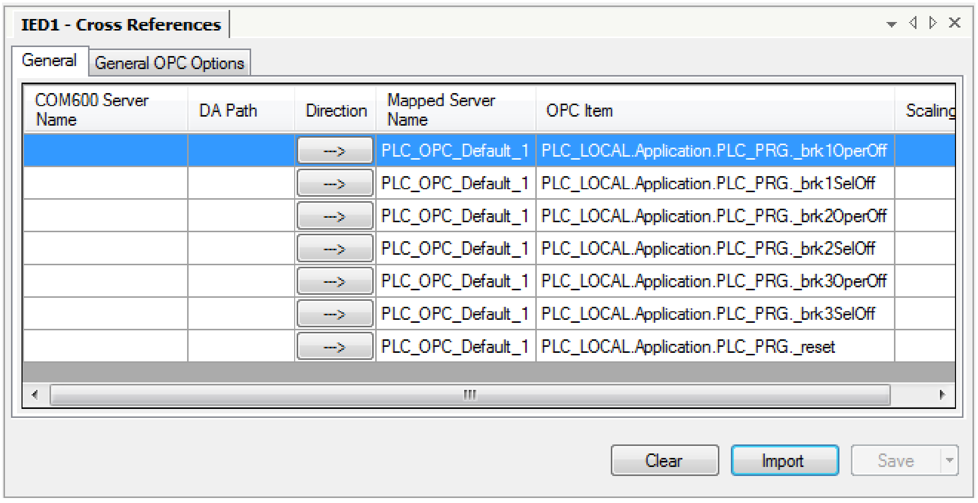
3.4. Making cross-references

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.



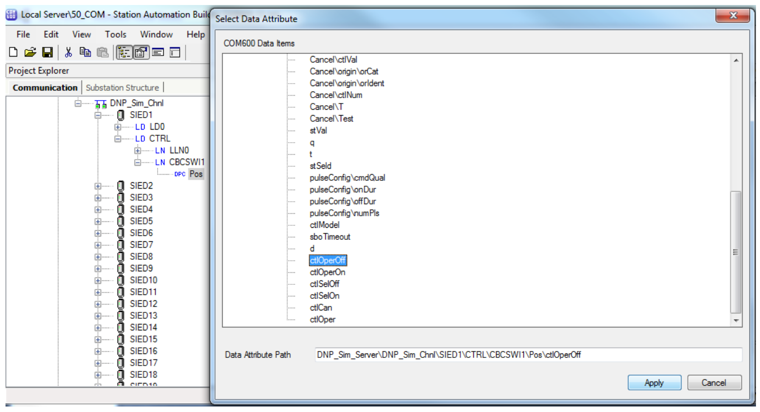
COM600 Server Name	DA Path	Direction	Mapped Server Name	OPC Item	Scaling
		-->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_PRG._brk1OperOff	
		-->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_PRG._brk1SelOff	
		-->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_PRG._brk2OperOff	
		-->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_PRG._brk2SelOff	
		-->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_PRG._brk3OperOff	
		-->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_PRG._brk3SelOff	
		-->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_PRG._reset	

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.



Data Attribute Path DNP_Sim_Server\DNP_Sim_Chnl\SIED1\CTRL\CBCSWI1\Pos\ctlOperOff

Adding_Cross-references.png

Figure 3.4-2 The Cross references tool

4. 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,

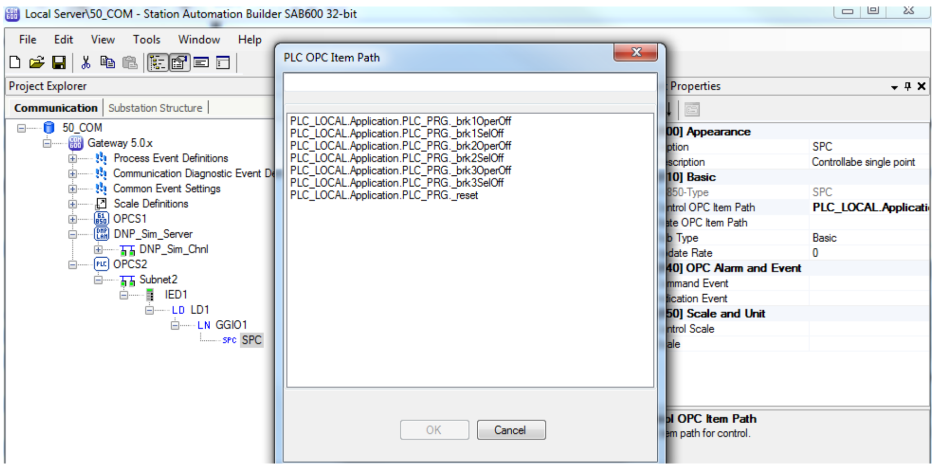
- Configuring a WebHMI element in COM600 SLD to visually display the state of a logic in PLC Application for operator consumption,
  Or,
- Transferring the status of a PLC Application to entities outside of COM600 through slave communication using OPC clients (DNP Slave/ Modbus slave/ IEC 61850 Proxy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


Local Server\50_COM - Station Automation Builder SAB600 32-bit
File Edit View Tools Window Help
Project Explorer
Communication | Substation Structure
50_COM
Gateway 5.0.x
Process Event Definitions
Communication Diagnostic Event De
Common Event Settings
Scale Definitions
OPCS1
DNP_Sim_Server
DNP_Sim_Chnl
OPCS2
Subnet2
IED1
LD LD1
LN GGIO1
SPC SPC
PLC OPC Item Path
OK
Cancel
Properties

```
PLC_LOCAL.Application.PLC_PRG._brk1OperOff
PLC_LOCAL.Application.PLC_PRG._brk1SelOff
PLC_LOCAL.Application.PLC_PRG._brk2OperOff
PLC_LOCAL.Application.PLC_PRG._brk2SelOff
PLC_LOCAL.Application.PLC_PRG._brk3OperOff
PLC_LOCAL.Application.PLC_PRG._brk3SelOff
PLC_LOCAL.Application.PLC_PRG._reset
```

00] Appearance	
ption	SPC
scription	Controllabe single point
10] Basic	
850-Type	SPC
ntrol OPC Item Path	**PLC_LOCAL.Applicati**
te OPC Item Path	
b Type	Basic
date Rate	0
40] OPC Alarm and Event	
mmand Event	
ication Event	
50] Scale and Unit	
ntrol Scale	
ale	

l OPC Item Path
m path for control.

Assigning_symbols_to_Virtual_Data_Objects.png

Figure 3.5-1 Assigning symbols to Virtual Data Objects

3.6. Downloading Logic Processor OPC Server configuration

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.

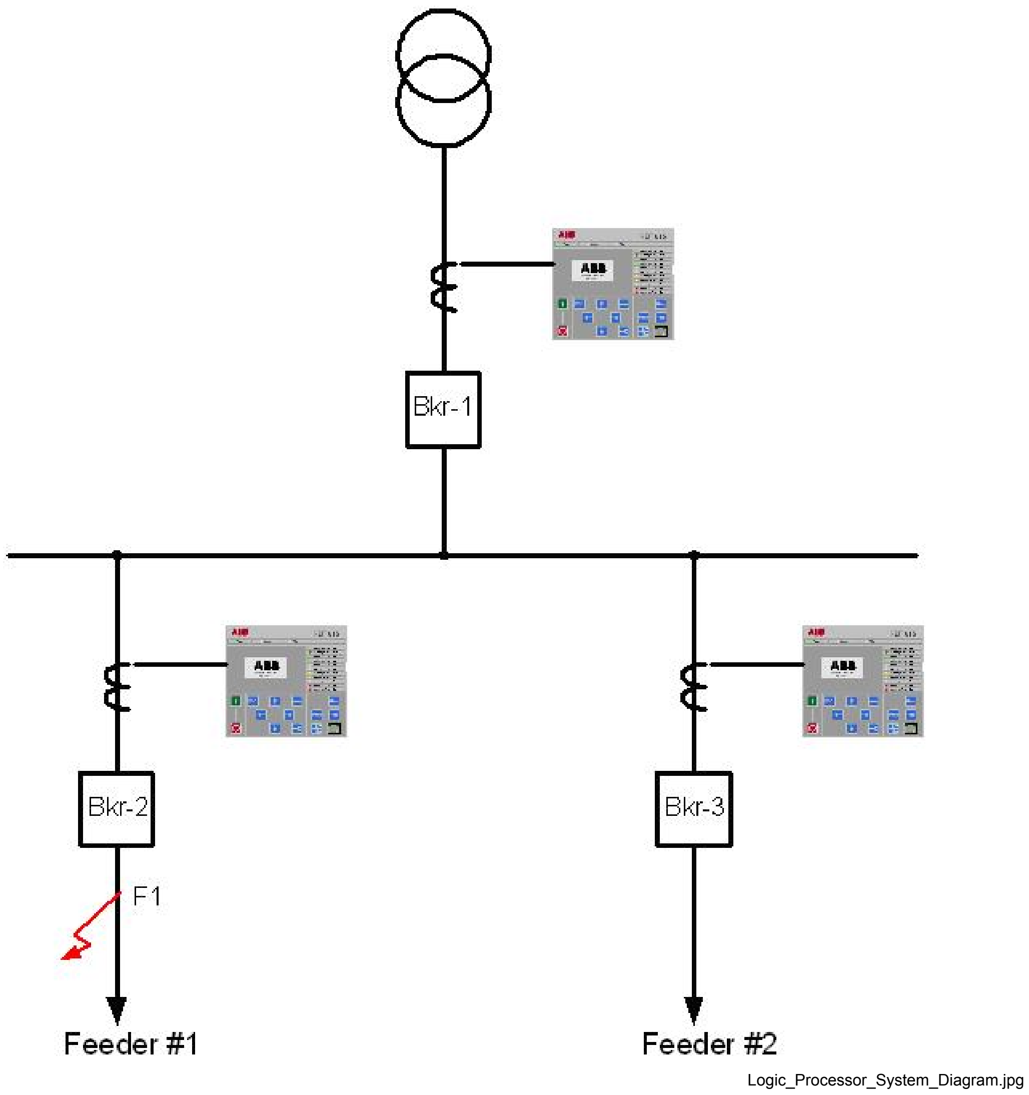


Figure 4.1-1 System diagram (Feeder #2 as the source)

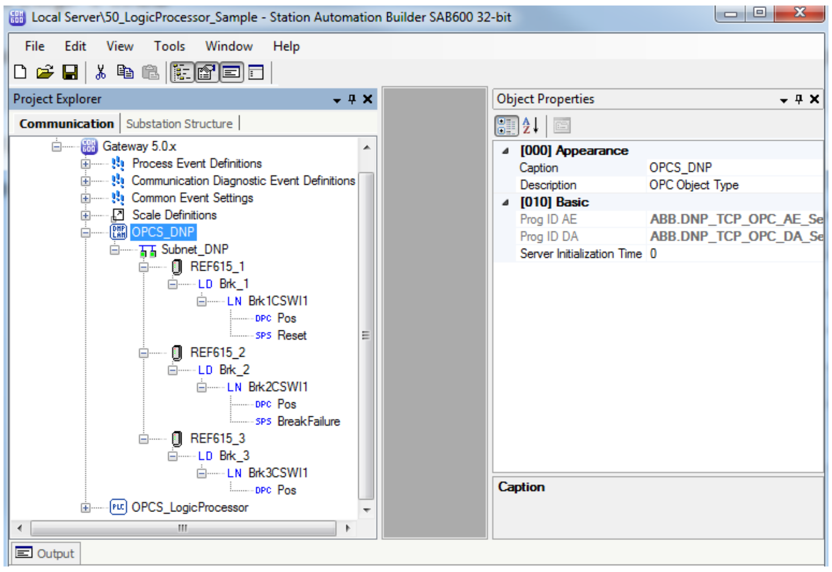
In the system shown in Figure 4.1-1 with three breakers, the failure protection plan is the following:

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.



```
Gateway 5.0.x
    Process Event Definitions
    Communication Diagnostic Event Definitions
    Common Event Settings
    Scale Definitions
    OPCS_DNP
        Subnet_DNP
            REF615_1
                LD Brk_1
                    LN Brk1CSWI1
                        DPC Pos
                        SPS Reset
            REF615_2
                LD Brk_2
                    LN Brk2CSWI1
                        DPC Pos
                        SPS BreakFailure
            REF615_3
                LD Brk_3
                    LN Brk3CSWI1
                        DPC Pos
    OPCS_LogicProcessor
```

[000] Appearance	
Caption	OPCS_DNP
Description	OPC Object Type
[010] Basic	
Prog ID AE	ABB.DNP_TCP_OPC_AE_Se
Prog ID DA	ABB.DNP_TCP_OPC_DA_Se
Server Initialization Time	0

Caption

Configuring_IEDs.png

Figure 4.2-1 Configuring IEDs on SAB600

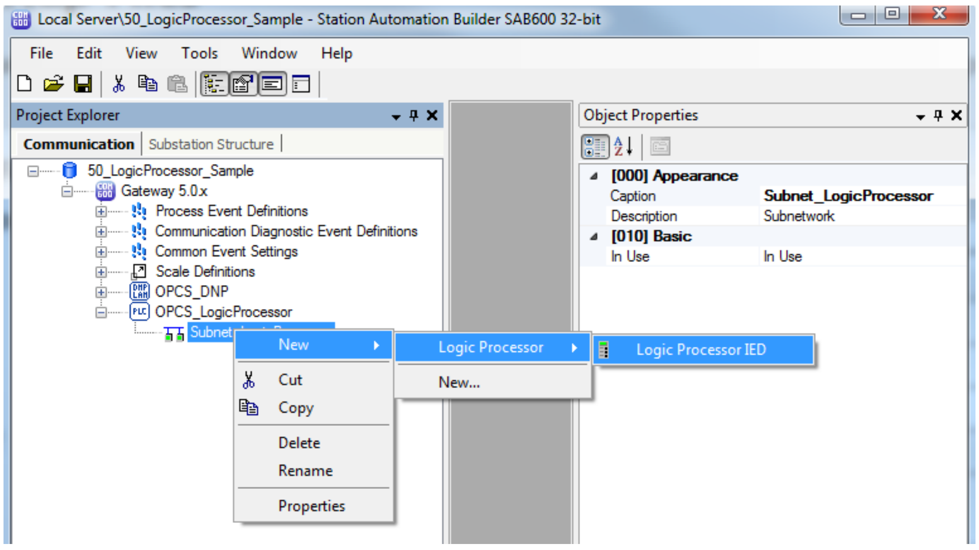
4. In breaker 1 IED, add a logic node Brk1CSWI1.
    a. Add one DPC data object (Pos) to read and control the switch.
    b. Add one SPS data object (Reset) to read Reset signal.
5. In breaker 2 IED, add a logic node Brk2CSWI1.

   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.



[000] Appearance	
Caption	Subnet_LogicProcessor
Description	Subnetwork
[010] Basic	
In Use	In Use

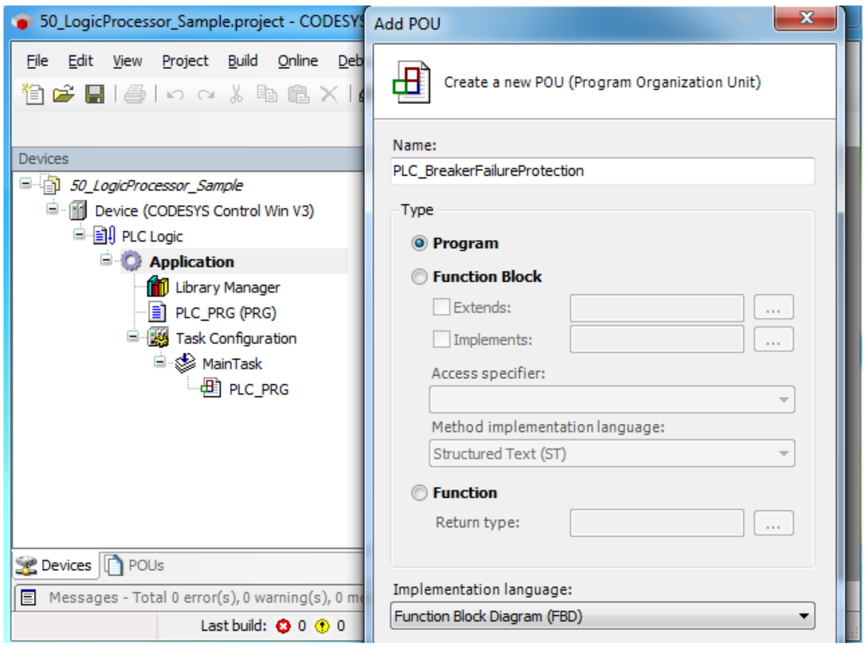
Add_Logic_Processor_IED.bmp

Figure 4.3-1 Adding 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.



Messages - Total 0 error(s), 0 warning(s), 0 m

Add POU

Create a new POU (Program Organization Unit)

Name: PLC_BreakerFailureProtection

Type

[✓ Program]

Function Block

Extends:

Implements:

Access specifier:

Method implementation language: Structured Text (ST)

Function

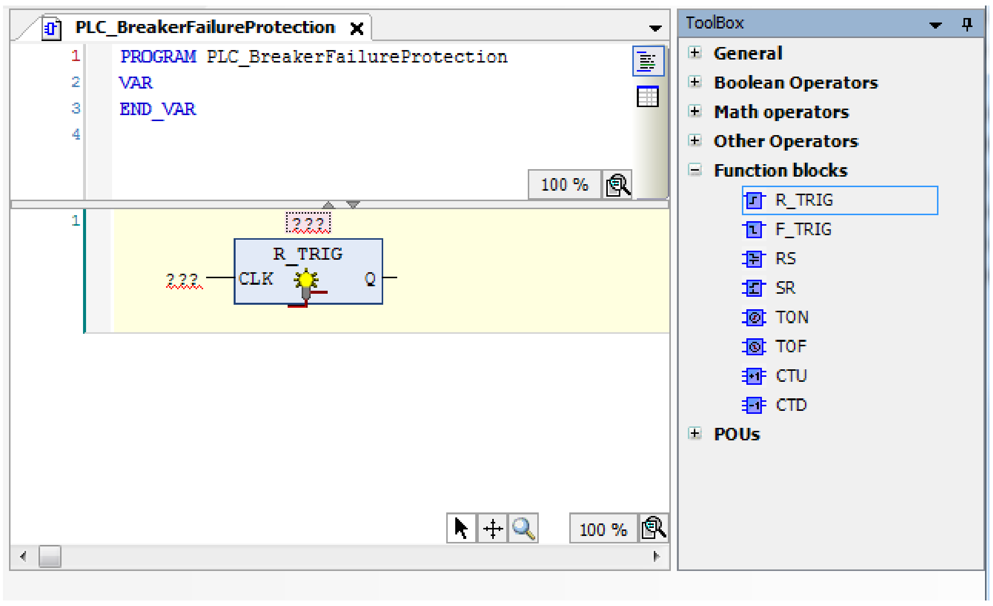
Return type:

Implementation language: Function Block Diagram (FBD)

Adding_Function_Block_Diagram.png

Figure 4.4-1 Adding a function block diagram

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.

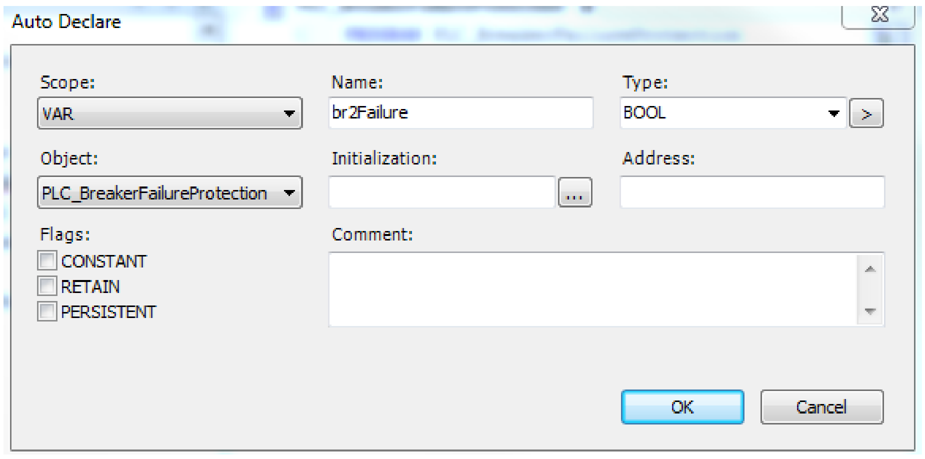


```
PROGRAM PLC_BreakerFailureProtection
VAR
END_VAR
```

Adding_R_TRIG.png

Figure 4.4-2 Adding R TRIG

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.



Auto Declare

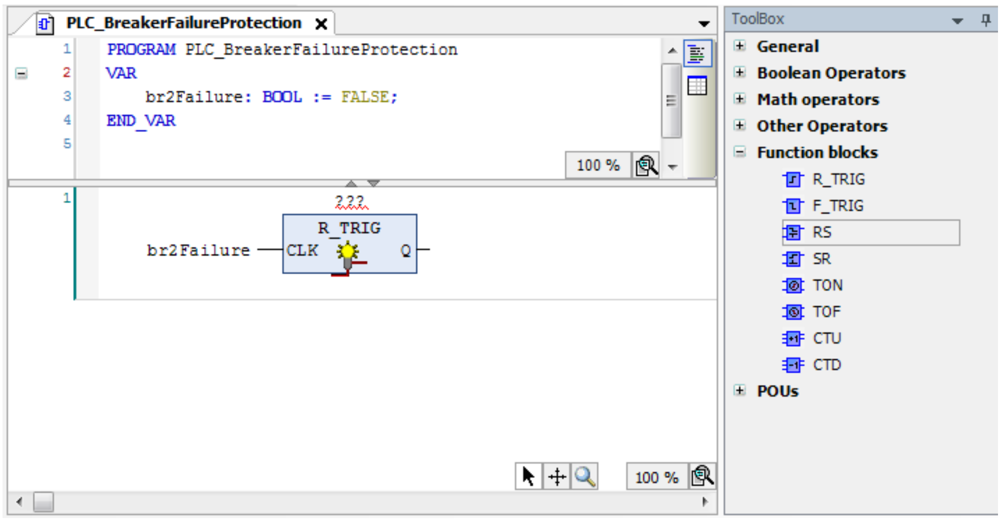
Scope: VAR
Name: br2Failure
Type: BOOL
Object: PLC_BreakerFailureProtection
Initialization:
Address:
Flags:
- CONSTANT
- RETAIN
- PERSISTENT

Comment:

Auto_Declare_Dialog.png

Figure 4.4-3 Auto Declare Dialog

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**.



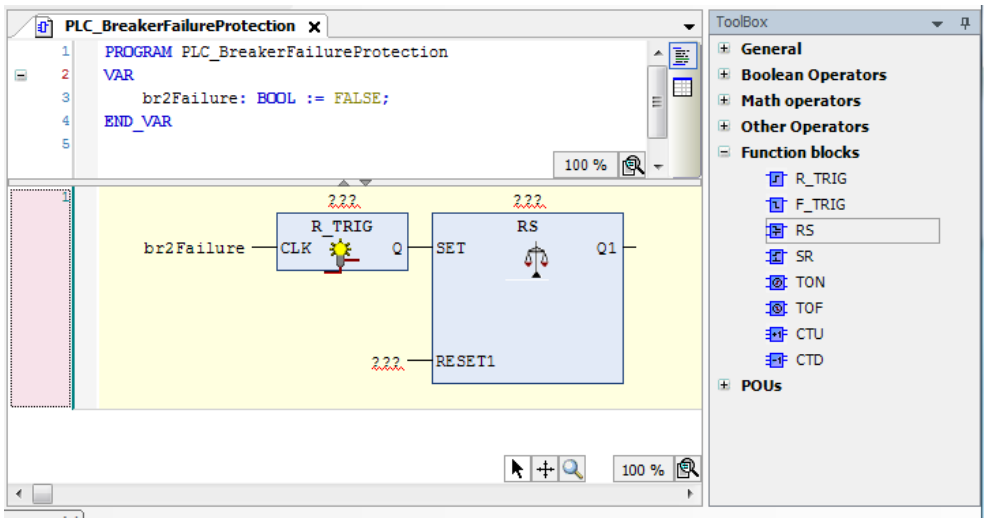
```
PROGRAM PLC_BreakerFailureProtection
VAR
    br2Failure: BOOL := FALSE;
END_VAR
```

Declaration_Part_FBD.png

Figure 4.4-4 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.

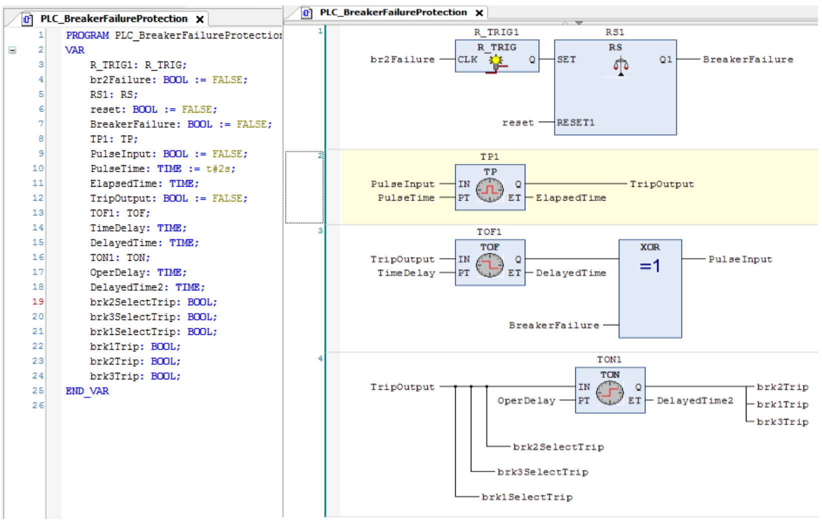


```
PROGRAM PLC_BreakerFailureProtection
VAR
    br2Failure: BOOL := FALSE;
END_VAR
```

Adding_AND_gate.bmp

Figure 4.4-5 Adding Reset Set function block

The logic can be built as follows.



```
PROGRAM PLC_BreakerFailureProtectio
VAR
    R_TRIG1: R_TRIG;
    br2Failure: BOOL := FALSE;
    RS1: RS;
    reset: BOOL := FALSE;
    BreakerFailure: BOOL := FALSE;
    TP1: TP;
    PulseInput: BOOL := FALSE;
    PulseTime: TIME := t#2s;
    ElapsedTime: TIME;
    TripOutput: BOOL := FALSE;
    TOF1: TOF;
    TimeDelay: TIME;
    DelayedTime: TIME;
    TON1: TON;
    OperDelay: TIME;
    DelayedTime2: TIME;
    brk2SelectTrip: BOOL;
    brk3SelectTrip: BOOL;
    brk1SelectTrip: BOOL;
    brk1Trip: BOOL;
    brk2Trip: BOOL;
    brk3Trip: BOOL;
END_VAR
```

Logic_trip.bmp

Figure 4.4-6 Failure Protection Logic

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.

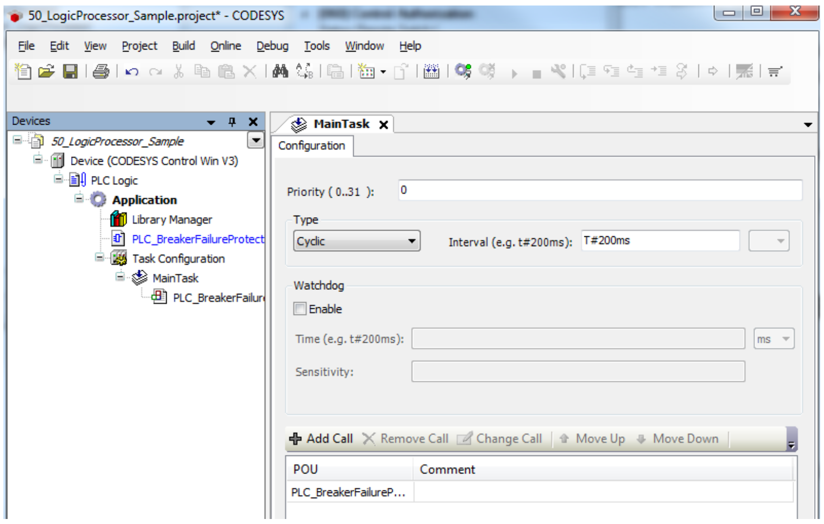
The last row is to send generated Tripping signal to open the breakers. A Timer Delay is added between the Select Off and Operator outputs to make sure it is a Select/Operate operation.

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

4.5. 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.



Priority (0..31): 0

Type: Cyclic — Interval (e.g. t#200ms): T#200ms

Watchdog: Enable

Time (e.g. t#200ms): ms

Sensitivity:

POU	Comment
PLC_BreakerFailureP...	

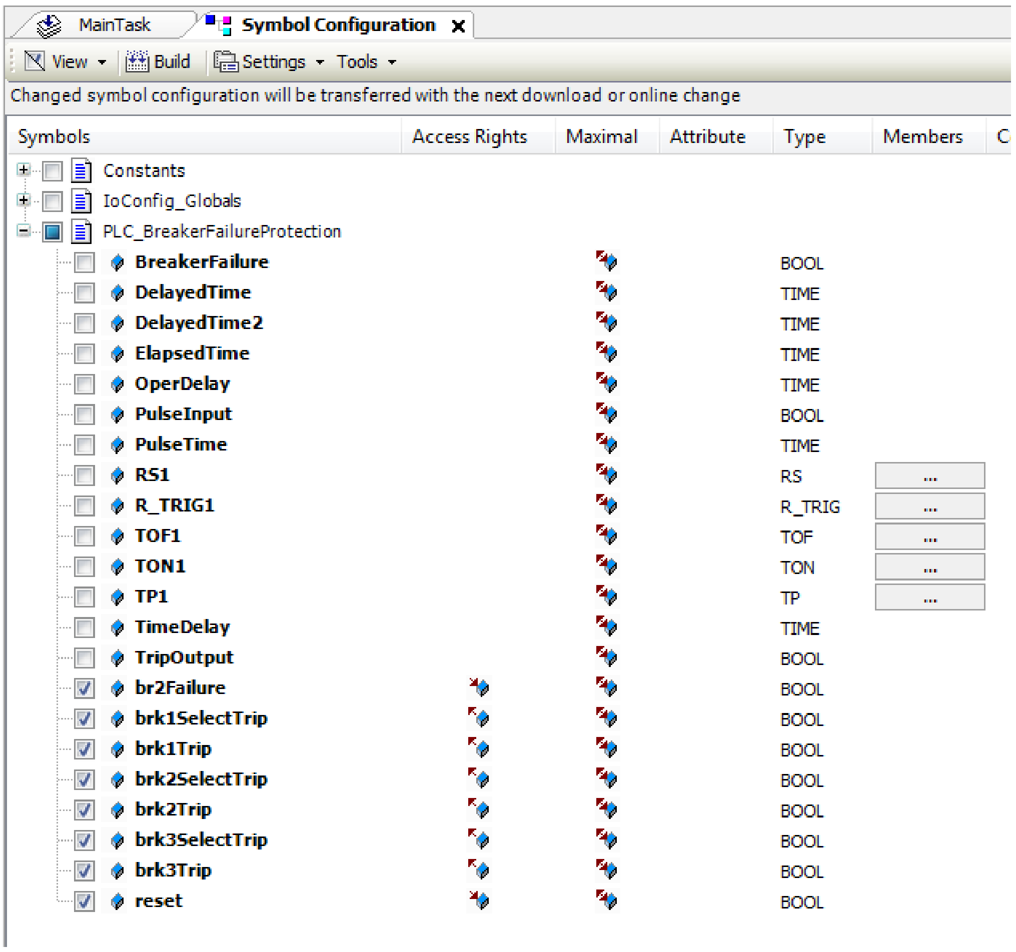
Add_POU_to_MainTask.png

Figure 4.5-1 Adding POU to MainTask

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”



Changed symbol configuration will be transferred with the next download or online change

Symbols	Access Rights	Maximal	Attribute	Type	Members	C
☐ Constants						
☐ IoConfig_Globals						
■ PLC_BreakerFailureProtection						
☐ BreakerFailure				BOOL		
☐ DelayedTime				TIME		
☐ DelayedTime2				TIME		
☐ ElapsedTime				TIME		
☐ OperDelay				TIME		
☐ PulseInput				BOOL		
☐ PulseTime				TIME		
☐ RS1				RS	...	
☐ R_TRIG1				R_TRIG	...	
☐ TOF1				TOF	...	
☐ TON1				TON	...	
☐ TP1				TP	...	
☐ TimeDelay				TIME		
☐ TripOutput				BOOL		
☑ br2Failure				BOOL		
☑ brk1SelectTrip				BOOL		
☑ brk1Trip				BOOL		
☑ brk2SelectTrip				BOOL		
☑ brk2Trip				BOOL		
☑ brk3SelectTrip				BOOL		
☑ brk3Trip				BOOL		
☑ reset				BOOL		

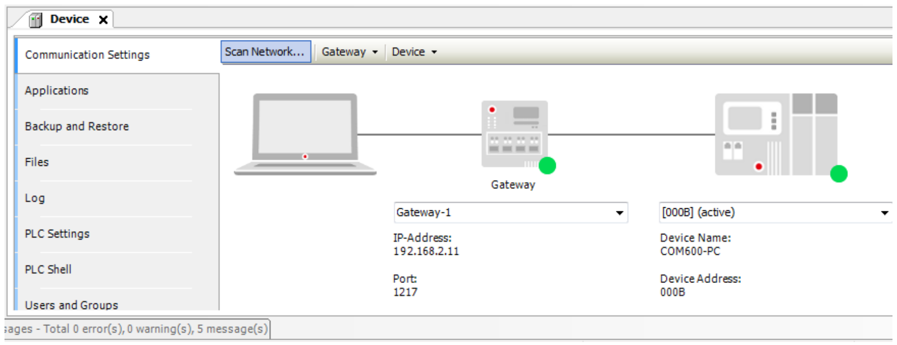
Add_Symbol_Configuration.png

Figure 4.6-1 Add Symbol Configuration

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.



Gateway-1

IP-Address:
192.168.2.11

Port:
1217

[000B] (active)

Device Name:
COM600-PC

Device Address:
000B

ages - Total 0 error(s), 0 warning(s), 5 message(s)

Device_Communication_Settings_Active_Path.png

Figure 4.7-1 Device Communication Settings after setting active path

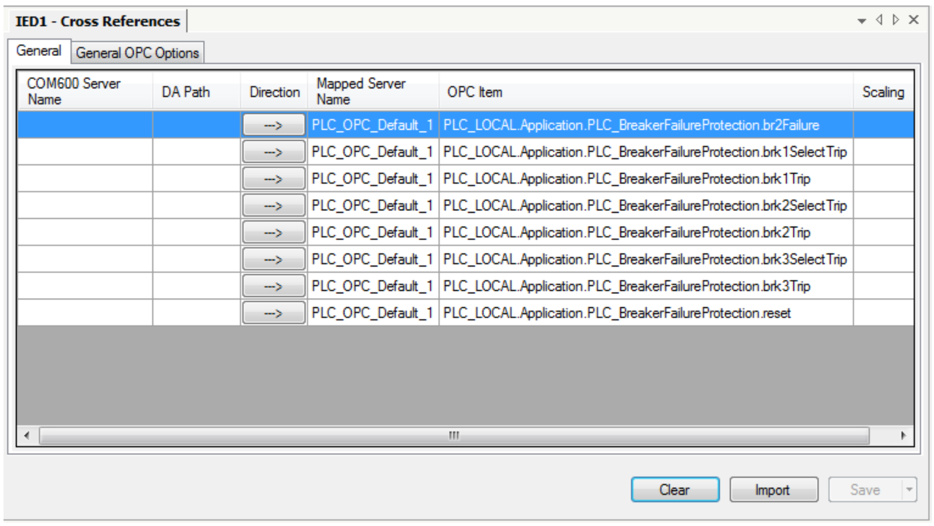
4.8. Building and activating the application

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.


IED1 - Cross References

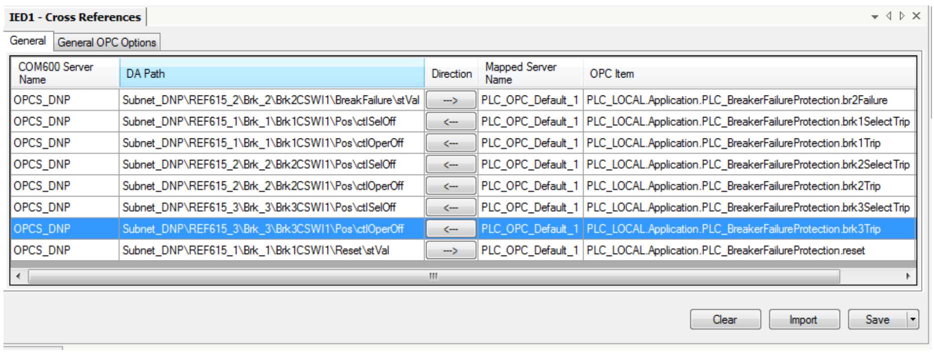
General | General OPC Options

COM600 Server Name	DA Path	Direction	Mapped Server Name	OPC Item	Scaling
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.br2Failure	
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk1SelectTrip	
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk1Trip	
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk2SelectTrip	
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk2Trip	
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk3SelectTrip	
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk3Trip	
		--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.reset	

Importing_LP_variables.png

Figure 4.9-1 Importing Logic Processor variables

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.


IED1 - Cross References

General | General OPC Options

COM600 Server Name	DA Path	Direction	Mapped Server Name	OPC Item
OPCS_DNP	Subnet_DNP\REF615_2\Brk_2\Brk2CSWI1\BreakFailure\stVal	--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.br2Failure
OPCS_DNP	Subnet_DNP\REF615_1\Brk_1\Brk1CSWI1\Pos\ctlSelOff	<---	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk1SelectTrip
OPCS_DNP	Subnet_DNP\REF615_1\Brk_1\Brk1CSWI1\Pos\ctlOperOff	<---	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk1Trip
OPCS_DNP	Subnet_DNP\REF615_2\Brk_2\Brk2CSWI1\Pos\ctlSelOff	<---	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk2SelectTrip
OPCS_DNP	Subnet_DNP\REF615_2\Brk_2\Brk2CSWI1\Pos\ctlOperOff	<---	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk2Trip
OPCS_DNP	Subnet_DNP\REF615_3\Brk_3\Brk3CSWI1\Pos\ctlSelOff	<---	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk3SelectTrip
OPCS_DNP	Subnet_DNP\REF615_3\Brk_3\Brk3CSWI1\Pos\ctlOperOff	<---	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.brk3Trip
OPCS_DNP	Subnet_DNP\REF615_1\Brk_1\Brk1CSWI1\Reset\stVal	--->	PLC_OPC_Default_1	PLC_LOCAL.Application.PLC_BreakerFailureProtection.reset

Cross-references_with_all_variables.png

Figure 4.9-2 Cross-references with all variables

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

Index

—
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