Contents:

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

2. Introduction .................................................................................................................. 11
   2.1. Functional overview ............................................................................................ 11
   2.2. Features ............................................................................................................... 12

3. Configuration ................................................................................................................. 14
   3.1. Overview of configuring Logic Processor ......................................................... 14
   3.2. Building communication structure objects ....................................................... 14
   3.3. Logic editor .......................................................................................................... 15
       3.3.1. Creating the logic in the CoDeSys programming environment ..................... 15
       3.3.2. Adding symbol configuration ......................................................................... 16
       3.3.3. Setting the active application ........................................................................ 17
       3.3.4. Configuring communication between Logic Editor and Logic Runtime .............. 19
       3.3.5. Building and activating the application ........................................................ 21
       3.3.6. Monitoring and debugging ............................................................................ 23
   3.4. Making cross-references ...................................................................................... 27
   3.5. Creating virtual data objects in the Logic Processor OPC server .................... 29
   3.6. Downloading Logic Processor OPC Server configuration ................................... 30

4. Application example .................................................................................................... 31
   4.1. Logic requirement ................................................................................................. 31
   4.2. Building object tree in SAB600 of three REF615s ............................................. 32
   4.3. Adding Logic Processor IED ............................................................................... 33
   4.4. Creating logic configuration ................................................................................ 34
   4.5. Adding symbol configuration .............................................................................. 39
   4.6. Setting the active application ............................................................................. 40
   4.7. Configuring communication between Logic Editor and Logic Runtime ............... 40
   4.8. Building and activating the application ............................................................. 41
   4.9. Making cross-references ..................................................................................... 42
| Index | .............................................................................................................. 45 |
1. About this manual

1.1. Copyrights

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Oy. ABB Oy assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Oy be liable for direct, indirect, special, incidental, or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Oy be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from ABB Oy, 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.

© Copyright 2009 ABB. All rights reserved.

1.2. 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.

1.3. General

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.

1.4. 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 > Menu-Item > 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

1.5. 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.

1.6. Terminology

The following is a list of terms associated with COM600 that you should be familiar with. The list contains terms that are unique to ABB or have a usage or definition that is different from standard industry usage.

<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>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>LON</td>
<td>A communication protocol developed by Echelon.</td>
</tr>
<tr>
<td>LON Application Guideline for substation automation; LAG</td>
<td>A proprietary method of ABB on top of the standard LON protocol.</td>
</tr>
</tbody>
</table>
### Term | Description
--- | ---
OPC | Series of standards specifications aiming at open connectivity in industrial automation and the enterprise systems that support industry.
OPC item | Representation of a connection to the data source within the OPC server. An OPC item is identified by a string `<object path>:<property name>`. Associated with each OPC item are Value, Quality, and Time Stamp.
Property | Named data item.
Report Control Block | The report control block controls the reporting processes for event data as they occur. The reporting process continues as long as the communication is available.
SPA | ABB proprietary communication protocol used in substation automation.
SPA device | Protection and/or Control Product supporting the SPA protocol version 2.5 or earlier.
Substation Configuration Language; SCL | XML-based description language for configurations of electrical substation IEDs. Defined in IEC 61850 standard.

### Abbreviations

The following is a list of abbreviations associated with COM600 that you should be familiar with. See also 1.6, Terminology.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>Alarms and Events</td>
</tr>
<tr>
<td>ASDU</td>
<td>Application Service Data Unit</td>
</tr>
<tr>
<td>BRCB</td>
<td>Buffered Report Control Block</td>
</tr>
<tr>
<td>DA</td>
<td>Data Access</td>
</tr>
<tr>
<td>DMCD</td>
<td>Data Message Code Definition</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>HMI</td>
<td>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>LAG</td>
<td>LON Application Guideline for substation automation</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LD</td>
<td>Logical Device</td>
</tr>
</tbody>
</table>
Abbreviation | Description
--- | ---
LMK | LonMark interoperable device communicating in LonWorks network. In this document, the term is used for devices that do not support the ABB LON/LAG communication.
LN | Logical Node
LSG | LON SPA Gateway
NCC | Network Control Center
NUC | Norwegian User Convention
NV | Network Variable
OLE | Object Linking and Embedding
OPC | OLE for Process Control
P&C | Protection & Control
RTS | Request To Send
SA | Substation Automation
SAB600 | Station Automation Builder 600
SCL | Substation Configuration Language
SLD | Single Line Diagram
SNTP | Simple Network Time Protocol
SOAP | Simple Object Access Protocol
RCB | Report Control Block
URCB | Unbuffered Report Control Block
XML | eXtended Markup Language

1.8. 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>DNP LAN/WAN Master (OPC)</td>
<td>1MRS756566</td>
</tr>
<tr>
<td>DNP Serial Master (OPC)</td>
<td>1MRS756567</td>
</tr>
<tr>
<td>DNP LAN/WAN Slave (OPC)</td>
<td>1MRS755496</td>
</tr>
<tr>
<td>DNP Serial Slave (OPC)</td>
<td>1MRS755495</td>
</tr>
<tr>
<td>External OPC Client Access</td>
<td>1MRS755564</td>
</tr>
</tbody>
</table>
### Name of the manual | MRS number
--- | ---
IEC 60870-5-101 Slave (OPC) | 1MRS755382
IEC 60870-5-101 Master (OPC) | 1MRS756703
IEC 60870-5-103 Master (OPC) | 1MRS752278
IEC 60870-5-104 Slave (OPC) | 1MRS755384
IEC 60870-5-104 Master (OPC) | 1MRS756704
IEC 61850 Master (OPC) | 1MRS755321
Logic Processor User's Manual | 1MRS756738
LON-LAG Master (OPC) | 1MRS755284
MNS IS Connectivity (OPC) | 1MRS756569
Modbus Serial Master (OPC) | 1MRS756126
Modbus Serial Slave (OPC) | 1MRS756913
Modbus TCP Master (OPC) | 1MRS756445
Modbus TCP Slave (OPC) | 1MRS756914
SPA Master (OPC) | 1MRS752275
SPA Router (OPC) | 1MRS755497

### 1.9. 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>
</tbody>
</table>
2. Introduction

2.1. 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 SP 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, Setting the active application).

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

### 2.2. 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 SP 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 (HMI, slave clients)
- Controls from HMI 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 V3.2 SP2 must be installed to be able to use the Logic Editor in SAB600. For information about CoDeSys V3.2 SP2 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. Select a Logic Processor OPC Server object and right-click it.
4. Add a **Logic Processor Subnetwork** object.
5. Select a Logic Processor Subnetwork object and right-click it.
6. Add a **Logic Processor IED** object.
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.
2. Select Logic Editor.
   The CoDeSys programming environment will start as a new application instance.
3. 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 SP 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.
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

The variables must be defined before configuration. Double-click the POU to be able to add variables and logic.

Select which variables from logic programming are cross-referenced with data in the COM600 communication structure by generating a symbol list.

To add a symbol configuration:
1. Select **Application** in the Device tree.
2. Right-click **Application**.
3. Select **Add Object**.
4. Select **Symbol configuration**.  
   The symbol configuration editor opens.
5. To get the currently available item pool, first click Refresh.
6. Add a variable from the Available variables field to the Selected variables field, select the variable and click the arrow button [>].
   You can add all variables by clicking the double-arrow button [>>].
7. The access right for a selected item can be modified in the Selected variables field by clicking the symbol in the Access Rights column.

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

3.3.3. Setting the active application

To set the active application:
1. Right-click the Main Task in the Devices window, and the editor view containing the configuration of the task opens.
In the Devices window the name “Application” is displayed in bold letters. This means that this application is set as “Active application”. Thus all commands and actions concerning the communication with the PLC refers to this application.

2. Click Add POU and the Input Assistant dialog opens.
3. In the Items tree view, select the desired POU and click OK. This program is added to main task.
4. In the Main Task Configuration dialog, set Priority to 0 and select Cyclic for the task type and “t# 200ms” for 200-millisecond interval time.
3.3.4. Configuring communication between Logic Editor and Logic Runtime

To download a logic program:
2. Add a Gateway object.
3. Add the IP Address of the COM600 computer to the Gateway dialog and click OK.
4. The Device dialog opens with the Communication Settings tab.
5. In Device dialog, click **Scan network**. PLC Device on COM600 appears. The Node name is the COM600 computer name.

6. Select the PLC device and click **Set active path**. This communication path is set as the active one, which means that all actions concerning communication refers exactly to this path.

### 3.3.5. Building and activating the application

To log in to the device:

1. Select Build -> Build Application 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. Select the device.

3. Select **Online > Login to ‘Application[Device:PLC Logic]’**.
To start the PLC program application:

1. Select Online > Start ‘Application [Device: PLC Logic]’.
2. When the logic application has started, the status window shows RUN in green.

3.3.6. Monitoring and debugging

Open the instance window of a POU, and monitor and debug the application in online mode.
You can write or force values to Logic Processor variables.

To write or force values:
1. Enter values in the Prepared value column of the Expression table.
2. Select Online > Write Values or Online > Force Values.
Figure 3.3.6-2 Write values
3. After PLC logic has been tested, you can log out from the device by selecting **Online > Logout from ‘Application [Device: PLC Logic]’**.
Logic is still running on PLC device.

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.

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 **PLC Server Path** column.

![Logic Processor Cross-References](Logic_Processor_Cross_Ref)

**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.

4. The Cross References tool automatically fills in a default data attribute to be used.

5. To change the used data attribute, select the whole row and click **Edit Row** to open the **Data attribute selection** dialog.

![Editing_row.bmp](Editing_row.bmp)

**Figure 3.4-2 The Cross references tool**

6. From the data attribute drop-down list, select the data attribute to be used. Click **Apply** to confirm the change in the table.
7. 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 (<-).

8. After completing the configuration of cross-reference, click **Apply** to save the setting, 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 OPC server

You can use virtual data objects for COM600 HMI or slave clients to access COM600 Logic Processor variables.

To add a logical node and a virtual data object:

1. Select a Logical Device object and right-click it.
2. Add a Logical Node object.
3. Select a Logical Node object and right-click it.
4. Add a data object.
5. Modify the object properties.
   - In the **OPC Item Path** property field, click the browse button with three dots to open the **PLC OPC Item Path** logic variable dialog.
6. Select the logic variable from the list and click **OK**.

The created data objects can be used with COM600 HMI and slave clients.

### 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.

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.

![System Diagram](Logic_Processor_System_Diagram.jpg)

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

In the system shown in Figure 3.3.2-1 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.
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 Open to confirm the object settings.
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 five catalogs: General, Boolean Operators, Math operators, Function blocks, and Ladder elements.
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.

---

**Figure 4.4-2 Adding R TRIG**

**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**.

![Declaration Part FBD.bmp](image)

*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.
The logic can be built as follows.

![ figure 4.4-5 Adding Reset Set function block ]

![ 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 symbol configuration

1. Select Application in the Device tree.
2. Right-click Application.
3. Select Add Object.
4. Select Symbol configuration.
   The symbol configuration editor opens.

5. To get the currently available item pool, first click Refresh.
6. Add a variable from the Available variables field to the Selected variables field, select the variable and click the arrow button [>]. You can add all variables by clicking the double-arrow button [>>]. In this case, only signals that are used in this application are selected, i.e. tripping signals: brk1SelectTrip, brk1Trip, brk2SelectTrip, brk2Trip, brk3SelectTrip, and Brk3Trip; breaker failure of breaker 2: brk2Failure; reset from breaker 1.

7. The access right for a selected item can be modified in the Selected variables field by clicking the symbol in the Access Rights column. During the compilation of the project, a symbol list is created, which gets exported to a file (XML) in the project directory, and also gets loaded to the device during the application download in the form of a child application.

4.6. Setting the active application

To set the active application:
1. Right-click the Main Task in the Devices window, and the editor view containing the configuration of the task opens. In the Devices window the name “Application” is displayed in bold letters. This means that this application is set as “Active application”. Thus all commands and actions concerning the communication with the PLC refers to this application.
2. Click Add POU and the Input Assistant dialog opens.
3. In the Items tree view, select “PLC_BreakerFailureProtect” and click OK. This program is added to main task.
4. In the Main Task Configuration dialog, set Priority to 0 and select Cyclic for the task type and “t# 200ms” for 200-millisecond interval time.

4.7. Configuring communication between Logic Editor and Logic Runtime

To configure a communication channel:
1. Double-click the device in the device window.
2. A dialog with the Communication Settings tab opens.
3. Select Gateway in the dialog and click Scan network to search for a device (PLC) on COM600 in the local network. The PLC found is displayed indented below the gateway.
4. Select the PLC (device) entry and click Set active path. This sets the communication channel as the active one, which means that all actions concerning the communication refers to this channel.
4.8. Building and activating the application

1. Select **Build -> Build Application** 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 Application** 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 **Online -> Start Application** to start the program on the PLC. The program starts running.

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

---

*Figure 4.7-1 Setting communication channel*
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.

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 PLC Server Path column.

3. From the communication structure, select a data object and drag and drop it to the proper row in the Cross References tool.
4. Select a DPC data object to map the breaker's Trip variable. The default data attribute of DPC is 'stVal'.
5. Select the whole row and click Edit Row to open the Data attribute selection dialog.

Figure 4.9-1 Importing Logic Processor variables

The Import button is used for fetching another symbol file.

Example_cross_reference.bmp
6. From the data attribute drop-down list, select 'ctlOperOff' to map 'brk1Trip'. Click **Apply** to confirm the change in the table. The control signal is sending from the PLC logic to IED, so click the Direction field and make it point to the left `<-

7. After completing the configuration of cross reference, click **Apply** to save the setting, and close the Cross References tool.
Downloading Logic Processor OPC Server configuration

For downloading Logic Processor OPC Server configuration, see 3.6, Downloading Logic Processor OPC Server configuration
Index

A
activating ...................................................................................................... 19, 21

B
building
communication structure ................................................................................. 14

cross-reference ............................................................................................. 27, 42

C
CoDeSys SP runtime .......................................................................................... 12
cross-reference ............................................................................................. 27, 42

L
logic ................................................................................................................... 34
Logic Processor
CoDeSys SP runtime ...................................................................................... 11
logic programming
logic editor ..................................................................................................... 12

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

V
variable
OPC symbol ..................................................................................................... 16, 39
virtual data object
variable .......................................................................................................... 29