Grid Automation Controller COM600 4.0
Logic Processor User's Manual

Power and productivity
for a better world™

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
2.3. Opening projects created with SAB600 version 3.4 13

3. Configuration 15

3.1. Overview of configuring Logic Processor 15
3.2. Building communication structure objects 15
3.3. Logic editor 16
3.3.1. Creating the logic in the CoDeSys programming environment 16
3.3.2. Adding symbol configuration 17
3.3.3. Setting the active application 18
3.3.4. Configuring communication between Logic Editor and Logic Runtime 20
3.3.5. Building and activating the application 22
3.3.6. Monitoring and debugging 24
3.4. Making cross-references 28
3.5. Creating virtual data objects in the Logic Processor OPC server 30
3.6. Downloading Logic Processor OPC Server configuration 31

4. Application example 32

4.1. Logic requirement 32
4.2. Building object tree in SAB600 of three REF615s 33
4.3. Adding Logic Processor IED 34
4.4. Creating logic configuration 35
4.5. Adding symbol configuration 41
4.6. Setting the active application 42
4.7. Configuring communication between Logic Editor and Logic Runtime 42
4.8. Building and activating the application 43

4.9. Making cross-references 44

Index 47

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.

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** > **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

1.5. 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

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.
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.
LON	A communication protocol developed by Echelon.
LON Application Guideline for substation automation; LAG	A proprietary method of ABB on top of the standard LON protocol.
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.
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.

1.7. Abbreviations

Abbreviation	Description
AE	Alarms and Events
ASDU	Application Service Data Unit
BRCB	Buffered Report Control Block
DA	Data Access
DMCD	Data Message Code Definition
DO	Data Object
GW	Gateway, component connecting two communication networks together
HMI	Human Machine Interface
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
LAG	LON Application Guideline for substation automation
LAN	Local Area Network
LD	Logical Device
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

Abbreviation	Description
NUC	Norwegian User Convention
NV	Network Variable
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
SNMP	Simple Network Management Protocol
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

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
DNP LAN/WAN Master (OPC)	1MRS756566
DNP Serial Master (OPC)	1MRS756567
DNP LAN/WAN Slave (OPC)	1MRS755496
DNP Serial Slave (OPC)	1MRS755495
External OPC Client Access	1MRS755564

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 *i*S 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

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

2. Introduction

2.1. Functional overview

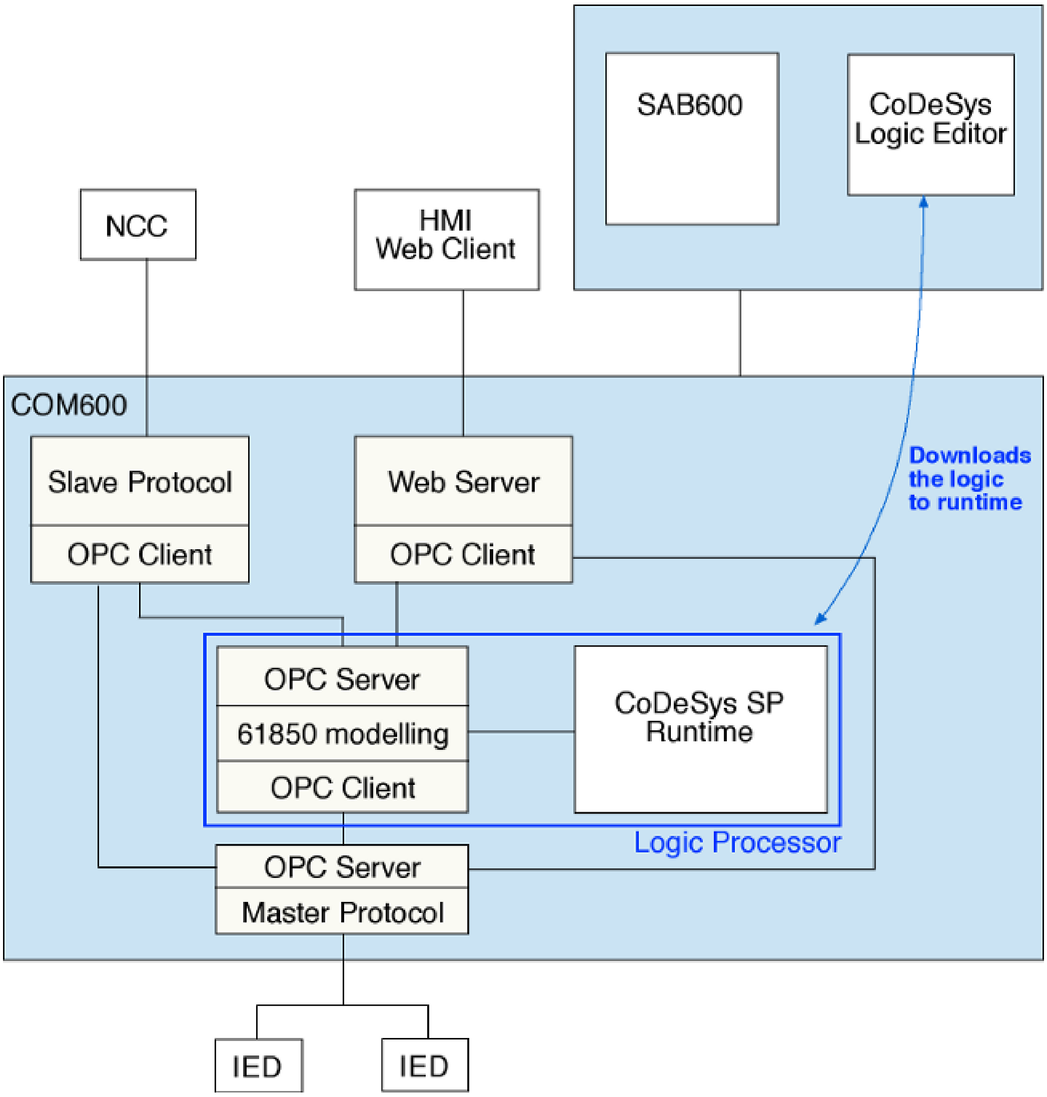
The Logic Processor function enables developing customer-specific automated applications for COM600. Applications are programmed using the IEC 61131-3 logic editor (CoDeSys). The COM600 unit has the logic engine (CoDeSys SP Runtime), which can load and execute the created IEC 61131-3 applications. Information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



Logic Processor.bmp

Figure 2.1-1 Functional overview of Logic Processor

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)

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

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

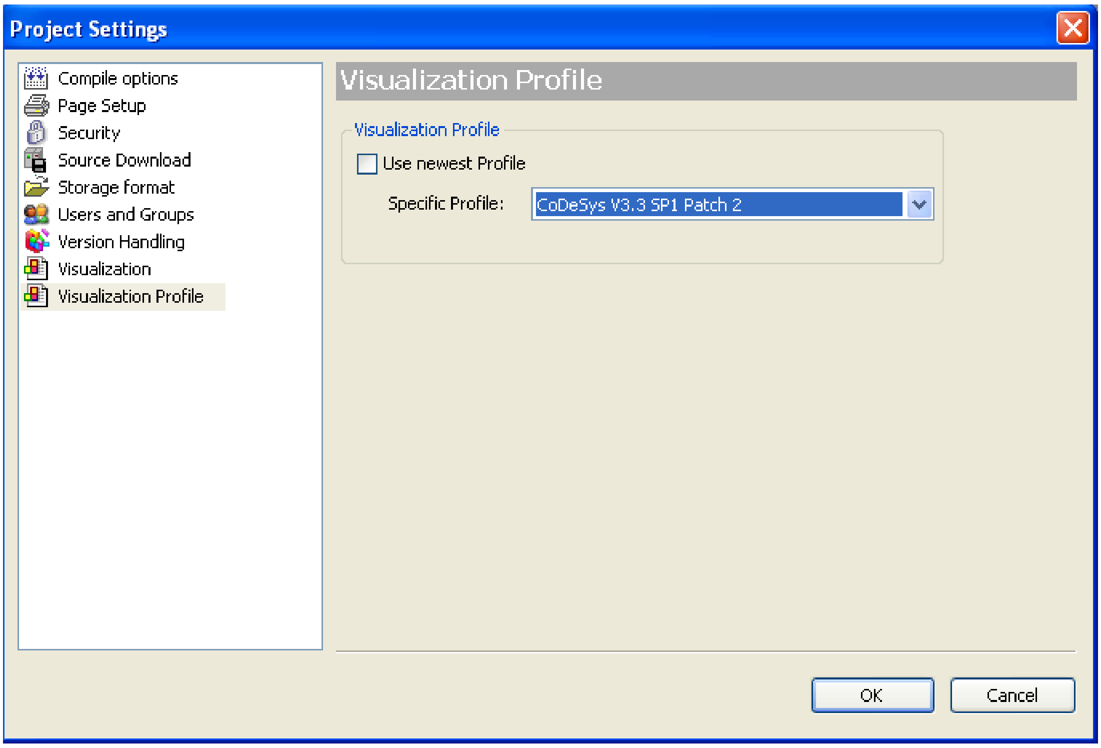
For example measurements, indications, and controls can be exchanged between the logic runtime and other COM600 components.

2.3. Opening projects created with SAB600 version 3.4

When opening projects created with 3.4 version of SAB600, following error(s) might be shown in the CoDeSys output window.

```
[ERROR]        codesys34: The file 'compileinfo' is missing in the repository
 for visual elements.  The repository with the problem is 'System', the
profile 'CoDeSys V3.4'.
```

This can be corrected by selecting a correct visualization profile in **Project** > **Project Settings**. See Figure 2.3-1 for the correct setting.



Visualization Profile

□ Use newest Profile

Specific Profile: CoDeSys V3.3 SP1 Patch 2

3sProjectSettings.bmp

Figure 2.3-1 Project settings

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.

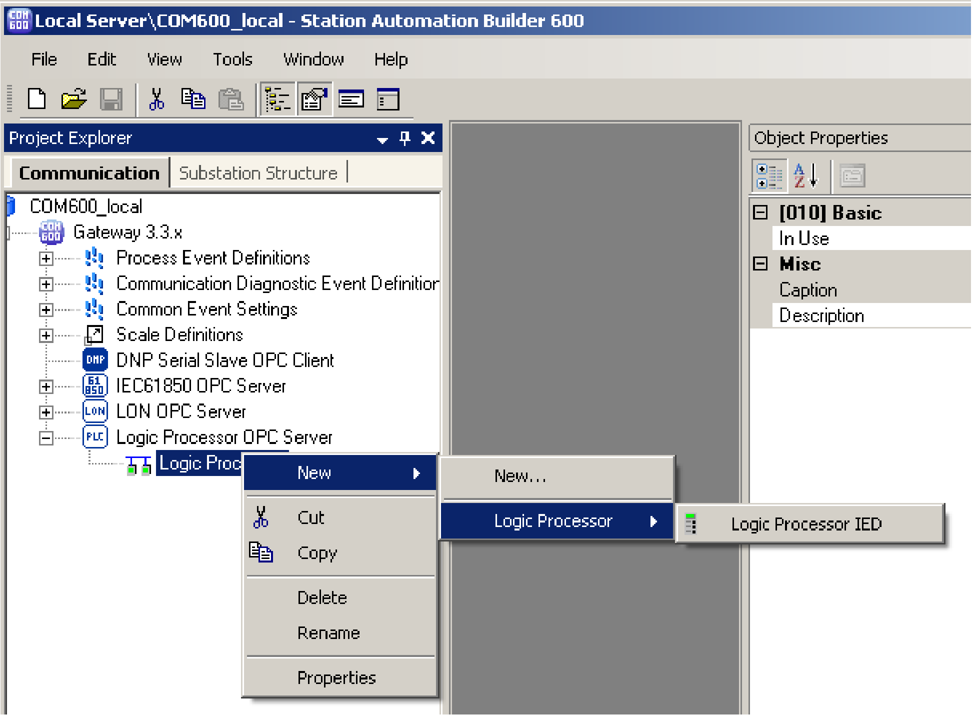
The Logic Processor configuration in COM600 can be divided into the following tasks:

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

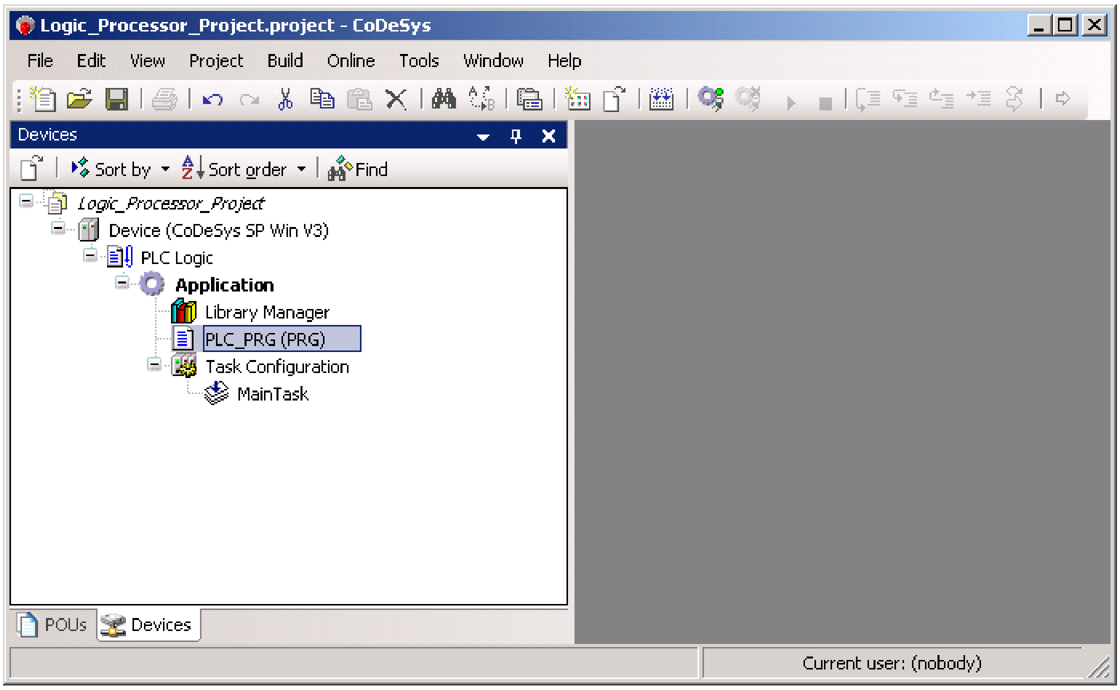
3.3. Logic editor

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 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.



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.

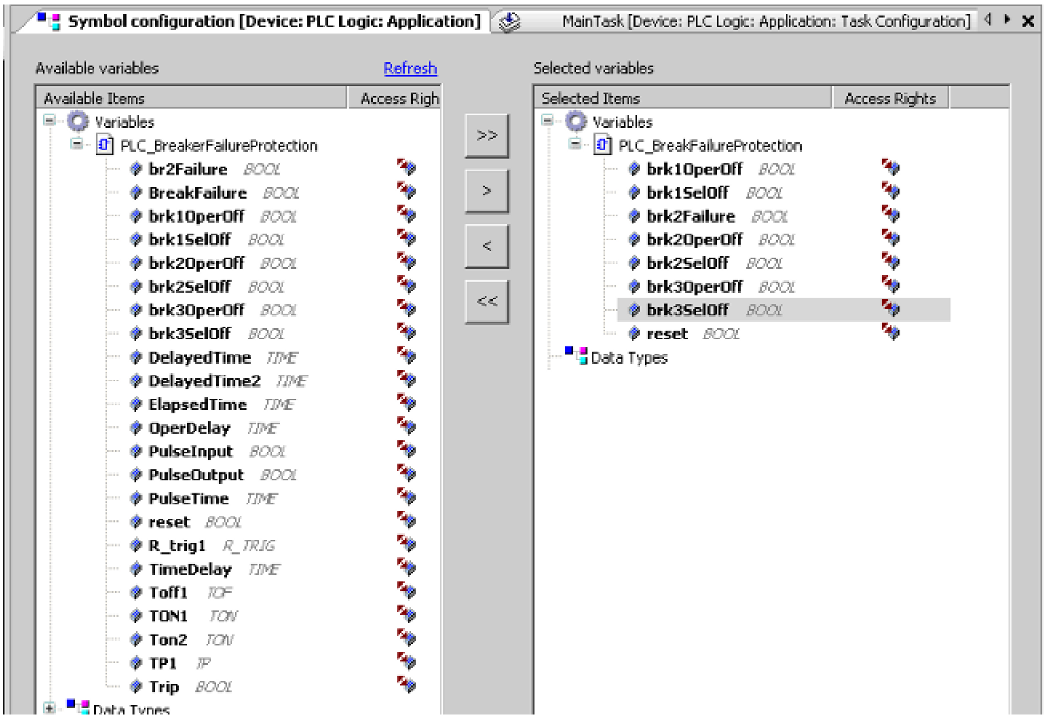
3.3.2. Adding symbol configuration

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.


Symbol configuration [Device: PLC Logic: Application]
MainTask [Device: PLC Logic: Application: Task Configuration]
Refresh
>>
>
<
<<

Available variables

Available Items	Access Righ
Variables	
PLC_BreakerFailureProtection	
br2Failure BOOL	
BreakFailure BOOL	
brk1OperOff BOOL	
brk1SelOff BOOL	
brk2OperOff BOOL	
brk2SelOff BOOL	
brk3OperOff BOOL	
brk3SelOff BOOL	
DelayedTime TIME	
DelayedTime2 TIME	
ElapsedTime TIME	
OperDelay TIME	
PulseInput BOOL	
PulseOutput BOOL	
PulseTime TIME	
reset BOOL	
R_trig1 R_TRIG	
TimeDelay TIME	
Toff1 TOF	
TON1 TON	
Ton2 TON	
TP1 TP	
Trip BOOL	
Data Types	

Selected variables

Selected Items	Access Rights
Variables	
PLC_BreakFailureProtection	
brk1OperOff BOOL	
brk1SelOff BOOL	
brk2Failure BOOL	
brk2OperOff BOOL	
brk2SelOff BOOL	
brk3OperOff BOOL	
brk3SelOff BOOL	
reset BOOL	
Data Types	

Adding_symbol_configuration.bmp

Figure 3.3.2-1 Adding symbol configuration

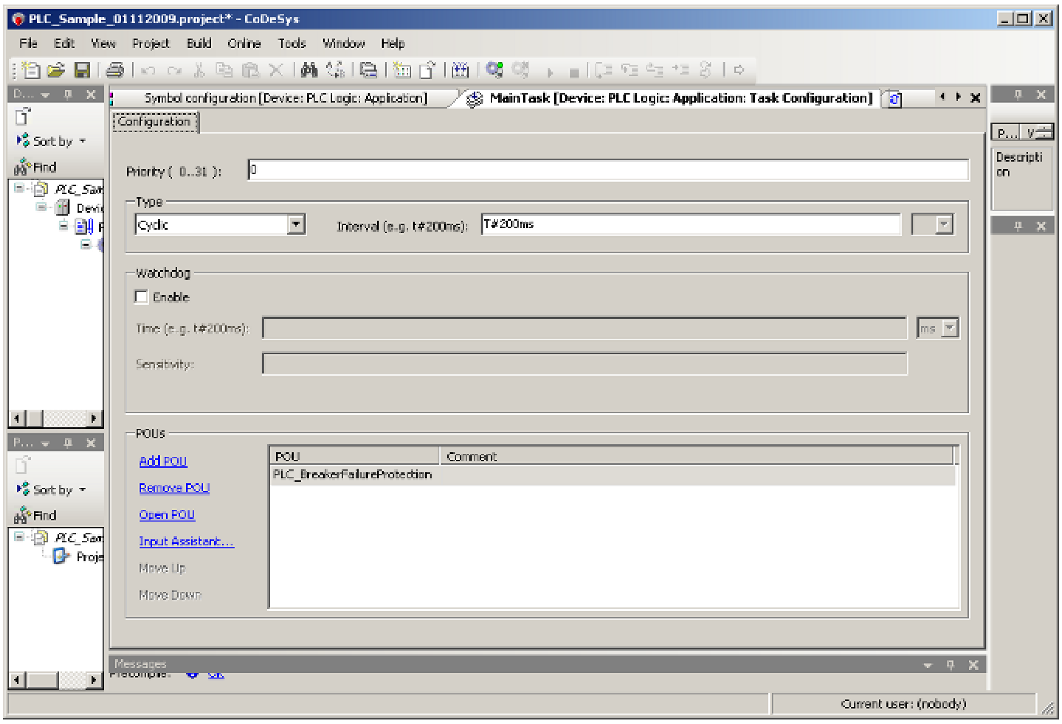
5. To get the currently available item pool, first click **Refresh**.
6. Add a variable from the **Available variables** field to the **Selected variables** field, select the variable and click the arrow button [>].
   You can add all variables by clicking the double-arrow button [>>].
7. The access 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.


PLC_Sample_01112009.project* - CoDeSys
File Edit View Project Build Online Tools Window Help
Sort by
Find
PLC_Sam
Devic
Symbol configuration [Device: PLC Logic: Application]
MainTask [Device: PLC Logic: Application: Task Configuration]
Configuration
Descripti on
Messages
Precompile: OK

Priority (0..31): 0

Type

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

Watchdog

☐ Enable

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

Sensitivity:

POUs

Add POU

Remove POU

Open POU

Input Assistant...

Move Up

Move Down

POU	Comment
PLC_BreakerFailureProtection	

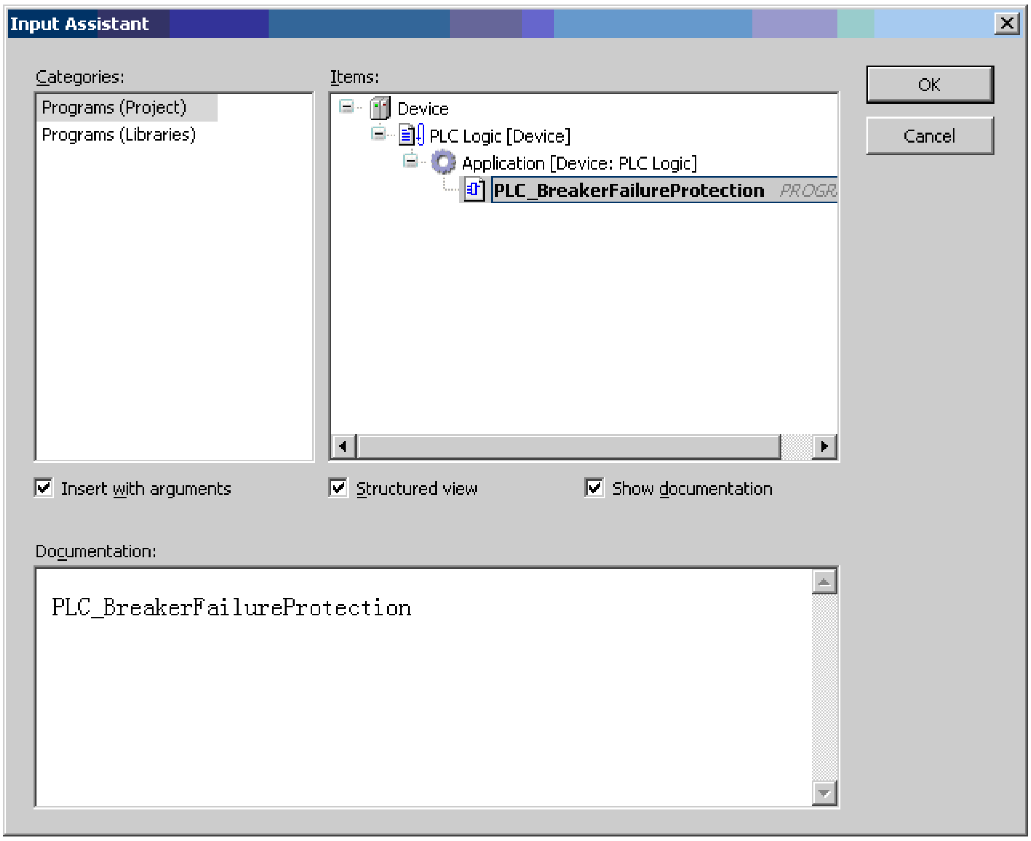
Current user: (nobody)

Adding_Program_Main_Task.bmp

Figure 3.3.3-1 Adding program to main task

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.



☑ Insert with arguments ☑ Structured view ☑ Show documentation

PLC_BreakerFailureProtection

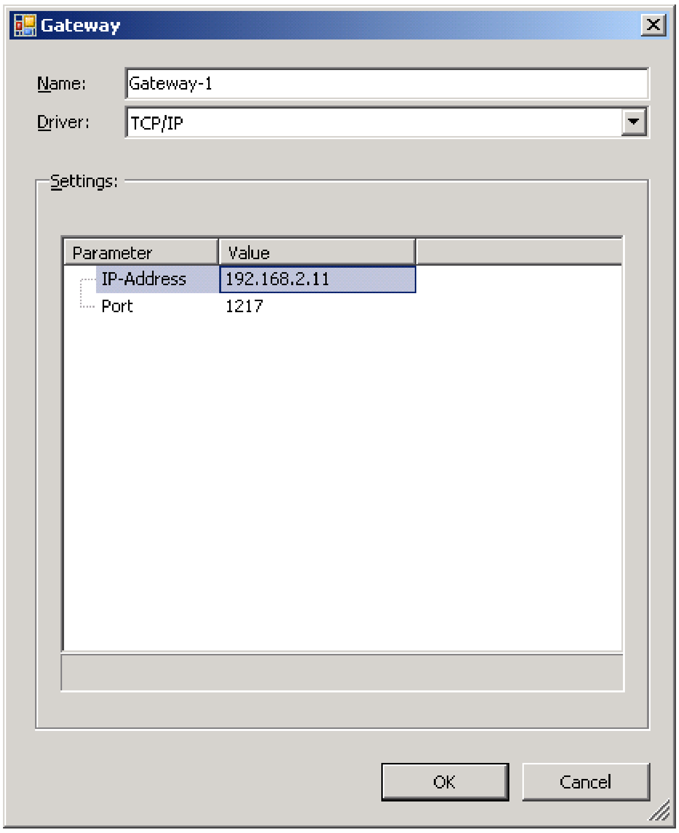
Selecting_Program.bmp

Figure 3.3.3-2 Selecting program to add to main task

3.3.4. Configuring communication between Logic Editor and Logic Runtime

To download a logic program:

1. Select a device by double-clicking the **Device** node in the Devices tree. CoDeSys Device opens.
2. Add a Gateway object.



Name: Gateway-1

Driver: TCP/IP

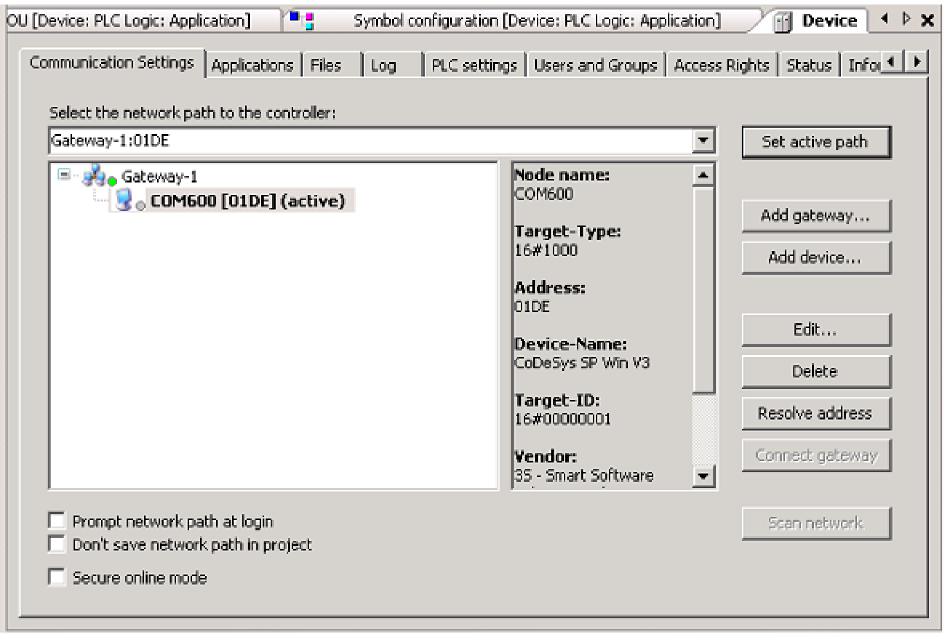
Settings:

Parameter	Value
IP-Address	192.168.2.11
Port	1217

Add_gateway.bmp

Figure 3.3.4-1 Adding the 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.



Select the network path to the controller: Gateway-1:01DE

Gateway-1
COM600 [01DE] (active)

Node name: COM600
Target-Type: 16#1000
Address: 01DE
Device-Name: CoDeSys SP Win V3
Target-ID: 16#00000001
Vendor: 3S - Smart Software

□ Prompt network path at login
□ Don't save network path in project
□ Secure online mode

Select_device.bmp

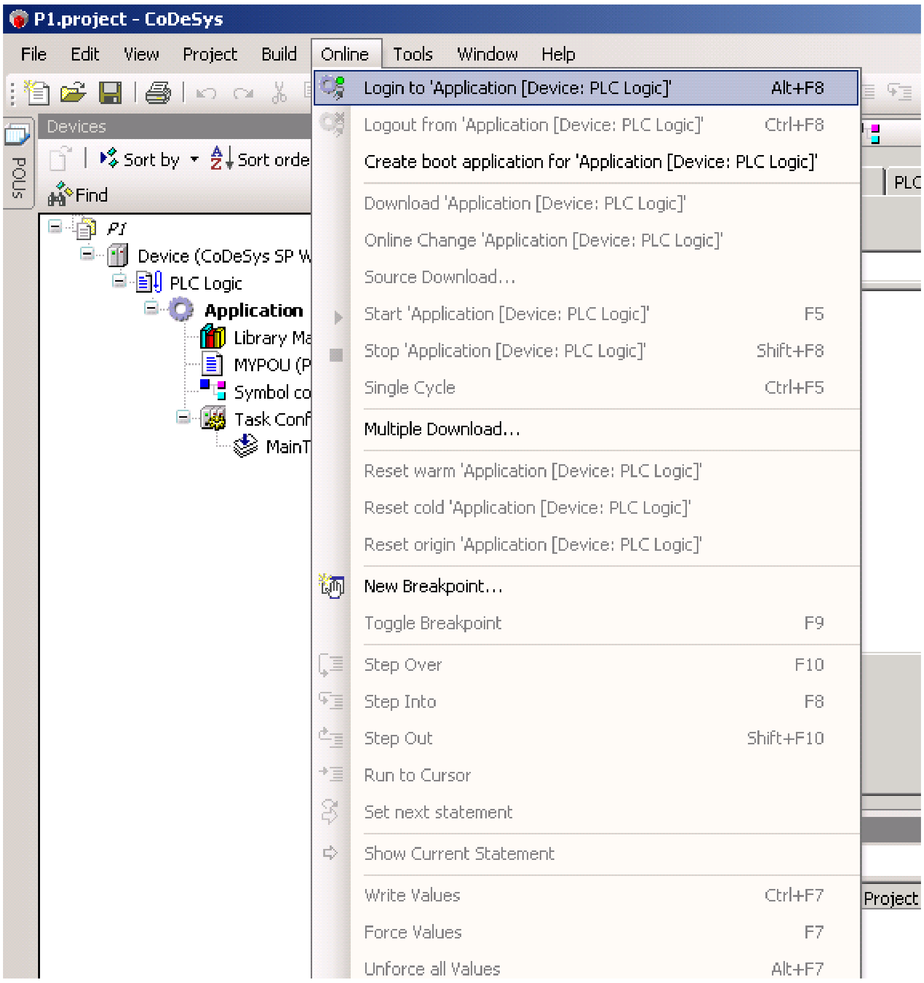
Figure 3.3.4-2 CoDeSys Device dialog

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]’**.

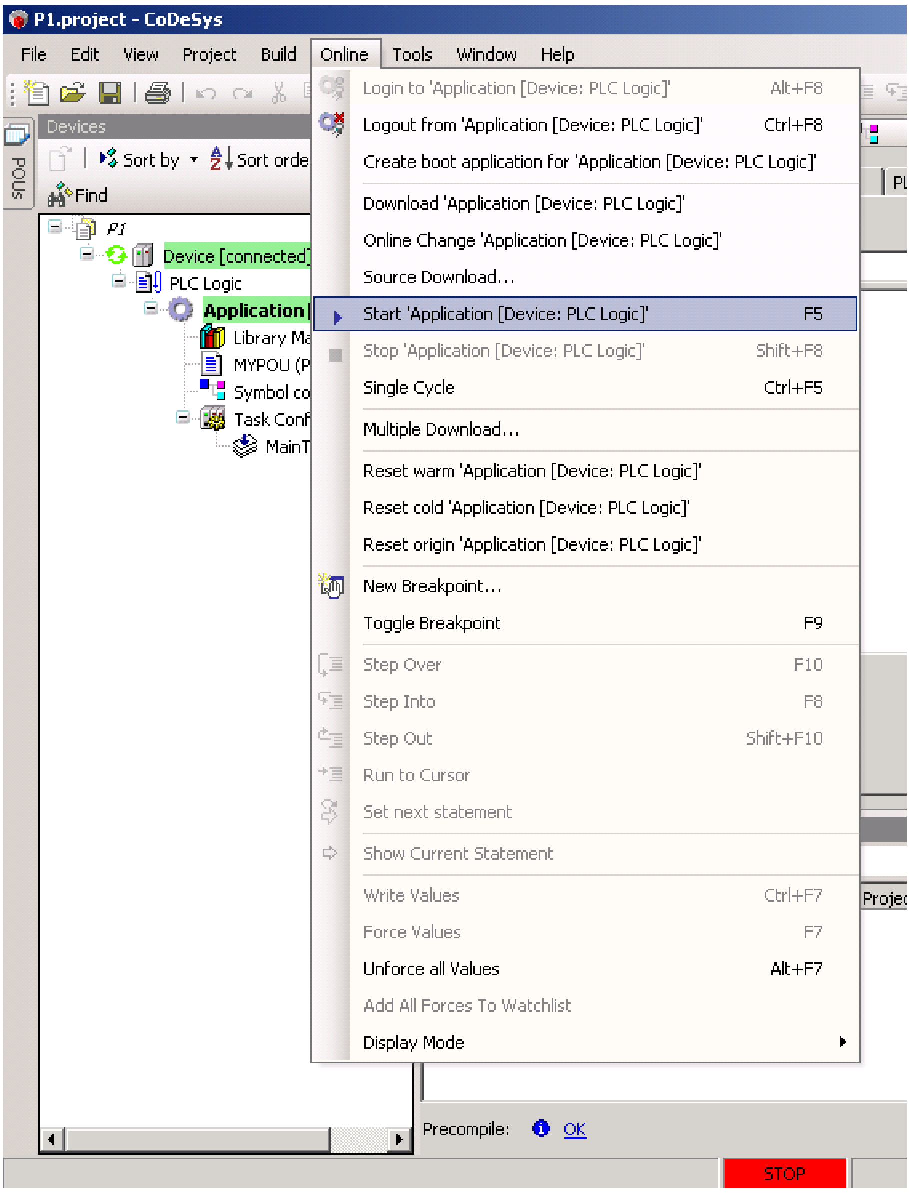

P1.project - CoDeSys
File
Edit
View
Project
Build
Online
Tools
Window
Help
Devices
POUs
Sort by
Sort orde
Find
P1
Device (CoDeSys SP W
PLC Logic
Application
Library Ma
MYPOU (P
Symbol co
Task Conf
MainT
Login to 'Application [Device: PLC Logic]' Alt+F8
Logout from 'Application [Device: PLC Logic]' Ctrl+F8
Create boot application for 'Application [Device: PLC Logic]'
Download 'Application [Device: PLC Logic]'
Online Change 'Application [Device: PLC Logic]'
Source Download...
Start 'Application [Device: PLC Logic]' F5
Stop 'Application [Device: PLC Logic]' Shift+F8
Single Cycle Ctrl+F5
Multiple Download...
Reset warm 'Application [Device: PLC Logic]'
Reset cold 'Application [Device: PLC Logic]'
Reset origin 'Application [Device: PLC Logic]'
New Breakpoint...
Toggle Breakpoint F9
Step Over F10
Step Into F8
Step Out Shift+F10
Run to Cursor
Set next statement
Show Current Statement
Write Values Ctrl+F7
Force Values F7
Unforce all Values Alt+F7
PLC
Project

Login_to_application.bmp

Figure 3.3.5-1 Logging in to application

To start the PLC program application:

1. Select **Online > Start 'Application [Device: PLC Logic]'** .


P1.project - CoDeSys
File
Edit
View
Project
Build
Online
Tools
Window
Help
Devices
Sort by
Sort orde
Find
P1
Device [connected]
PLC Logic
Application
Library Ma
MYPOU (P
Symbol co
Task Conf
MainT
POUs
Login to 'Application [Device: PLC Logic]' Alt+F8
Logout from 'Application [Device: PLC Logic]' Ctrl+F8
Create boot application for 'Application [Device: PLC Logic]'
Download 'Application [Device: PLC Logic]'
Online Change 'Application [Device: PLC Logic]'
Source Download...
Start 'Application [Device: PLC Logic]' F5
Stop 'Application [Device: PLC Logic]' Shift+F8
Single Cycle Ctrl+F5
Multiple Download...
Reset warm 'Application [Device: PLC Logic]'
Reset cold 'Application [Device: PLC Logic]'
Reset origin 'Application [Device: PLC Logic]'
New Breakpoint...
Toggle Breakpoint F9
Step Over F10
Step Into F8
Step Out Shift+F10
Run to Cursor
Set next statement
Show Current Statement
Write Values Ctrl+F7
Force Values F7
Unforce all Values Alt+F7
Add All Forces To Watchlist
Display Mode
Projec
Precompile: OK
STOP

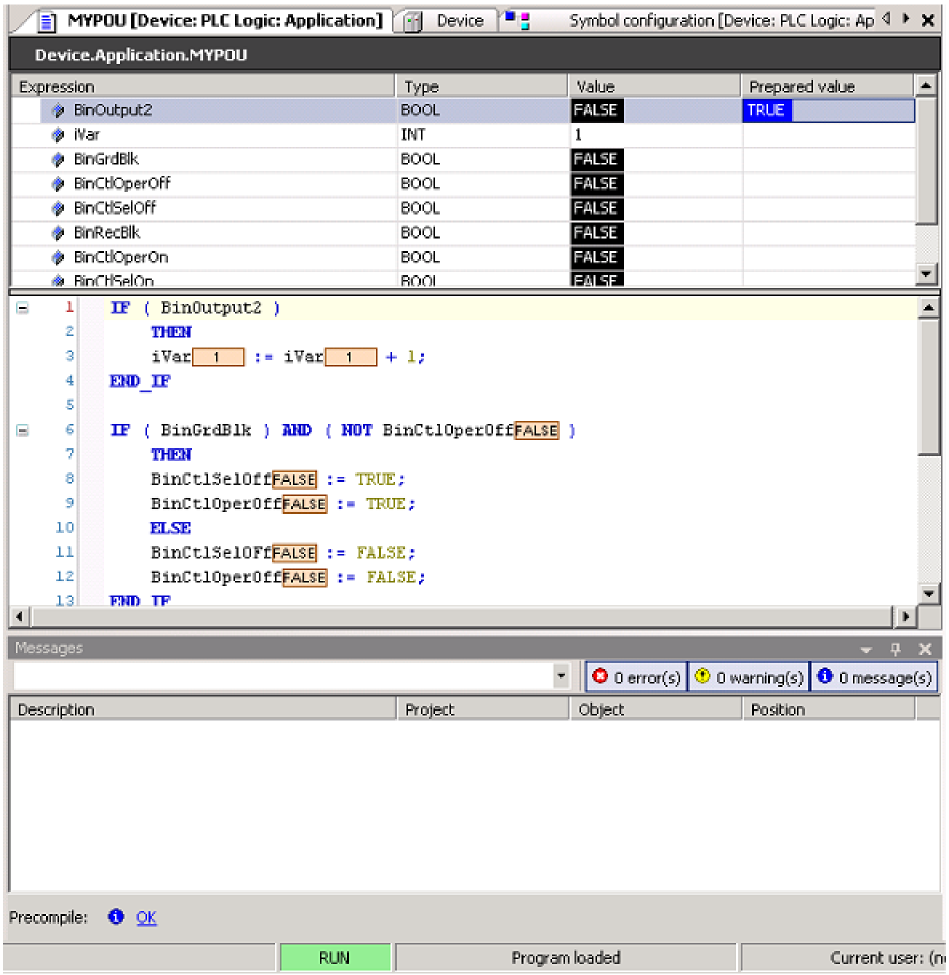
Start_application.bmp

Figure 3.3.5-2 Starting application on the device

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.


MYPOU [Device: PLC Logic: Application]
Device
Symbol configuration [Device: PLC Logic: Ap

Device.Application.MYPOU

Expression	Type	Value	Prepared value
BinOutput2	BOOL	FALSE	TRUE
iVar	INT	1	
BinGrdBlk	BOOL	FALSE	
BinCtlOperOff	BOOL	FALSE	
BinCtlSelOff	BOOL	FALSE	
BinRecBlk	BOOL	FALSE	
BinCtlOperOn	BOOL	FALSE	
BinCtlSelOn	BOOL	FALSE	

```
IF ( BinOutput2 )
    THEN
    iVar 1 := iVar 1 + 1;
END_IF

IF ( BinGrdBlk ) AND ( NOT BinCtlOperOff FALSE )
    THEN
    BinCtlSelOff FALSE := TRUE;
    BinCtlOperOff FALSE := TRUE;
    ELSE
    BinCtlSelOff FALSE := FALSE;
    BinCtlOperOff FALSE := FALSE;
END_IF
```

Description	Project	Object	Position

Precompile: OK

RUN | Program loaded | Current user: (n

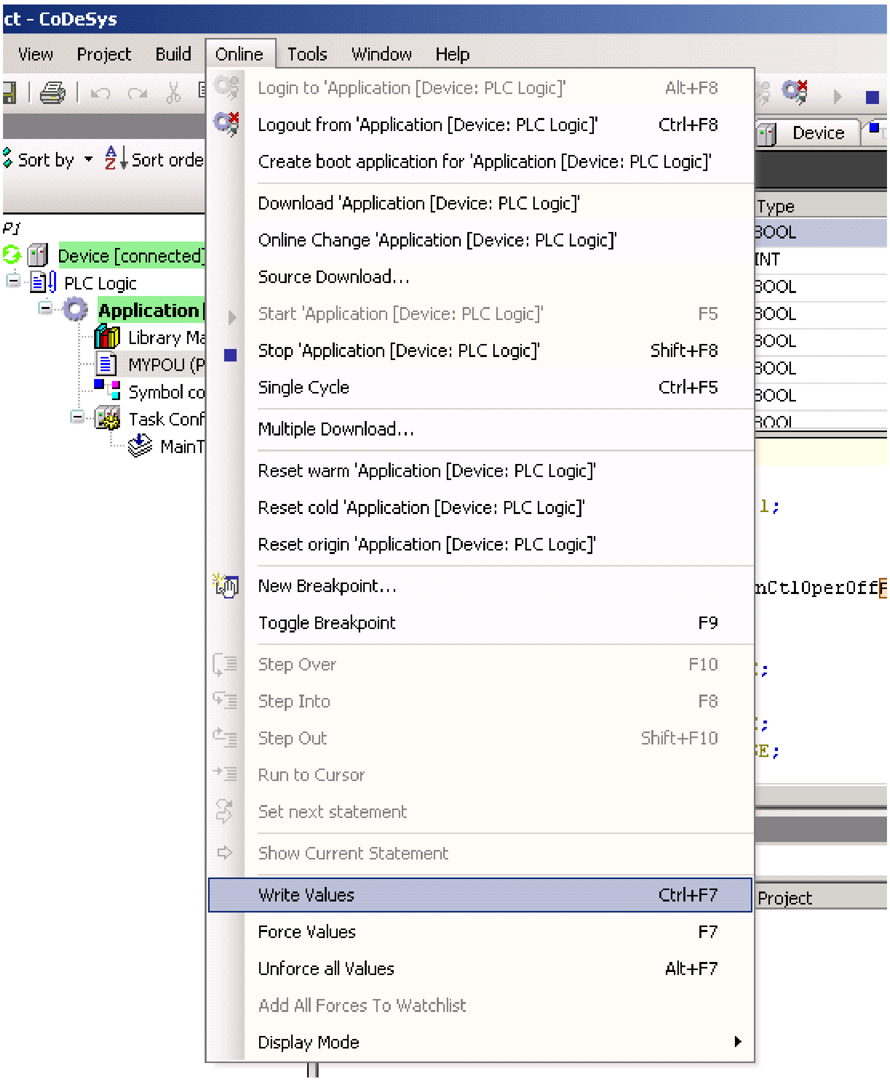
Monitoring_and_debugging.bmp

Figure 3.3.6-1 Monitoring and debugging

You can write or force values to Logic Processor variables.

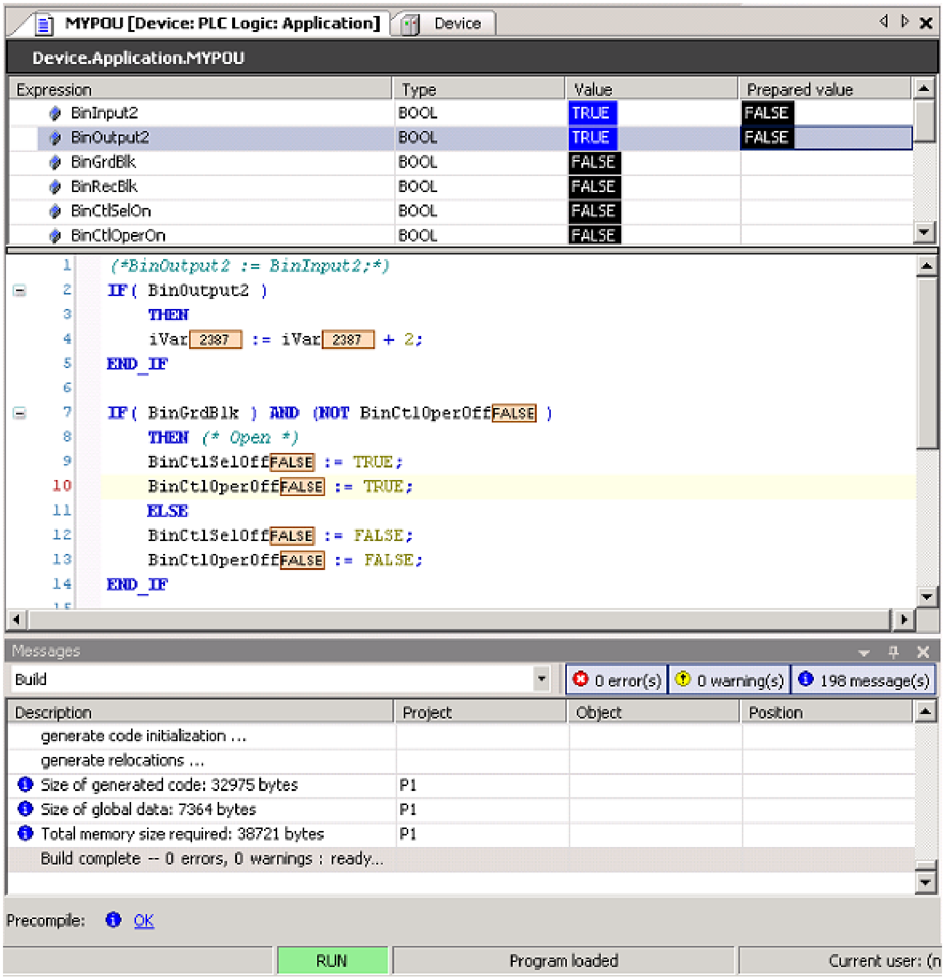
To write or force values:

1. Enter values in the **Prepared value** column of the Expression table.
2. Select **Online** > **Write Values** or **Online** > **Force Values**.


ct - CoDeSys
View Project Build Online Tools Window Help
Sort by
Sort orde
Device
Type
P1
Device [connected]
PLC Logic
Application
Library Ma
MYPOU (P
Symbol co
Task Conf
MainT
BOOL
INT
BOOL
BOOL
BOOL
BOOL
BOOL
BOOL
Login to 'Application [Device: PLC Logic]' Alt+F8
Logout from 'Application [Device: PLC Logic]' Ctrl+F8
Create boot application for 'Application [Device: PLC Logic]'
Download 'Application [Device: PLC Logic]'
Online Change 'Application [Device: PLC Logic]'
Source Download...
Start 'Application [Device: PLC Logic]' F5
Stop 'Application [Device: PLC Logic]' Shift+F8
Single Cycle Ctrl+F5
Multiple Download...
Reset warm 'Application [Device: PLC Logic]'
Reset cold 'Application [Device: PLC Logic]'
Reset origin 'Application [Device: PLC Logic]'
New Breakpoint...
Toggle Breakpoint F9
Step Over F10
Step Into F8
Step Out Shift+F10
Run to Cursor
Set next statement
Show Current Statement
Write Values Ctrl+F7
Force Values F7
Unforce all Values Alt+F7
Add All Forces To Watchlist
Display Mode
mCtlOperOff
Project

Write_values.bmp

Figure 3.3.6-2 Write values



Expression	Type	Value	Prepared value
BinInput2	BOOL	TRUE	FALSE
BinOutput2	BOOL	TRUE	FALSE
BinGrdBlk	BOOL	FALSE	
BinRecBlk	BOOL	FALSE	
BinCtlSelOn	BOOL	FALSE	
BinCtlOperOn	BOOL	FALSE	

```
(*BinOutput2 := BinInput2;*)
IF( BinOutput2 )
    THEN
    iVar 2387 := iVar 2387 + 2;
END_IF

IF( BinGrdBlk ) AND (NOT BinCtlOperOffFALSE )
    THEN (* Open *)
    BinCtlSelOffFALSE := TRUE;
    BinCtlOperOffFALSE := TRUE;
    ELSE
    BinCtlSelOffFALSE := FALSE;
    BinCtlOperOffFALSE := FALSE;
END_IF
```

Description	Project	Object	Position
generate code initialization ...			
generate relocations ...			
Size of generated code: 32975 bytes	P1		
Size of global data: 7364 bytes	P1		
Total memory size required: 38721 bytes	P1		
Build complete -- 0 errors, 0 warnings : ready...			

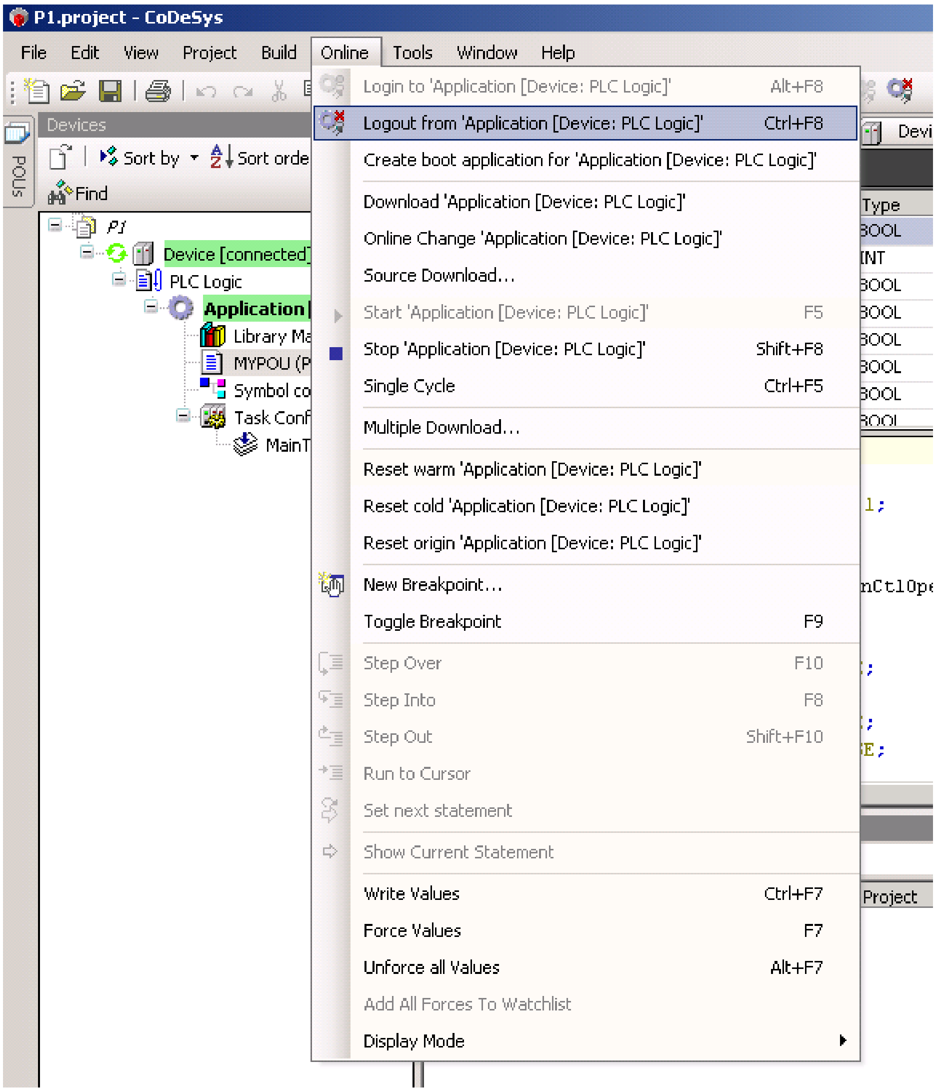
Precompile: OK

RUN | Program loaded | Current user: (n

Variable_is_written.bmp

Figure 3.3.6-3 Variable is written

3. After PLC logic has been tested, you can log out from the device by selecting **Online > Logout from ‘Application [Device: PLC Logic]’**.


P1.project - CoDeSys
File
Edit
View
Project
Build
Online
Tools
Window
Help
Devices
Sort by
Sort orde
Find
POUs
P1
Device [connected]
PLC Logic
Application
Library Ma
MYPOU (P
Symbol co
Task Conf
MainT
Login to 'Application [Device: PLC Logic]' Alt+F8
Logout from 'Application [Device: PLC Logic]' Ctrl+F8
Create boot application for 'Application [Device: PLC Logic]'
Download 'Application [Device: PLC Logic]'
Online Change 'Application [Device: PLC Logic]'
Source Download...
Start 'Application [Device: PLC Logic]' F5
Stop 'Application [Device: PLC Logic]' Shift+F8
Single Cycle Ctrl+F5
Multiple Download...
Reset warm 'Application [Device: PLC Logic]'
Reset cold 'Application [Device: PLC Logic]'
Reset origin 'Application [Device: PLC Logic]'
New Breakpoint...
Toggle Breakpoint F9
Step Over F10
Step Into F8
Step Out Shift+F10
Run to Cursor
Set next statement
Show Current Statement
Write Values Ctrl+F7
Force Values F7
Unforce all Values Alt+F7
Add All Forces To Watchlist
Display Mode
Devi
Type
BOOL
INT
BOOL
BOOL
BOOL
BOOL
BOOL
BOOL
Project

Logout_from_device.bmp

Figure 3.3.6-4 Logging out from the Device

Logic is still running on PLC device.

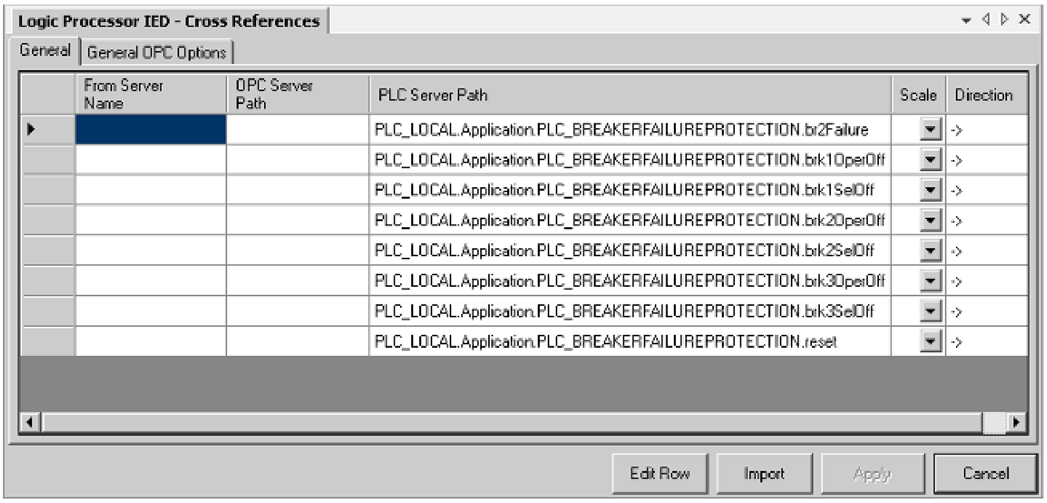
3.4. Making cross-references

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.



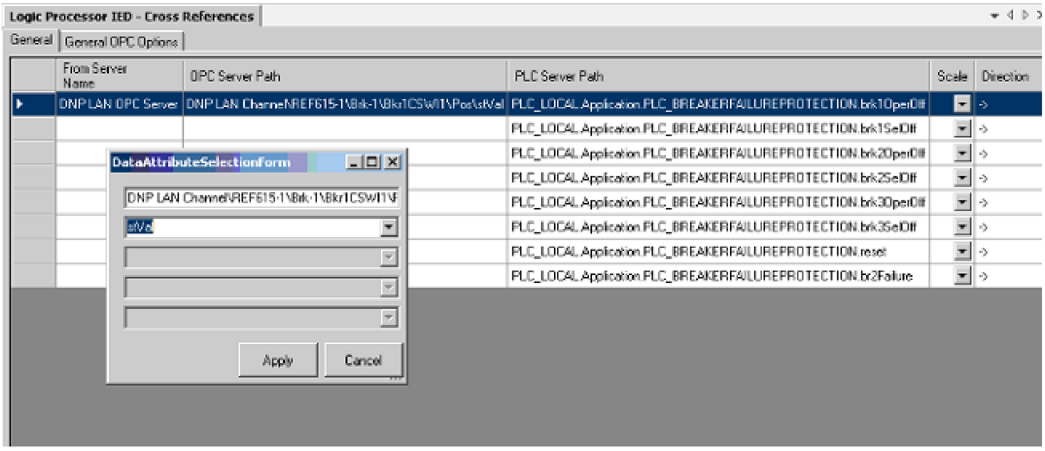
From Server Name	OPC Server Path	PLC Server Path	Scale	Direction
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.br2Failure		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1OperOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1SelOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2OperOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2SelOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3OperOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3SelOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.reset		->

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.


Logic Processor IED - Cross References
General
General OPC Options
DataAttributeSelectionForm
Apply
Cancel

From Server Name	OPC Server Path	PLC Server Path	Scale	Direction
DNP LAN OPC Server	DNP LAN Channel\REF615-1\Brk-1\Bkr1CSWI1\Pos\stVal	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1OperOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1SelOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2OperOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2SelOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3OperOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3SelOff		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.reset		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.br2Failure		->

DNP LAN Channel\REF615-1\Brk-1\Bkr1CSWI1\F

stVal

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:

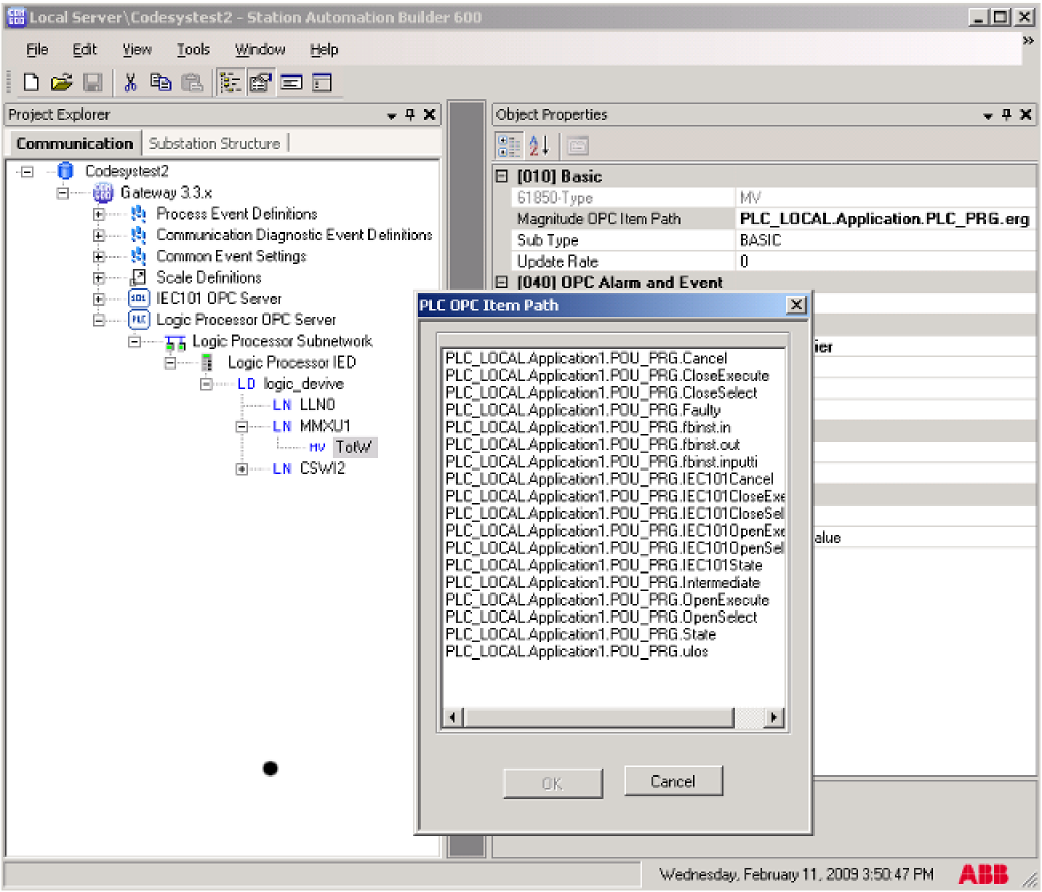
- With SPS and SPC: attribute **EaCnt**, that is incremented with 1 each time **stVal** is updated with the value **True**. The value is reset to 0 each time **stVal** is updated with the value **False**. This allows logic processor to catch several consecutive updates of the object with the value **True**.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



[010] Basic	
61850-Type	MV
Magnitude OPC Item Path	PLC_LOCAL.Application.PLC_PRG.erg
Sub Type	BASIC
Update Rate	0
[040] OPC Alarm and Event	

```
PLC_LOCAL.Application1.POU_PRG.Cancel
PLC_LOCAL.Application1.POU_PRG.CloseExecute
PLC_LOCAL.Application1.POU_PRG.CloseSelect
PLC_LOCAL.Application1.POU_PRG.Faulty
PLC_LOCAL.Application1.POU_PRG.fbinst.in
PLC_LOCAL.Application1.POU_PRG.fbinst.out
PLC_LOCAL.Application1.POU_PRG.fbinst.inputti
PLC_LOCAL.Application1.POU_PRG.IEC101Cancel
PLC_LOCAL.Application1.POU_PRG.IEC101CloseExe
PLC_LOCAL.Application1.POU_PRG.IEC101CloseSel
PLC_LOCAL.Application1.POU_PRG.IEC101OpenExe
PLC_LOCAL.Application1.POU_PRG.IEC101OpenSel
PLC_LOCAL.Application1.POU_PRG.IEC101State
PLC_LOCAL.Application1.POU_PRG.Intermediate
PLC_LOCAL.Application1.POU_PRG.OpenExecute
PLC_LOCAL.Application1.POU_PRG.OpenSelect
PLC_LOCAL.Application1.POU_PRG.State
PLC_LOCAL.Application1.POU_PRG.ulos
```

Add_OPC_Item_Path.bmp

Figure 3.5-1 Selecting the logic variable to the OPC Item Path

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

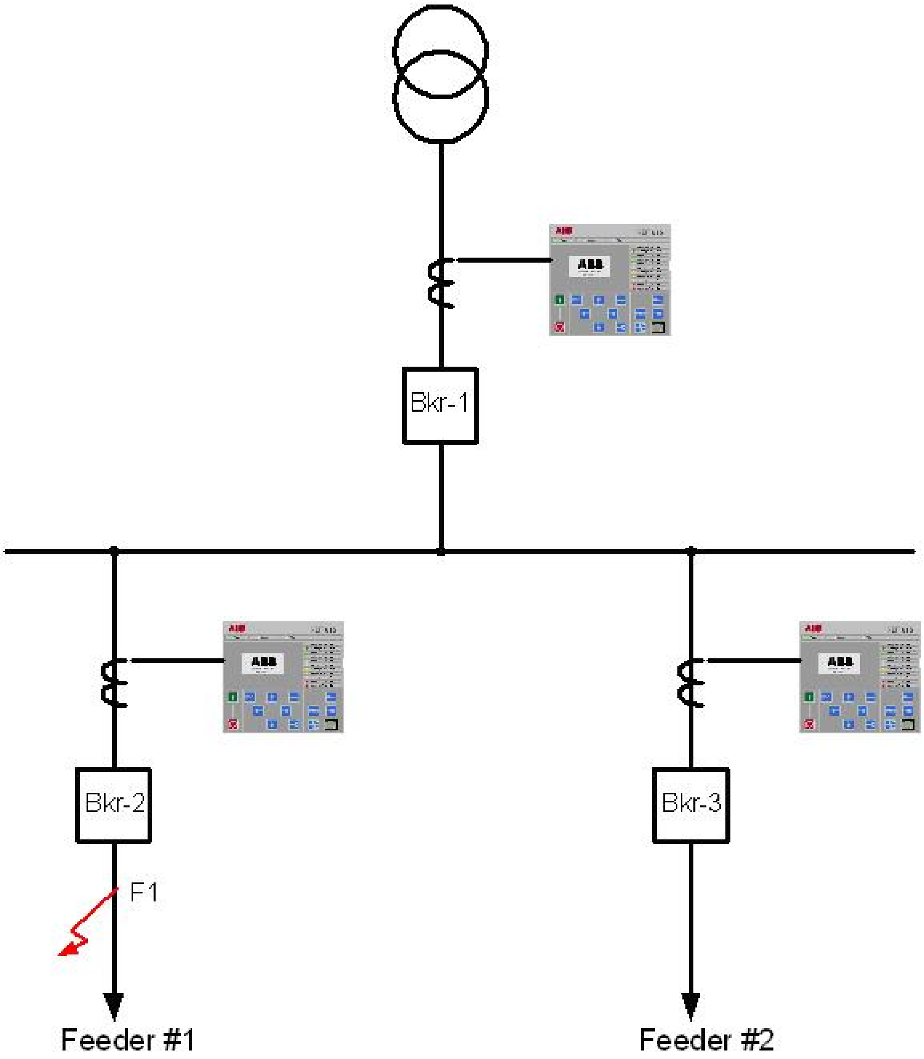
When communication configuration of IEDs, logic configuration of Logic Processor, and the cross-references between them are done, go to SAB600, select Management on Gateway, and download the configuration to COM600.

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.



Logic_Processor_System_Diagram.jpg

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

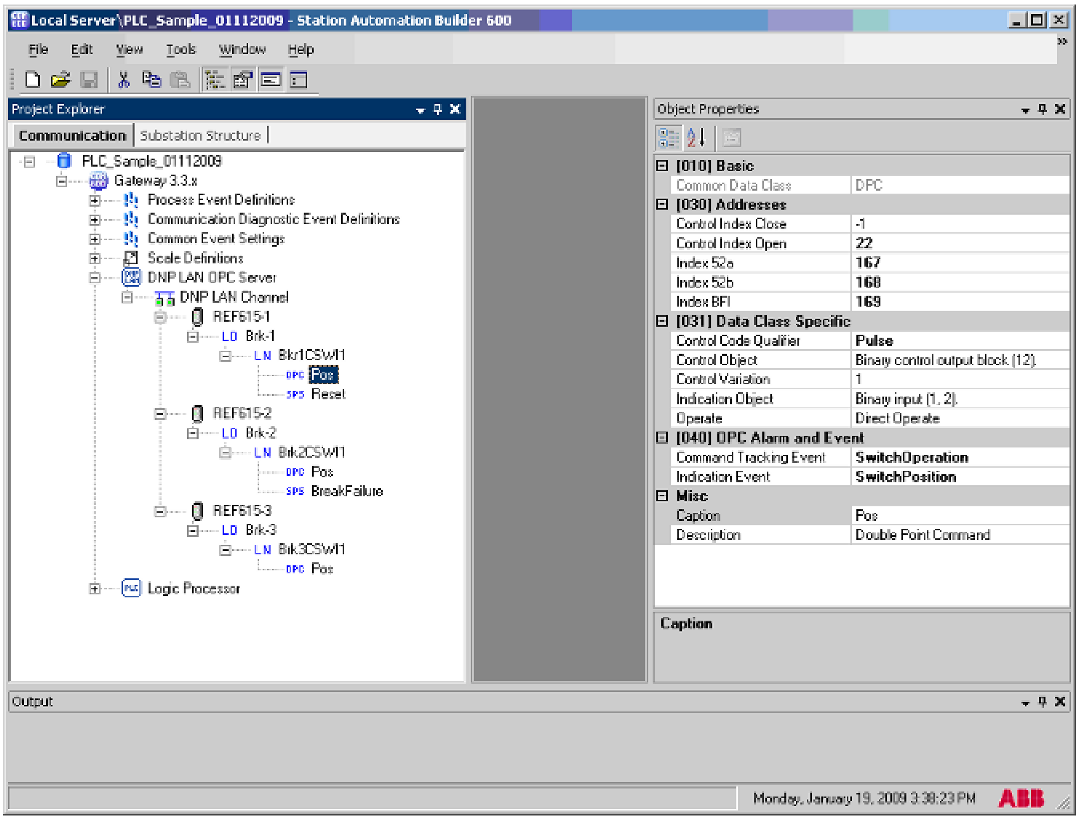
In the system shown in Figure 3.3.2-1 with three breakers, the failure protection plan is the following:

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.



[010] Basic	
Common Data Class	DPC
[030] Addresses	
Control Index Close	-1
Control Index Open	22
Index 52a	167
Index 52b	168
Index BFI	169
[031] Data Class Specific	
Control Code Qualifier	Pulse
Control Object	Binary control output block (12)
Control Variation	1
Indication Object	Binary input (1, 2)
Operate	Direct Operate
[040] OPC Alarm and Event	
Command Tracking Event	SwitchOperation
Indication Event	SwitchPosition
Misc	
Caption	Pos
Description	Double Point Command

Configuring_IEDs.bmp

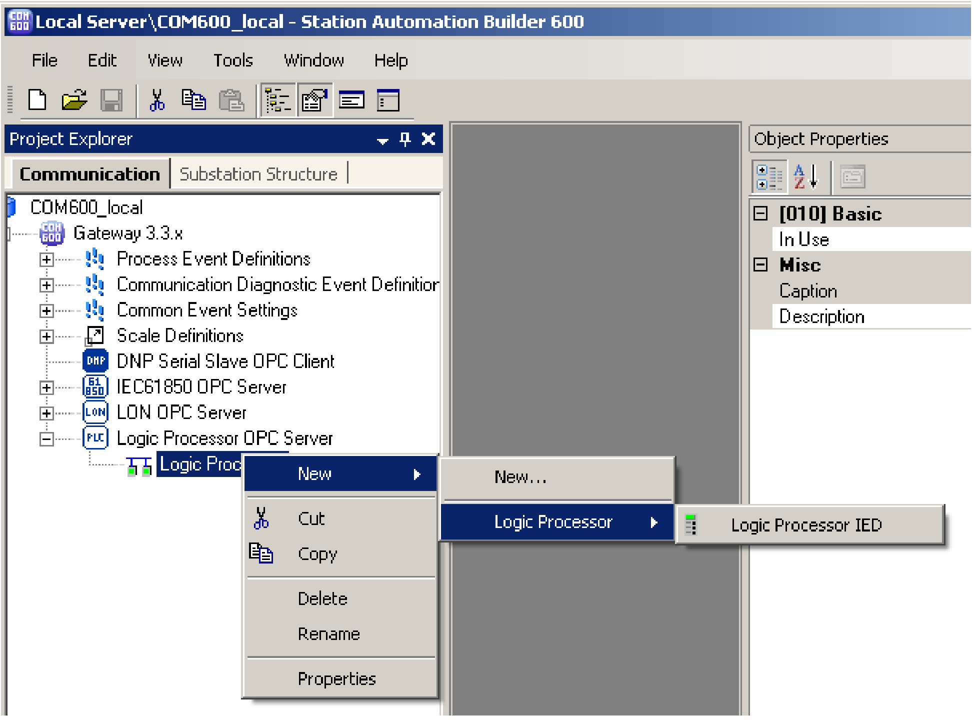
Figure 4.2-1 Configuring IEDs on SAB600

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.



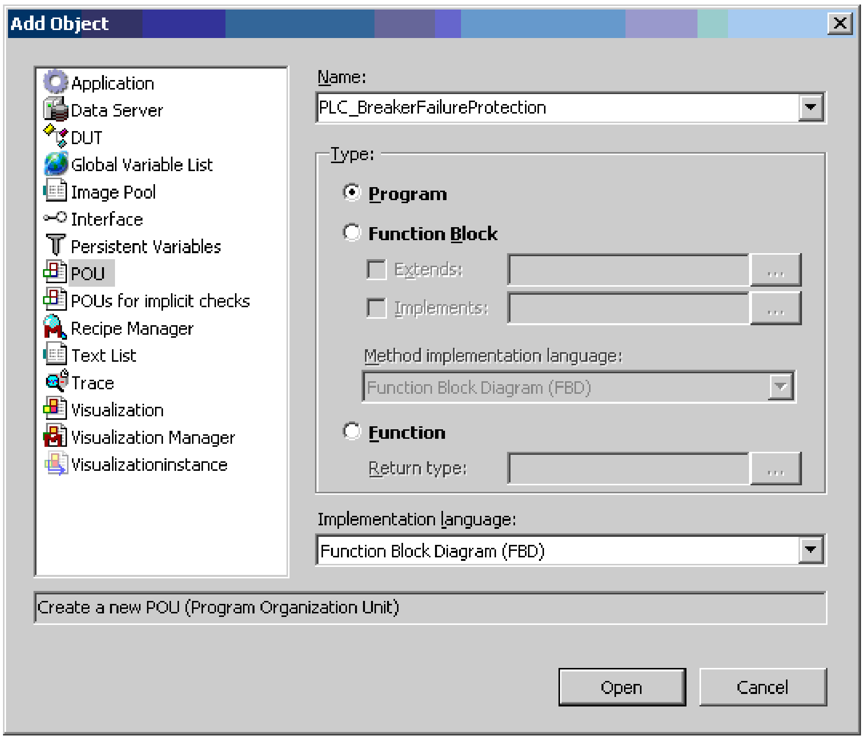
Add_Logic_Processor_IED.bmp

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.



Add Object

Application
Data Server
DUT
Global Variable List
Image Pool
Interface
Persistent Variables
POU
POUs for implicit checks
Recipe Manager
Text List
Trace
Visualization
Visualization Manager
Visualizationinstance

Name: PLC_BreakerFailureProtection

Type:

● **Program**

○ **Function Block**

□ Extends:

□ Implements:

Method implementation language: Function Block Diagram (FBD)

○ **Function**

Return type:

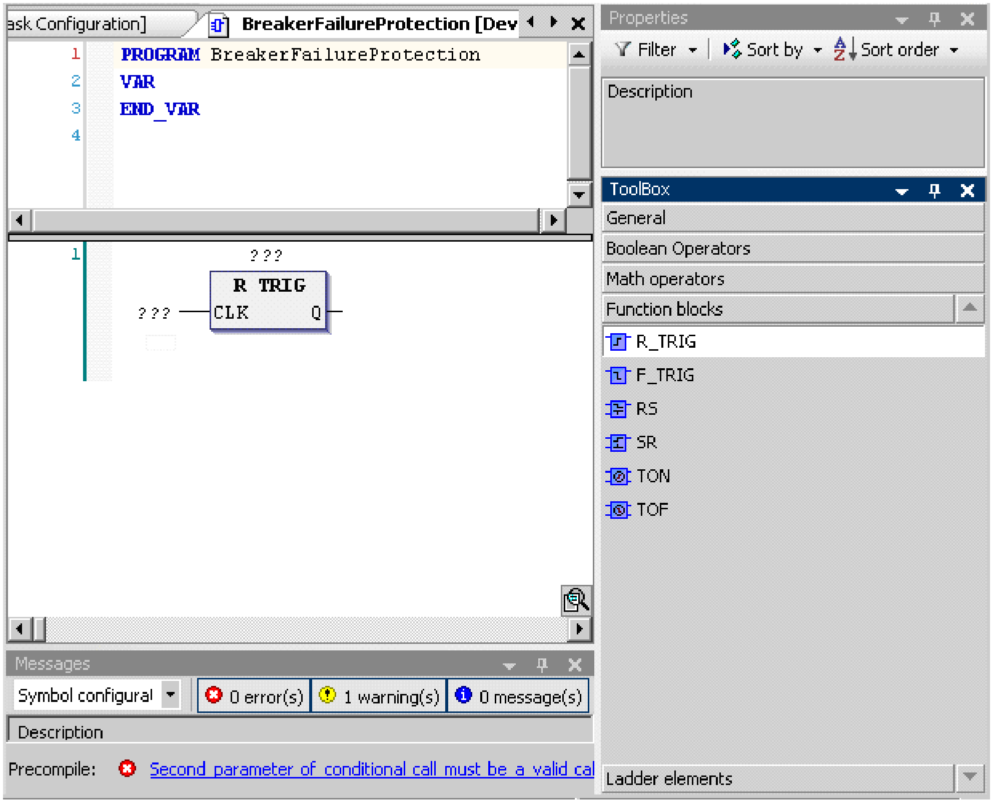
Implementation language: Function Block Diagram (FBD)

Create a new POU (Program Organization Unit)

Adding_Function_Block_Diagram.bmp

Figure 4.4-1 Adding a function block diagram

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.



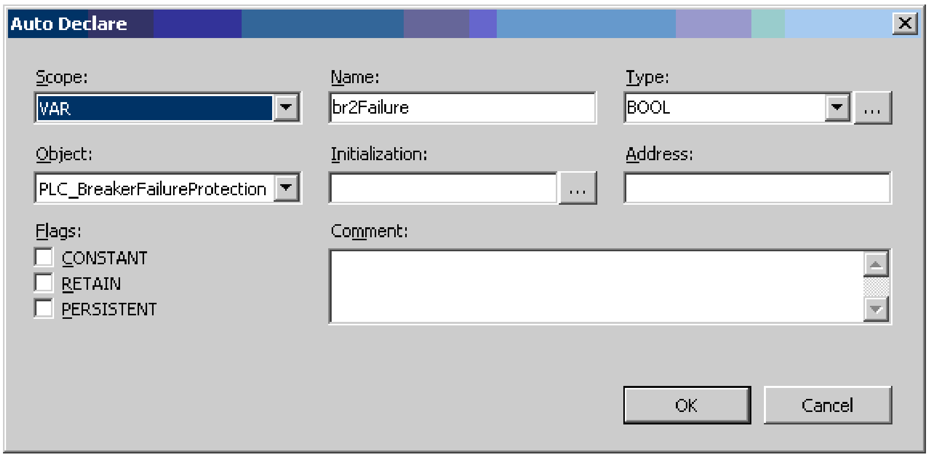
```
PROGRAM BreakerFailureProtection
VAR
END_VAR
```

Precompile: Second parameter of conditional call must be a valid cal

Adding_R_TRIG.bmp

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.



Scope: VAR

Name: br2Failure

Type: BOOL

Object: PLC_BreakerFailureProtection

Initialization:

Address:

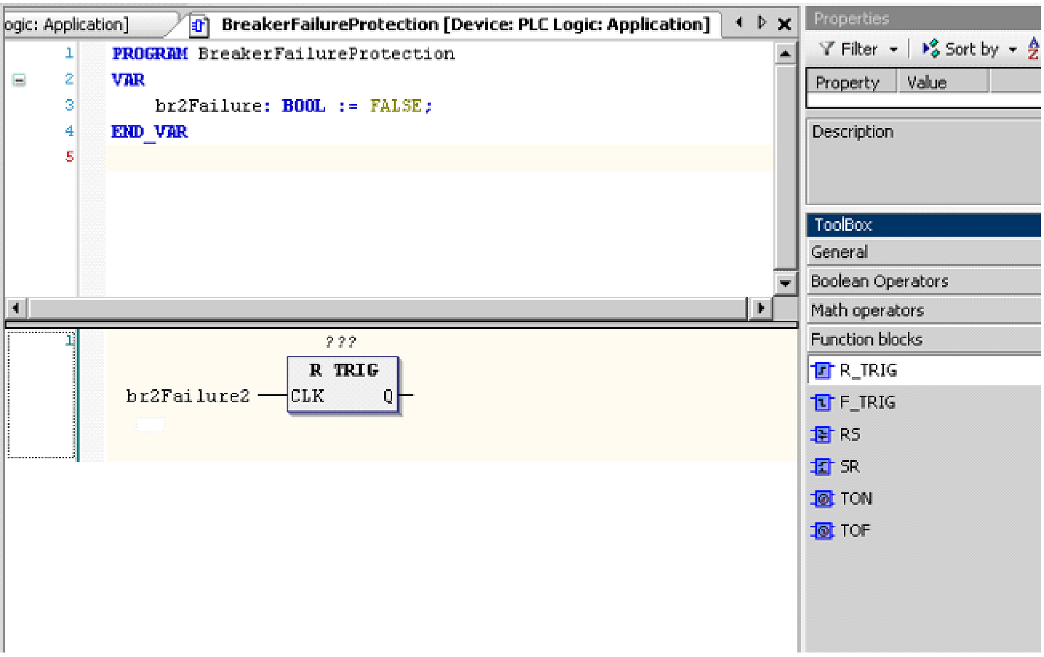
Flags:
□ CONSTANT
□ RETAIN
□ PERSISTENT

Comment:

Auto_Declare_Dialog.bmp

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**.



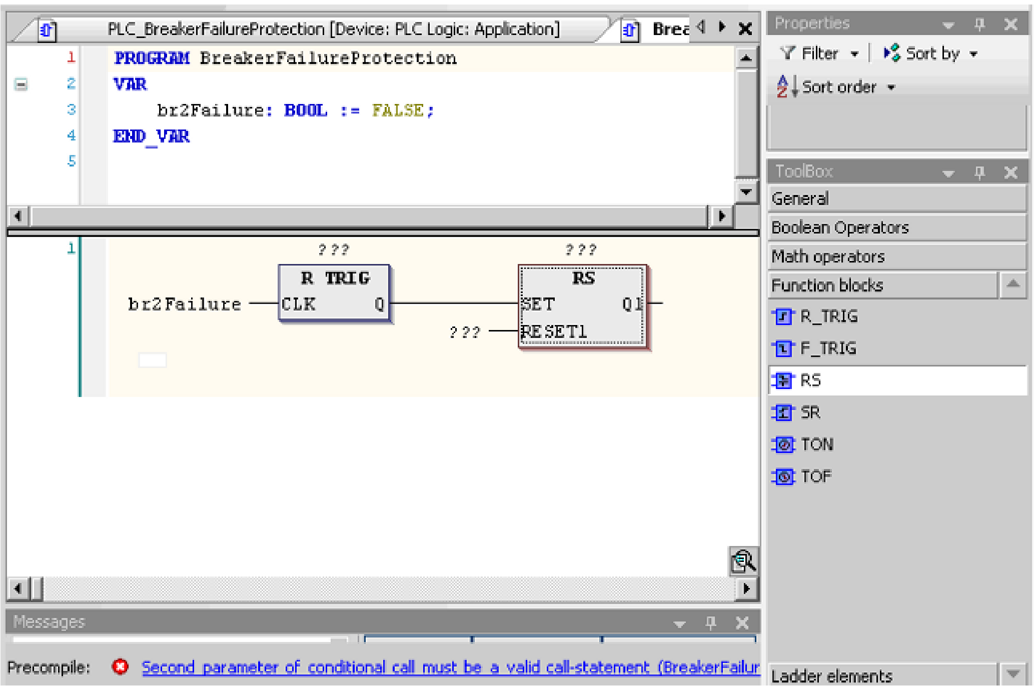
```
PROGRAM BreakerFailureProtection
VAR
    br2Failure: BOOL := FALSE;
END_VAR
```

Declaration_Part_FBD.bmp

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.



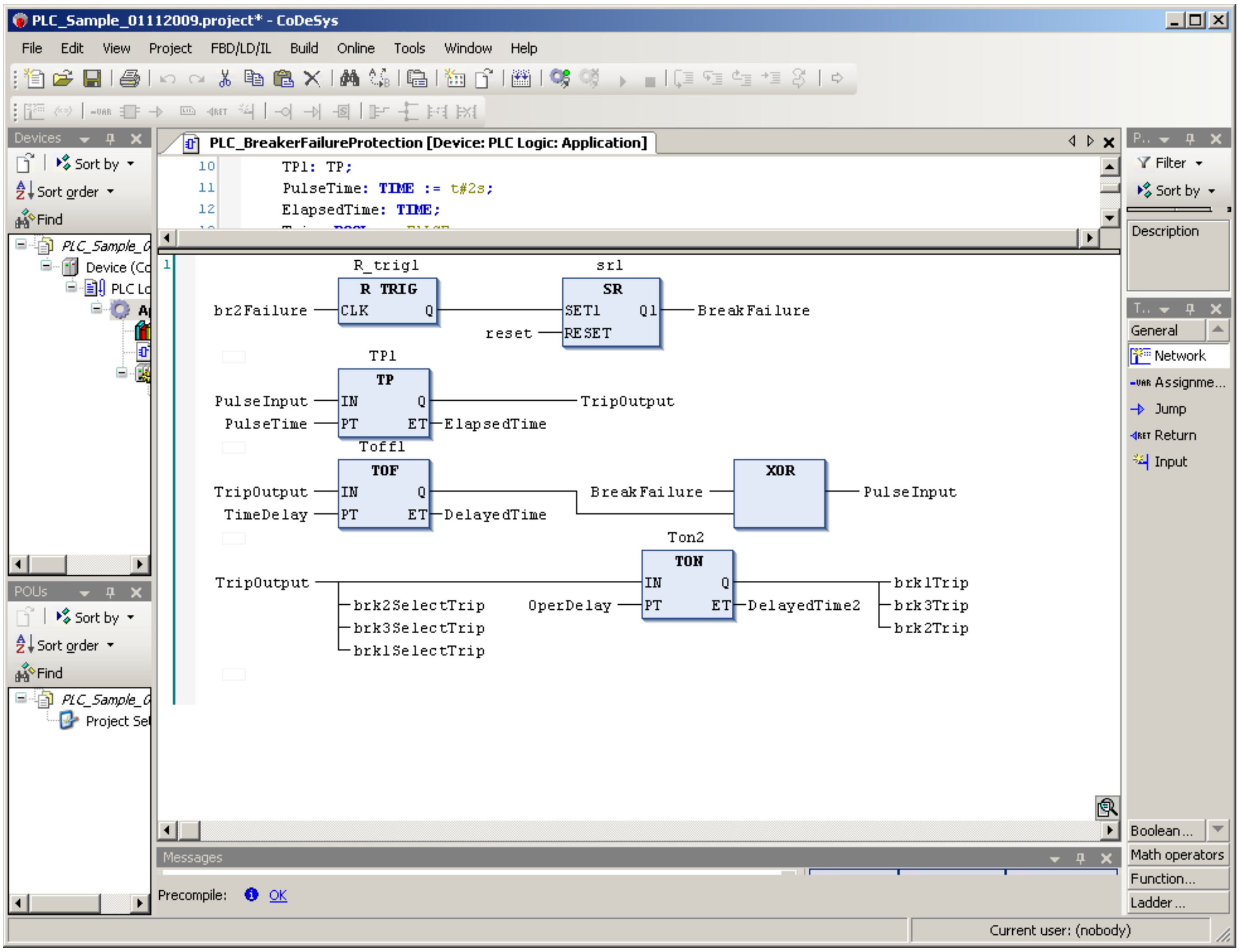
```
PROGRAM BreakerFailureProtection
VAR
    br2Failure: BOOL := FALSE;
END_VAR
```

Precompile: Second parameter of conditional call must be a valid call-statement (BreakerFailur

Adding_AND_gate.bmp

Figure 4.4-5 Adding Reset Set function block

The logic can be built as follows.



```
TP1: TP;
PulseTime: TIME := t#2s;
ElapsedTime: TIME;
```

Precompile: OK

Current user: (nobody)

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.

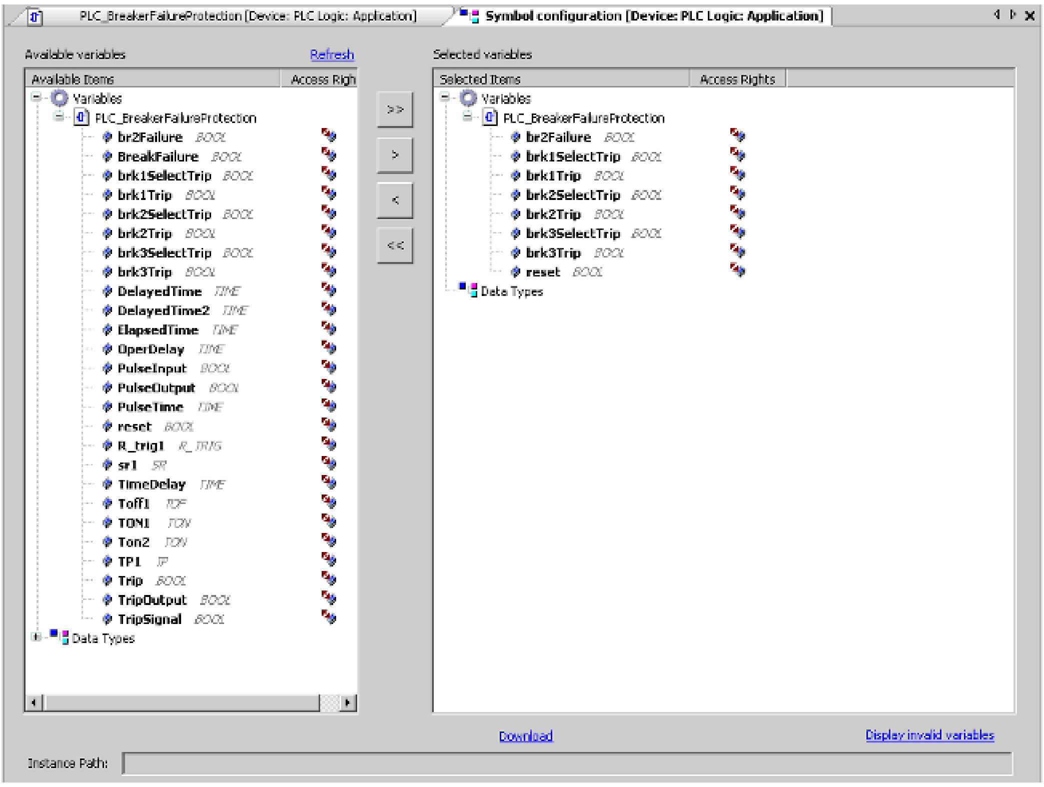
The second and third row are to generate an On/Off square wave (Tripping signal) from a break failure status by using Timer Pulse, Timer and XOR gate function blocks.

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 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.


PLC_BreakerFailureProtection [Device: PLC Logic: Application]
Symbol configuration [Device: PLC Logic: Application]
Available variables
Refresh
Selected variables
>>
>
<
<<
Download
Display invalid variables
Instance Path:

Available Items	Access Righ
Variables	
PLC_BreakerFailureProtection	
br2Failure BOOL	
BreakFailure BOOL	
brk1SelectTrip BOOL	
brk1Trip BOOL	
brk2SelectTrip BOOL	
brk2Trip BOOL	
brk3SelectTrip BOOL	
brk3Trip BOOL	
DelayedTime TIME	
DelayedTime2 TIME	
ElapsedTime TIME	
OperDelay TIME	
PulseInput BOOL	
PulseOutput BOOL	
PulseTime TIME	
reset BOOL	
R_trig1 R_TRIG	
sr1 SR	
TimeDelay TIME	
Toff1 TOF	
TON1 TON	
Ton2 TON	
TP1 TP	
Trip BOOL	
TripOutput BOOL	
TripSignal BOOL	
Data Types	

Selected Items	Access Rights
Variables	
PLC_BreakerFailureProtection	
br2Failure BOOL	
brk1SelectTrip BOOL	
brk1Trip BOOL	
brk2SelectTrip BOOL	
brk2Trip BOOL	
brk3SelectTrip BOOL	
brk3Trip BOOL	
reset BOOL	
Data Types	

Adding_symbol_configuration_ex.bmp

Figure 4.5-1 Adding symbol configuration

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

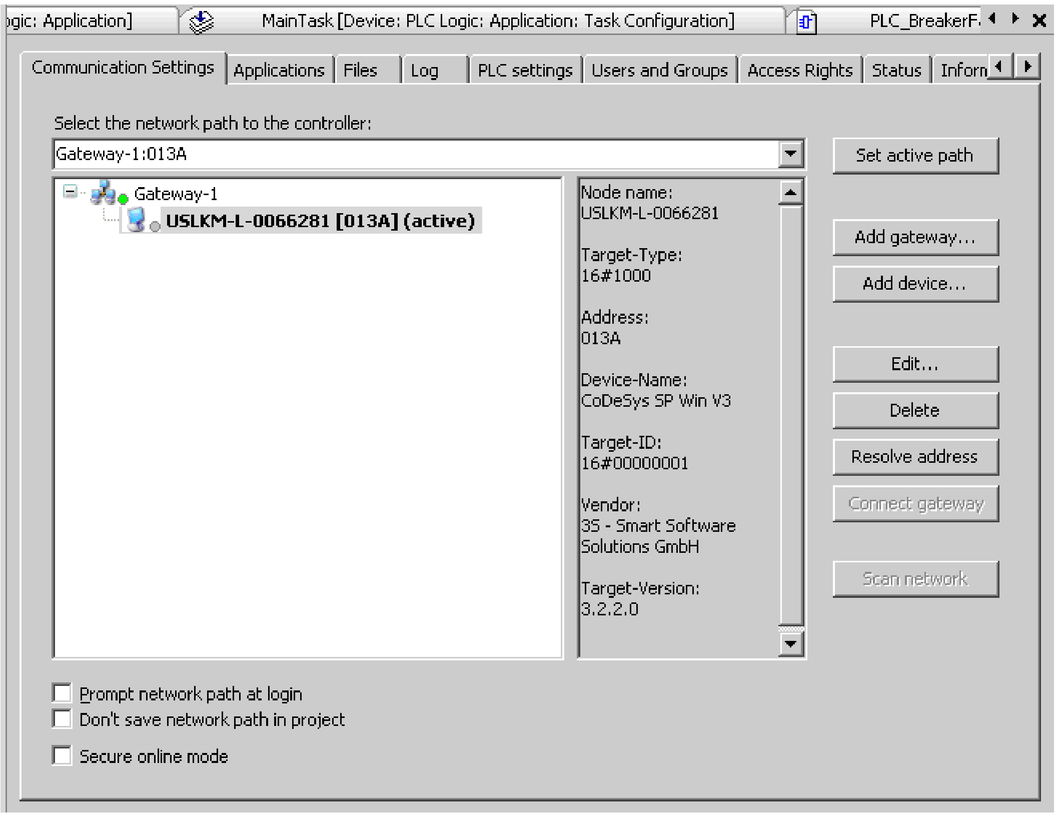
To set the active application:

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.



Select the network path to the controller:

Gateway-1:013A

- Gateway-1
  - **USLKM-L-0066281 [013A] (active)**

Node name:
USLKM-L-0066281

Target-Type:
16#1000

Address:
013A

Device-Name:
CoDeSys SP Win V3

Target-ID:
16#00000001

Vendor:
3S - Smart Software Solutions GmbH

Target-Version:
3.2.2.0

☐ Prompt network path at login
☐ Don't save network path in project
☐ Secure online mode

Setting_communication_channel.bmp

Figure 4.7-1 Setting communication channel

4.8. Building and activating the application

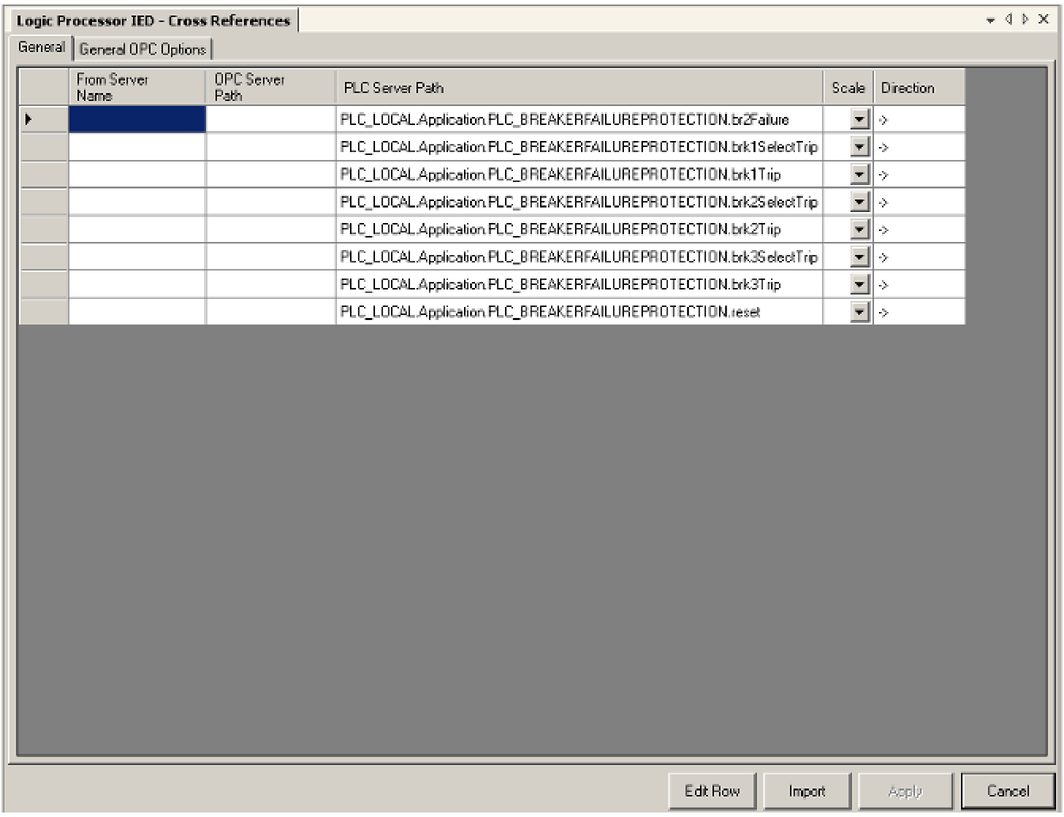
1. Select **Build -> Build Application** to compile the project.
   All objects belonging to the application is syntactically checked. Any potential error messages or warnings are displayed within the message view.
2. If the project is built successfully without error, select **Online -> Login to Application** 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.

4.9. 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.



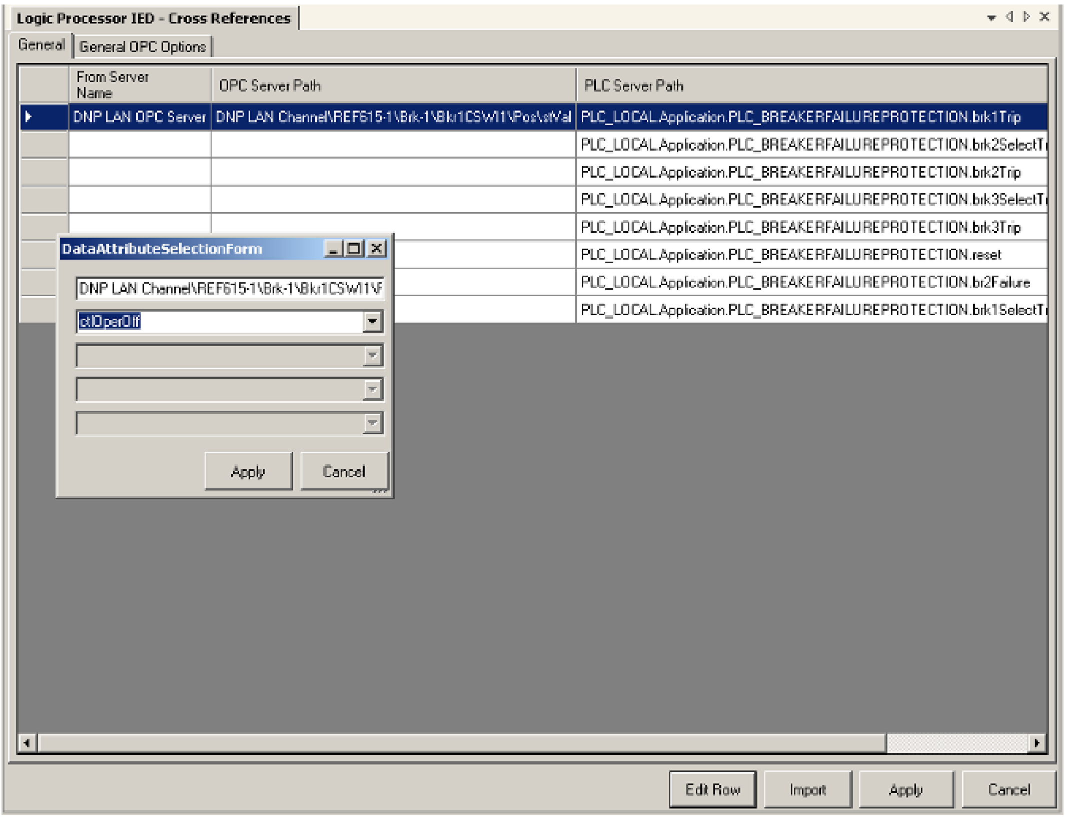
From Server Name	OPC Server Path	PLC Server Path	Scale	Direction
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.br2Failure		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1SelectTrip		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1Trip		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2SelectTrip		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2Trip		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3SelectTrip		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3Trip		->
		PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.reset		->

Example_cross_reference.bmp

Figure 4.9-1 Importing Logic Processor variables

   The **Import** button is used for fetching another symbol file.

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.



Logic Processor IED - Cross References

General | General OPC Options

	From Server Name	OPC Server Path	PLC Server Path
▶	DNP LAN OPC Server	DNP LAN Channel\REF615-1\Brk-1\Bkr1CSWI1\Pos\stVal	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1Trip
			PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2SelectT
			PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2Trip
			PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3SelectT
			PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3Trip
			PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.reset
			PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.br2Failure
			PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1SelectT

DataAttributeSelectionForm

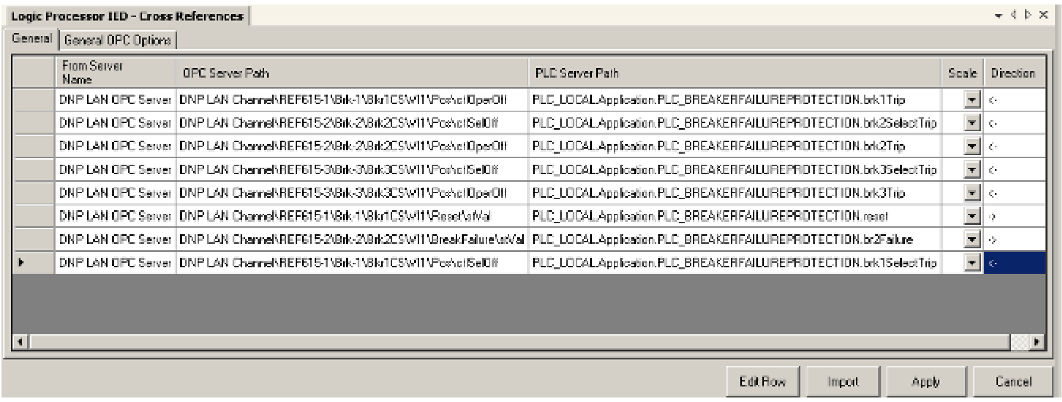
DNP LAN Channel\REF615-1\Brk-1\Bkr1CSWI1\F

ctlOperOff

Editor_Row.bmp

Figure 4.9-2 Cross references tool

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 <-.



Logic Processor IED - Cross References

General | General OPC Options

	From Server Name	OPC Server Path	PLC Server Path	Scale	Direction
	DNP LAN OPC Server	DNP LAN Channel\REF615-1\Brk-1\Bkr1CSWI1\Pos\ctlOperOff	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1Trip		<-
	DNP LAN OPC Server	DNP LAN Channel\REF615-2\Brk-2\Brk2CSWI1\Pos\ctlSelOff	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2SelectTrip		<-
	DNP LAN OPC Server	DNP LAN Channel\REF615-2\Brk-2\Brk2CSWI1\Pos\ctlOperOff	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk2Trip		<-
	DNP LAN OPC Server	DNP LAN Channel\REF615-3\Brk-3\Brk3CSWI1\Pos\ctlSelOff	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3SelectTrip		<-
	DNP LAN OPC Server	DNP LAN Channel\REF615-3\Brk-3\Brk3CSWI1\Pos\ctlOperOff	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk3Trip		<-
	DNP LAN OPC Server	DNP LAN Channel\REF615-1\Brk-1\Bkr1CSWI1\Reset\stVal	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.reset		->
	DNP LAN OPC Server	DNP LAN Channel\REF615-2\Brk-2\Brk2CSWI1\BreakFailure\stVal	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.br2Failure		->
▶	DNP LAN OPC Server	DNP LAN Channel\REF615-1\Brk-1\Bkr1CSWI1\Pos\ctlSelOff	PLC_LOCAL.Application.PLC_BREAKERFAILUREPROTECTION.brk1SelectTrip		<-

Cross_reference_all_variables.bmp

Figure 4.9-3 Cross-references with all variables

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 20, 22

B

building
  communication structure 15

C

CoDeSys 16
CoDeSys SP runtime 12
cross-reference 28, 44

L

logic 35
Logic Processor
  CoDeSys SP runtime 11
logic programming
  logic editor 12

M

monitoring and debugging 24

V

variable
  OPC symbol 17, 41
virtual data object
  variable 30

Contact us

ABB Oy
Distribution Automation
P.O. Box 699
FI-65101 VAASA, FINLAND
Tel. +358 10 22 11
Fax. +358 10 224 1094

ABB Inc.
Distribution Automation
655 Century Point
Lake Mary, FL 32746, USA
Tel: +1 407 732 2000
Fax: +1 407 732 2335

www.abb.com/substationautomation

Power and productivity
for a better world™