The Drive & Control Company

Rexroth
Bosch Group

IndraWorks 14VRS
IndraLogic 2G

ProVi Diagnostics

Application Description
R911344037

Edition 02

Title
IndraWorks 14VRS
IndraLogic 2G
ProVi Diagnostics

Type of Documentation
Application Description

Document Typecode
DOK-IWORKS-PROVI2G*V14-AP02-EN-P

Internal File Reference
RS-a4d0f20e6f1f15940a6846a500ed0d3f-2-en-US-5

Change Record

Edition	Release Date	Note
01	2015-05	First edition
02	2015-11	Update of the "Variables supported for the ProVi SFC diagnostics"

Editorial Department
Engineering Automation Systems PLC Programming Interface ViGe (KaWa/MePe)

Table of Contents

Table of Contents

Table of Contents

Page

1 About this documentation

1.1 Validity of the documentation

Overview on target groups and product phases

In the following illustration, the framed activities, product phases and target groups refer to this documentation.

Example: In the product phase "Engineering", the target group "programmer" can execute the activities "parameterize", "program" and "configure" using this documentation.

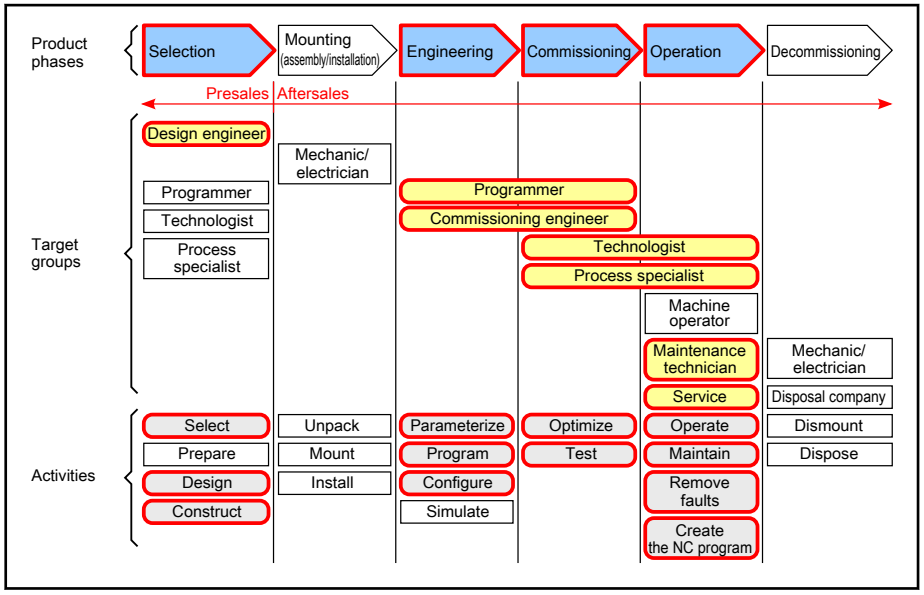


Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

1.2 Required and supplementing documentation XLC/MLC

Documentation titles with type codes and part numbers

System overview
Rexroth IndraLogic XLC 14VRS System Overview DOK-XLC***-SYSTEM**V14-PRRS-EN-P, R911341483 This documentation provides an overview on the possible hardware/software components of the automation system IndraLogic XLC of the mentioned version. The documentation facilitates assembling a system.
Rexroth IndraMotion MLC 14VRS System Overview DOK-MLC***-SYSTEM**V14-PRRS-EN-P, R911341504 This documentation provides an overview of the hardware/software components of the automation system IndraMotion MLC in the mentioned version. The documentation facilitates assembling a system.

Tab. 1-1: XCL/MLC documentation overview - System overview

About this documentation

First steps
Rexroth IndraMotion MLC 14VRS First Steps
DOK-MLC***-F*STEP**V14-CORS-EN-P, R911341517
This documentation describes the first steps of the IndraMotion MLC and the RobotControl. It includes the hardware and software prerequisites as well as the creation of a project.
Rexroth IndraLogic XLC 14VRS First Steps
DOK-XLC***-F*STEP**V14-CORS-EN-P, R911341489
This documentation describes the first steps of the IndraLogic XLC. It includes the hardware and software prerequisites as well as the creation of a project.
Rexroth IndraMotion MLC 14VRS Project Conversion
DOK-XLCMLC-PROCONV*V14-APRS-EN-P, R911341494
This documentation describes the project conversion of the IndraMotion MLC.

Tab. 1-2: XCL/MLC documentation overview - First steps

Engineering
Rexroth IndraWorks 14VRS Software Installation
DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286
This documentation describes the IndraWorks installation.
Rexroth IndraWorks 14VRS Engineering
DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566
This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Function Description
DOK-XLCMLC-FUNC****V14-APRS-DE-P, R911341700
This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.

Tab. 1-3: XCL/MLC documentation overview - Engineering

Diagnostics and service
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481
This documentation includes all control diagnostics implemented in the control systems IndraLogic XLC and IndraMotion MLC.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Parameters
DOK-XLCMLC-PARAM***V14-RERS-EN-P, R911341479
This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.

Diagnostics and service
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Commissioning
DOK-XLCMLC-STARTUP*V14-CORS-EN-P, R911341502
This documentation describes the steps to commission and service the IndraMotion MLC and IndraLogic XLC systems. It includes checklists for frequent tasks and a detailed description of the steps.
Rexroth IndraWorks IndraMotion Service Tool
DOK-IWORKS-IMST*******-APRS-DE-P, R911341383
This documentation describes the IndraMotion Service Tool (IMST). IMST is a web-based diagnostic tool used to access a control system via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. The PC has to use at least Internet Explorer 8, Firefox 3.5 or a higher version. The following control variants are supported: • IndraMotion MLC L25/L45/L65/L75/XM2/VPx • IndraLogic XLC L25/L45/L65/L75

Tab. 1-4: XLC/MLC documentation overview - Diagnostics and service

Firmware
Rexroth IndraDrive MPx-20, Functions [1)]
DOK-INDRV*-MP*-20VRS**-AP01-EN-P, R911345608
This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-20, MPM-20, MPC-20 and MPE-20.
Rexroth IndraDrive MPx-16 to MPx-20 und PSB, Parameters
DOK-INDRV*-GEN1-PARA**-RERS-EN-P, R911328651
This documentation contains the descriptions of all parameters implemented in the firmware for drive controllers of the IndraDrive range. It supports the drive controller parameterization.
Rexroth IndraDrive MPx-16 to MPx-20 and PSB, Diagnostic Messages
DOK-INDRV*-GEN1-DIAG**-RERS-EN-P, R911326738
This documentation contains the descriptions of all diagnostic messages implemented in the following firmware products: • Drive controller firmware variants MPx-16 to MPx-20 • PSB firmware variants for "HMU05" and "KMV03" type supply units • Firmware variants for "HMV01" type supply units It assists machine operators and installation programmers with troubleshooting.
Hardware
Rexroth IndraDrive Control Sections CSB02, CSE02, CSH02, CDB02
DOK-INDRV*-Cxx02******-PRRS-EN-P, R911338962
Rexroth IndraDrive Cs Drive Systems with HCS01
DOK-INDRV*-HCS01******-PRRS-EN-P, R911322210
Rexroth IndraDrive Mi Drive Systems with KCU02, KSM02, KMS02/03, KMV03
DOK-INDRV*-KCU02+KSM02-PRxx-EN-P, R911335703
Rexroth IndraDrive ML Drive Systems with HMU05
DOK-INDRV*-Hxx05******-PRxx-EN-P, R911344279

1) In preparation

Tab. 1-5: Overview of documentations for drive controllers

About this documentation

PLC
Rexroth IndraWorks 14VRS IndraLogic 2G PLC Programming System
DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571
This documentation describes the PLC programming tool IndraLogic 2G and its usage. It includes the basic use, first steps, visualization, menu items and editors.
Rexroth IndraWorks 14VRS Basic Libraries IndraLogic 2G
DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920
This documentation describes the system-comprehensive PLC libraries.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries
DOK-XLCMLC-FUNLIB**V14-LIRS-EN-P, R911341491
This documentation describes the function blocks, functions and data types of the RIL_CommonTypes, ML_Base and ML_PLCopen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.

Tab. 1-6: XCL/MLC documentation overview - PLC

Field buses
Rexroth IndraWorks 14VRS Field Buses
DOK-IWORKS-FB******V14-APRS-EN-P, R911341485
This documentation describes the field bus and local periphery connections supported by the IndraLogic XLC, IndraMotion MLC and IndraMotion MTX systems. The focus of this documentation is on the configuration, parameterization, commissioning and the diagnostics of the different periphery connections. This documentation is the basis for the online help.
Rexroth IndraWorks 14VRS Field Buses Libraries
DOK-IWORKS-FB*LIB**V14-LIRS-EN-P, R911343575
This documentation describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDPSlave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.
Rexroth IndraWorks 12VRS FDT Container
DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398
This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.
Sercos System Manual for I/O Devices
DOK-CONTRL-ILS3*******-APxx-EN-P, R911333512
This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a Sercos interface.

xx Corresponding edition

Tab. 1-7: XCL/MLC documentation overview - Field buses

HMI
Rexroth IndraWorks 14VRS HMI
DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569
This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.
Rexroth IndraWorks 14VRS WinStudio
DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585
This "User Manual and Technical Reference Book" facilitates working with the "Rexroth WinStudio" software for optimal results. This document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS HMI Connection
DOK-XLCMLC-HMI*****V14-APRS-EN-P, R911341497
This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.

Tab. 1-8: XCL/MLC documentation overview - HMI

Technology
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template
DOK-XLCMLC-TF*GAT**V14-APRS-EN-P, R911341487
This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.
Rexroth IndraMotion MLC 14VRS Technology Libraries
DOK-MLC***-TF*LIB**V14-LIRS-EN-P, R911341511
This documentation describes the function blocks, functions and data types of the "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.
Rexroth IndraWorks Energy Efficiency Management 14VRS
DOK-IWORKS-4EE*****V14-APRS-EN-P, R911339229
This documentation describes the HMI connection of the energy management system and the function blocks, functions and data types of the "RIL_4EE " library. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS RegisterControl (Library)
DOK-MLC***-REGI*CO*V14-LIRS-EN-P, R911341509
This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.
Rexroth IndraMotion MLC 14VRS RegisterControl (Application Description)
DOK-MLC***-REGI*CO*V14-APRS-EN-P, R911341507
This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react to errors and query diagnostics. This documentation is written for machine setters and machine operators.
Rexroth IndraMotion MLC 14VRS Winder Function Application
DOK-MLC***-TF*WIND*V14-APRS-EN-P, R911341513
This application-related system documentation describes the application of the winder technology functions.

About this documentation

Technology
Rexroth IndraWorks 13VRS CamBuilder
DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291
This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.
Rexroth IndraMotion MLC 14VRS Robot Control V2
DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588
This documentation provides information about the Robot Control V2. The focus is on PLCopen programming. The program structure, variables, functions, motion statements and the required system parameters are described. This documentation also includes the description of the "ML_Robot" library. Robot Control V2 is supported by a kinematic interface. The components of this interface are described in the "ML_KinTech" library.

Tab. 1-9: XCL/MLC documentation overview - Technology

Open Core Engineering - OCE
Rexroth IndraLogic XLC IndraMotion MLC 14VRS First Steps MLPI
DOK-XLCMLC-MLPI****V14-CORS-EN-P, R911341033
This document describes the installation and the commissioning of the MLPI interface.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Automation Interface
DOK-XLCMLC-AUT*INT*V14-APRS-EN-P, R911341499
This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.

Tab. 1-10: XCL/MLC documentation overview - Open Core Engineering - OCE

SafeLogic
Rexroth IndraWorks SafeLogic 14VRS First Steps
DOK-IWORKS-SL*STEP*V14-CORS-EN-P, R911341520
This documentation describes the initial commissioning of the Safety function module and its external Safety periphery. Taking the example of a project, all required steps to commission a simple Safety application are executed.
Rexroth IndraMotion MLC IndraLogic XLC 14VRS SafeLogic System Overview
DOK-XLCMLC-SL**SYS*V14-PRRS-EN-P, R911341696
This documentation describes the project planning, configuration, creation and commissioning of safety-related devices.
Rexroth IndraWorks 14VRS SafeLogic Project Configuration
DOK-IWORKS-SL**PRJ*V14-APRS-EN-P, R911341694
This documentation describes the configuration of Safety applications.
Rexroth IndraControl SafeLogic Function Module
DOK-CONTRL-SL**FM*****-ITRS-DE-P, R911336576
This documentation is provided with the hardware and describes the commissioning, the installation, the decommissioning and disposal of the hardware.

Tab. 1-11: XCL/MLC documentation overview - SafeLogic

Hardware
Rexroth IndraControl L45/L65/L85 Control
DOK-CONTRL-ICL45L65L85-PRxx-EN-P, R911332116
This documentation describes the IndraControl L45/L65/L85 controls.
Rexroth IndraControl L25
DOK-CONTRL-IC*L25*****-PRxx-EN-P, R911328474
This documentation describes the IndraControl L25 controls.
Rexroth IndraControl Lxx 14VRS Function Modules
DOK-CONTRL-FM*LXX**V14-APRS-EN-P, R911341583
This documentation describes all function modules of the Lxx controls including engineering and diagnostics.

xx Corresponding edition

Tab. 1-12: XCL/MLC documentation overview - Hardware

Hydraulics
Rexroth IndraMotion MLC 14VRS Hydraulic Functions
DOK-MLC***-TF*HMOT*V14-LIRS-EN-P, R911341907
This documentation describes the function blocks, functions and data types of the MH_TechHydrBase and MH_TechHydrMotion libraries. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS Sequential Programming
DOK-XLCMLC-SEQPROG*V14-LIRS-EN-P, R911341909
This documentation describes the sequential programming options, also when using GAT, and the MH_HydrControlProg library.

Tab. 1-13: XCL/MLC documentation overview - Hydraulics

1.3 Use of the safety instructions

1.3.1 Structure of the safety instructions

The safety instructions are structured as follows:

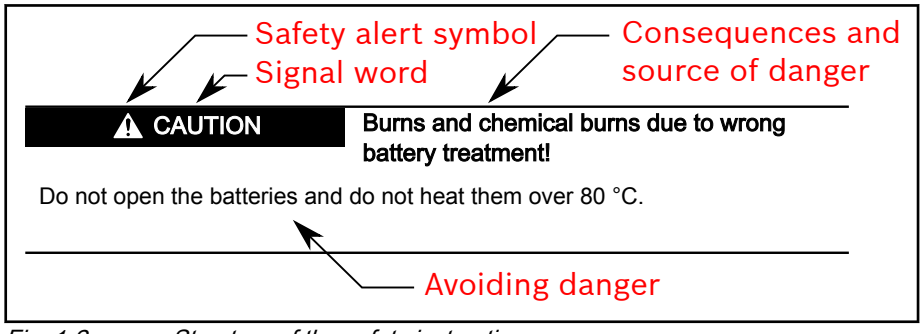


Fig. 1-2: Structure of the safety instructions

1.3.2 Explaining signal words and safety alert symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

The signal word is used to draw attention to the safety instruction and also provides information on the severity of the hazard.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words danger,warning and caution is used to alert the reader to personal injury hazards.

About this documentation

⚠ DANGER

In the event of non-compliance with this safety instruction, death or serious injury **will** occur.

⚠ WARNING

In the event of non-compliance with this safety instruction, death or serious injury **will** occur.

⚠ CAUTION

In the event of non-compliance with this safety instruction, minor or moderate injury can occur.

NOTICE

In the event of non-compliance with this safety instruction, material damage can occur.

1.3.3 Symbols used

Hints are represented as follows:

This is an information.

Tips are represented as follows:

This is a tip.

1.3.4 Signal graphic explanation on the device

Prior to the installation and commissioning of the device, refer to the device documentation.

1.4 Names an abbreviations

Term	Explanation
CANopen	Field bus
DeviceNet	Field bus
Ethernet	Communication interface
IWE	IndraWorks Engineering
IWO	IndraWorks Operation
NC	Numerical Control
OEM	Original Equipment Manufacturer
Profibus DP	Field bus
Sercos	Sercos (Serial Realtime Communication System) interface is a world-wide standardized interface for the communication between controls and drives

Tab. 1-14: Names and abbreviations used

1.5 Customer feedback

Customer requests, comments or suggestions for improvement are of great importance to us. Please email your feedback on the documentations to Feedback.Documentation@boschrexroth.de. Directly insert comments in the electronic PDF document and send the PDF file to Bosch Rexroth.

2 System description

2.1 Rexroth ProVi diagnostics

For an economical and efficient system operation, classified diagnostic messages about the system state have to be provided to the machine operator or to the person commissioning the system. For PLC-controlled systems, Rexroth ProVi diagnostics provides the required tool.

ProVi messages are issued by the PLC and displayed in the Rexroth IndraWorks HMI. Depending on the configuration, messages are logged in a logbook.

After the “Important instructions on use" and the "Safety instructions", the chapter "First Steps" describes how easy ProVi messages can be programmed using an example. The following sections describe all programming points from the insertion of a ProVi message into the PLC program up to the representation of the message with a criteria analysis in the "Diagnostics" window of the HMI device.

3 ProVi diagnostics – First steps

General information

This chapter describes how to program the ProVi diagnostics in a PLC project. The example is based on the project of the chapter "Writing a little program" in the documentation "PLC Programming with Rexroth IndraLogic" (part no. R911305036), extending this PLC project by ProVi messages.

Task

In the program, the correct operation of the machine must repeatedly be confirmed in certain intervals. If not confirmed within the interval, a warning is issued first, followed by the stop of the machine shortly afterwards.

Currently, the warning and the stopped machine are only displayed in the IndraLogic visualization.

Extend the program so that the warning and the error are displayed as ProVi message on an IndraWorks HMI device.

Preparation

The project "FirstSteps.pro" has to be opened from an IndraWorks project.

Enabling diagnostics for IndraLogic 2G

In case of the IndraLogic 2G, the programming interface for the control is integrated in IndraWorks.

To enable the diagnostics for the PLC project, select the control in the project tree and select the entry **ProVi SFC diagnostics** in the menu item **Diagnostics**:

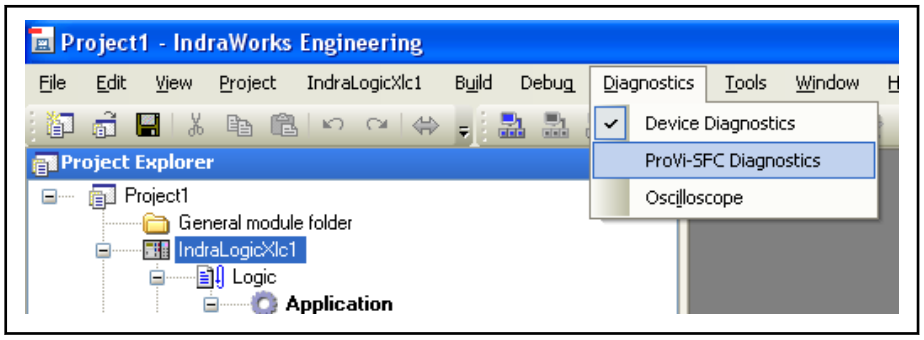


Fig. 3-1: Opening the diagnostic configuration

The enabled diagnostics can be identified by the selected menu item.

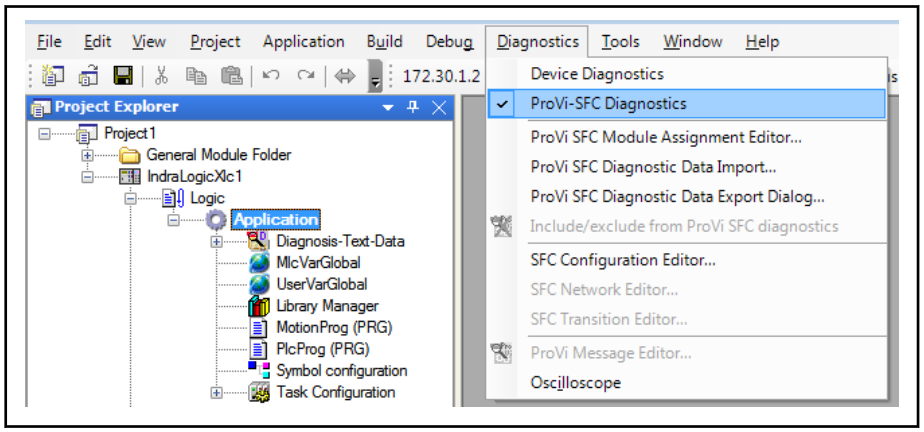


Fig. 3-2: ProVi SFC diagnostics

After the diagnostics has been enabled, the submenu items "ProVi message editor" and "ProVi SFC text data editor" are enabled under the respective menu item.

Please note that the menu item "ProVi message editor..." is only enabled if the cursor in the program is located at a position relevant for the ProVi message.

ProVi diagnostics – First steps

Editing ProVi messages

1. Open the ProVi input dialog via the menu item **Diagnostics ▸ ProVi message editor...**. Please take into consideration that the cursor in the PLC program is not on the assignment triggering the ProVi message. Only then, the menu item is enabled and the ProVi message is automatically inserted at the correct position. The "ProVi message editor" of the IndraLogic 2G can also be opened via right click.

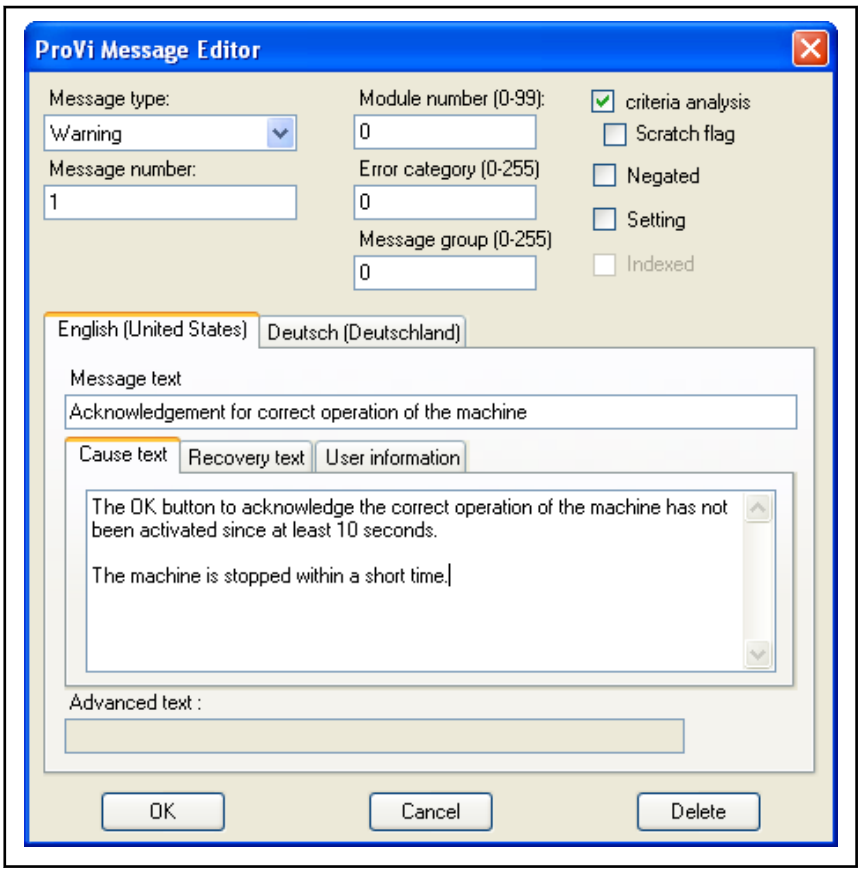


Fig. 3-3: Entering ProVi message

2. Select, for instance, "Warning" as message type from the "Message type" drop-down list.

   The following text inputs relate to the German message texts. They can also be applied to foreign language message texts.

☞ The text has to be entered in the languages which are to be displayed in IndraWorks Operation.

Text changes are only applied after "Create diagnostic data" in IndraWorks Operation.

---

3. In the "German (Germany)" tab, enter the message text into the input field "Message text", for example:

   "Acknowledgement for correct operation of the machine missing."

4. Select the tab "Cause text" and enter the cause text, for example:

   "The **OK** button to acknowledge the correct operation of the machine has not been pressed for at least 10 seconds. The machine will be stopped soon."

5. Select the tab "Recovery text" and enter the recovery text, for example:

   If the machine still functions correctly, click on **OK**.

6. Close the ProVi input dialog with **OK**.
7. The result of the ProVi is automatically inserted in the correct position in the network or the correct line in case of ST.

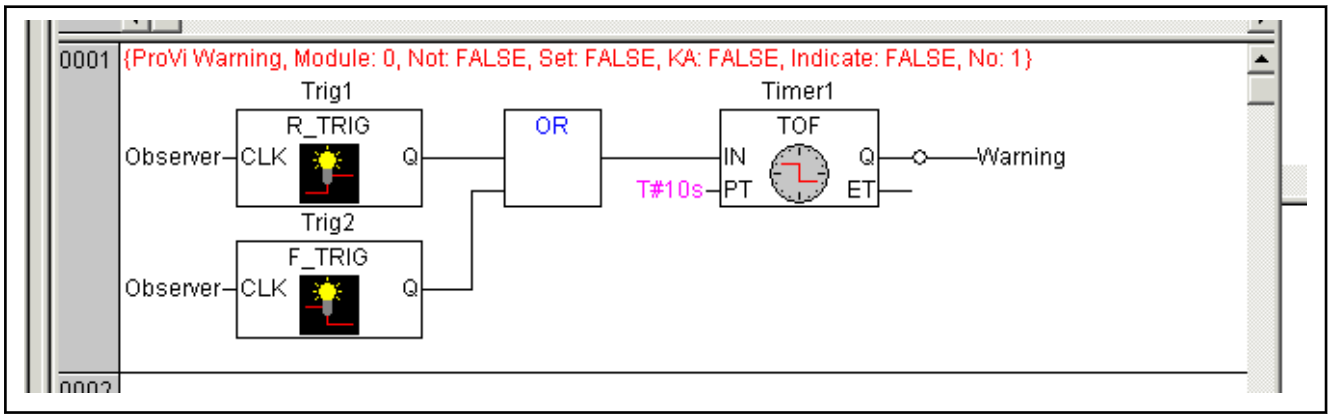


Fig. 3-4: ProVi message in network 1

Proceed in the same manner for the second network using the following exemplary texts:

Field	Content
Message type	"Error"
Message text	Machine is not running correctly
Reason text	**OK** was not pressed for at least 15 seconds. *This can be caused by the following:* 1. Machine is not running correctly. 2. No reaction by the machine operator. Machine was stopped!
Recovery text	*Eliminate the error cause:* 1. Eliminate the machine error. 2. Ensure that the machine is operated by the machine operator. Then, click **OK**

Tab. 3-1: ProVi message for the second network

Diagnostic data generate IndraLogic 2G

Diagnostic data is automatically generated while compiling the PLC program.

IndraWorks Operation can be started if the following conditions apply:

- Diagnostic data generated correctly
- Project compiled without errors
- Project loaded successfully to the control
- Control was started
- Visualization data was transmitted and enabled successfully

The messages are displayed after starting IndraWorks Operation under the menu item **Diagnostics**:

ProVi diagnostics – First steps

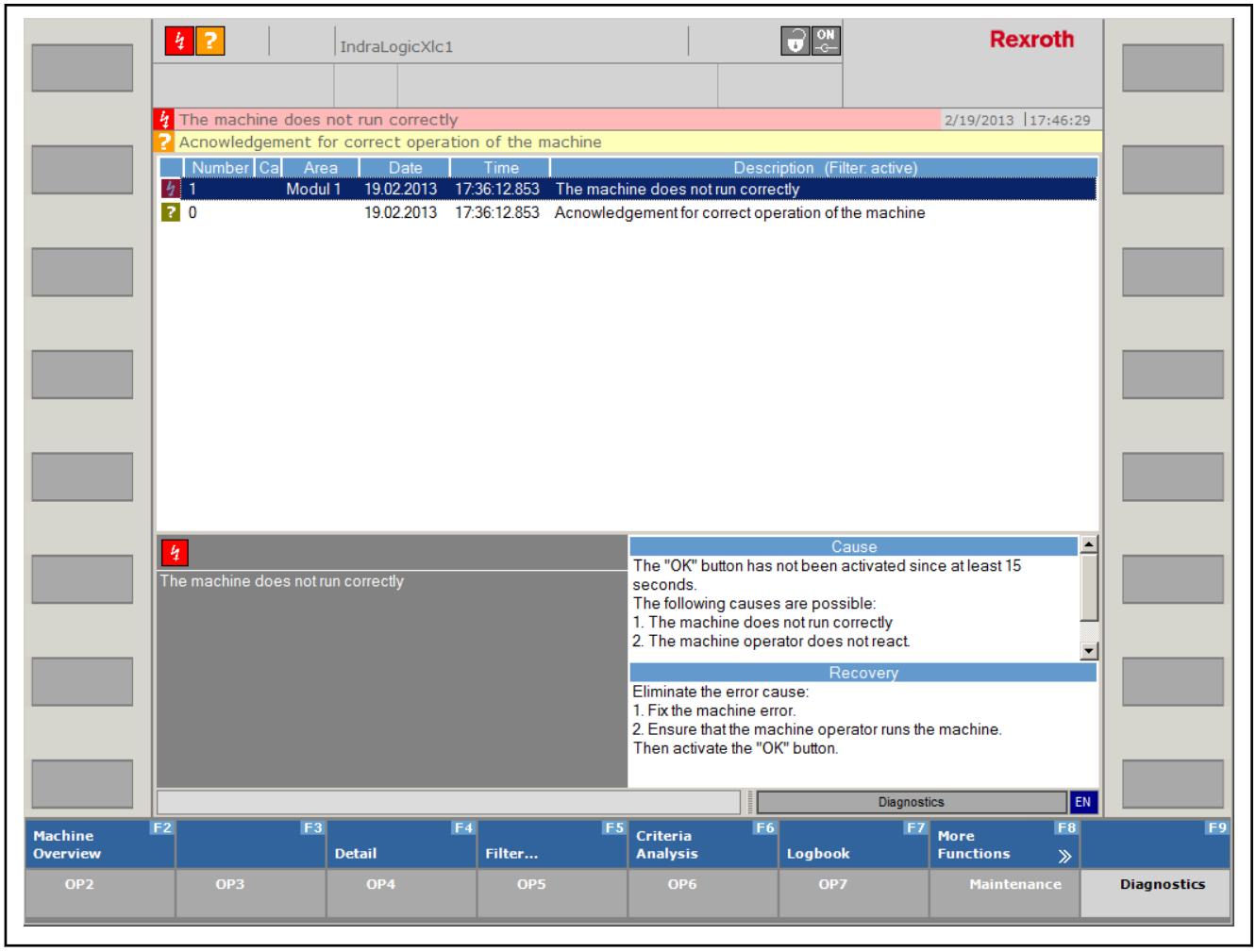


Fig. 3-5: Displaying the diagnostics on the HMI interface

4 ProVi messages

4.1 Overview

ProVi messages are messages issued by the PLC. These messages can be displayed in IndraWorks Operation. ProVi messages can also be recorded in a logbook.

ProVi messages can be divided into five different message types:

- Error
- Note
- Warning
- Start condition
- Setup diagnostics

All message types can be grouped into different modules.

Every message can additionally be assigned to an error category and to a message group.

ProVi messages can be programmed as non-expiring. Thus, the message remains present until it is reset via a function block call.

The message texts can be entered in several languages into IndraLogic.

4.2 What is a ProVi message?

A ProVi message is a PLC message that is issued depending on the current result (CR) of a logic expression.

All necessary inputs can be written directly into the PLC program.

When is a message triggered?

A message can be triggered in two different ways:

1. At a positive edge of the current result
2. At a negative edge of the current result (negated ProVi message)

How long is a message pending?

Two different reasons can cause the message not to be pendent anymore:

1. If the condition (CR = TRUE or FALSE) no longer applies to the message.

   If the program code issuing the message is not edited anymore, the message remains displayed, even if the actual condition is no longer met.

2. If the message is reset (non-expiring ProVi message).

   The messages can be reset using function blocks. Different criteria can be defined to determine the message(s) to be reset.

   If the condition for the message is still fulfilled, the message is triggered again immediately.

Where is the message analyzed?

In contrast to many other diagnostic systems, the messages are analyzed directly when editing the corresponding program code and not at the end of the cycle. Thus, scratch flags can also be used as triggering variable or to analyze the sequence of issued messages within one cycle.

The original message is not necessarily linked to a variable. All that is required is a Boolean CR. It can, for example, also be used directly as a jump.

ProVi messages

Grouping messages

ProVi messages can be grouped according to different criteria:

1. Message type
   - Error
   - Note
   - Warning
   - Start condition
   - Setup diagnostics

☞ The message type is the uppermost group and has to be specified. All other groups are optional and do not have to be used.

2. Module

   This grouping is used as a logical assignment of the message to a certain machine component, e.g. the PLC controls several logically disconnected components of a machine and the diagnostics of one component is to be disconnected from the other component.

☞ If no modules are to be used, enter "0" into the ProVi input dialog.

1 to 99 are valid module numbers.

3. Error category

   Assigning the message to a specific error category, e.g. E-STOP, immediate stop, no clearance.

☞ If no error category is to be used, ignore the corresponding input in the ProVi input dialog.

0 to 255 are valid numbers.

4. Message group

   Assigning the message to a certain message group, e.g. electrical error, mechanical error.

☞ If no message group is to be used, ignore the corresponding input in the ProVi input dialog.

0 to 255 are valid numbers.

Multilingual texts

The ProVi message texts can be entered in several languages. The languages to be available can be configured in the IndraWorks project.

The texts can be exported and imported for translation within IndraWorks (see chapter 4.7.4 "Message text export for external processing" on page 53 and chapter "Importing message texts" on page 60).

Placeholders

Placeholders can be used in ProVi message texts (see chapter 4.7.6 "Placeholders in message text" on page 63). These placeholders are substituted by the respective values for display. Placeholders can be instance names, comments or the status of a variable for example.

Indirect addressing

When defining a ProVi message, the message number to be issued is normally indicated. For indirect addressing, a PLC variable is indicated whose status indicates the message number at the time of issuing.

At present, indirect addressing is not yet available.

Criteria analysis

If the criteria analysis is enabled for a ProVi message, the contacts that caused the issuing of the message are shown in the diagnostics. Only the affected network is analyzed. The variables may not be declared in the declaration ranges VAR_IN_OUT and VAR_TEMP.

Scratch flag

The “Scratch flag” option, which can be enabled in addition to the criteria analysis, considers the assignments within the POU. The auxiliary flags used for clarity reasons are resolved.

4.3 Programming ProVi messages

4.3.1 Prerequisites

To program a ProVi message, the diagnostics for the PLC project has to be enabled (see chapter 4.8 "Configuring diagnostics" on page 68). IndraLogic has to be opened via an IndraWorks project.

To use ProVi messages, no symbols are required in the symbol configuration. The only exception is the control configuration. If the variables used are directly declared at the I/O assemblies, they have to be output in the system configuration.

The diagnostic communication operates with internal control variables. Thus, do not subsequently change control and application name. If the name has to be changed, close the IndraWorks Operation Desktop and create and load diagnostic data again.

4.3.2 Definition of the supported variables for the ProVi SFC diagnostics

Declarable variable types

The ProVi SFC diagnostics supports the following variable types:

1. Standard types:

BOOL	DINT	LREAL
BYTE	LINT	DT
WORD	USINT	TOD
DWORD	UINT	TIME
LWORD	UDINT	POINTER
SINT	ULINT	STRING
INT	REAL	STRING[8] (with length specification)

To issue messages, only Boolean operations may be used irrespective of the programming code.

E.g. Boolean variables in ST:

```
{ProVi Error, Module: 1, Not: FALSE, Set: FALSE, CA: TRUE, Indicate: FALSE, No: 123}
bMyBool := bMyBool_1 OR bMyBool_2;
```

or different integer variables with the point operator:

```
{ProVi Warning, Module: 1, Not: FALSE, Set: FALSE, CA: TRUE, Indicate: FALSE, No: 456}
iMyInt.5 := dwMyDWord.6 AND byMyByte.7 OR bMyBool;
```

All standard types mentioned above can use variables as placeholders in message texts. However, these placeholders had to be declared in the project.

2. *Arrays:*
   - `ARRAY[`from .. to`] OF` XXX

     or `ARRAY[`from .. to`] OF ARRAY[`from .. to`] OF` XXX

     (XXX = standard types, function blocks or structures) "from" "to" - have to be standard types.

☞ *Multi-dimensional arrays are only supported up to the third dimension:*

- `ARRAY[`from.. to, from .. to`] OF` (standard types, function blocks or structures)
- `ARRAY[`from.. to, from .. to, from .. to`] OF` (standard types, function blocks or structures)

The array elements have to be addressed directly or via enum for the feasibility display and the online criteria analysis:

- `sMy_Struct.Array[5, 5].Struct.bMyBool`
- `sMy_Struct.Array[five, 5].Struct.bMyBool`

Indirect addressing is not supported:

- `sMy_Struct.Array[iMyInt].Struct.bMyBool`

Function blocks, in which messages are issued, may be instantiated using arrays. When declaring these arrays, only decimal specifications, local or global project constants from the project are permitted. The used constants have to be initialized with a decimal value.

Example of a supported declaration:

```
aMy_FB_Array: ARRAY[1..17] OF My_FB
aMy_FB_Array: ARRAY[1..Constant_local] OF MY_FB;
aMy_FB_Array:  ARRAY[Glob_ConstMin..Glob_ConstMax]  OF MY_FB;
```

Example of unsupported declarations:

```
aMy_FB_Array:           ARRAY[OtherProg.ConstMin..Library.GVL.ConstMax] OF MY_FB;
```

Constants from other POUs or libraries are not supported.

Example of the implementation or calling of such declared function blocks:

`Array_1[Array_2[Int]]` -"Array_2" has to be globally declared

`Array[Standard_Type].Array[Standard_Type]`

3. **Structures:**

   Example of a supported definition:

   ```
   TYPE MyStruct :
   Struct
   ```

   - `Standard types`
   - `Struct`

- `Array[Standard_Type..Standard_Type, Standard_Type..Standard_Type] OF ...`

```
End_Struct
END_TYPE
```

Example of supported declarations:

```
sMy_Struct : MyStruct;
```

Example of the implementation:

```
sMy_Struct.Array[5, five].Struct.Standard_Typ
```

Instantiation

Example:

`Program.InstanceFB.BOOL`

`Program.InstanceFB.InstanceFB_2.BOOL`

`Program.InstanceFB.InstanceFB_2.InstanceFB_3.BOOL`

`GVL.InstanceFB.BOOL`

`Program.InstanceFB[Index,Index2,Index3].bool` - "Index" has to be a number or an enum if this variable is used for the feasibility display or the criteria analysis.

☞ Using the criteria analysis (CA) and the feasibility display, the following applies to all variables involved:

- Only operations of Boolean variables can be analyzed
- The logic to be diagnosed has to be as simple as possible
- The variables have to be declared in the project
- Complex calculations, comprehensive and nested function block calls should be outsourced to the function blocks if possible
- Complex or indexed structure or array elements should be replaced by simple Boolean variables
- The deeper the nesting and the higher the number of elements, the longer the diagnostic data generation. Saving the states in case of a diagnostic issue requires longer accordingly

---

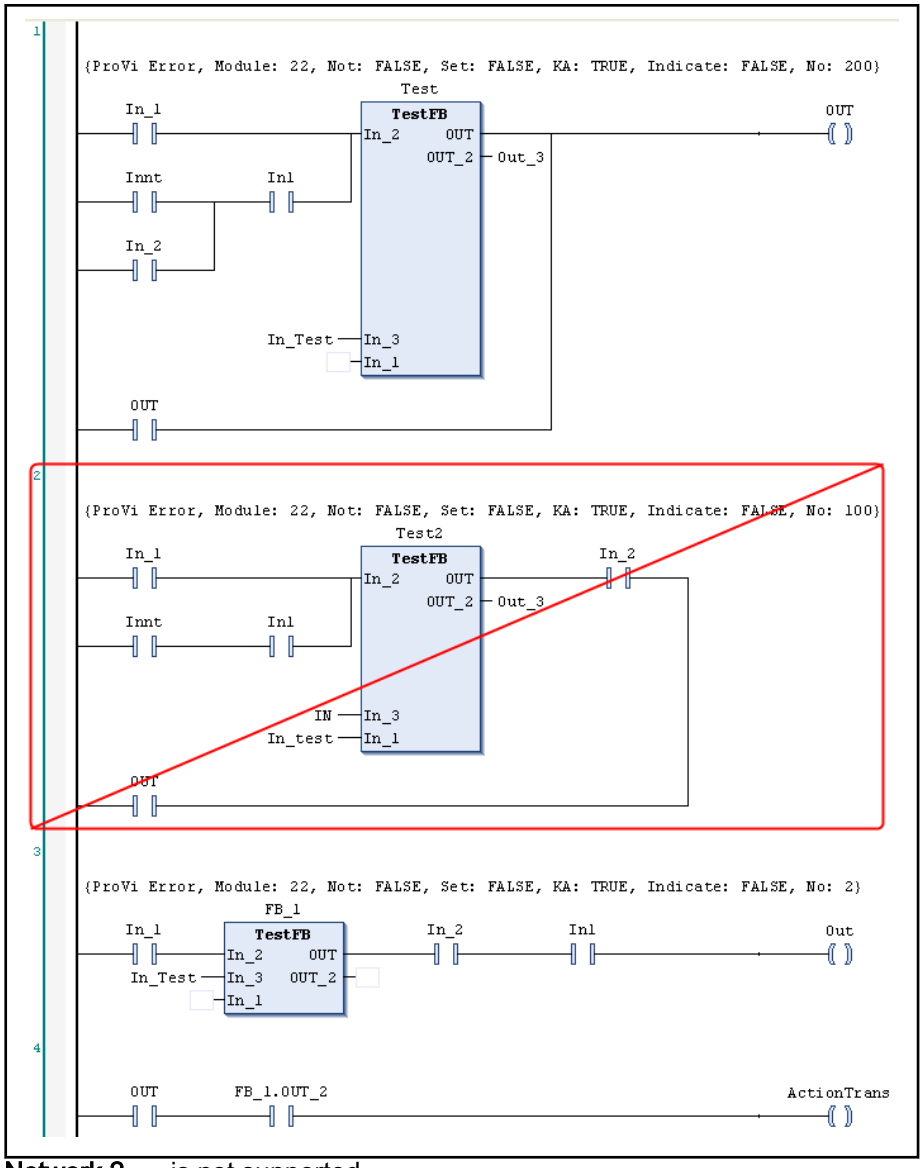
Restrictions

The following restrictions apply to the criteria analysis (CA) and the feasibility display:

- VAR_IN_OUT, VAR_TEMP variables, bit access and array variables with indirect addressing can be used to issue the ProVi or SFC messages. These messages are not supported by the criteria analysis. The frozen variable states are not determined and no criteria analysis is calculated. If the states of these VAR_IN_OUT, VAR_TEMP variables, bit access and array variables with indirect addressing should be used, these variables have to be projected via the intermediate flag.

  When these variables are used, a warning is output during compilation. If this warning is ignored, an error message is displayed in the visualization device under CA. This error message reports that the criteria analysis could not be calculated due to missing variables. In operating screens, a question mark signals this problem in the feasibility display.

- Non-Boolean variable types do not have an effect on the criteria analysis
- Box with "EN" (also with Boolean output) must not be used in diagnostic networks. Optionally, function blocks or functions can be used. The first input and the first output are considered by the diagnostics as if the input and the output would be connected internally
- POUs for the feasibility display created in the programming code "ST" may not contain any multi-line assignments, since the feasibility can otherwise not be calculated
- *Using function blocks, functions and line branching:*
  - The ProVi SFC diagnostics does not support any derived function blocks and may not be programmed as such
  - Only one function block may be used per network
  - Only the first input and the last output are considered (short-circuited). The current result of the input and output has to be assigned to a Boolean variable
  - The logic of the first input may only consist of Boolean variables
  - The first output has to be assigned
  - The function block may be provided with an unlimited number n of inputs/outputs
  - There may be no further logic at the inputs / outputs 2 to n
  - The inputs and outputs 2 to n may also remain unassigned
  - To ensure a unique network limit identification, the inputs/outputs may not be used at the first place in the subsequent network
  - Each function block output can be used to create a transition condition (last network in the screen)



Network 2 is not supported

Fig. 4-1: Example the supported and unsupported function block usage

- When using line branching in diagnostic networks, note that only the uppermost branch is responsible for issuing the message. The criteria analysis is only calculated for this branch. Do not confuse line branching with multiple assignments. The diagnostics supports multiple assignments. All assigned variables are displayed for the criteria analysis

Excluding from ProVi SFC diagnostics

When using assignments not supported by the ProVi SFC diagnostics, these networks or assignments can be completely excluded from the diagnostics. Open this functionality via the menu item "Diagnostics" or via the context menu in the relevant editors. The following pragma `{ProVi SFC diag: Exclude}` is inserted as response and for identification.

ProVi messages

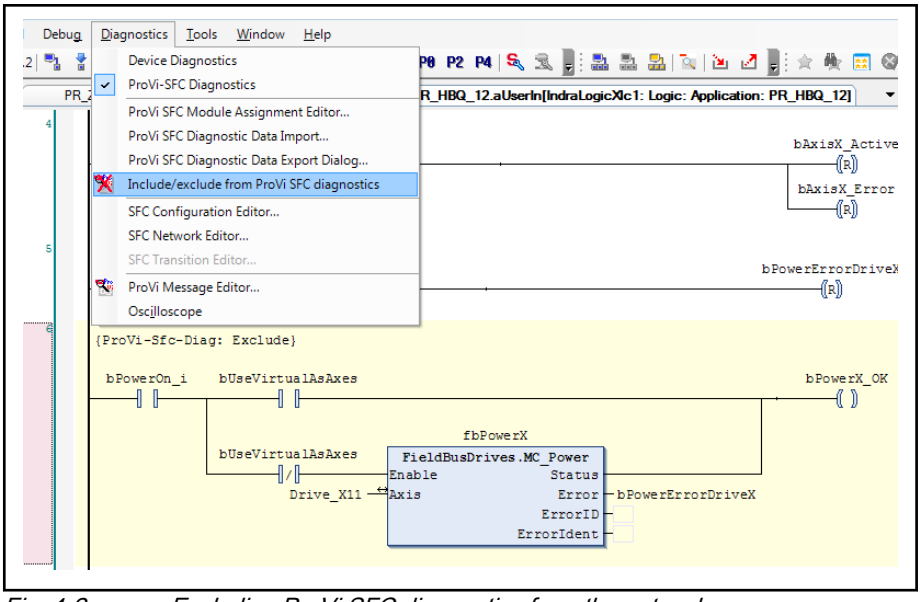


Fig. 4-2: Excluding ProVi SFC diagnostics from the network

If a network or an assignment is excluded from the ProVi SFC diagnostics like this, this exclusion applies to all diagnostic messages.

Diagnostic limit values

Do not exceed the following limits when using the ProVi SFC diagnostics:

- Maximum number of POUs with diagnostics = 2048
- Maximum number of instances that can be created by a diagnostics POU = 512
- Maximum number of messages in one POU = 4096
- Maximum number of SFCs with operation modes including instances = 256
- Maximum number of instances that can be created by an SFC with operation modes = 32
- Maximum number of steps in an SFC with operation modes = 512
- Maximum number of networks with diagnostics in an SFC with operation modes = 131072
- Maximum number of transitions in an SFC with operation modes = there is currently no limit

If the specified limits are exceeded, an error is output.

4.3.3 How is a ProVi Message defined?

A ProVi message is defined by a string (ProVi string) in curly brackets.

```
1  {ProVi Error, Module: 1, Not: FALSE, Set: FALSE, KA: FALSE, Indicate: FALSE, No: 1}
2  Output1 := Input1 AND Input2;
```

Fig. 4-3: Definition of a ProVi string

The ProVi string must not be commented for the IndraLogic 2G!

The ProVi string can either be created via a dialog (see chapter 4.3.6 "ProVi input dialog" on page 32) or entered manually into the PLC program (see the following chapter).

4.3.4 ProVi string syntax

ProVi string structure

{ProVi **Type**, Module: **ModuleNo**, Not: **Value**, Set: **Value**, CA: **Value**, Indicate: **Value**, [No: **MessageNo**][Variable: **VariableName**]}.

The bold type elements have to be substituted by the respective values.

Only one of the parameters "No" or "Variable" is to be used. Which one depends on the parameter value "Indicate".

Parameters	Description	Value range
Type	ProVi message type (error, info (note), warning, start condition, setup diagnostics)	Error, Info, Warning, Startup, Setup
Modules	Module number of the ProVi message	The value range for the module assignment is 1-99. If there is no module assignment, the value is "0"
Not	Indicates whether the ProVi message is triggered at the positive or at the negative edge	FALSE: Positive edge (not negated) TRUE: Negative edge (negated)
Set	Indicates how long the ProVi message is pending	FALSE: As long as the condition is pending (expiring) TRUE: Till the message is reset (non-expiring)
CA	Indicates whether the criteria analysis is enabled for the ProVi message	FALSE: No criteria analysis TRUE: With criteria analysis EXTENDED: With scratch flag calculation
Indicate	Indicates whether the message is issued with indirect addressing	FALSE: No indirect addressing TRUE: Indirect addressing
No	Message number of the ProVi message, only for Indicate = FALSE	All positive values that can be displayed with 32 bits (DWORD)
Variable	Variable indicating the message number in case of indirect addressing, only if Indicate = TRUE	One valid variable string as used in the PLC program

Tab. 4-1: ProVi string parameters

Indirect addressing is not yet supported. That means if "Indicate: TRUE", an error message is output when creating diagnostic data.

4.3.5 Where to program a ProVi message?

A ProVi message can be programmed either in the implementation, in the actions or in the transitions of a POU (program organization unit).

The POU has to be either of type "FB" or of type "Program". ProVi messages cannot be programmed in functions, POU methods, POU properties or derived function blocks.

ProVi messages

All ProVi messages have to be programmed in one task. ProVi messages in different tasks are not permitted and might cause a failure of the PLC program (it does not run).

This is checked when creating the diagnostic data. If a program is not moved to a different task afterwards, this case should never occur.

ST, IL, FBD (function block diagram) and LD (ladder diagram) are possible programming codes, with the definition of a ProVi message being different in the individual codes.

The PLC programming code CFC is not supported by the ProVi SFC diagnostics.

Structured text

In structured text (ST), a ProVi message can either be indicated in the line in front of the assignment or in the same line after an assignment.

```
1  {ProVi Error, Module: 1, Not: FALSE, Set: FALSE, KA: FALSE, Indicate: FALSE, No: 1}
2  Output1 := Input1 AND Input2;
3
4  Output2 := Input3 OR Input4; {ProVi Error, Module: 1, Not: FALSE, Set: FALSE, KA: FALSE, Indicate: FALSE, No: 1}
5
```

Fig. 4-4: ProVi message in structured text

Both at the same time, one assignment split across several lines or one ProVi string without assignment are not allowed and cause a warning when creating diagnostic data (see fig. 4-5 "Error in the ProVi string in structured text" on page 30).

```
3  {ProVi Error, Module: 1, Not: FALSE, Set: FALSE, KA: TRUE, Indicate: FALSE, No: 1}
4  Output1 := Input1 AND Input2; {ProVi Error, Module: 1, Not: FALSE, Set: FALSE, KA: TRUE, Indicate: FALSE, No: 2}
5
6  {ProVi Error, Module: 1, Not: FALSE, Set: FALSE, KA: TRUE, Indicate: FALSE, No: 3}
7  Output1 := Input1 OR Input2;
8             AND Input3 OR Input4;
9
10 {ProVi Error, Module: 1, Not: FALSE, Set: FALSE, KA: TRUE, Indicate: FALSE, No: 4}
11 WHILE Counter DO
12
```

Fig. 4-5: Error in the ProVi string in structured text

It should be avoided to program ProVi messages in loops and depending on the control variable, since no unique assignment of the causing variable can be made in these cases.

Only one and the same ProVi message is always issued and revoked. In case of short loops, only the state of the last run is visible for the user.

Instruction list

In the instruction list (IL), a ProVi string can only be specified for the following complete network as in FBD and LD.

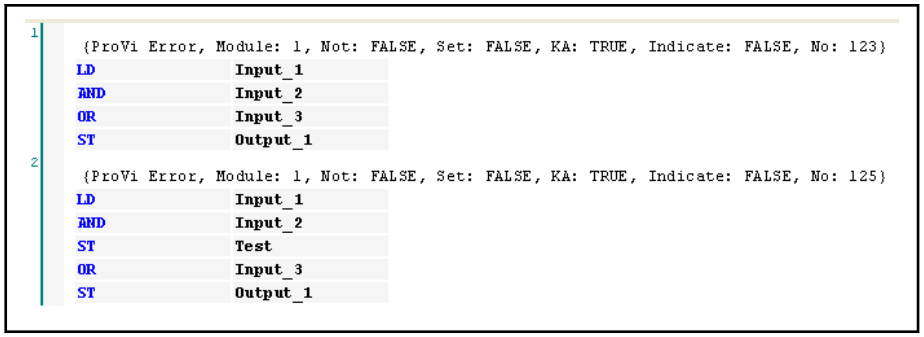


Fig. 4-6: ProVi message in the instruction list

ST, STN, S, R, JMPC, JMPCN, JMPN, RETC, RETCN and RET are possible assignments. All other possible entries are not considered and do no cause a message to be issued. The last assignment is always relevant for issuing a message within a network. It is not possible to issue messages depending on interim assignments.

Function block diagram and ladder diagram

In case of the languages "FBS" and "LD", the ProVi message can only be specified for an entire network and is automatically inserted at the correct position by the ProVi message editor.

The string can also be entered manually instead of the label.

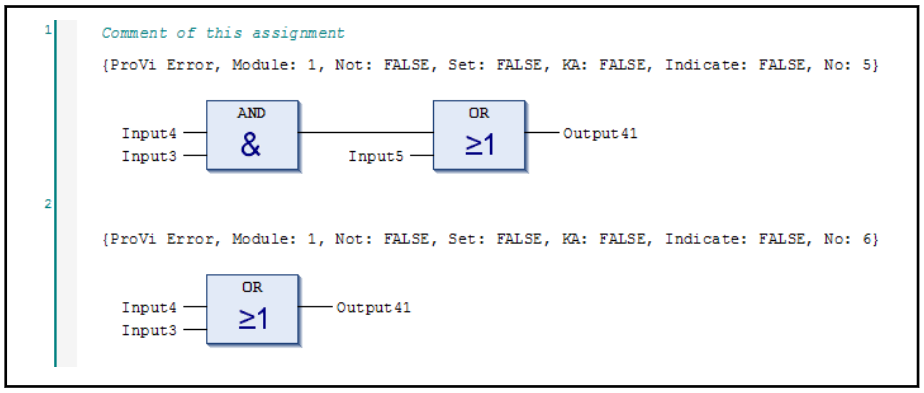


Fig. 4-7: ProVi message in function block code

An empty network and a network with only one assignment are not both possible at the same time.

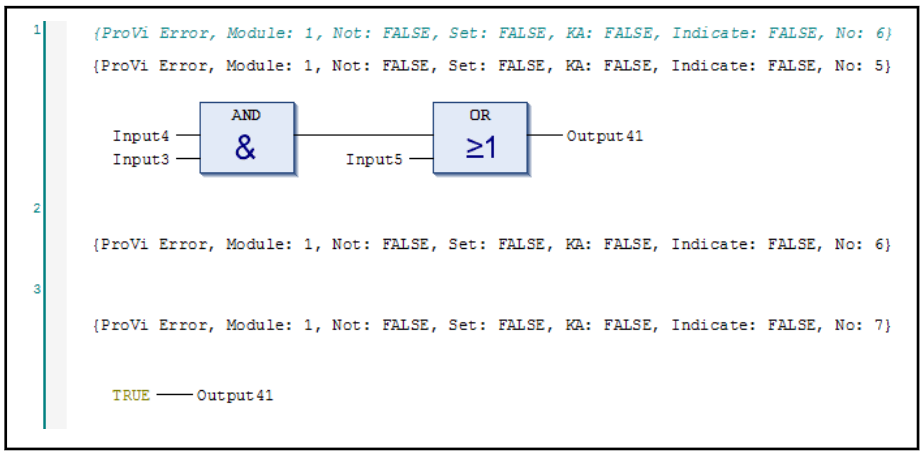


Fig. 4-8: Error in the ProVi string in the function block code

The ProVi string has to be a stand-alone string. That means that the ProVi string cannot be in the same line with either a label or a comment.

Function block diagram and ladder diagram

The assigned variables can be negated in the "FBD" and "LD" languages. This negation is restricted with the usage of diagnostics.

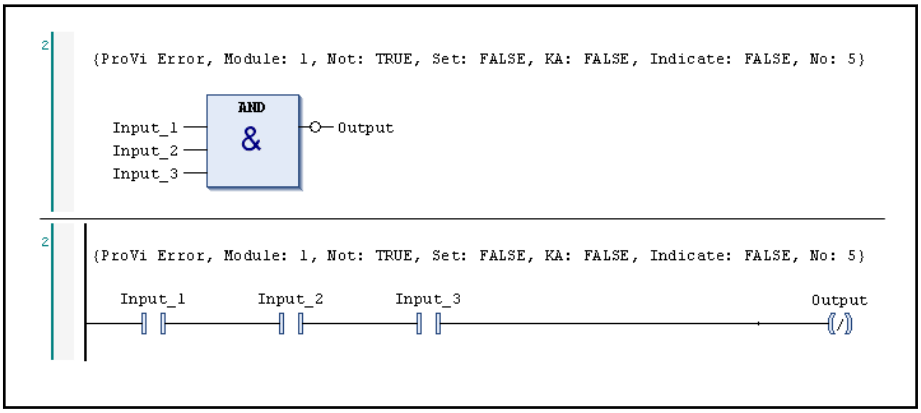


Fig. 4-9: Negated assignment

Observe that even the CR relevant for triggering the ProVi message is negated. Multiple assignments per network are also possible. Always the first assignment in the sequence from up to low is relevant. Thus, there may be no negated assignment at the first and last position.

When using the diagnostics and the negation, the respective POU must not be converted into other languages.

4.3.6 ProVi input dialog

The ProVi input dialog defines the properties of a ProVi message, the message texts, the error category and the message groups.

Opening the ProVi input dialog

The dialog is opened via the menu item **ProVi message editor...**.

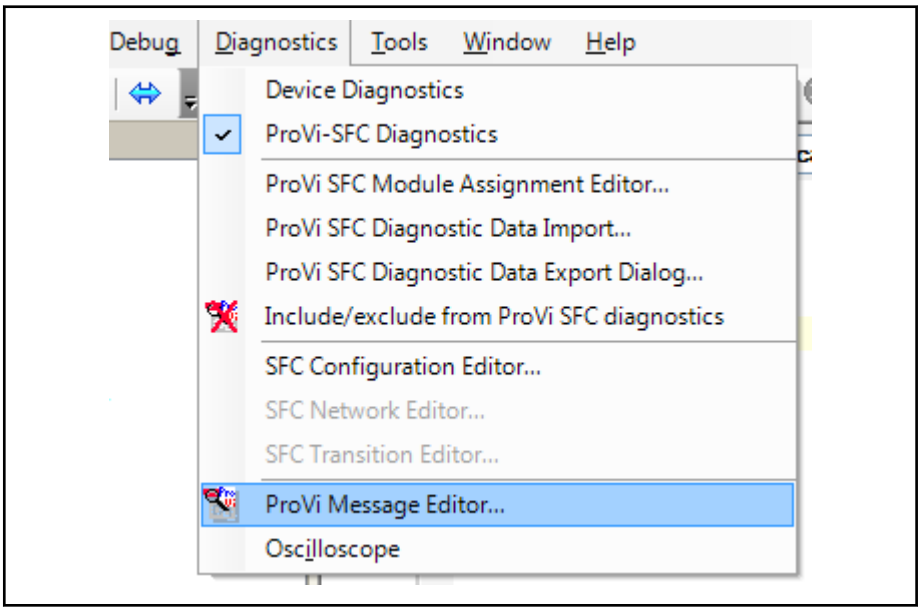


Fig. 4-10: Menu item "Edit ProVi message"

The properties of a ProVi message are stored in the ProVi string. The message texts, the error category as well as the message groups are saved in the PLC project.

The ProVi string is directly inserted at the position highlighted in the PLC program by the cursor. Please ensure that the cursor is set at the respective position **before opening** the ProVi Message Editor.

The ProVi input dialog can also be started via right-click. Also observe that the cursor is at the position relevant for the ProVi message. Thus, in front of or in the same line behind the assignment relevant to trigger the ProVi message.

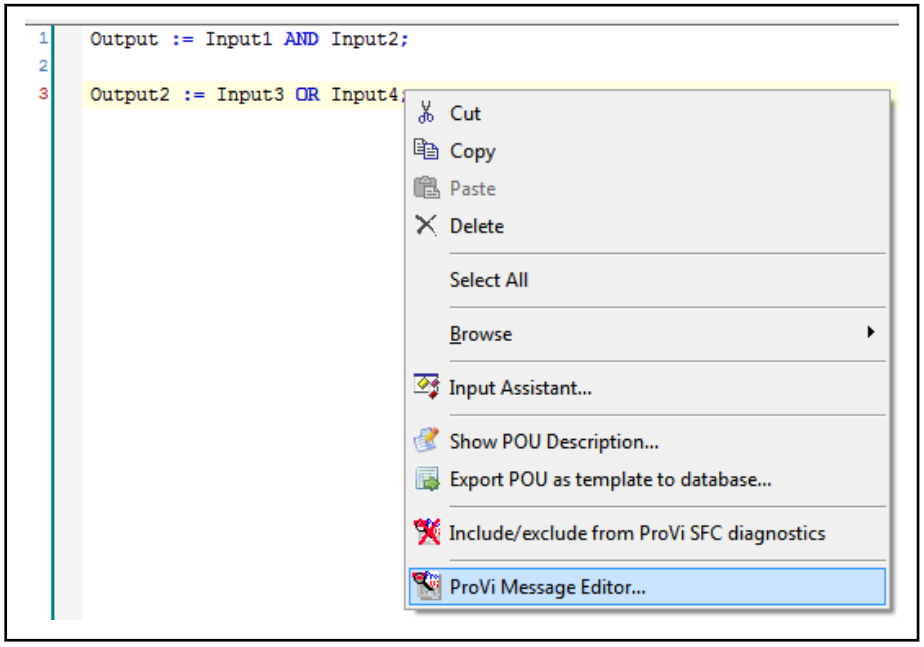


Fig. 4-11: ProVi input dialog, opened via context menu

Creating a ProVi message

Proceed as follows to create a new ProVi message:

- Open the ProVi input dialog
- Enter data
- Close the dialog with **OK**

Modifying a ProVi message

Proceed as follows to change a ProVi message:

- Position cursor on ProVi string
- Open the ProVi input dialog
- Modify data in the input dialog
- Close the dialog with **OK**

The ProVi string can also be edited directly in the PLC code. However, the message texts, the error category and the message groups cannot be modified in this manner.

Deleting a ProVi message

Proceed as follows to delete a ProVi message:

- Delete the ProVi string in the PLC code

- or -

- Position cursor on ProVi string
- Open the ProVi input dialog
- Close the dialog with **Delete**

The difference is that when deleting via the ProVi dialog , the message texts, the data of the error category and the message group are also deleted.

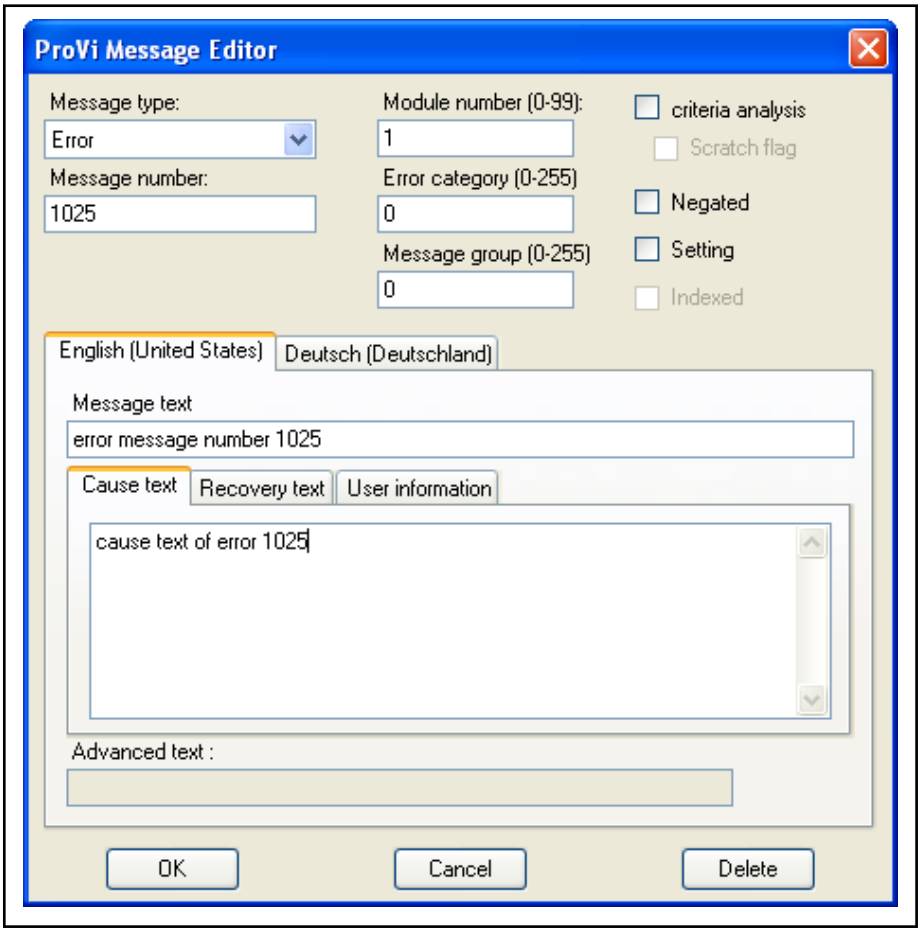


Fig. 4-12: ProVi input dialog

ProVi message type

The ProVi message type can be specified in the drop-down list "Message type".

If a new type is selected, a new message number is automatically suggested. This is always the highest number possible.

Module number

The module number is used to group messages (see chapter 4.2 "What is a ProVi message?" on page 21).

Value range: 1 to 99. If there is no module assignment, select "0".

Error category and message group

To group messages, the fields "Error category" and "Message group" can also be used.

Value range: 0 to 255.

Criteria analysis

If the option "Criteria analysis" is enabled, the error causing contacts or variables can be calculated or displayed for the ProVi message in IndraWorks Operation at runtime. Only the affected network or the affected assignment is analyzed.

Scratch flag

Using the enabled option "Scratch flag", the scratch flags used in the affected network are resolved. Assignments to these flags in the POU are also considered.

Negated
If this option is enabled, the ProVi message is triggered at the negative edge of the CR. If this option is not enabled, the ProVi message is triggered at the positive edge.

Non-expiring
If this option is enabled, the ProVi message remains pending until it is reset. If this option is not selected, the message is not pending anymore as soon as the condition for the ProVi message is not fulfilled anymore.

Message number
The ProVi message can be selected in the drop-down list "Message number". If the PLC project already contains data for this number (texts, error category or message group), this data is displayed automatically in the ProVi input dialog.

Multilingualism
The message texts can be entered in multiple languages. The input language available can be configured in the IndraWorks project.

☞ It is also possible to enter only one language first and then export the texts, have them translated before reimporting them (see chapter 4.7.4 "Message text export for external processing" on page 53 and chapter "Importing message texts" on page 60).

☞ Up to 10 languages can be edited in the ProVi input dialog. If a higher number is required, the text data editor is the appropriate tool (see chapter 4.7 "Text data editor for diagnostic messages" on page 46).

Message text
The text for the message to be displayed is available in the drop-down list "Message text". There is no restriction in the text length, but it must not consist of several lines.

In order to display additional data, use placeholders in the message text (see chapter 4.7.6 "Placeholders in message text" on page 63).

☞ The space on display elements is often limited. Therefore, the message texts should be as short as possible.

☞ Changes to existing texts are only visible after "Generate Diagnostic Data" and loading IndraWorks Operation.

Cause text (reason text), recovery text, user notes
Cause texts provide more detailed information on the message. There is no restriction regarding the length, and in contrast to the message text, the cause text can consist of more than one line.

4.4 Criteria analysis

When configuring the ProVi messages, the criteria analysis can be enabled or disabled for each ProVi message.

ProVi messages

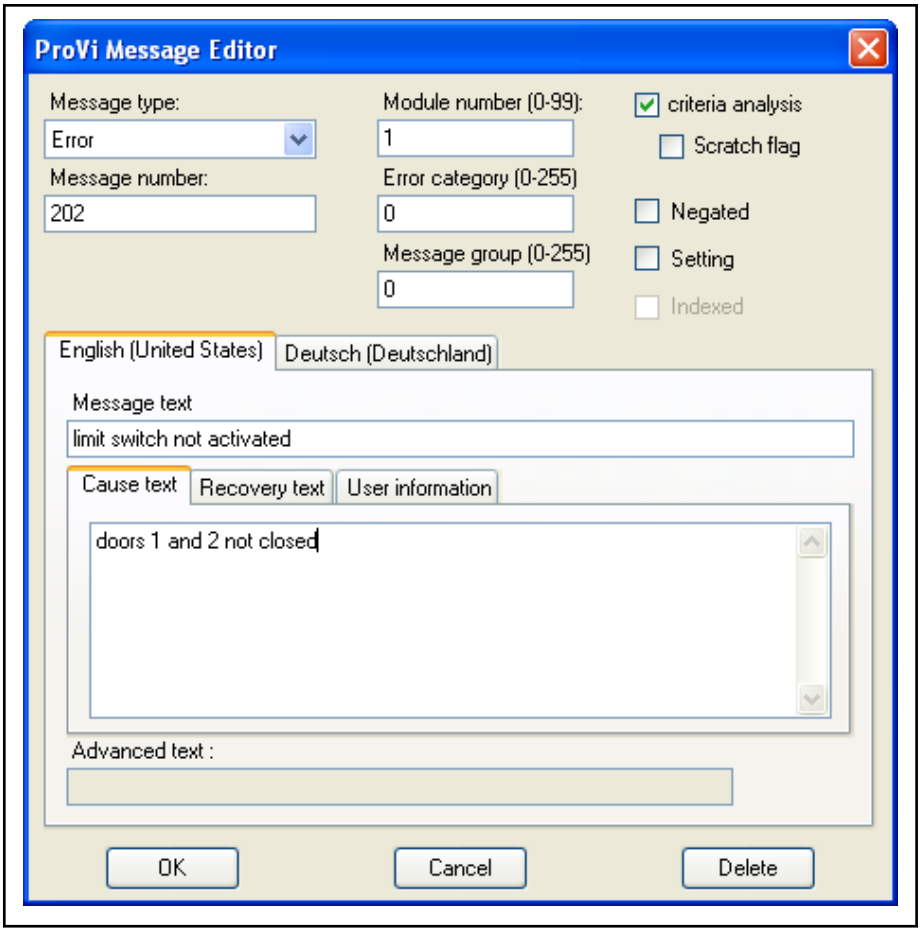


Fig. 4-13: ProVi input dialog

To enable the criteria analysis, select the checkbox "Criteria analysis". Additional activities are not necessary. The analysis is performed automatically.

The pending messages are listed in the diagnostic summary.

Additional information on a highlighted message (via mouse click or cursor keys) is displayed in the bottom section of the window.

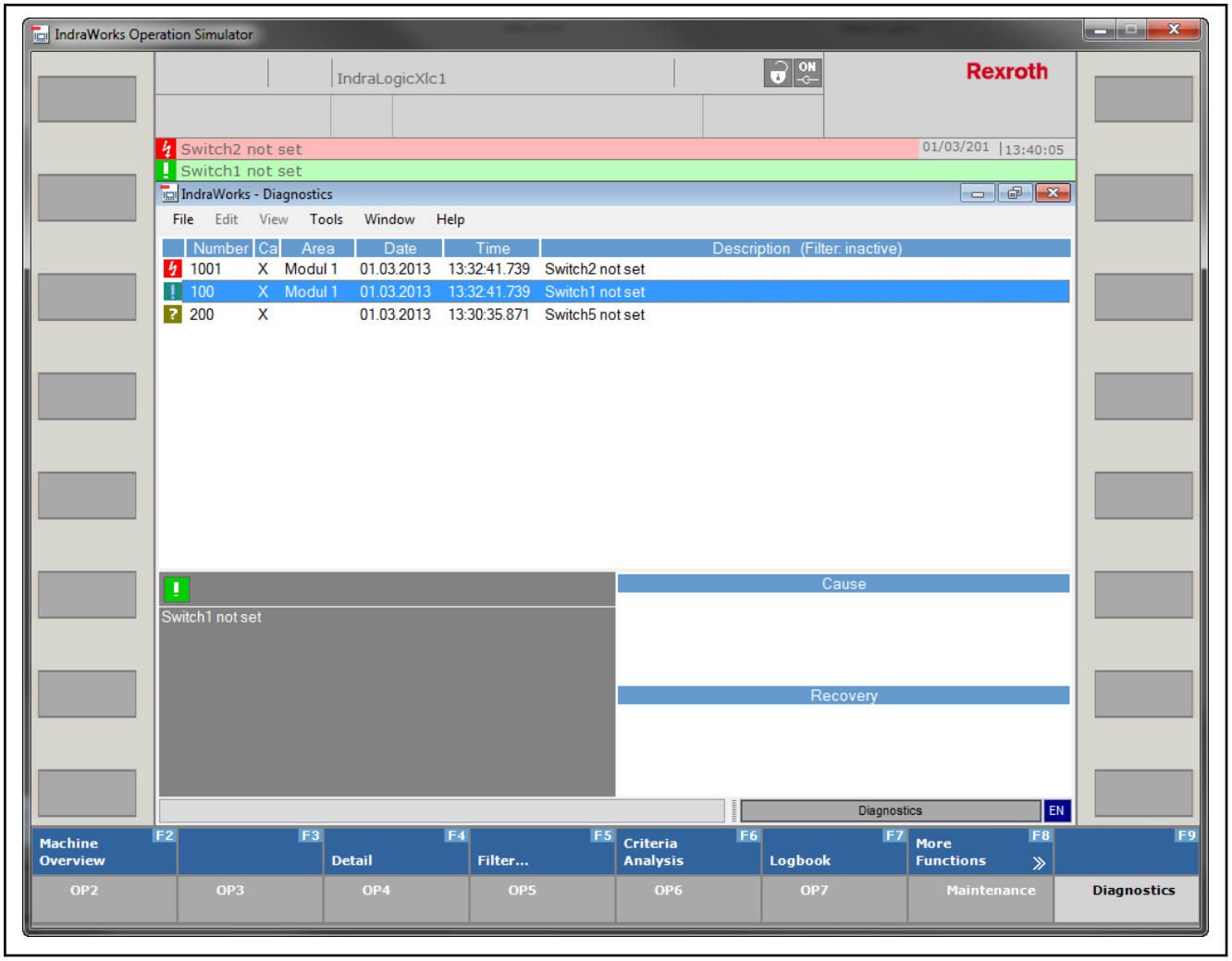


Fig. 4-14: Diagnostic overview

The criteria analysis for the highlighted ProVi message is enabled if there is an "X" in the "CA" column. Press the function key "Detail" (<F4>) to display additional information on this message.

ProVi messages

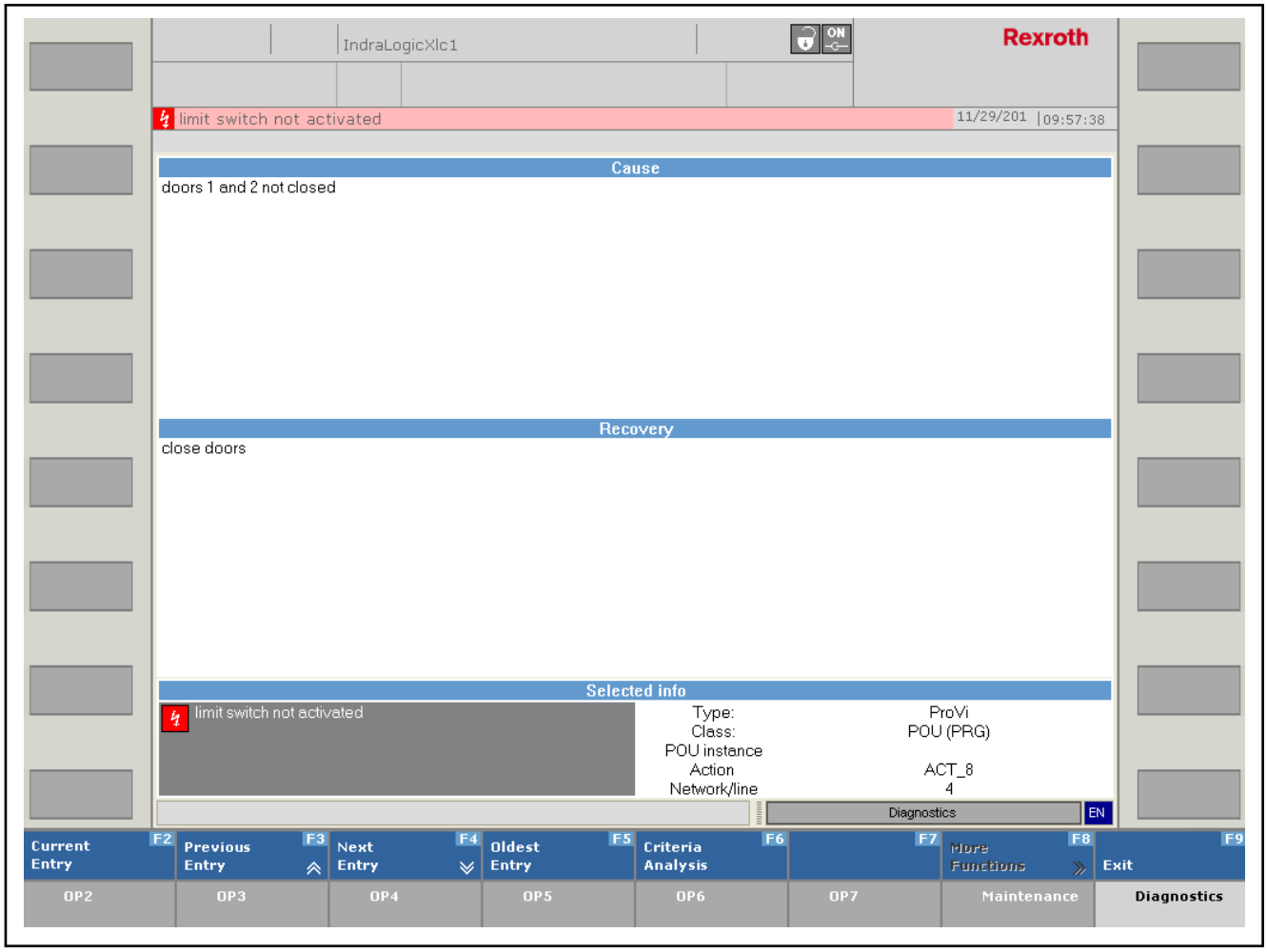


Fig. 4-15: Detailed view

The windows "Cause" and "Recovery" are now enlarged.

The following additional information is shown in the "Selected info" window:

Type: ProVi

Class: In this case POU (PRG) contains the POU name and the POU class (program, function) in brackets.

POU instance: Contains the instance name of the POU if available.

Action: Contains the action name (here: ACT_8) in which the ProVi message was programmed, if available.

Network/line: Contains the line number or the network number in which the message occurs (here: 4).

The function key "Criteria analysis" (<F6>) can be used in order to switch from the detailed display to the criteria analysis.

The criteria causing the error are identified for the selected error. These criteria are displayed in IL view in a new window. When opening the criteria analysis window, the reduced view is selected. "Reduced" means that only those criteria that caused the error are displayed.

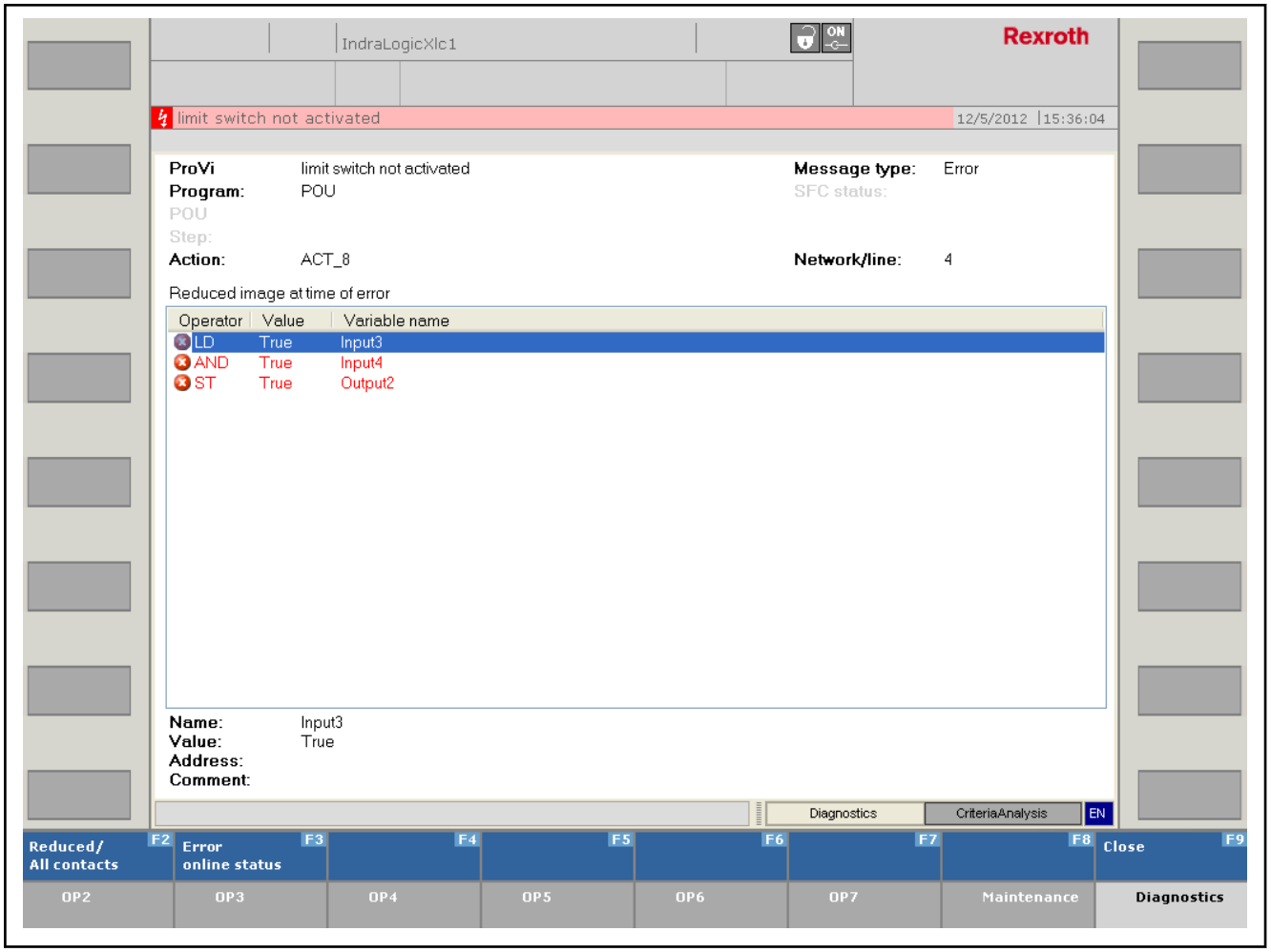


Fig. 4-16: Criteria analysis display

The criteria analysis window is divided into three sections:

1. The upper section contains general information on the message, which can also be seen in the detailed display. This facilitates keeping an overview on the message the criteria analysis belongs to.
2. The middle section shows the relevant criteria (variables) of the pending message.

   The operator is displayed in IL syntax. The state of the variables can be "TRUE" or "FALSE". The state when the message occurs is shown (snapshot).

   Press the function key "All" (<F2>) to show all criteria of the network in which the message was issued.

   The relevant criteria are shown in red.

   The criteria analysis does NOT display the logic as it has been programmed but it shows an interpretation of the programmed logic. Sporadically, some operators are negated during switching. However, this does not change the result or the logic.
3. The lower section contains additional information on the variables used. This additional information refers to the highlighted line in the central third of the criteria analysis window. The criteria analysis window is closed via the function key "Close" (<F9>).

Setting the scratch flag

Scratch flags are partial results in logic operations that are used again in subsequent networks. Thus, the PLC program is more clearly structured and easier to maintain. Often, many partial results are merged again in a separate network at a later point.

The criteria analysis calculation has previously considered the scratch flags. Now, the flags in the network line are taken into consideration for the analysis.

Only if the option "Scratch flag" has been enabled, the flags the scratch flag is composed of are also taken into consideration.

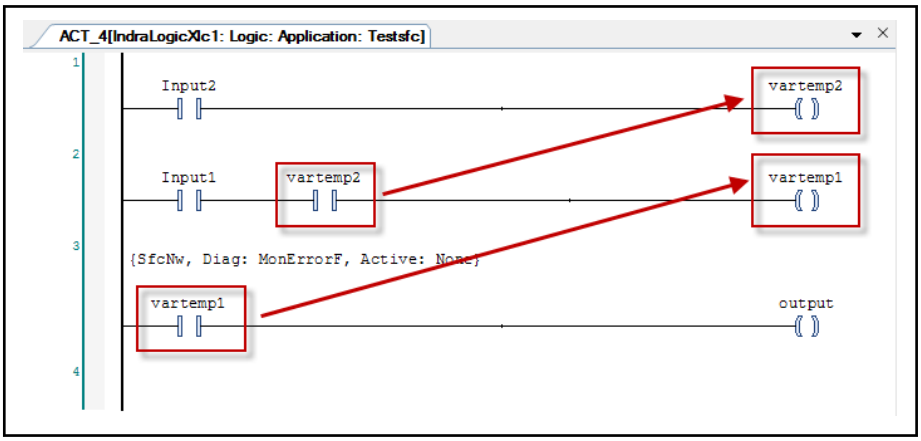


Fig. 4-17: Scratch flags in the network

Network2 causes a monitor error if the state of the assignment is "False".

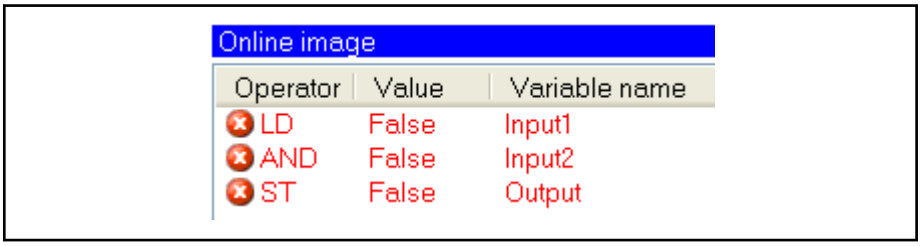


Fig. 4-18: Criteria analysis scratch flag

The scratch flag causing the monitor error was replaced by the logic generating the scratch flag. The states of this logic caused the monitor error. The nesting depth of the scratch flag is not limited. The scratch flags are only determined within the POU, but not beyond the POU limits.

If the scratch flag is used as non-expiring/expiring in previous networks, this network is NOT included in the scratch flag calculation but only evaluated as the variable used as non-expiring/expiring. The previous network can ONLY be a direct assignment.

The scratch flags are always set automatically. If no scratch flags are available, the criteria analysis corresponds to the previous analysis.

4.5 ProVi messages for VCP devices

No ProVi messages are available on the compact terminals (VCP devices). For the message type "Error" with a value range from "1" to "2047", multi-language messages can be displayed.

Therefore, the following steps have to be executed:

1. Compiling VCP device in the project and then creating diagnostic data, compiling and loading the control.

2. Exporting text data for VCP devices (see chapter 4.7.3 "Message text export for VCP devices" on page 53).
3. Importing these text data into the VCP project using the VI-Composer.

   For information on the VI-Composer, refer to the manual "Rexroth VCP Operating Concept" (part no. R911310666).

In the first step, a PLC variable array is created with the name `ILDP.VCP[0..255]` in which the binary states of the ProVi messages "1" to "2047" are mapped. If the value of the binary state is "1", a ProVi error message is currently present. The byte number results from the integer division of the "message number - 1" by "8". The bit address results from the rest of the division (method of counting 0..7).

Example:

- Message 1 corresponds to ILDP.VCP[0] bit 0
- Message 38 corresponds to ILDP.VCP[4] bit 5
- Message 515 corresponds to ILDP.VCP[64] bit 2
- Message 2029 corresponds to ILDP.VCP[253] bit 4

The PLC variable `ILDP.VCP` as well as the number of bytes can be directly specified in the parallel message system in the VI-Composer. Therefore, the PLC variable is specified as `starting address` and the byte number as `size`. Thus, no binary conversion is required for the display on the VCP device.

The PLC variables can be cyclically retrieved from the control by the VCP device.

The control variable provides information on the preparation of VCP device data to `ILDP.b_VCP`.

The length of the message texts for VCP devices is restricted to 255 characters.

Due to the restrictions of the VI-Composer, only the ProVi error texts from 1 to 2047 can be imported into the VCP. The ProVi message "0" is thus not exported.

4.6 HTML information

4.6.1 General information

More detailed information on a ProVi message can be provided to the operator in text or HTML format for cause and remedy texts.

The HTML link is shown in the area of the recovery text of the overview screen "Diagnostics".

Select the ProVi message and press "Detail" (<F4>) to display the HTML file content as part of the detailed view.

Even for messages of the machine state display (MSD messages), more detailed HTML messages are possible.

4.6.2 Entering file link into the database

The entry of the file link is entered as part of the recovery text. A keyword is used to identify the file, such as <FILE>123.html</FILE>. The keywords and the file name can be either in upper or lower case. Only the file name is required for the link.

ProVi messages

Entries for ProVi messages

Entering the HTML link for ProVi messages is executed in the ProVi message editor.

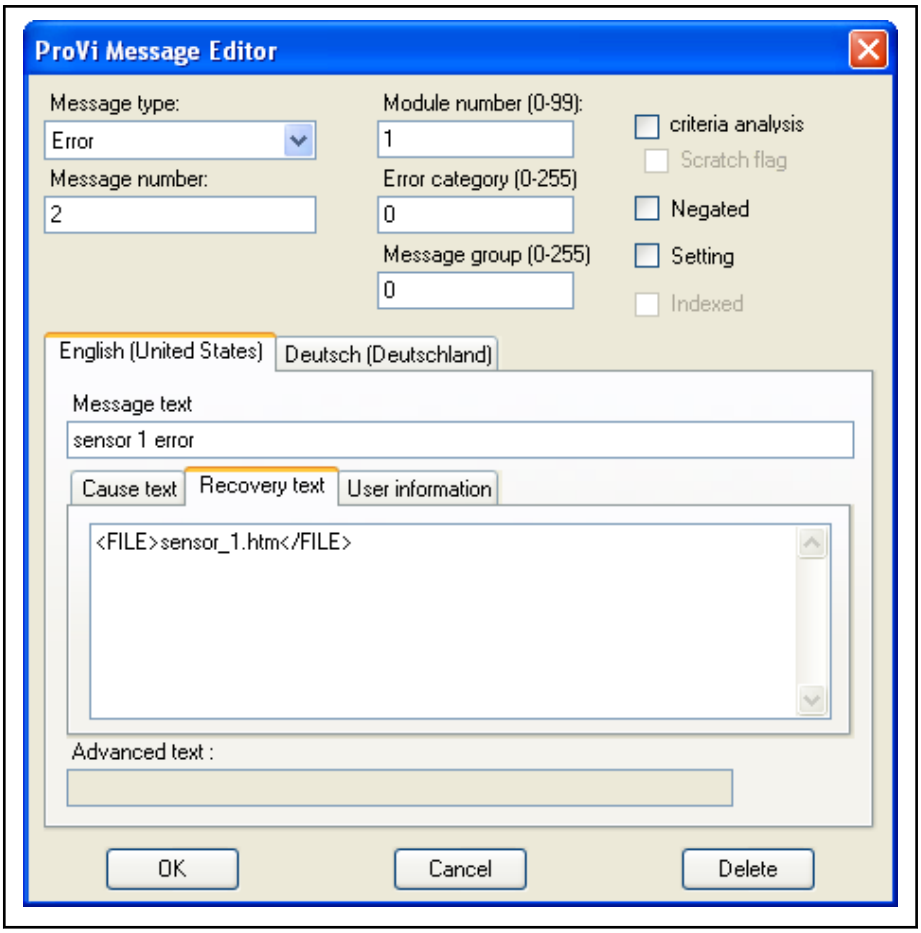


Fig. 4-19: Entry of the file link for the ProVi message

Entries for MSD messages

To enter the HTML link for MSD message texts, a reference to an HTML text with an advanced diagnostics is entered instead of a plain recovery text. HTML links must not contain any text separators (see "##" in the following example).

Example:

(E123, sensor 1 !## Sensor 1 is active ## <FILE>Sensor_1.htm</FILE>)

4.6.3 Managing HTML files

The HTML files are managed by the user. The folder structure described below has to be created by the user. The files for HTML information have to be stored on the local PC in the directory "RootOfIndraWorksProject\NameOfDevice\Help\HTML \LanguageOfHelpFiles(e.g. de-DE or en-US)". The directory names can be either written in upper or lower case.

The files are opened according to the predefined language in the HMI Operation Desktop.

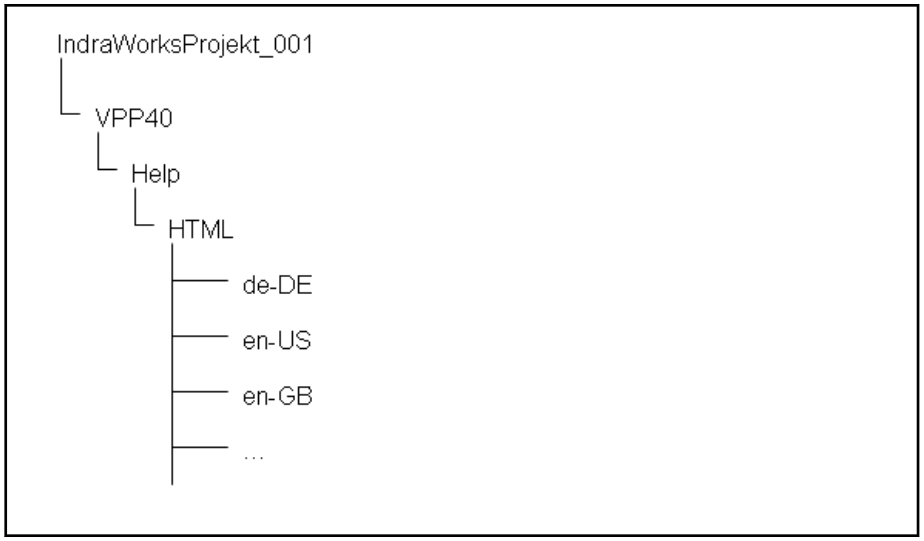


Fig. 4-20: HTML file management

4.6.4 HTML information in the "Diagnostics" workspace

If a ProVi message is selected, a note is displayed in the "Diagnostic" screen informing the user that further information is available in the detailed view.

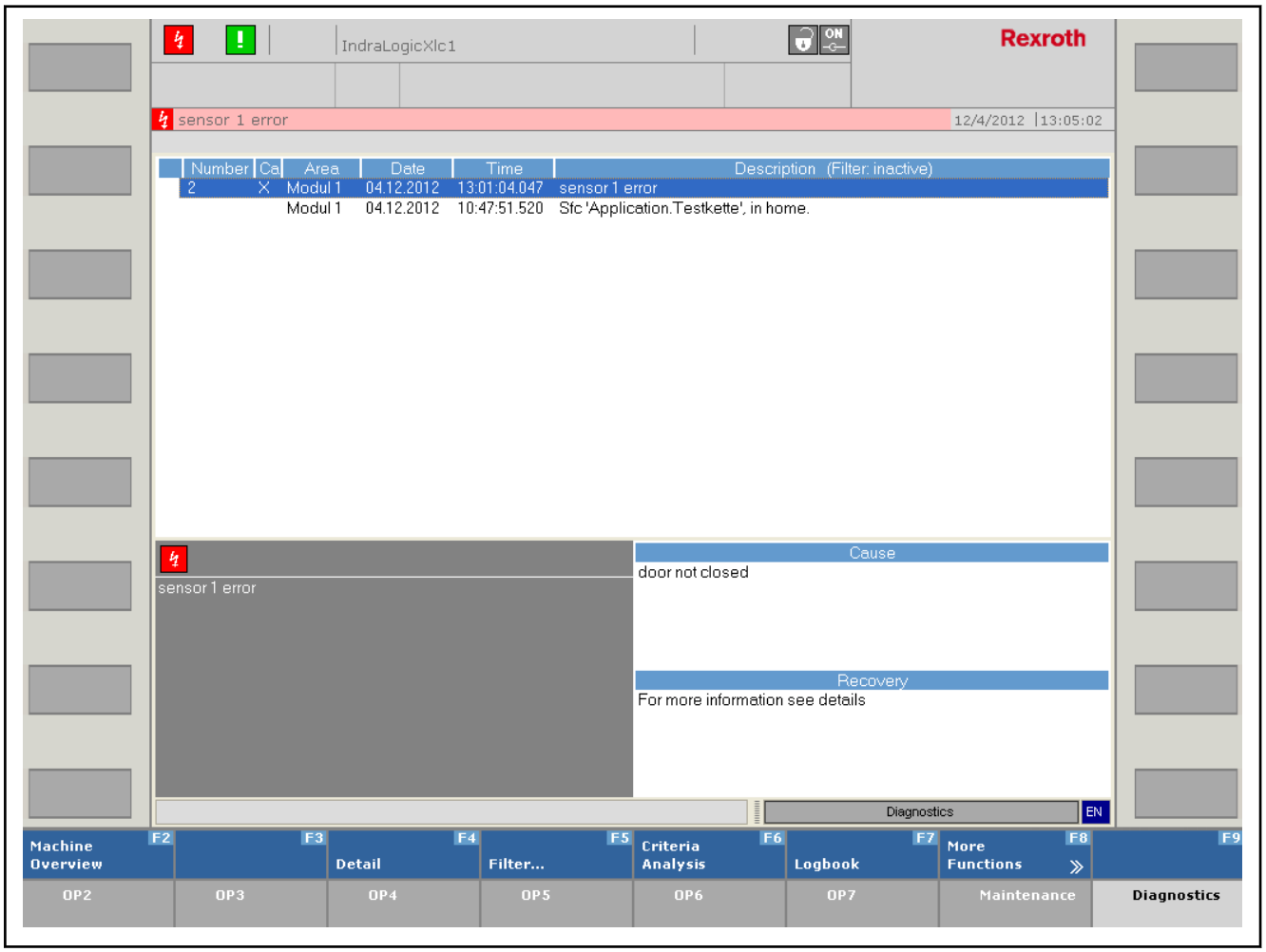


Fig. 4-21: Overview on HTML information

ProVi messages

If there is HTML information, the areas "Cause" and "Recovery" in the detailed view of the diagnostic image is substituted by HTML sites.

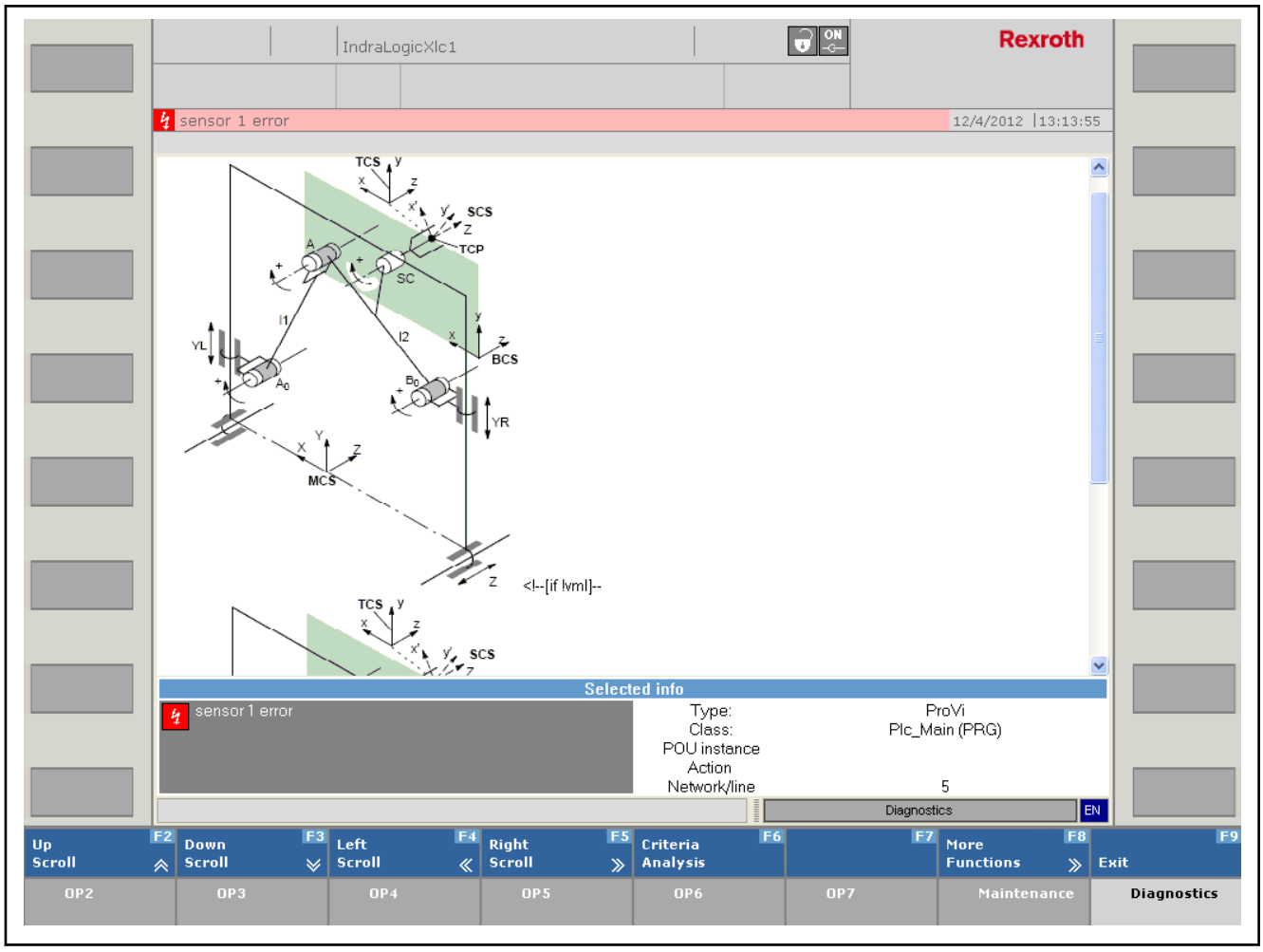


Fig. 4-22: Detailed view with F panel for scrolling

In the detailed view with HTML information, an individual F-panel is shown. The F keys from <F2> to <F5> are used to scroll the HTML site. To navigate between the individual diagnostics, it is switched to a second panel level via <F8> (»).

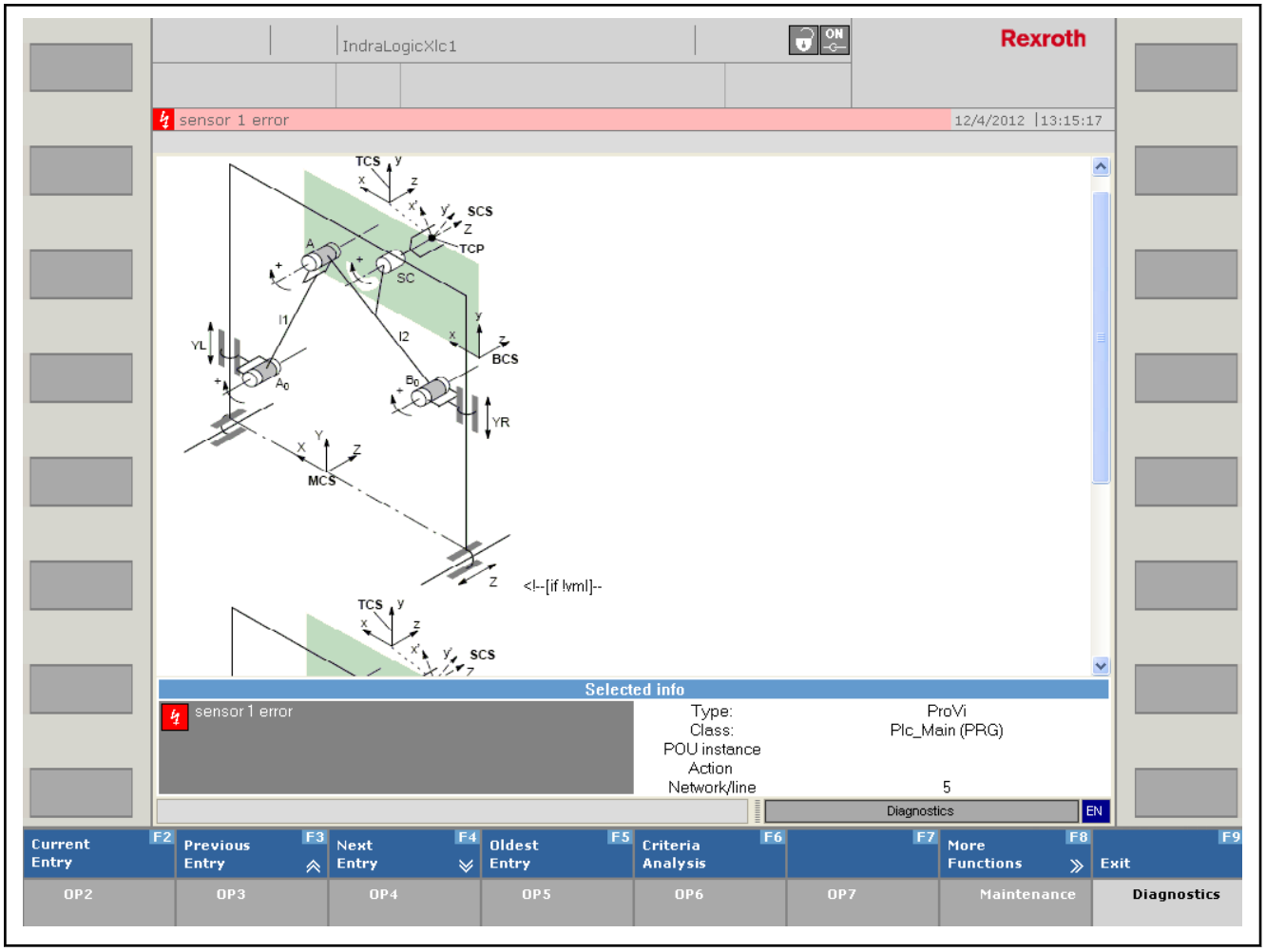


Fig. 4-23: Detailed view with F-panel for navigation

If the detailed view with HTML information is opened from the overview, the F-panel to scroll the HTML site is displayed. Use the level change to go to the panel for navigation. If the same view is opened from the detailed view (e.g. via "Next entry", "Previous entry", …), the "Navigation panel" remains active and the F-panel for scrolling is opened via the level change.

ProVi messages

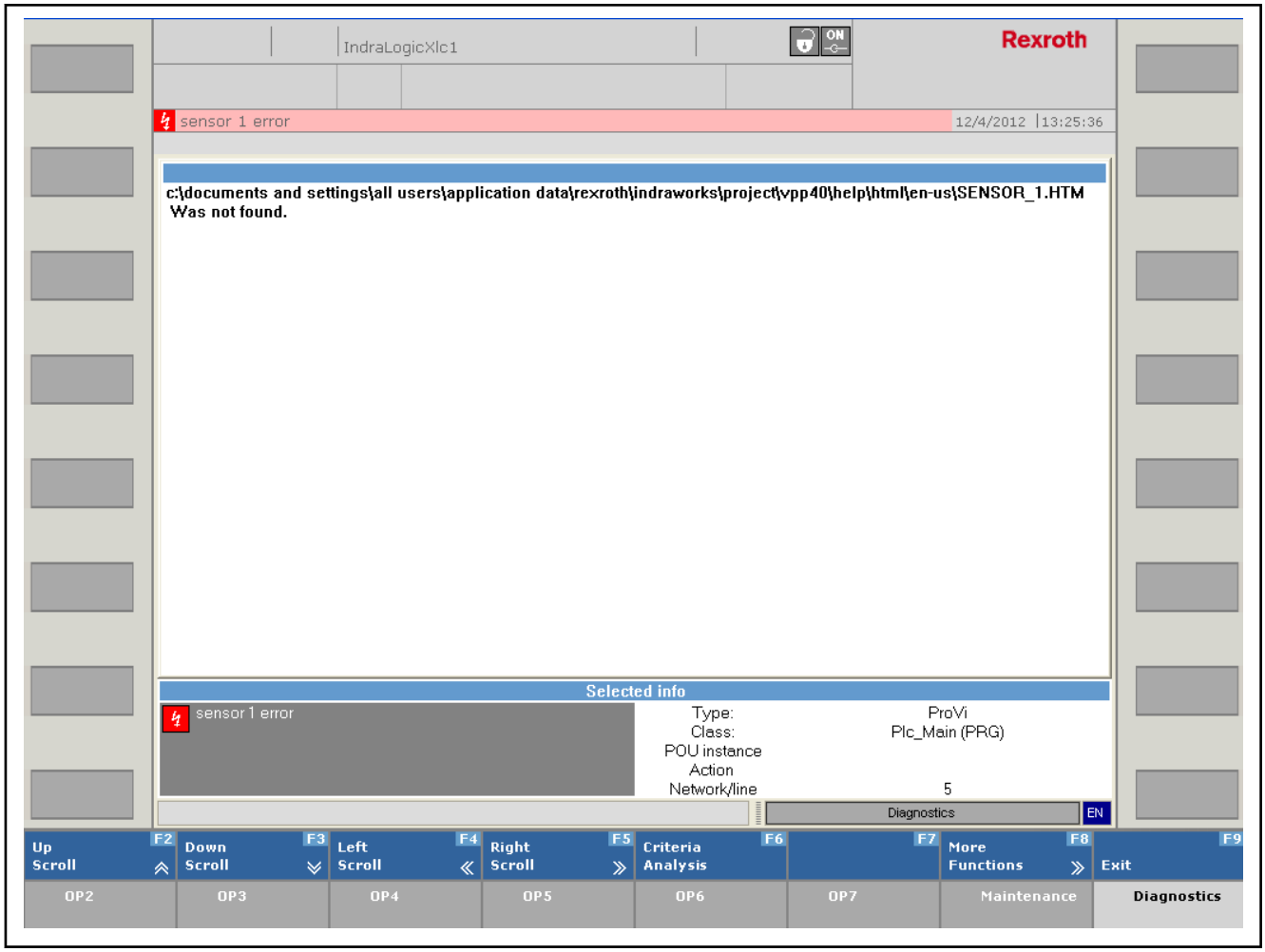


Fig. 4-24: Page "File was not found"

If the HTML file was not found, a default site containing the message "File was not found" plus path information is displayed.

Level change F-key: It is only enabled if the ProVi message contains HTML information.

4.7 Text data editor for diagnostic messages

4.7.1 Overview

Apart from the ProVi input dialog, edit message texts and additional information texts such as

- Cause texts
- Recovery texts
- User notes

New texts can be entered and existing texts can be edited using the text data editor. The relation between the texts can also be defined. For example, the cause text with the ID=5 can be assigned to the message 1025. The texts are displayed in all languages as configured in the IndraWorks project.

Changes to existing texts are only visible after "Generate Diagnostic Data" and loading the IndraWorks Operation Desktop.

Message pool texts

In addition, the message pool texts can also be edited in the text data editor. These are preformulated sentences of e.g. annexes or text lists for machine or plant-specific use. The message pool texts can be used several times if necessary. With a corresponding assignment, these texts are displayed in the message instead of the message text.

Default texts

Another diagnostics is the SFC (Sequential Function Chart). For the message output, the default texts are available, refer to tab. 4-2 "Overview on the default texts available" on page 47. These texts consist of a standardized message as well as placeholders (see chapter 4.7.6 "Placeholders in message text" on page 63), which are replaced at runtime by the respective SFC data, such as SFC name, SFC instance, SFC name or action name.

The default texts must not be changed. They are used for translation into other languages.

When translating, ensure that the placeholders are kept. Otherwise important information on the displayed texts is missing. To easily edit in all languages - except for German and English which have already been provided - a "text skeleton" is displayed in the text data editor which only needs to be filled in around the placeholders.

Texts are only saved if they are entered in a language other than German or English or if the German or English texts deviate from the original default texts. The texts are managed in the text database via "tokens". These tokens are created automatically and cannot be changed. This column is write-protected in the text data editor.

SfcExecutingMessage	SFC {0} , Step {1} currently processed
SfcHomingMessage	SFC {0} , homing
SfcHomeOkMessage	SFC {0} , in home position
SfcMonErrorMessage	SFC {0} , monitor error, action {1} ({2}) ,{3}
SfcTimeoutMessage	SFC {0} , time error step {1} {2}
SfcStartConditionError	SFC {0} error, miss. Start condition {1}
SfcHomePositionError	SFC {0} error, missing home position {1}
SfcStartConditionMessage	SFC {0} missing start condition {1}
SfcHomePositionMessage	SFC {0} missing home position {1}
NoProViTextAvailable	No ProVi message texts available

Tab. 4-2: Overview on the default texts available

Individualized SFC texts

A table integrated into the text data editor contains the "Individualized SFC texts". These texts are an optional supplement to the default texts. Individual message texts can be added to the errors "Monitor error", "Timeout error", "Missing startup condition" and "Missing home position" of each SFC instance. This allows to provide further information on an exactly defined error to the machine operator. The "Individualized SFC texts" can only be created in the text data editor. Only plain texts without placeholders are valid.

The individualized SFC texts are added to the default text in the diagnostic display and then output. The texts are managed in the text database via "tokens". These "tokens" result from the names, instances and step or action names of the SFCs. The tokens are created automatically and cannot be changed.

If SFC names, instance names, step names or action names are modified, there are new tokens. Already created texts can no longer be addressed. Old texts remain as "unused" in the text database.

SFCName.(SFCInstanceName)_StepName	Token for timeout errors
SFCName.(SFCInstanceName)_ActionName	Token for monitor errors
SFCName.(SFCInstanceName)_Start	Token for missing start condition
SFCName.(SFCInstanceName)_Home	Token for missing home position

Tab. 4-3: Overview on the automatically created "tokens" for each SFC instance

Exporting and importing message texts

Use the text data editor to export texts for external editing or to re-import them to the text database . For more information, refer to chapter 4.7.4 "Message text export for external processing" on page 53.

In the text data editor, even the message texts can be exported for the VCP devices (see chapter 4.7.3 "Message text export for VCP devices" on page 53).

The user texts as well as the configured languages are managed centrally in the IndraWorks project. This management also includes the ProVi message texts.

Opening the text data editor for IndraLogic 2G

The text data editor is located in the Project Explorer below the "Application" object in the "Diagnostic Text Data" folder. Double-click on the object "ProVi SFC Text Data Editor" or go to the context menu items **ProVi SFC Text Data Editor ▸ Open** to open the text data editor. Optionally, open the text data editor after selection in the Project Explorer via the main menu **ProVi SFC Text Data Editor ▸ Open**:

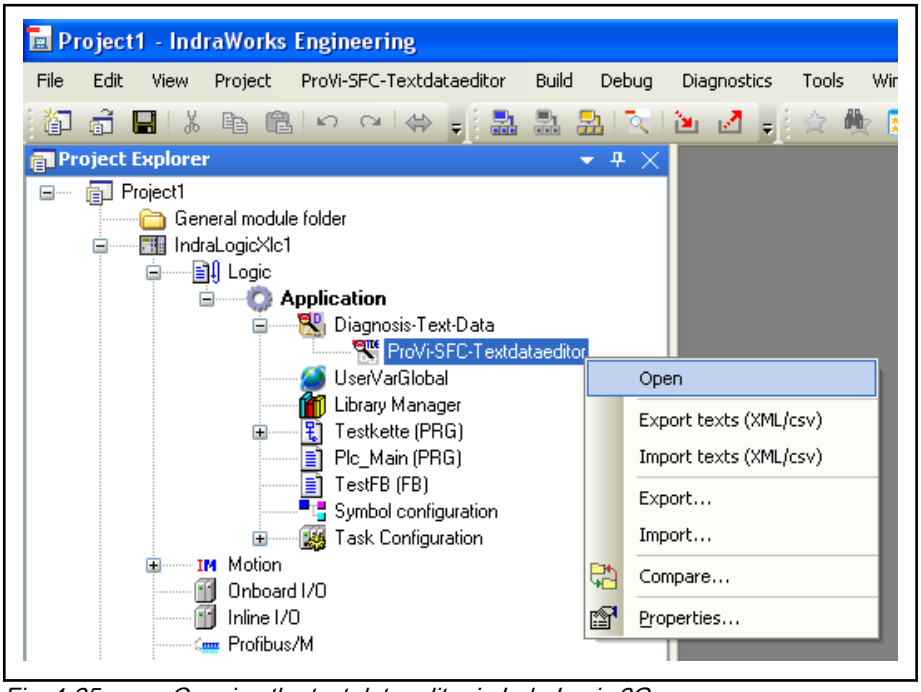


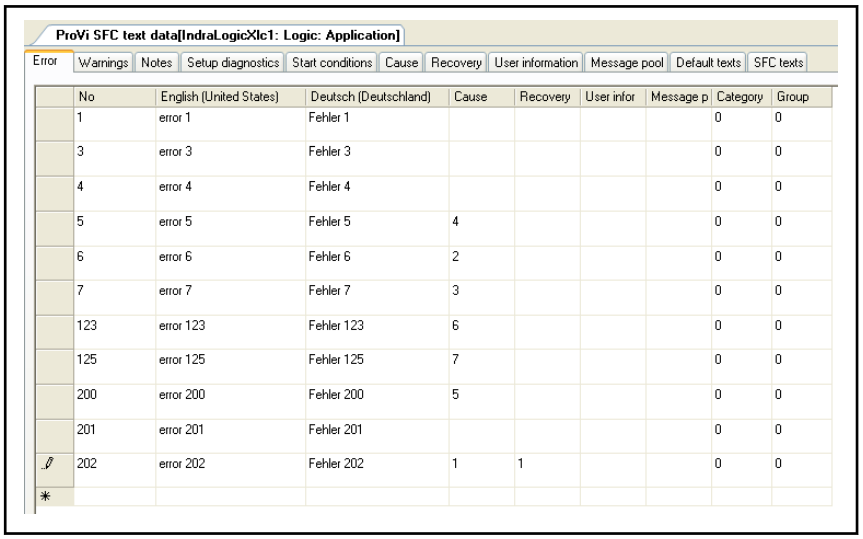
Fig. 4-25: Opening the text data editor in IndraLogic 2G

4.7.2 Editing message texts

To edit message texts, proceed as follows:

1. The opened text data editor provides a tab for each message type. Toggle between the different text types and edit the texts

   Select the message type of the message to be edited via the tab, in this example, the "Error" messages.



No	English (United States)	Deutsch (Deutschland)	Cause	Recovery	User infor	Message p	Category	Group
1	error 1	Fehler 1					0	0
3	error 3	Fehler 3					0	0
4	error 4	Fehler 4					0	0
5	error 5	Fehler 5	4				0	0
6	error 6	Fehler 6	2				0	0
7	error 7	Fehler 7	3				0	0
123	error 123	Fehler 123	6				0	0
125	error 125	Fehler 125	7				0	0
200	error 200	Fehler 200	5				0	0
201	error 201	Fehler 201					0	0
202	error 202	Fehler 202	1	1			0	0

Fig. 4-26: Text data editor with tabs for different message text types

2. **Message number, cause text ID, recovery text ID, user notes ID, message pool ID**

   The first column contains the message number to identify the message, followed by the corresponding message text in the configured languages. The following columns contain - via the number (ID) - the assignment of the additional texts, such as cause and recovery texts, user information and message pool texts for the respective message if defined.

   Error category ID, message group ID

   The last two columns contain the number (ID) of the error category and of the message group.

   To enter a new message, click on the next number field that has not yet been labeled and enter any free message number.

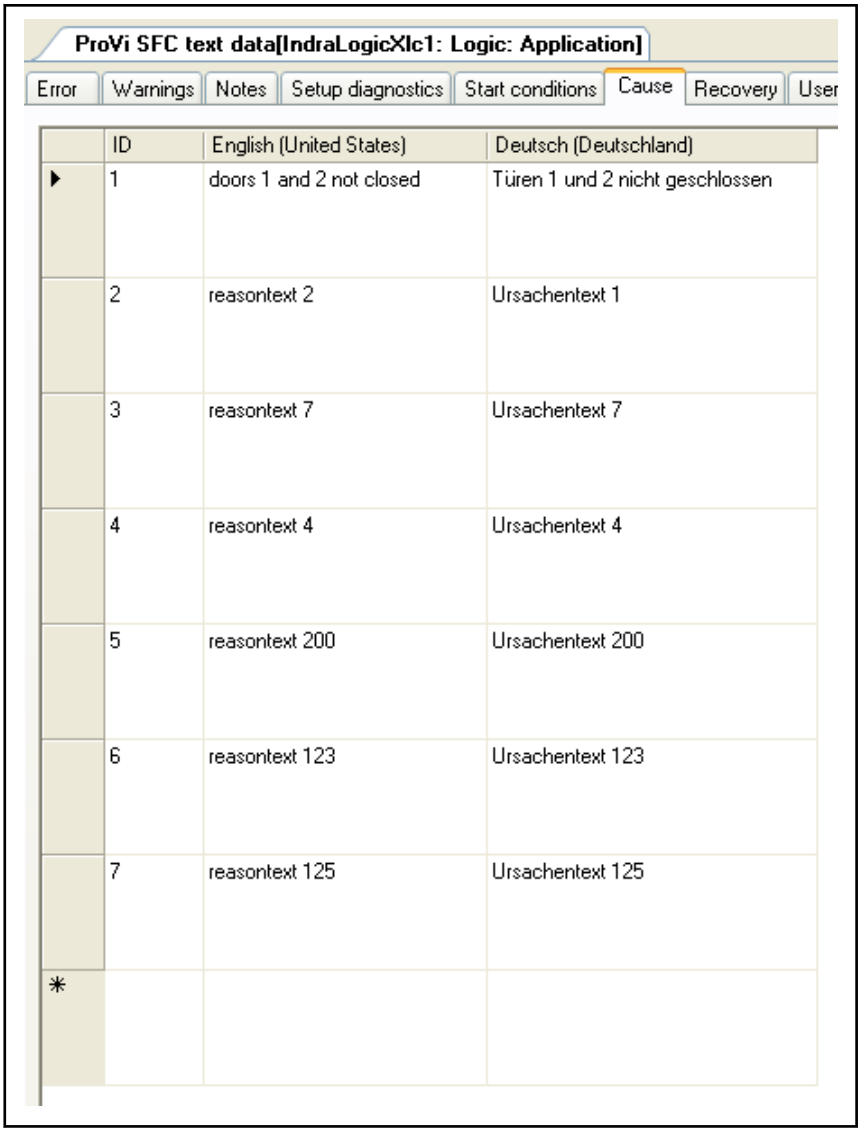
   If you enter an ID which has not yet been used, the non-visible text windows are predefined with empty text.

   If one line is deleted (highlight the line and press <Del>), the remaining additional texts do not have any reference in the database and can be used again for other messages.

The entries in the windows for info, warning, start condition and setup diagnostics are made in the same manner.

3. To edit cause texts, recovery texts or user information, select the desired window in the tab. The example in the following figure shows the window for cause texts:

ProVi messages



	ID	English (United States)	Deutsch (Deutschland)
▶	1	doors 1 and 2 not closed	Türen 1 und 2 nicht geschlossen
	2	reasontext 2	Ursachentext 1
	3	reasontext 7	Ursachentext 7
	4	reasontext 4	Ursachentext 4
	5	reasontext 200	Ursachentext 200
	6	reasontext 123	Ursachentext 123
	7	reasontext 125	Ursachentext 125
*			

Fig. 4-27: Window "Cause text"

The texts belonging to the respective ID can be edited in the configured languages. The assignment to the corresponding message is carried out by the text preceding the ID.

As long as there is any reference from the message types to a text line, this text line cannot be deleted. When trying to delete it, a note appears.

☞ When deleting or changing texts, please note that these can also be in multiple use.

Changes are only applied after exiting the input cell.

---

4. To edit message pool texts, please proceed as for cause texts, recovery texts or user information.

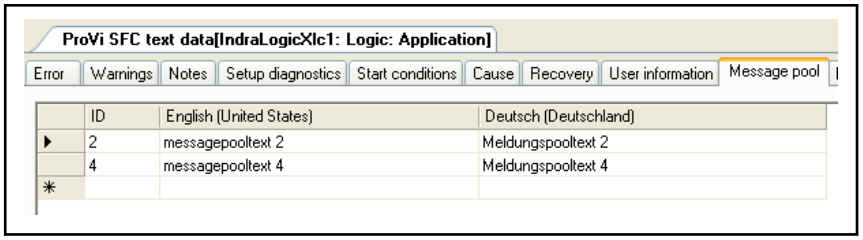


Fig. 4-28: Window "Message pool text"

The message pool contains a number of ProVi message texts that can be used multiple times. These message texts can also be assigned to messages using their ID. They are shown in the display instead of the declared message text.

"@@@@" ID for user texts:

When creating an IndraWorks project, a project language is specified. A text has to be available in this language before being entered in another language. If this is not the case, or if a text is edited in another language before being entered in the project language, this text is marked with "@@@@" and additionally stored in the project language. "@@@@" identifies for later editing.

5. Select the tab **Default texts** to edit these texts.

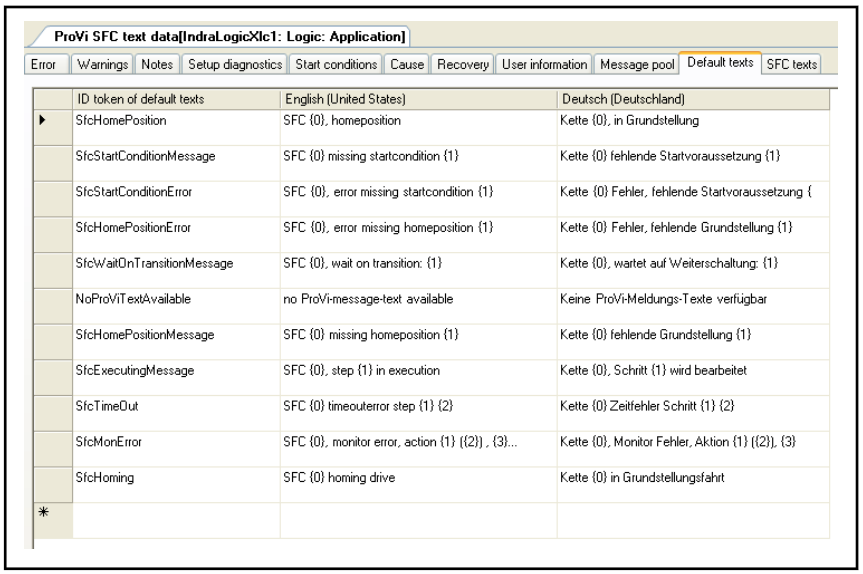


Fig. 4-29: View of default texts

The list shows all currently available default texts. The "tokens" to identify the texts in the left-hand column are predefined and cannot be changed.

The German and English texts are the recommended default texts and should not be changed. When carrying out modifications, ensure that the placeholders are kept (for a more detailed description, refer to chapter 4.7.6 "Placeholders in message text" on page 63). When translating into other languages, a "text skeleton" is shown as support containing the respective number of necessary placeholders.

When deleting a line (highlighting a line and confirming with "Del"), a modified default text is reset to the original default text. They are automatically shown again after the deletion of a line. The "token" is kept.

Instead of entering text into the text data editor, the default texts can also be exported and imported again after translation (see chapter 4.7.4 "Message text export for external processing" on page 53 or chapter "Importing message texts" on page 60).

☞ After importing a new language, close the window and open it again.

6. To edit the "individual SFC texts", select **SFC texts** via the tab.

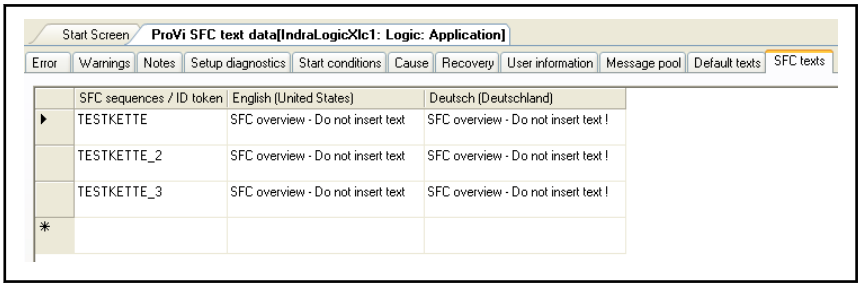


Fig. 4-30: Overview on SFCs

The view "SFC text" (Sequential Function Chart) shows a list with all SFCs defined in the project. This list is created automatically and based on the data created during the last "Create diagnostic data". Double-click on the first column of the desired SFC for the advanced view. The advanced view shows all steps and actions as well as the corresponding individualized SFC texts of the instance:

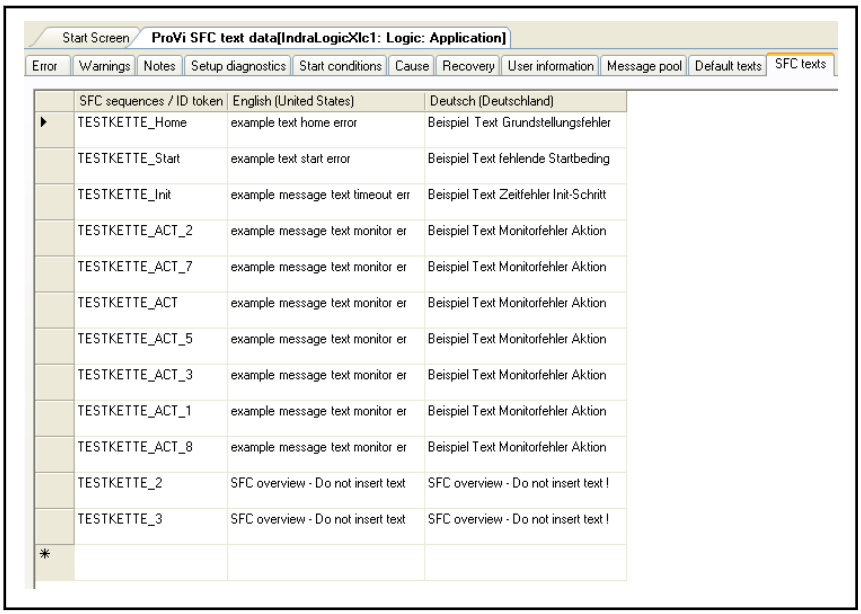


Fig. 4-31: Advanced view of an SFC

The texts can be edited in the advanced view. The modified texts are stored in the text database when the project is saved.

If a line is deleted (highlighting the line and confirming with <Del>), the texts of this line are deleted. The token is kept.

To close the enlarged view, double-click into the left column.

Instead of entering text in the text data editor, the "individualized SFC texts" can also be exported and imported again after translation (see chapter 4.7.4 "Message text export for external processing" on page 53 or chapter "Importing message texts" on page 60).

After importing a new language, close the window and open it again.

4.7.3 Message text export for VCP devices

On VCP devices, ProVi messages of the message type "Error" can be shown in a validity range between "1" and "2047". Therefore, export the respective message texts.

An individual text file in CSV format has to be created for each language. The language is represented by a respective language code in the file name.

Example: "TextExportVCP_en.csv"

The exported files can be imported into the VCP project using the VI-Composer. They contain line by line error message numbers in ascending order followed by the message text limited to 255 characters. For information on the VI-Composer, refer to the manual "Rexroth VCP Operating Concept".

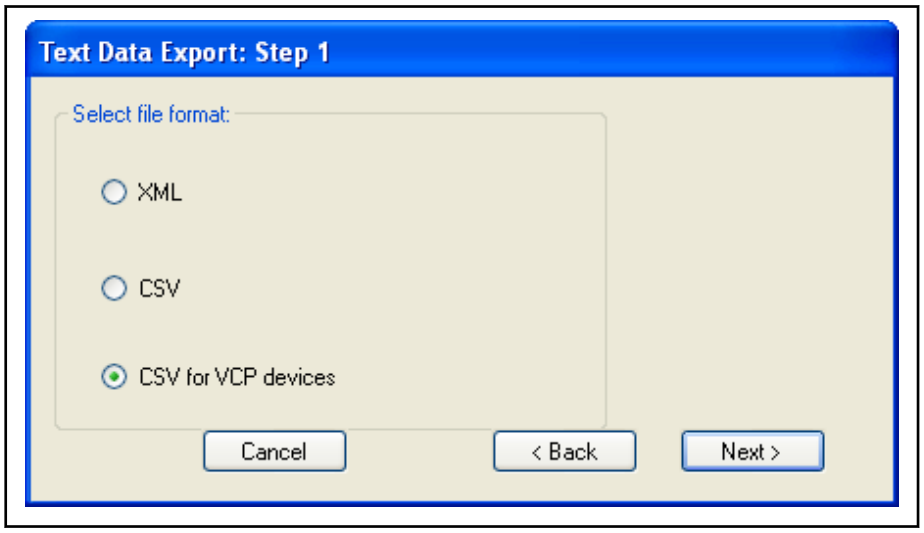


Fig. 4-32: Text data export CSV for VCP devices

To open the export functionality, start the text data editor in the context menu or in the main menu under **ProVi SFC text data editor** the menu item **Export texts (xml/csv)** and select the file format "CSV for VCP devices". The file names and the storage location of the file can be freely selected. The language ID as well as the file extension ".csv" are automatically added.

4.7.4 Message text export for external processing

General information

To edit message texts externally, there is an export function and an import function. Messages can be modified or new messages can be added.

There are two data formats available:

- CSV
- XML

In CSV format, texts can be edited externally using a spreadsheet program.

In XML format, a text or an XML editor can be used.

In case of both export formats, the link between message texts and additional message texts remains. The default texts and individualized SFC texts are also exported into the file.

Open the text data editor in the context menu or in the main menu via **ProVi SFC text data editor Export texts (XML/CSV)** to start the export dialog for all configured languages:

ProVi messages

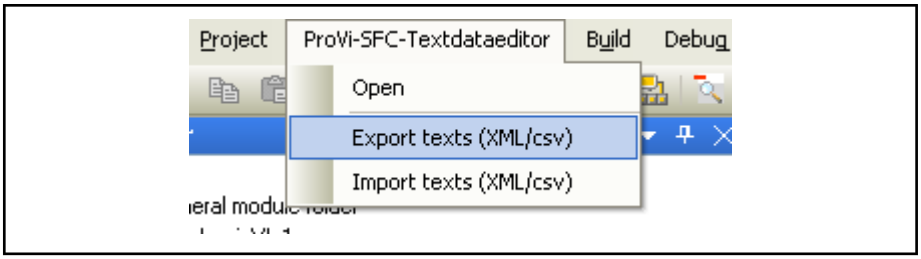


Fig. 4-33: Exporting message texts

In the dialog, select either the export format "XML" or "CSV".

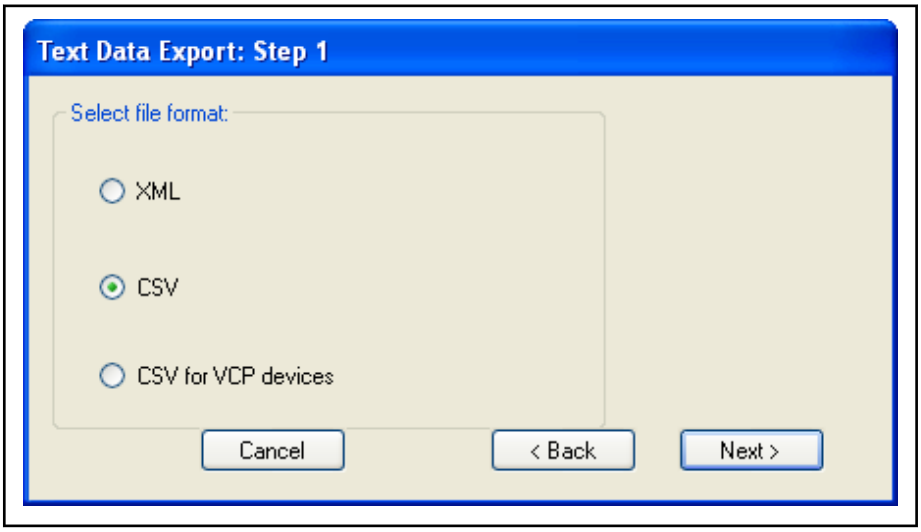


Fig. 4-34: Selecting the export format

Export file in CSV format

The CSV format is an ASCII file organized in columns and lines (UTF16) using a semicolon as column separator. The file can also be saved in the default ASCII format for further editing with Microsoft Excel® (for examples refer to "CSV exemplary files in text and excel presentation" on page 55 and fig. 4-36 "View of a CSV file in Excel" on page 56).

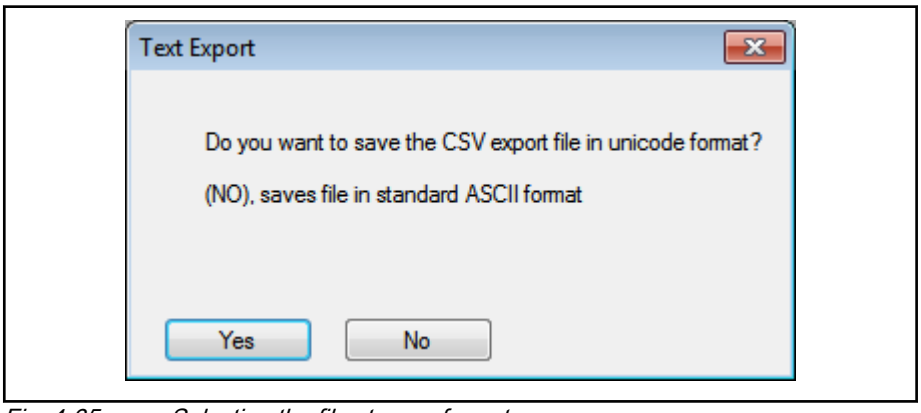


Fig. 4-35: Selecting the file storage format

If there are semicolons or inverted commas, the whole text is embedded in inverted commas. Inverted commas in the text are marked with preceding inverted commas. This marking is removed during import.

File structure

The file contains a defined header and below a two-line file body consisting of:

- Message line block with all texts assigned to a message.

- Text line block for additional texts not assigned to any message.

For any text group, the file contains the number of columns of the languages configured in the IndraWorks project. If three languages are configured for instance, three columns are included for all text types (message text, cause text, recovery text, etc.). This applies to each line in the file. If there are no texts in the respective language for a particular text type, the space between the separating semicolons is empty and thus, two semicolons follow each other.

Since the messages and all their related texts (cause text, recovery text etc.) are organized in lines, texts used several times are inserted in each assigned message line. This has to be considered when modifying such texts externally (see chapter "Importing message texts" on page 60).

Header structure

The header consists of all kinds of message components, starting with the message type, the corresponding ID (number) and the message texts in the configured languages. This is followed by the text blocks for cause, recovery, user information, message pool, error category and message group. Each consists of the ID and the texts in the configured languages. In case of the default texts as well as the individualized SFC texts, the message type contains the ID token and the ID column is empty. The remaining structure varies only in the number of languages.

The header does not depend on the language and is always in English.

File body structure

For each text type, one column is available in each configured language.

If there is no value in the message or the text line for a header, a semicolon is needed as column identification.

Message line block

The message line block contains all entries defined in the header. The message type is always given in English irrespective of the language: Error, Warning, Info, Setup, Startup.

Text line block

In contrast to the message line block, the entries in the (non-obligatory) text line block only consist of text type, ID and the texts in the configured languages. The text type is provided in English (irrespective of the language): Cause, Recovery, UserNotes, MessagePool. The text line block also contains the "default texts" and "individualized SFC texts" with token as text type and empty ID column.

The sequence of the two text blocks must not be changed.

CSV exemplary files in text and excel presentation

The following example shows a CSV export file with two configured languages for display in a text editor. The header is shown in two lines due to the word wrap of the text editor.

ProVi messages

```
Type;ID;de-DE;en-US;Reason-ID;de-DE;en-US;Recovery-ID;de-DE;en-US;UserNote-ID;de-DE;en-US;MessagePool-ID;de-DE;en-US;Category-ID;de-DE;en-US;Group-ID;de-DE;en-US;
Error;0;Meldungstext Fehler 0;messagetext error 0;1;Ursachentext 1;reasontext 1;0;Behebungstext 0;recoverytext 0;8;;;;;;0;;;4;;;
Error;1;Meldungstext Fehler 1;messagetext error 1;3;Ursachentext 3;reasontext 3;4;;;3;Hinweistext 3;usernotetext 3;;;;0;;;0;;;
Warning;2;Text Warnung 2;messagetext warning 2;2;Ursachentext 2;reasontext 2;2;;;2;Hinweistext 2;usernotetext 2;;;;1;;;1;;;
Info;0;Text Info 2;messagetext Info 2;;2;Ursache 2;reason 2;2;Behebung 2;recovery 2;2;Hinweistext 2;usernotetext 2;;;;1;;;1;;;
Reason;5;Ursachentext 5;reasontext 5;
Reason;6;Ursachentext 6;reasontext 6;
Recovery;1;Behebungstext 1;recoverytext 1;
Recovery;3;Behebungstext 3;recoverytext 3;
UserNotes;1;Hinweistext 1;usernotetext 1;
MessagePool;2;Meldungspooltext 2;messagepooltext 2;
```

Tab. 4-4: Example of a CSV export file in ASCII view

The following example shows a CSV export file in Microsoft Excel®.

Q39

	A	B	C	D	E	F	G	H	I
1	Type	ID	en-US	de-DE	Reason-ID	en-US	de-DE	Recovery-ID	en-US
2	Error	1	error 1	Fehler 1					
3	Error	3	error 3	Fehler 3					
4	Error	4	error 4	Fehler 4					
5	Error	5	error 5	Fehler 5	4	reasontext 4	Ursachentext test 4		
6	Error	6	error 6	Fehler 6	2	reasontext 2	Ursachentext 1		
7	Error	7	error 7	Fehler 7	3	reasontext 7	Ursachentext 7		
8	Error	123	error 123	Fehler 123	6	reasontext 123	Ursachentext 123		
9	Error	125	error 125	Fehler 125	7	reasontext 125	Ursachentext 125		
10	Error	200	error 200	Fehler 200	5	reasontext 200	Ursachentext 200		
11	Error	201	error 201	Fehler 201					
12	Error	202	error 202	Fehler 202	1	doors 1 and 2 not closed	Türen 1 und 2 nicht geschlossen	1	close doors
13	Error	203	Fehler						
14	Error	2	sensor 1 error	Sensor 1 Fehler	8	Error	Fehler	4	123.htm
15	Info	0							
16	Info	1							
17	Recovery	3	Magazin befüllen						
18	Recovery	2	Siehe 123.htm						
19	Recovery	5							
20	MessagePoo	2	messagepooltext	Meldungspooltext 2					
21	MessagePoo	4	messagepooltext	Meldungspooltext 4					

Fig. 4-36: View of a CSV file in Excel

Export file in XML format

If "XML" is selected as export format, the following window is displayed:

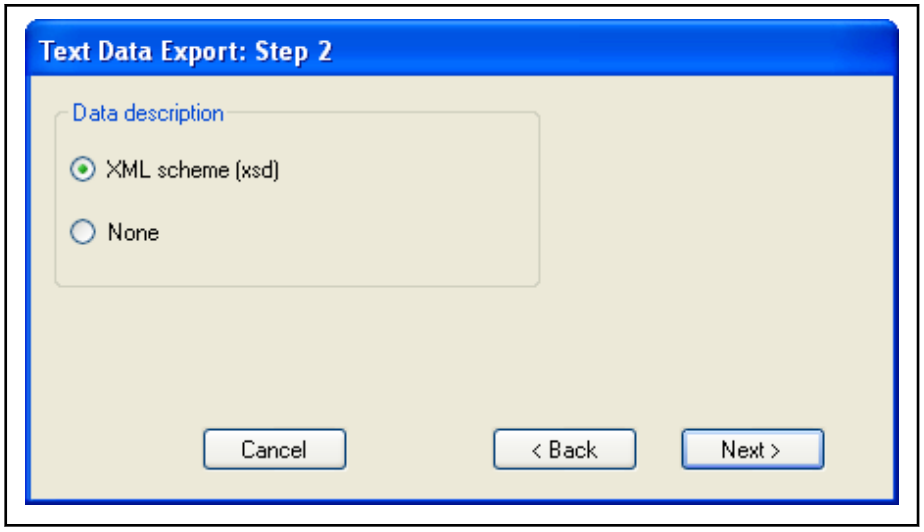


Fig. 4-37: Selecting during XML file export

As for the exported XML file, it is possible to store a scheme file in XSD format. Using this file, changes made in the exported file can be checked for a valid XML structure at import.

XML file structure

For an example, refer to "XML exemplary file shown as text" on page 59. The file consists of a fixed frame, which must not be changed, with the two nodes "<ProVi>" and "<Filter>". These nodes mark the two blocks "Message data" and "Filter data". The message data is divided into the message area consisting of the references to additional texts via ID and the message texts and the additional text area. The filter data contains the category texts and the group texts.

The texts are exported to the languages configured in the IndraWorks project. Missing texts are exported as empty texts. The individual languages are arranged one after the other. The language is labeled in the node: <Language ID="?"> (see below).

All node texts are fix and in English.

ProVi data set

The data sets of the message types and of the text types are located below the <ProVi> node if there are messages or texts of these types.

- <MessageType> from {Error, Warning, Info, Setup, Startup}
- <TextType> from {Reason, Recovery, UserNotes, MessagePool}

Message data sets

The messages for this type are listed below the message type and consist of the message block labeled with the message number with references to additional texts and filter texts using the ID and the message texts in the text block.

<Message ID="?"> Question marks stand for the message number. There are references and text block as subnode below:

The <Language ID> is a country code from the "Table of Language Culture Names" published in MSDN® as well.

Example: "en-US" for English - United States

"es-ES" stands for Spanish - Spain

<Reason-ID>	If available (text ID has to exist!)
<Recovery-ID>	If available (text ID has to exist!)
<UserNotes-ID>	If available (text ID has to exist!)

ProVi messages

<MessagePool-ID>	If available (text ID has to exist!)
<Category ID>	If available (text ID has to exist!)
<Group ID>	If available (text ID has to exist!)
<Text>	This text node contains the language-specific texts as subnode and has to be labeled as attribute with a compliant country code as ID. Structure: <Language ID="**de-DE**"><![CDATA[**here is the text**]]></Language>

Tab. 4-5: Message block

Text data sets

The texts of cause, recovery, user notes and message pool available for this type are located below the text type:

<Text ID="?">	The question mark stands in place of the text ID. This text node contains the language-specific texts as subnode and has to be labeled as attribute with a compliant country code as ID. Structure: <Language ID="**de-DE**"><![CDATA[**here is the text**]]></Language>

Tab. 4-6: Text data set

Filter data set

The category and group texts are stored below the <Filter> node if these texts are available.

- <Category>
- <Group>

<Text ID="?">	The question mark stands in place of the text ID. This text node contains the language-specific texts as subnode and has to be labeled as attribute with a compliant country code as ID. Structure: <Language ID="**de-DE**"><![CDATA[**here is the text**]]></Language>

Tab. 4-7: Text data set

SFC texts

Another node contains the "individualized SFC texts" under the node name <SFCTexts>.

<Text Token="?">	The question mark stands in place of the text token. This text node contains the language-specific texts as subnode and has to be labeled as attribute with a compliant country code as ID. Structure: <Language ID="**de-DE**"><![CDATA[**here is the text**]]></Language>

Tab. 4-8: SFC text data set

Default texts

The default texts are located below the node name <DefaultTexts>.

<Text Token="?">	The question mark stands in place of the text token. This text node contains the language-specific texts as subnode and has to be labeled as attribute with a compliant country code as ID. Structure: <Language ID="**de-DE**"><![CDATA[**here is the text**]]></Language>

Tab. 4-9: Default texts data set

XML exemplary file shown as text

The following example shows an XML export file with two configured languages for display in a text editor. In rough view, some lines are marked with "...". These lines have to be supplemented with the fitting entries in an XML file to be exported.

XML export file

```
<?xml version="1.0" encoding="UTF-8"?>
<ILDModify xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:noNamespaceSchemaLocation="ILD.Modify.xsd">
  <ProVi>
    <Error>
      <Message ID="0">
        <Reason-ID>1</Reason-ID>
        <Recovery-ID>2</Recovery-ID>
        <UserNotes-ID>1</UserNotes-ID>
        <Category-ID>0</Category-ID>
        <Group-ID>0</Group-ID>
        <Text>
          <Language ID="de-DE"><![CDATA[Meldungstext Fehler 0]]></Language>
          <Language ID="en-US"><![CDATA[messagetext error 0]]></Language>
        </Text>
      </Message>
      <Message ID="1">
      ...
      </Message>
    </Error>
    <Warning>
      <Message ID="2">
      ...
      </Message>
    </Warning>
    <Reason>
      <Text ID="1">
        <Language ID="de-DE"><![CDATA[Ursachentext 1]]></Language>
        <Language ID="en-US"><![CDATA[reasontext 1]]></Language>
      </Text>
      <Text ID="2">
      ...
      </Text>
      <Text ID="5">
      ...
      </Text>
    </Reason>
    <Recovery>
      <Text ID="1">
        <Language ID="de-DE"><![CDATA[Behebungstext 1]]></Language>
        <Language ID="en-US"><![CDATA[recoverytext 1]]></Language>
      </Text>
      <Text ID="2">
        ...
      </Text>
    </Recovery>
    <UserNotes>
      <Text ID="1">
        <Language ID="de-DE"><![CDATA[Hinweistext 1]]></Language>
        <Language ID="en-US"><![CDATA[usernotetext 1]]></Language>
      </Text>
      <Text ID="2">
        ...
      </Text>
    </UserNotes>
    <MessagePool>
      <Text ID="2">
        <Language ID="de-DE"><![CDATA[Meldungspooltext 2]]></Language>
        <Language ID="en-US"><![CDATA[messagepooltext 2]]></Language>
      </Text>
```

ProVi messages

```
    </MessagePool>
  </ProVi>
  <Filter>
    <Category>
      <Text ID="0">
        <Language ID="de-DE"><![CDATA[]]></Language>
        <Language ID="en-US"><![CDATA[]]></Language>
      </Text>
      <Text ID="1">
        ...
      </Text>
    </Category>
    <Group>
      <Text ID="0">
        <Language ID="de-DE"><![CDATA[]]></Language>
        <Language ID="en-US"><![CDATA[]]></Language>
      </Text>
    </Group>
  </Filter>
```

Importing message texts

1. In the context menu of the text data editor or in the main menu, select **Export texts (XML/CSV) ProVi SFC text data editor** to import all previously exported and edited message texts in all configured languages.

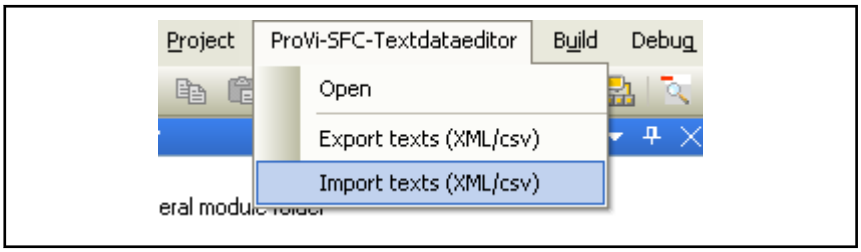


Fig. 4-38: Importing message texts

2. In the following dialog, specify the files to be imported. Select the format via the "File type" field:

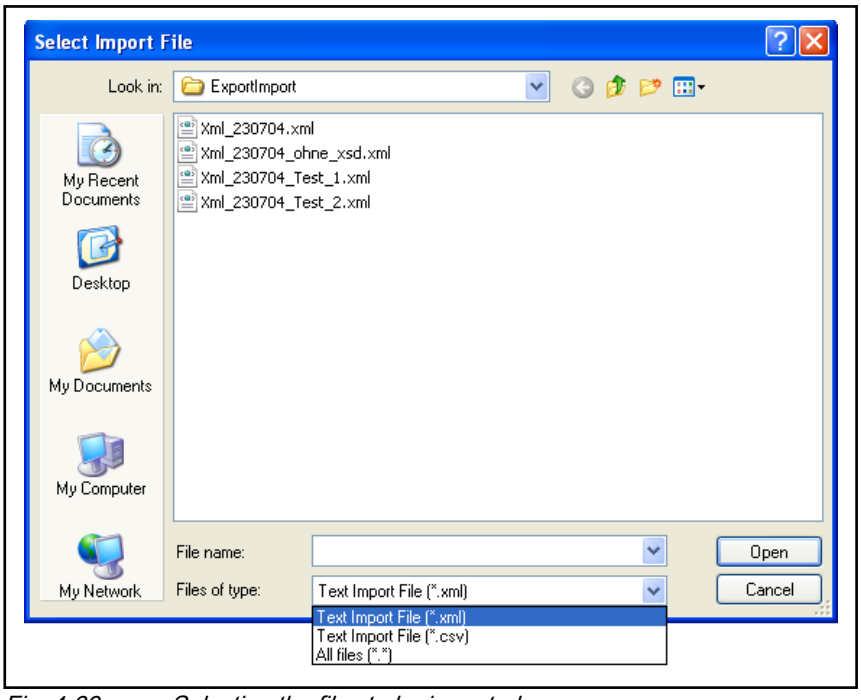


Fig. 4-39: Selecting the files to be imported

The file format of the import file has to match the export file format.

Importing XML files

When importing XML files, their structure is always checked for validity using an XSD scheme.

Importing CSV files

When importing CSV files, also specify whether the file is available in Unicode format or (e.g. after editing with Microsoft Excel®) in standard ASCII format:

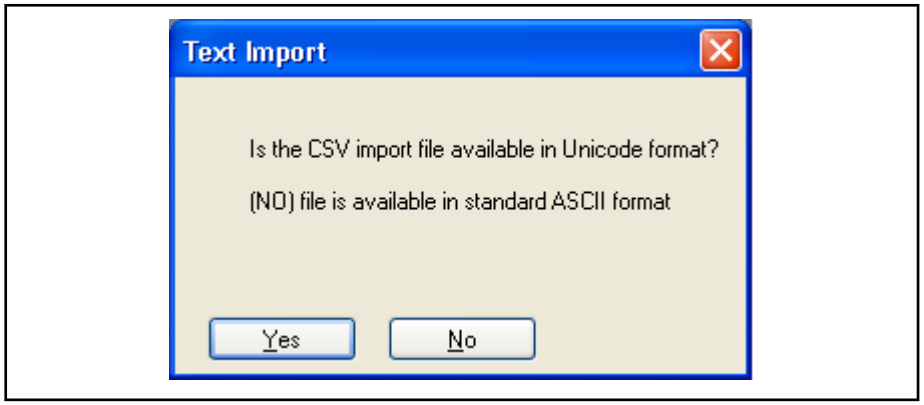


Fig. 4-40: CSV file format

4.7.5 Messages while importing message texts

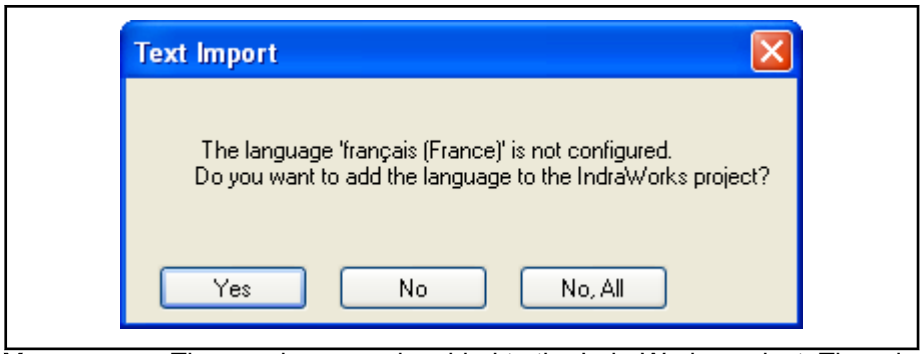
Adding additional languages

It is possible to add further languages via import, even if these were not configured before in the IndraWorks project.

If the file to be imported contains texts in other languages, select whether to import these new languages and thus whether to add them to the IndraWorks project (see fig. 4-41 "Query when importing an XML file" on page 61 and fig. 4-42 "Query when importing a CSV file" on page 62).

XML import of a new language

Example of a query for the additionally imported"French" language when importing an XML file:



Yes The new language is added to the IndraWorks project. There is no other query with regard to this language

No The new language is not added to the IndraWorks project. There is a new query when the next text is entered in this language

No, all The new language is not added to the IndraWorks project. There is no new query when the next texts are entered in this language

Fig. 4-41: Query when importing an XML file

New language for CSV import

If the CSV file to be imported contains texts in a new language, this new language **must** be added for the import.

ProVi messages

Fig. 4-42: Query when importing a CSV file

Importing changed texts in CSV format

As the messages are organized in lines, texts used several times were inserted in all corresponding message lines during export. If such texts are modified externally, all these texts have to be modified. Alternatively, only the first text is changed and applied during reimport. Any further query on this texts is denied. A query whether an import text is applied is displayed whenever the text in the database does not match the text to be imported.

Example:

Example of a query while importing a modified and already existing text:

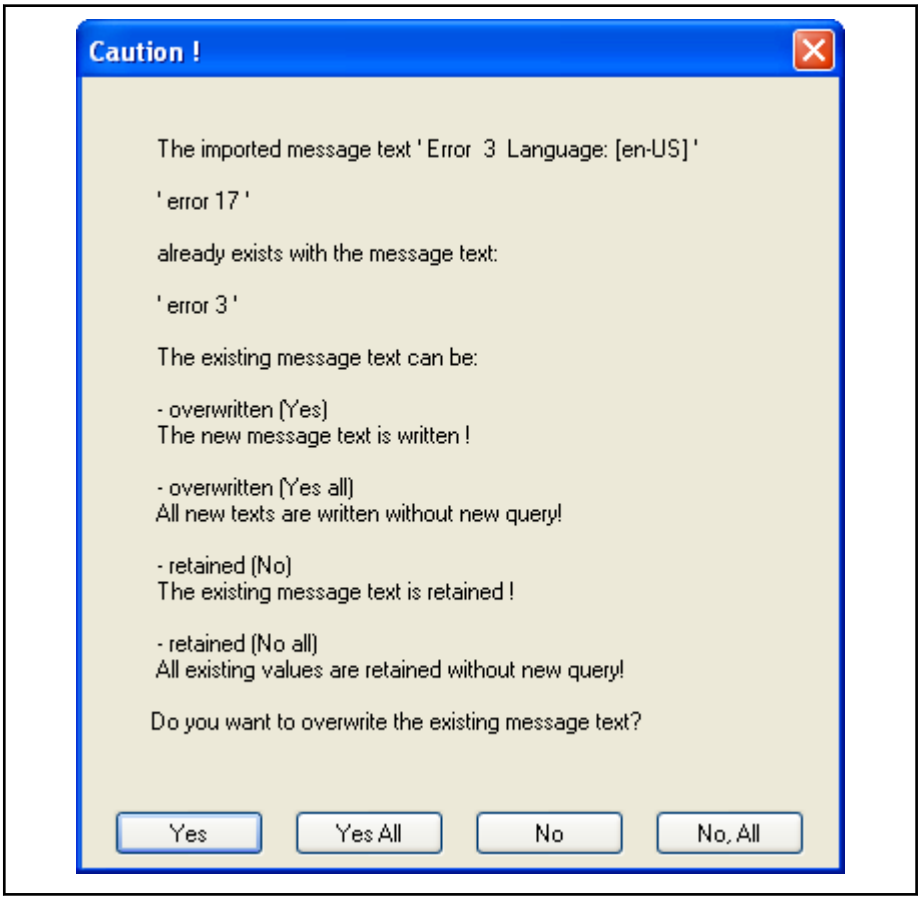


Fig. 4-43: Query when re-importing a modified text

Example:

Example of a query while importing of a modified user note ID of an already existing message:

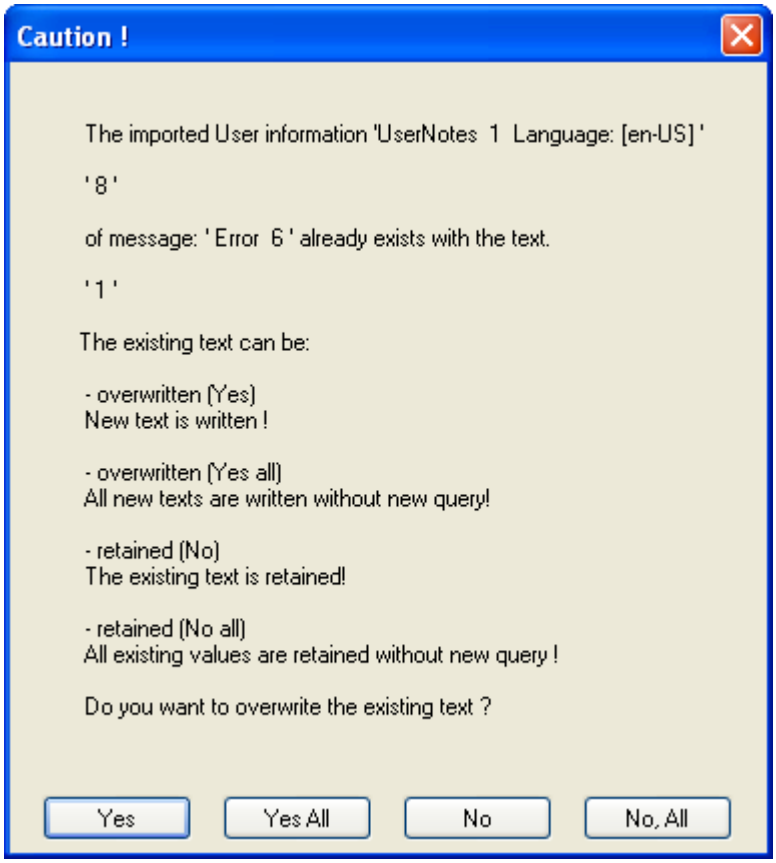


Fig. 4-44: Query when re-importing a modified user note ID

4.7.6 Placeholders in message text

General information

Placeholders can be inserted into the message text that - at a later point in time (i.e. when the message is triggered) - are substituted by current values.

Even variables from the "VAR CONSTANT" declaration area can be used as placeholders in messages.

Enumerations (only required in the PLC editor at runtime) can only be interpreted and displayed using their numerical values. The string cannot be displayed. This works only with the "string variable" (advantage: multilingualism possible).

Variable status Variable values can be displayed in the message. The status of the variables is identified when the message is triggered (frozen status).

Variable texts PLC variables, whose values are interpreted as text numbers, can be interpreted as placeholders. The placeholder is substituted by this message text from the text data editor. The text originates from the same ProVi message type as the ProVi message.

Example:

Declaration: **Error text: INT := 20;**

Message text number 20: "The right door is open"

Text of current message: "Error XYZ further information: **{@TEXT(error text)}**"

Output message: "**Error XYZ further information: The right door is open**"

The status of the variables is identified when the message is triggered (frozen status).

There must not be any further placeholders in the text the variable refers to. If there are still placeholders, these are not replaced.

Fixed placeholders

To identify the part of the PLC program from which the message is triggered (e.g. which machine part), there are some defined placeholders:

- INSTANCE_NAME:
  POU instance from which the message is generated
- INSTANCE_FULLNAME:
  Complete instance path of the POU from which the ProVi message is issued
- INSTANCE_COMMENT:
  Instance comment
- PROGRAM_NAME:
  Name of the program POU from which the ProVi message is issued
- MODULE_NO:
  Module number from which the ProVi message is generated
- VAR_NAME:
  Variable name assigned to the ProVi message
- VAR_COMMENT:
  Variable comment
- VAR_ADDRESS:
  Absolute variable address

Placeholder syntax

{@ COMMAND [%Format]}

Placeholder definitions start with **"{@"** and end with **"}"**. Information in square brackets is optional.

COMMAND *The following placeholders are possible as COMMAND*

- INSTANCE_NAME - POU instance name
- INSTANCE_FULLNAME - Complete POU instance path
- INSTANCE_COMMENT - POU instance comment
- PROGRAM_NAME - Program name
- MODULE_NO module number
- VAR_NAME - Variable name
- VAR_COMMENT - Variable comment

- VAR_ADDRESS - Variable address
- TEXT(VarName) - Variable texts. The text is used under the respective number (status of the "VarName" variable).
- VAR(VarName) - Variable status

Example:

Error XYZ in **{@INSTANCE_NAME}** of **{@VAR(Signal_3)}**

Array elements and structure elements are also possible variable names (VarName). Indexed arrays can also be addressed.

☞ The placeholders TEXT(Varname) and VAR(Varname) may only to be used in the message text. If these placeholders are to be used in other texts such as cause or recovery text, the same placeholders have to be in the message text as well. If these placeholders are not in the message text, they are not replaced in other texts as well.

The specified variable is first searched in the POU in which the ProVi message was issued. If the variable was not found in the POU, global data is scanned. Complete instance names can also be specified e.g. Program1.FB2.Signal_3.

Format The placeholder text output can be formatted via the format. Specifying the format is optional. If no format is specified, the display is as for the status display in IndraLogic (numeric values are shown in decimals, BYTE, WORD, DWORD with 16# as prefix, time values with T# as prefix, etc.).

☞ Formatting is only useful for the VAR placeholder. If formatting is also used for other placeholders, these placeholders are formatted like a STRING variable (type s).

Format syntax

{@ Command [%Format]}

The format structure is:

[Flags][MinLen][{.|,}accuracy] type

The parts in square brackets are optional.

Flags Specifying the flags is optional.

Flag	Description	Default
-	Left-aligned, also if MinLen is given	Right-aligned
0	If "MinLen" is also specified, the character string is filled with zeros up to minimum length. This flag can only be used for x, X, o, b, u types. If both flags are specified, "0" is ignored	No filling

Tab. 4-10: Flags when formatting the placeholder output

MinLen Positive decimal value specifying the minimum number of characters to be output. The remaining text is filled with blank characters (see also flags – and 0). Specifying the minimum length is optional.

Precision Positive decimal value with a leading point specifying the accuracy of the output. The effect of the parameter depends on the type specified. Specifying the accuracy is optional.

ProVi messages

Type	Description	Default
c	No effect	
b, o, x, X	Specifies the minimum number of places to be output. If the value is smaller, it is filled with zeros on the left	At least one place
d, u	Like b, o, x, X, but a comma is also allowed instead of the dot. In such a case, the value is issued as a fixed-point number and the accuracy indicates the number of decimal places	At least one place and no fixed-point number
e, E, f	Number of decimal places behind the comma	6 decimal places behind the comma
g, G	Maximum number of decimal places to be issued	6 decimal places
s, t	Maximum number of characters to be issued	All characters

Tab. 4-11: Precision when formatting the placeholder output

Type If a format is specified, the type has to be specified.

Each type is only permitted for certain variable types.

Type	Variable type	Output format
c	BYTE	ANSI character
d	SINT, INT, DINT	Decimal with sign. If the value is positive, no sign is output
u	BOOL, BYTE, WORD, DWORD, USINT, UINT, UDINT	Decimal without sign
b	BOOL, BYTE, WORD, DWORD, USINT, UINT, UDINT	Binary without sign
o	BOOL, BYTE, WORD, DWORD, USINT, UINT, UDINT	Octal without sign
x	BOOL, BYTE, WORD, DWORD, USINT, UINT, UDINT	Hexadecimal without sign, in lower case
X	BOOL, BYTE, WORD, DWORD, USINT, UINT, UDINT	Hexadecimal without sign, in upper case
f	REAL	Non-integer value with sign as [-]ddd.ddd
e	REAL	Non-integer value with sign as [-]d.dddd e {+|-}ddd
E	REAL	Non-integer value with sign as [-]d.dddd E {+|-}ddd
g	REAL	Non-integer value with sign. It is output as "e" or "f ". The "e" format is used if the exponent is smaller than -4 or if the value is greater than or equal to the given accuracy
G	REAL	Like "g", but "e" is output as "E"
s	STRING	String, character string
t	TIME, TOD, DATE, DT	Date or time. For this type, an additional format specification is possible (see below), e.g. t[MM/dd/yyyy hh-mm-ss tt]

Tab. 4-12: Type for formatting the placeholder output

Formatting type "t" Date and time can be output in any form using this formatting. The formatting can take place in any order with any separators. The different format identifi-

ers are permitted only for specified variable types. The formatting for "**t**" is optional and written in square brackets.

Format identifier	Variable type	Description
h	DT, TOD	Outputs the hours 1-12 (the format identifier "tt" should additionally be used)
hh	DT, TOD	Outputs the hours 1-12 with leading zeros (the format identifier "tt" should additionally be used)
H	DT, TIME, TOD	Outputs the hour from 0-23
HH	DT, TIME, TOD	Outputs the hour from 0-23 with leading zeros
m	DT, TIME, TOD	Outputs the minutes
mm	DT, TIME, TOD	Outputs the minutes with leading zeros
s	DT, TIME, TOD	Outputs the seconds
ss	DT, TIME, TOD	Outputs the seconds with leading zeros
f	DT, TIME, TOD	Shows the split seconds as one digit (100 ms)
ff	DT, TIME, TOD	Shows the split seconds as two digits (10ms)
fff	DT, TIME, TOD	Shows the split seconds as three digits (1 ms)
tt	DT, TOD	Outputs AM or PM
y	DT, DATE	Outputs the year as two-digit number
yy	DT, DATE	Outputs the year as two-digit number with leading zeros
yyyy	DT, DATE	Outputs the year as four-digit number
M	DT, DATE	Outputs the month
MM	DT, DATE	Outputs the month with leading zeros
d	DT, TIME, DATE	Outputs the day of the current month
dd	DT, TIME, DATE	Outputs the day of the current month with leading zeros

Tab. 4-13: Formats of date and time values in the placeholder

Examples:

Variable declaration:	**Signal_3: BYTE := 20;**	
Use as placeholder **without** format:		**{@VAR(Signal_3)}**
Display:	**16#14**	
Usage as placeholder **with** format:		**{@VAR(Signal_3)%010b}**
Format breakdown:	**%**	**Beginning of format**
	0	**Flag**
	10	**MinLen**
	b	**Type "binary"**
Display:	**0000010101**	
Usage as placeholder **with** format:		**{@VAR(Signal_3)%u}**
Format breakdown:	**%**	**Beginning of format**

	u	**Type "decimal"**
Display:	**20**	

Tab. 4-14: Formatting type "t"

4.8 Configuring diagnostics

4.8.1 General information

Specific settings are required to use the diagnostics. First, enable the diagnostics for the current PLC project via the diagnostic configuration.

4.8.2 Diagnostics for IndraLogic 2G

Enabling diagnostics

The programming interface for the control in IndraWorks is integrated in IndraLogic 2G.

To enable the diagnostics for the PLC project, select the control in the project tree and select **ProVi SFC Diagnostics** in the menu item **Diagnostics**:

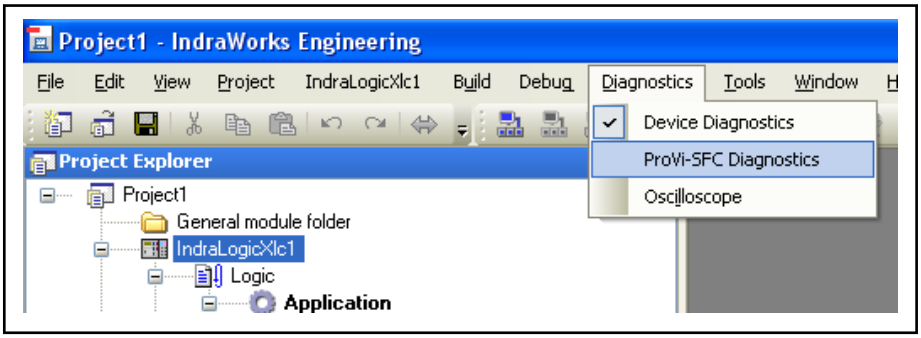


Fig. 4-45: Opening the diagnostic configuration

The enabled diagnostics can be identified by the selected menu item.

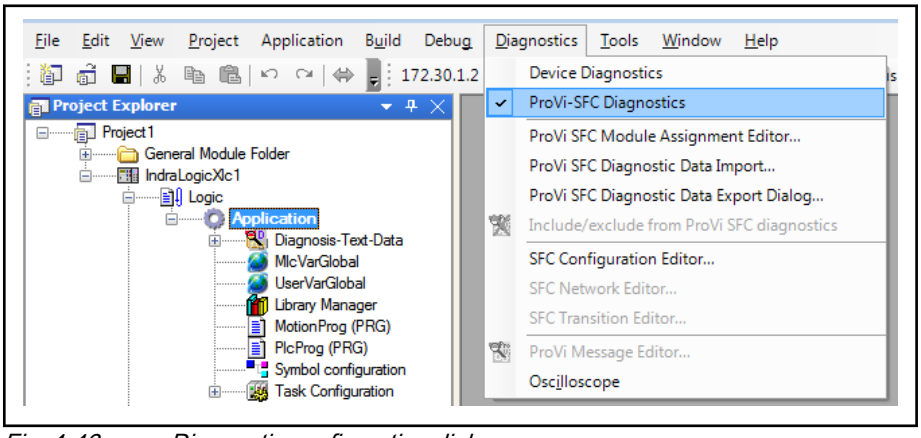


Fig. 4-46: Diagnostic configuration dialog

4.9 Creating diagnostic code in PLC projects

4.9.1 General Information on the IndraLogic 2G diagnostics

To use diagnostics (ProVi messages and SFCs with operation modes), an additional code is required in the PLC program. If the diagnostics was activated and ProVi messages or SFCs with operation modes were also used, additional code has to be generated. The code is automatically created in IndraLogic 2G during compilation.

Diagnostic data was generated correctly

If the diagnostic data is correctly generated in the build process, the following message is reported:

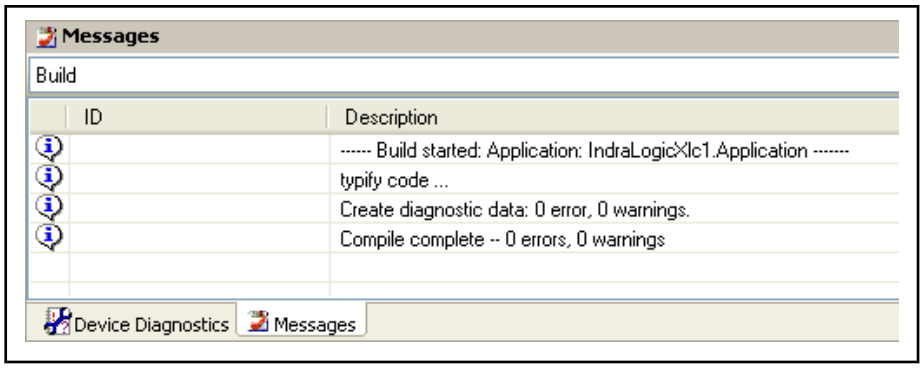


Fig. 4-47: Output in case of a correct generation of diagnostic data

Errors and warnings when generating diagnostic data

If errors or warnings are issued while generating diagnostic data, these are shown in the IndraWorks task list.

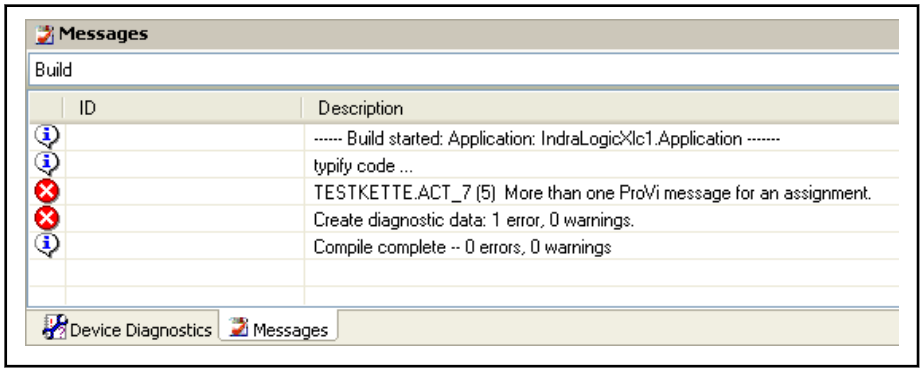


Fig. 4-48: Output of errors when creating diagnostic data

In the error output, it is not possible to jump to the incorrect code. However, it is specified.

Automatic update of the PLC project

After creating the diagnostic data, the PLC project is automatically updated.

The following chapters describe the individual code parts inserted.

4.9.2 Inserting diagnostic variables

As soon as a POU contains diagnostics, a local variable "ILDId" is inserted automatically in the VAR range of the declaration. These variable values must not be changed in the PLC program.

```
27
28      ILDId : WORD; (*Internal for Diagnosis! Do not use or change!*)
29  END_VAR
30
```

Fig. 4-49: Automatically declared diagnostic variable

4.9.3 Opening the diagnostic initialization

If the diagnostics was enabled, an initialization call for the diagnostics is inserted in the main program of the task with diagnostics (in most cases PlcProg). This call may neither be removed nor changed. If this call is not needed anymore, or if it has to be executed at a different place, it is removed or moved automatically.

```
1  ILD_Internal_Init(FALSE); (*Internal for Diagnosis, do not change*)
2
```

Fig. 4-50: Opening the diagnostic initialization

This call cannot be inserted in programs programmed in the PLC programming languages "CFC" or "AS". If such programs are used as the first program of a task with diagnostics (main program), an error message is issued when creating diagnostic data.

☞ There may only be one task with diagnostics per PLC. Diagnostics is not "Thread-save".

4.10 Logbook configuration

If diagnostics are used in a project, all messages issued by the PLC can also be logged in a logbook. Therefore, a logbook has to be created first in IndraWorks Engineering and configured if necessary.

To create a logbook, a visualization device has to be available in an IndraWorks project. This device has a "Logbooks" subitem.

1. Right-click on this node to create a new logbook.

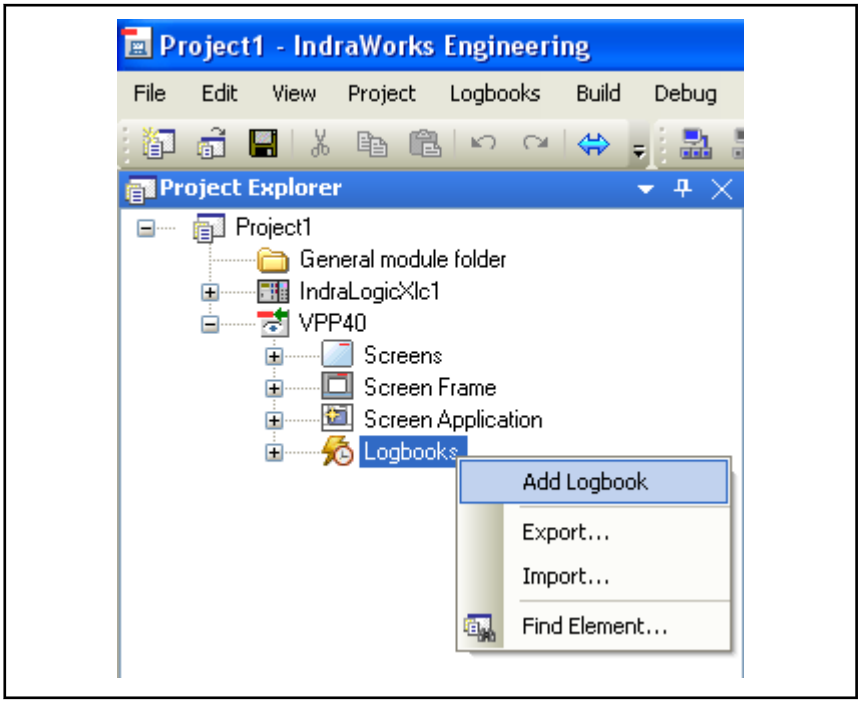


Fig. 4-51: Add Logbook

2. Rename or delete the logbook via the context menu of the new logbook.

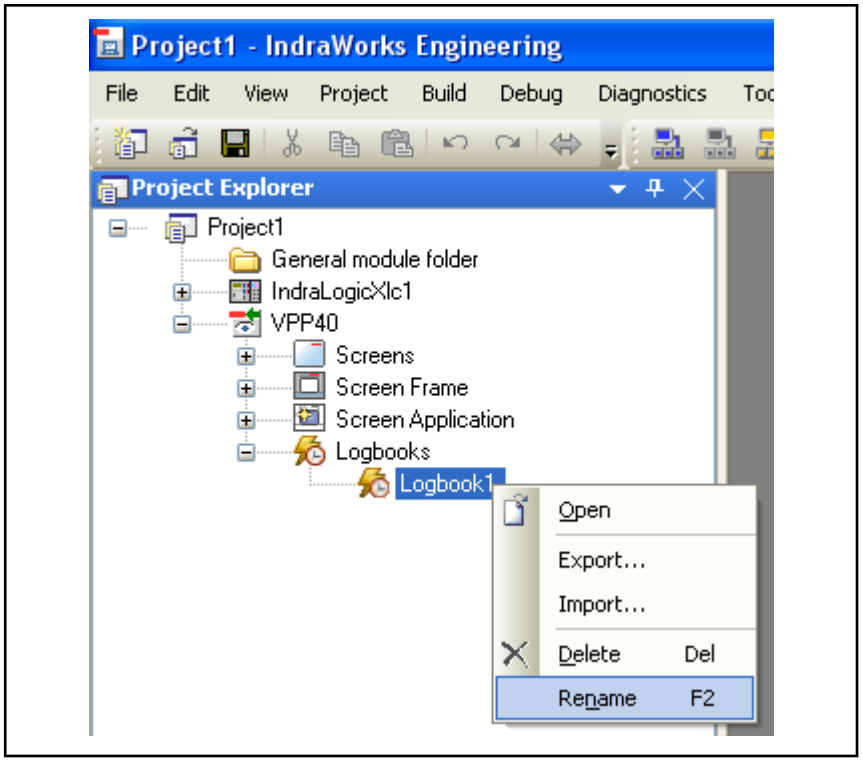


Fig. 4-52: Delete or rename logbook

3. Double-click on the new logbook to open a tab for further settings.

ProVi messages

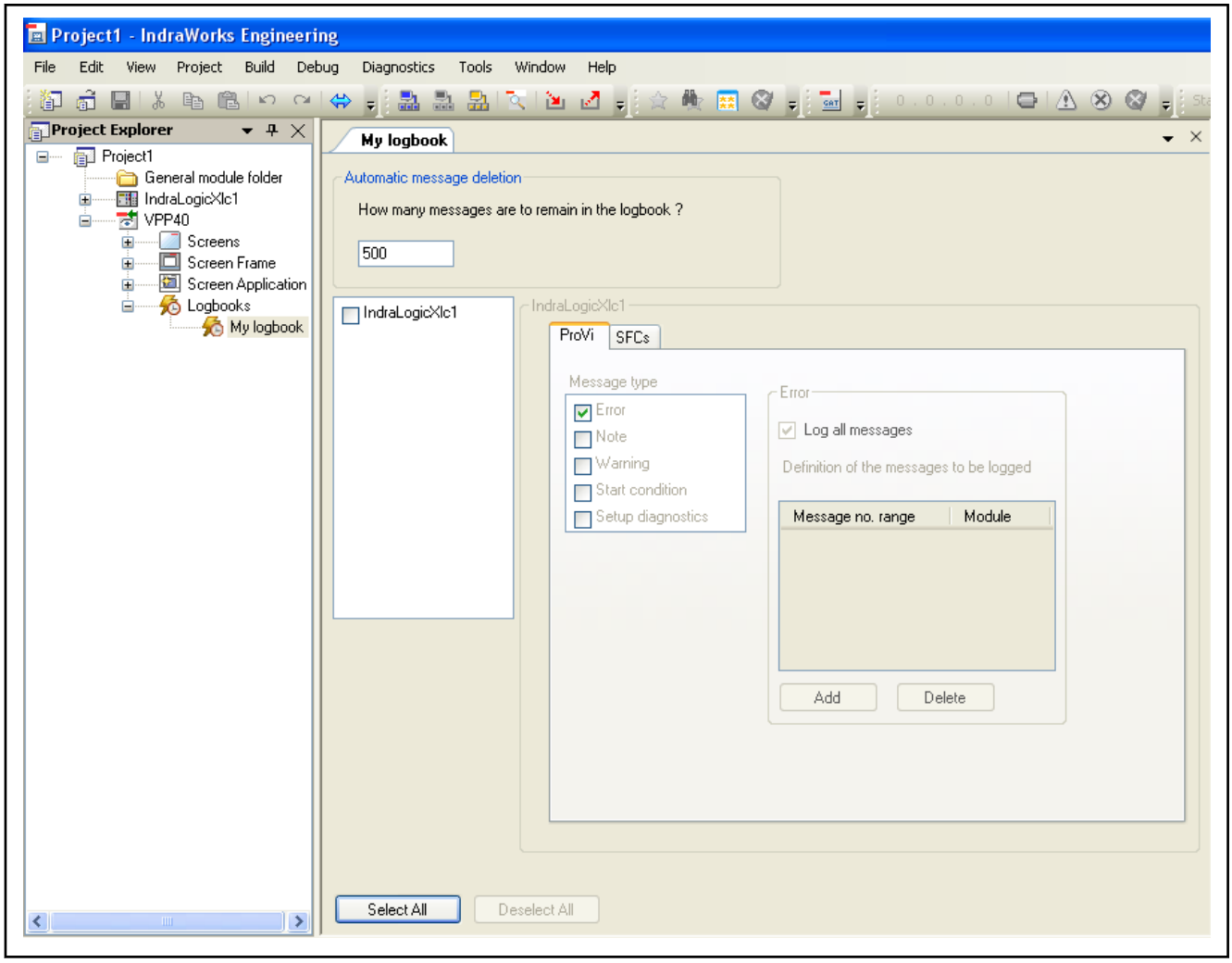


Fig. 4-53: Logbook configuration; default setting

Several logbooks can thus be created. Up to 1000 messages can be stored in each logbook. 500 is set as default value. If the set limit is reached, the first 50 messages are deleted and new messages are added at the end. Thus, the latest messages are always documented in the logbook.

The ProVi or SFC messages to be logged in the logbook can be configured (see fig. 4-53 "Logbook configuration; default setting" on page 72). Select the desired control first. If there is more than one control in the project, all controls are shown for selection. It is also possible to select several controls in one logbook.

As soon as a control is selected, further control-specific settings can be made.

In general, every message can be logged in the logbook. With the default setting, all ProVi **errors** and all SFC **errors** are documented. If required, further message types can be added for ProVi as well as for SFCs, but they can also be deselected.

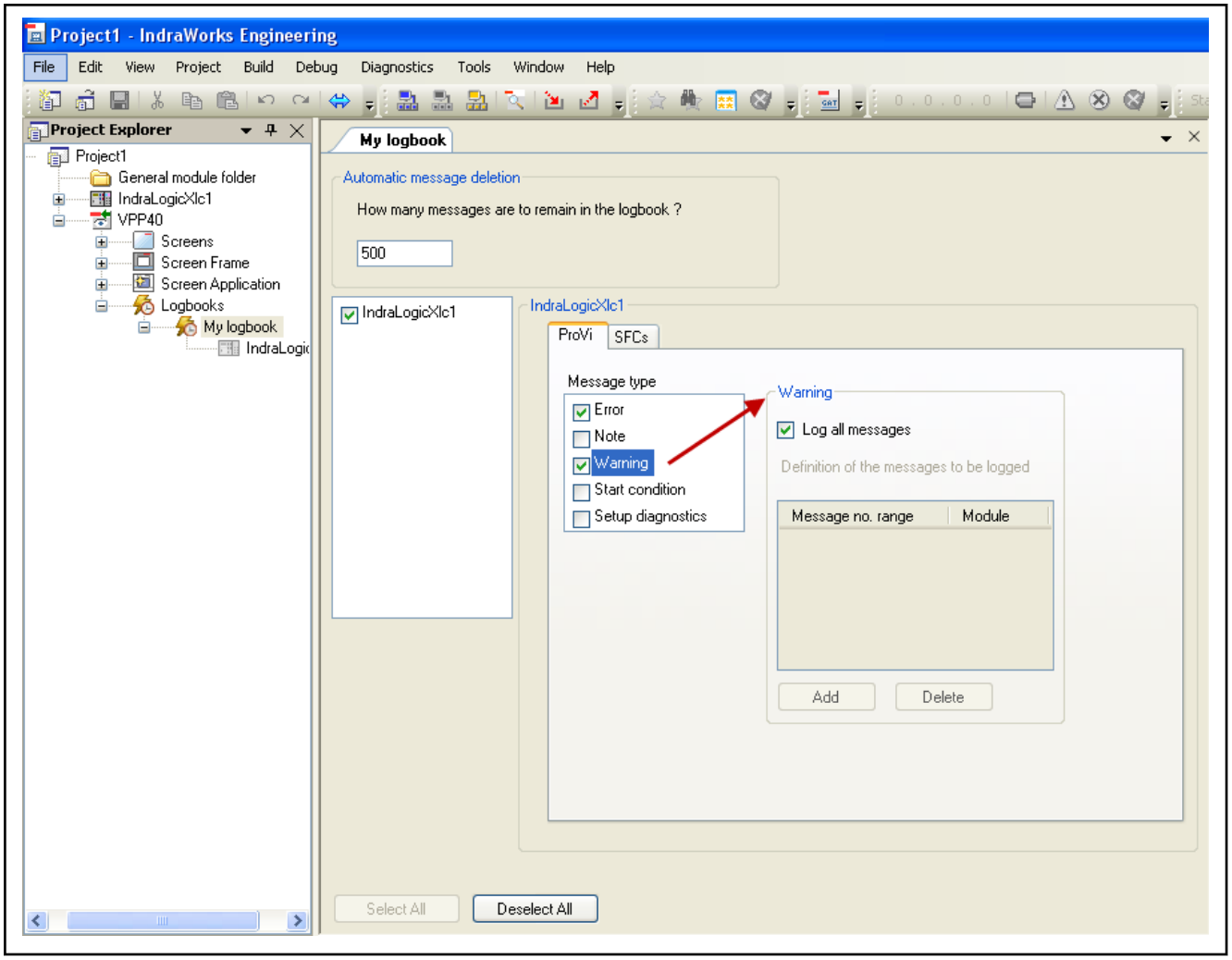


Fig. 4-54: Logbook configuration; message type selection

When selecting individual message types, further restrictions such as "Message number range" and "Module" can be made.

Deselecting the "Log all messages" option (this option always applies to the highlighted error type) and confirming the **Add** button, opens another dialog. Different limits can be entered.

ProVi messages

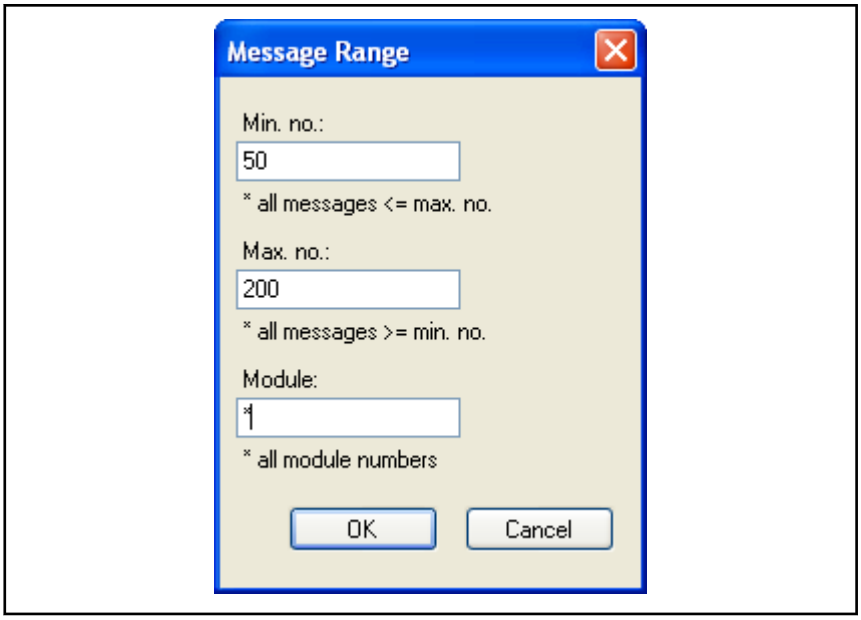


Fig. 4-55: Dialog to configure the message ranges

All messages outside the specified limit range are not logged.

In this example, all messages of type "Warning", whose message number is greater than or equal to 50 and less than or equal to 200, are logged.

If only messages of a certain module number are to be logged, the corresponding module number has to be entered into the module box and a "*" has to be specified in the boxes “Min. no”. and “Max. no.”. After confirming the entry, the entered values are displayed in the main overview.

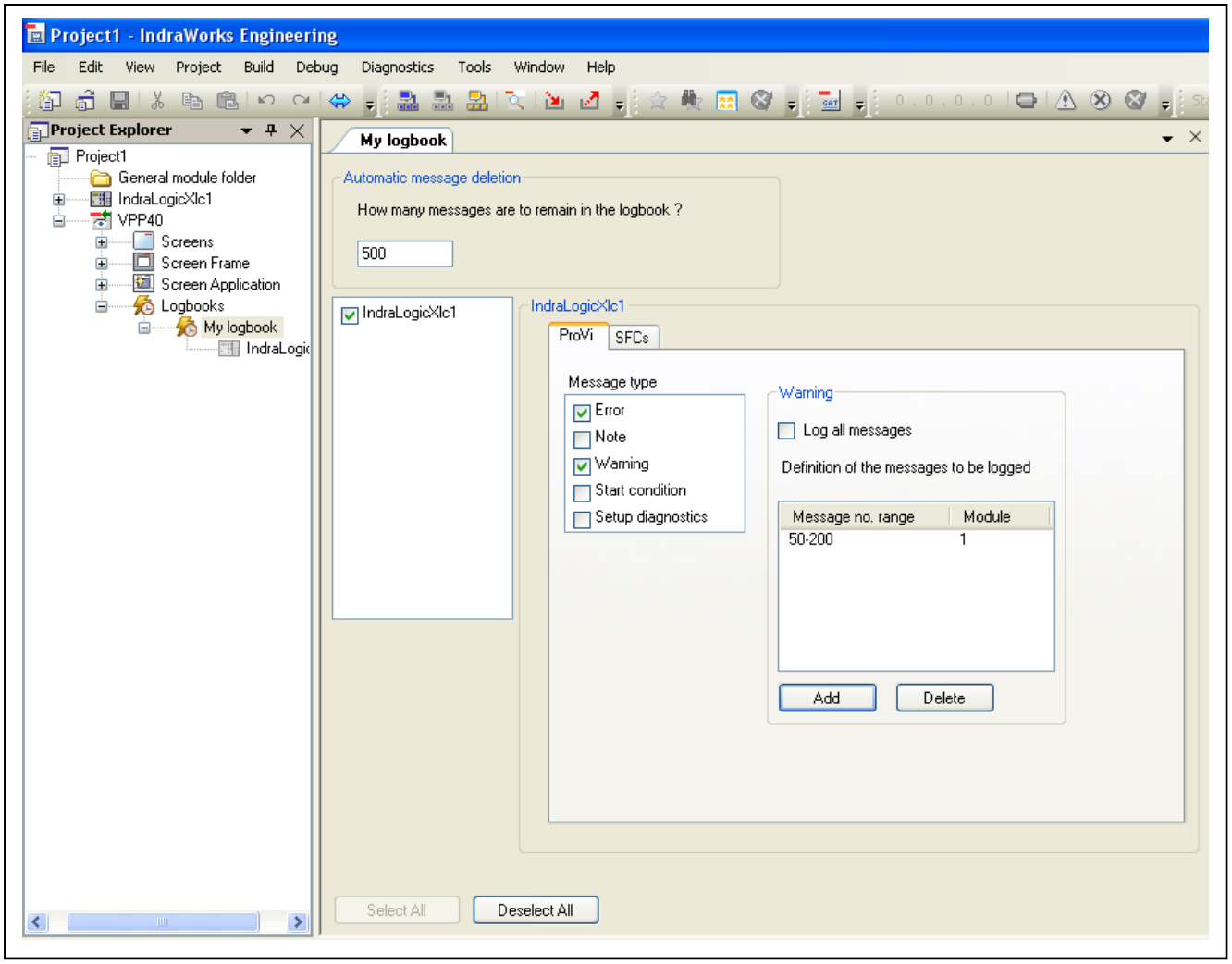


Fig. 4-56: Already configured message range

4. To save all settings correctly, close the tab or save the whole project.

The documentation "Rexroth IndraWorks HMI", chapter "Logbook" describes the access to the individual logbooks at runtime, i.e. from the IndraWorks Operation Desktop.

4.11 Module assignment editor

☞ Consider the following section only if you have divided your PLC project into different modules or if you use several instances of a function block with ProVi or SFC diagnostics.

If a function block with diagnostics is declared several times, this leads to the question in which module the diagnostics of the individual instances is to be shown. A unique module number can be assigned for each use of a diagnostics in the module assignment editor. Thus, the messages programmed for a specific module can also be assigned with a different module number later on.

Opening module assignment editor

Open the module assignment editor via the node **Diagnostic text data ▸ Module assignments** in the Project Explorer.

Enter the module assignments into the table considering the syntax.

Module assignment syntax

POU[.Instance][.Diagnostictype][.Messagetype][.OrgModulNo=NewModule-No

The specifications in square brackets are optional (from left to right). Example: If no "Diagnostic type" is selected, no selection can be made for the parameters "Message type" and "Module number" located next to the right (for further examples see tab. 4-15 "Examples of the module number assignment" on page 77).

For the syntax of module assignments with the diagnostic type SFC, note that the parameters "Message type" and "Module number" located on the right, cannot be selected.

It is mandatory to enter the new module number to provide a purpose for the module assignment. If the column of the new module number remains empty, the respective line is lost when closing. Valid entries of the module numbers are between 0 and 99!

Entering the instance

POU can be entered either manually or via drag&drop using the process variable browser.

- Manually: The complete instance name has to be specified for the POU, e.g. "Station_01.Drilling_Module1"
- Process variable browser: Drag the instance from the “Process Variables” window to the previously selected cell of the “POU/Instance” column

Please ensure a correct spelling of the instance names. Instance names are not checked automatically.

Example:

ProVi messages

There is a function block (FB_Drilling) that controls a drill completely and also contains the diagnostic messages of the drill. The control should contain two modules, each controlling one drill. For both drills, instances of the same function block are used.

The program, in which both drills are declared, is called Station_01.

```
4   VAR
5     Drilling_Module1: FB_Drilling;
6     Drilling_Module2: FB_Drilling;
7
```

Fig. 4-57: Declaring two FB_Drilling instances

The ProVi messages (error and information) are programmed in the function block.

When programming, module number 1 was specified for these diagnostics (see "Module assignment syntax" on page 76). However, the diagnostics for the first drill is to be displayed in module 1 and the diagnostics for the second drill is to be displayed in module 2.

Module assignment editor [IndraLogicXlc1: Logic: Application]

	POU/Instance	Diagnostic Type	Message Type	Module number	New module number
▶	Station_01.Drillin_Module1	PROVI	Error	1	2
	Station_01.Drillin_Module1	PROVI	Note	1	2
*		All	All		

Fig. 4-58: Diagnostic module assignment

In the module assignment editor, module number 2 is assigned to all diagnostics from Drilling_Modul2; in this case, the two ProVi messages “Error” and “Note”.

If no entry is made in this editor, the original module number 1 is used.

Examples of module assignments:

Character string	Description
Station_01.Drilling_Module2.PROVI.Error.1=3	The ProVi error with the module number 1 of this instance is displayed in module 3
Station_01.Drilling_Module2.PROVI.Error=3	All ProVi errors in this instance are displayed in module 3
Station_01.Drilling_Module2.PROVI=3	All ProVi messages in this instance are displayed in module 3
Station_01.Drilling_Module2=5	All diagnostics in this POU instance are displayed in module 5. This also applies to all instances of the POUs declared in this POU
Station_01=5	All diagnostics in this program are displayed in module 5. This also applies to all instances of the POUs declared in this POU (in this example e.g. Drilling_Module1 and Drilling_Module2)

Tab. 4-15: Examples of the module number assignment

The last, i.e. the most accurate module assignment is always applied to an instance path.

The following assignments are assumed:

- Station_01.Drilling_Module2.PROVI.Error.1 =4
- Station_01.Drilling_Module2=3
- Station_01=2

The exemplary diagnostics is displayed in the following modules:

- Instance Station_01.Drilling_Module1
  ProVi note 1 in module 2
  ProVi note 1 in module 2
- Instance Station_01.Drilling_Module2
  ProVi note 1 in module 4
  ProVi note 1 in module 3

4.12 Exporting and importing diagnostic data

4.12.1 General information

If ProVi or SFC diagnostics and the corresponding message texts were used, they can also be exported or imported from other projects. There are two options to export diagnostic data. First, the diagnostic data can be exported alone. Individual POUs can be selected in the editor and the individual components of the diagnostic data can be selected. Second, the complete PLC project is exported with the diagnostic data. It cannot be intervened.

4.12.2 Exporting diagnostic data

To export diagnostic data, there two variants provided with different functions.

If, for example, only the message texts from a PLC project are needed to be imported to a different but very similar machine, the "Diagnostic Data Export" dialog can be used. Select the menu function **Diagnostics ▸ ProVi SFC diagnostic data export** to open the dialog for the diagnostic data export.

ProVi messages

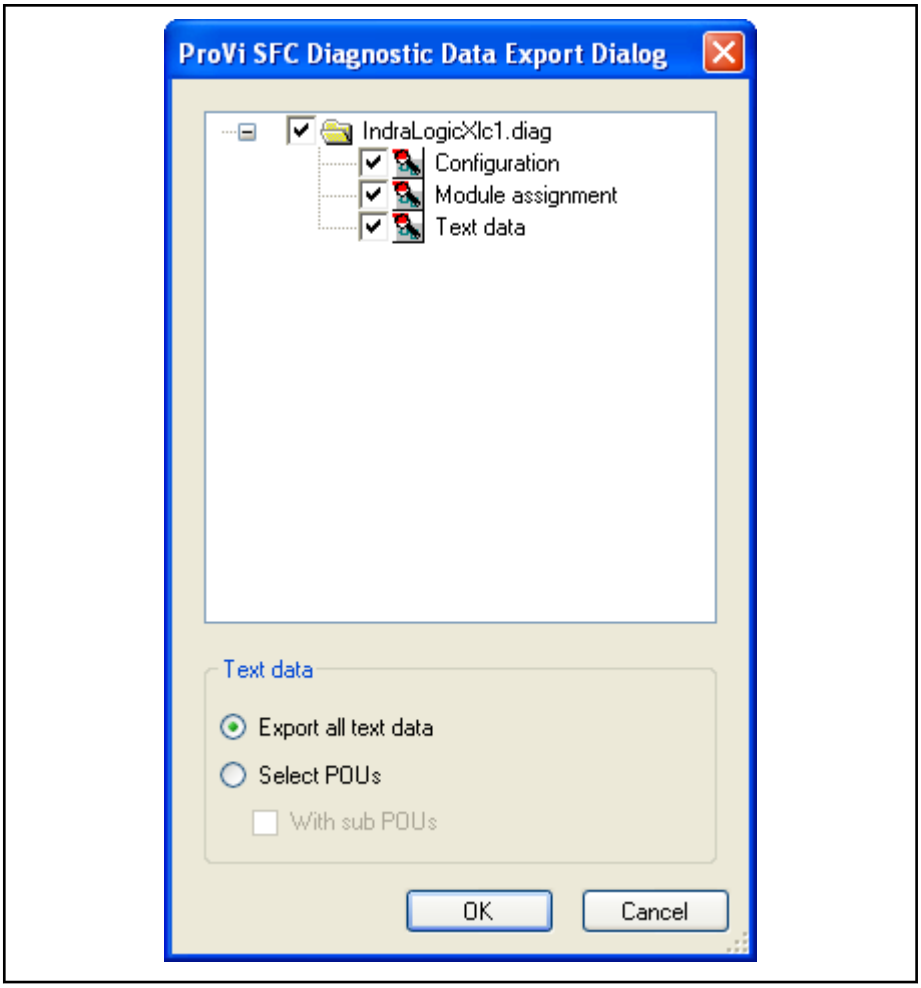


Fig. 4-59: Dialog: Diagnostic data export

The exported data is saved in a file with the extension "*.dex" and contains three main segments (configuration, module assignment, text data). The storage location of this file is queried separately after confirming with **OK**. In this dialog, select the elements to be exported.

Configuration The only configuration data currently available is the setting whether the project contains diagnostic data.

Module assignment The settings from the dialog "Diagnostic Module Assignment" are exported.

Text data The text data belonging to the respective PLC project is exported. All texts are exported, even if they are not used in any ProVi message.

Text data – Selecting POUs With the "Select POUs" option, only texts used in the selected POUs are exported.

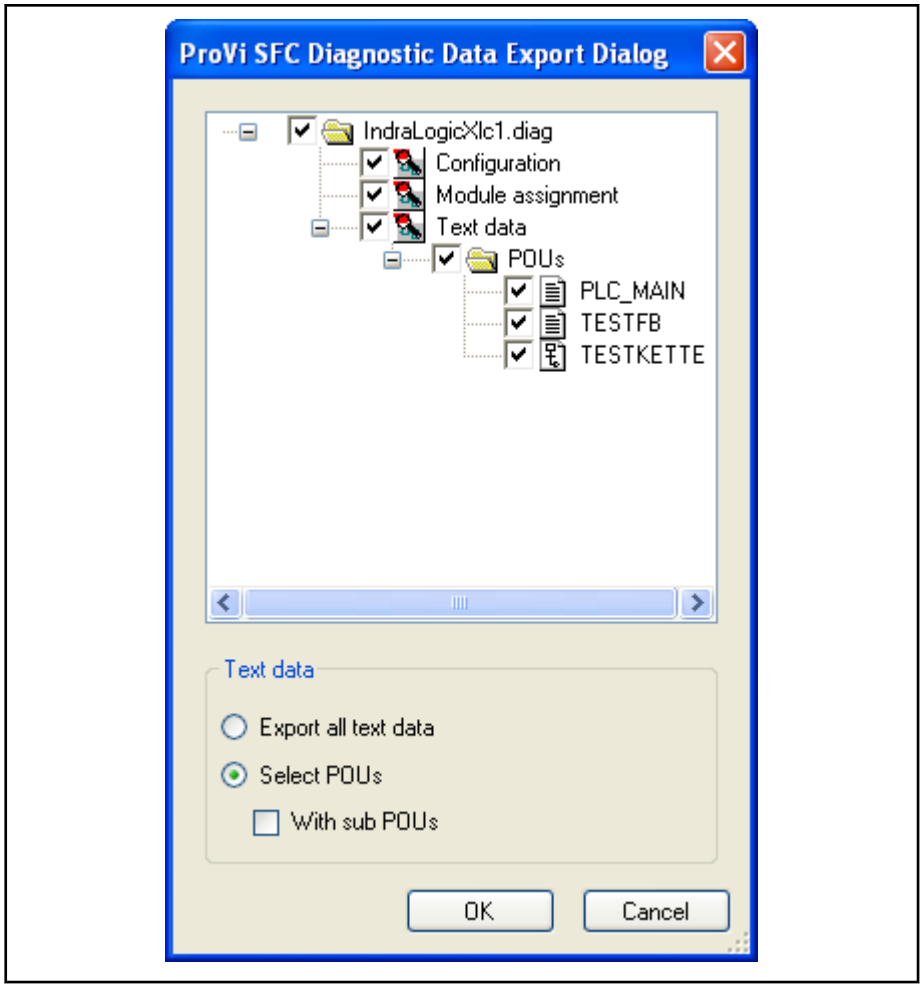


Fig. 4-60: Exporting texts of specific POUs

To export the texts currently used, run the "Create diagnostic data" command before exporting.

If the "With sub POUs" option is selected, those POU texts used by the selected POUs are additionally exported.

4.12.3 Importing diagnostic data

Exported diagnostic data of a PLC project can be imported into a different project.

There are two import options:

1. Only diagnostic data is imported.
2. A complete PLC project with or without diagnostic data is imported.

Importing only diagnostic data

If only the diagnostic data should be imported into a PLC project, select **Diagnostics ▸ ProVi SFC diagnostic data import** in the menu.

In the dialog for the diagnostic data import, select the *.dex file to be imported.

It is not possible to select individual components of the diagnostic data for the import.

If there is data in the export file that is already in the PLC project, a query, whether this data is to be overwritten, is sent.

4.13 Diagnostic data added to version control

4.13.1 Adding a project with diagnostics to version control

Projects with a ProVi SFC diagnostics can be added to version control. Both nodes "Diagnostic text data" and "ProVi SFC text data editor" are marked as versionized. Apart from the versionized diagnostic data, the library manager node is also changed by inserting the diagnostic library "RIL-Diagnosis.lib".

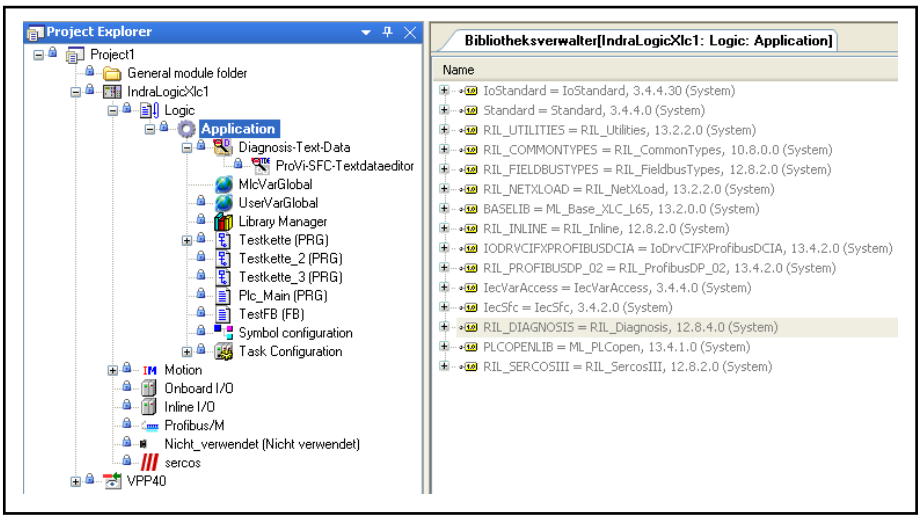


Fig. 4-61: Project with diagnostics added to version control

The "ProVi SFC Text data editor" node is checked out when changes made to the diagnostic data.

Dialogs changing the diagnostic data apart from the text data editor: ProVi dialog, the module assignment dialog and the import of diagnostic data or diagnostic texts. Changes are applied to the version control by checking in the "ProVi SFC text data editor" node.

Standard versioning functions such as "Hijack", "Undo Hijacking", "Checkout", "Undo Checkout" and "Check In" are also available for the diagnostic elements.

Currently, diagnostic data cannot be compared, deleted, merged or printed.

4.13.2 Enabling or disabling the diagnostics in a versionized project

If a project with diagnostics is added to version control, the nodes "Diagnostic text data" and "ProVi SFC text data editor" are inserted as new elements after the "Application" node has been checked out in the Project Explorer. The library manager is also checked out as the diagnostic library "RIL_Diagnosis.lib" is inserted into the project. The ProVi SFC diagnostics is permanently applied to the project by checking in the "Application" and the "Library manager" node.

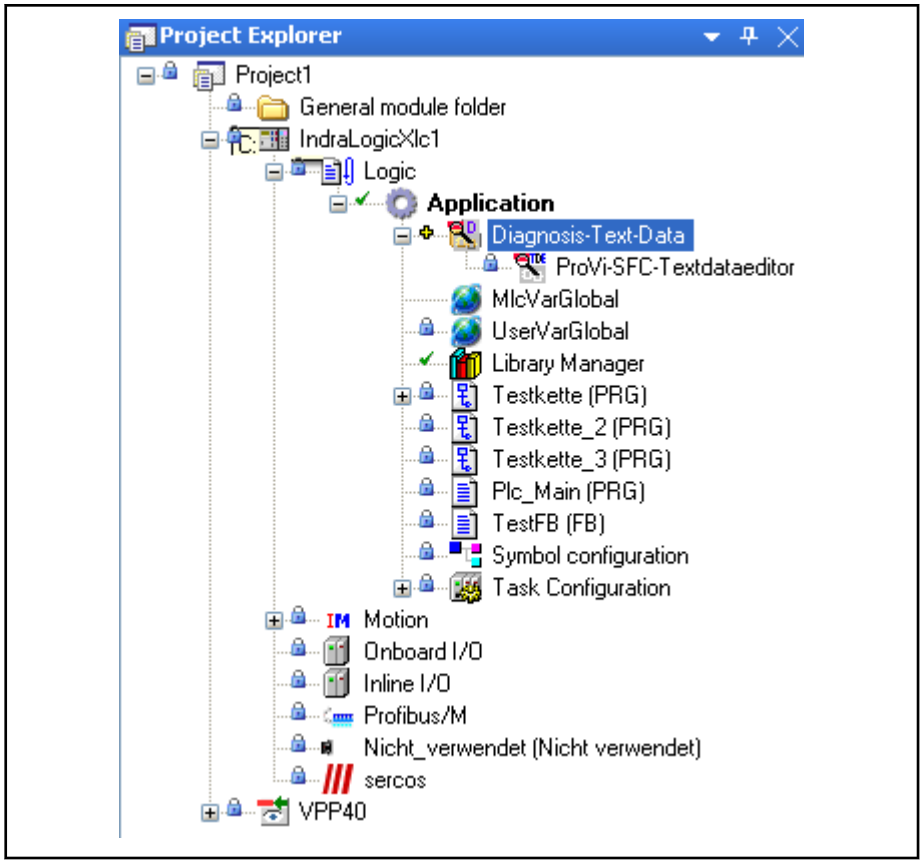


Fig. 4-62: Project added to version control with re-enabled diagnostics

When disabling the diagnostics, both diagnostic nodes are removed again after checking out the "Application" node. The RIL-Diagnosis.lib remains in the project. Thus, the library manager is not checked out anymore when enabling the diagnostics again in this project.

The diagnostic data is retained after disabling the diagnostics (as the ProVi pragmas) and is available after enabling the diagnostics again.

5 RIL_Diagnosis library

5.1 Overview

If the diagnostics is activated for a PLC project and if at least one ProVi message is configured, the "RIL_Diagnosis.library" is integrated automatically.

This library provides types and function blocks to access the diagnostics.

Note: Diagnostic function blocks are not "thread-safe"!

When using tasks, diagnostic function blocks may only be used in one task. Otherwise, the diagnostics might not work properly.

Data types and function blocks:

5.2 ProViType

This data type defines the ProVi types for transfer to the function blocks of this library.

☞ Always use these definitions instead of directly transferring the real value.

ProViTypeError	ProVi error type
ProViTypeInfo	ProVi note type
ProViTypeWarning	ProVi warning type
ProViTypeStartup	ProVi start condition type
ProViTypeSetup	ProVi setup diagnostic type

5.3 Change check

5.3.1 ProViMessageChanged

This function block is used to check whether the diagnostics have changed for a specific ProVi type. The output parameter is set to "TRUE" if new messages are added or if already triggered messages are not pending anymore.

This function block is also used to collect messages that are pending only for a short period of time, e.g. which are displayed for one PLC cycle only.

To check different ProVi types, create one function block for each type.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of the ProVi message to be checked

Output parameters:

Changed (BOOL):	TRUE	Some messages changed
	FALSE	No messages changed

5.4 Resetting messages

5.4.1 Overview

These function blocks reset diagnostic messages.

You can specify whether only the messages programmed as non-expiring are to be reset or whether all messages are to be reset.

If all messages are reset and if the condition for a message is still fulfilled, this message is immediately triggered again. This also means that this message is provided with a new time stamp, that a new log book entry is created and a new email is sent.

There are different function blocks to reset messages. These function blocks differ with regard to the messages to be reset.

General input parameter

All function blocks obtain at least these three input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	

General output parameter

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

Module number

Each type also has another function block containing a module number (DWORD) as additional input. For these function blocks, only messages with the corresponding module number are reset.

Function blocks to reset messages:

- chapter 5.4.2 "ResetProVi" on page 85
- chapter 5.4.3 "ResetProViType" on page 86
- chapter 5.4.4 "ResetProViTypeModule" on page 86
- chapter 5.4.5 "ResetProViCategory" on page 86
- chapter 5.4.6 "ResetProViCategoryModule" on page 87
- chapter 5.4.7 "ResetProViCategoryArea" on page 87
- chapter 5.4.8 "ResetProViCategoryAreaModule" on page 88
- chapter 5.4.9 "ResetProViGroup" on page 88
- chapter 5.4.10 "ResetProViGroupModule" on page 89
- chapter 5.4.11 "ResetProViGroupArea" on page 89
- chapter 5.4.12 "ResetProViGroupAreaModule" on page 90
- chapter 5.4.13 "ResetProViMessage" on page 90
- chapter 5.4.14 "ResetProViMessageModule" on page 91
- chapter 5.4.15 "ResetProViMessageArea" on page 91
- chapter 5.4.16 "ResetProViMessageAreaModule" on page 91

5.4.2 ResetProVi

This function block resets all messages.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Only messages programmed as non-expiring are reset.

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.3 ResetProViType

This function block resets all messages of a specific type.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.4 ResetProViTypeModule

This function block resets all messages of a specific type belonging to the specified module.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Module (DWORD):	Module number of the ProVi messages to be reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.5 ResetProViCategory

This function block resets all messages of a specific error category.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Category (BYTE):	Only messages of this error category are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.6 ResetProViCategoryModule

This function block resets all messages of a specific error category belonging to the specified module.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Module (DWORD):	Module number of the ProVi messages to be reset	
Category (BYTE):	Only messages of this error category are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.7 ResetProViCategoryArea

This function block resets all messages of a specific type belonging to a specific range of error categories.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset
MinCategory (BYTE):	Only messages, whose error category is greater than or equal to this value, are reset
MaxCategory (BYTE):	Only messages, whose error category is lower than or equal to this value, are reset

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.8 ResetProViCategoryAreaModule

This function block resets all messages of a specific type belonging to a specific range of error categories and a specific module number.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Module (DWORD):	Module number of the ProVi messages to be reset	
MinCategory (BYTE):	Only messages, whose error category is greater than or equal to this value, are reset	
MaxCategory (BYTE):	Only messages, whose error category is lower than or equal to this value, are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.9 ResetProViGroup

This function block resets all messages of a specific message group.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset
Group (BYTE):	Only messages of this message group are reset

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.10 ResetProViGroupModule

This function block resets all messages of a specific message group belonging to the specified module.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Module (DWORD):	Module number of the ProVi messages to be reset	
Group (BYTE):	Only messages of this message group are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.11 ResetProViGroupArea

This function block resets all messages of a specific type, which belong to a specific range of message groups.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
MinGroup (BYTE):	Only messages, whose message group is greater than or equal to this value, are reset	
MaxGroup (BYTE):	Only messages, whose message group is lower than or equal to this value, are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.12 ResetProViGroupAreaModule

This function block resets all messages of a specific type belonging to a specific range of message groups and a specific module number.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Module (DWORD):	Module number of the ProVi messages to be reset	
MinGroup (BYTE):	Only messages, whose message group is greater than or equal to this value, are reset	
MaxGroup (BYTE):	Only messages, whose message group is lower than or equal to this value, are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.13 ResetProViMessage

This function block resets all messages with a specific message number.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Message (DWORD):	Only messages with this message number are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.14 ResetProViMessageModule

This function block resets all messages with a specific message number belonging to the specified module.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Module (DWORD):	Module number of the ProVi messages to be reset	
Message (DWORD):	Only messages with this message number are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.15 ResetProViMessageArea

This function block resets all messages of a specific type belonging to the specified range of message numbers.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
MinMessage (BYTE):	Only messages, whose message number is greater than or equal to this value, are reset	
MaxMessage (BYTE):	Only messages, whose message number is lower than or equal to this value, are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.4.16 ResetProViMessageAreaModule

This function block resets all messages of a specific type belonging to the specified range of message numbers and to a specific module number.

Input parameters:

Execute (BOOL):	Reset ProVi messages at a rising edge	
ResetAll (BOOL):	TRUE	Reset all messages
	FALSE	Reset only messages programmed as non-expiring
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages to be reset	
Module (DWORD):	Module number of the ProVi messages to be reset	
MinMessage (BYTE):	Only messages, whose message number is greater than or equal to this value, are reset	
MaxMessage (BYTE):	Only messages, whose message number is lower than or equal to this value, are reset	

Output parameters:

Active (BOOL):	TRUE	It is reset
	FALSE	It is not reset

5.5 Determining, whether messages are pending

5.5.1 Overview

Use these function blocks to determine whether at least one of the indicated ProVi messages is pending.

There are several function blocks to determine whether messages are pending. These function blocks differ with regard to the messages searched for.

General input parameter

All function blocks obtain at least this input parameter:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages

General output parameter

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

Module number

Each type also has another function block containing a module number (DWORD) as additional input. In case of these function blocks, it is only searched for messages with the corresponding module number.

Function blocks to determine whether messages are pending:

- chapter 5.5.2 "PendingProViType" on page 93
- chapter 5.5.3 "PendingProViTypeModule" on page 93
- chapter 5.5.4 "PendingProViCategory" on page 93
- chapter 5.5.5 "PendingProViCategoryModule" on page 94
- chapter 5.5.6 "PendingProViCategoryArea" on page 94
- chapter 5.5.7 "PendingProViCategoryAreaModule" on page 94
- chapter 5.5.8 "PendingProViGroup" on page 95
- chapter 5.5.9 "PendingProViGroupModule" on page 95

5.5.2 PendingProViType

This function block determines whether messages of a specific type are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.3 PendingProViTypeModule

This function block determines, whether messages of a specific type belonging to a certain module, are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Module (DWORD):	Module number of the ProVi messages to be searched for

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.4 PendingProViCategory

This function block determines whether messages of a specific error category are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Category (BYTE):	Only messages of this error category are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.5 PendingProViCategoryModule

This function block determines whether messages of a specific error category belonging to a specific module are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Module (DWORD):	Module number of the ProVi messages to be searched for
Category (BYTE):	Only messages of this error category are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.6 PendingProViCategoryArea

This function block determines whether messages belonging to a specific range of error categories are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
MinCategory (BYTE):	Only messages, whose error category is greater than or equal to this value, are determined
MaxCategory (BYTE):	Only messages, whose error category is lower than or equal to this value, are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.7 PendingProViCategoryAreaModule

This function block determines whether messages belonging to a specific range of error categories and a specific module are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Module (DWORD):	Module number of the ProVi messages to be searched for
MinCategory (BYTE):	Only messages, whose error category is greater than or equal to this value, are determined
MaxCategory (BYTE):	Only messages, whose error category is lower than or equal to this value, are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.8 PendingProViGroup

This function block determines whether messages of a certain message group are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Group (BYTE):	Only messages of this message group are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.9 PendingProViGroupModule

This function block determines whether messages of a specific message group belonging to a specific module are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Module (DWORD):	Module number of the ProVi messages to be searched for
Group (BYTE):	Only messages of this message group are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.10 PendingProViGroupArea

This function block determines whether messages belonging to a specific range of message groups are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
MinGroup (BYTE):	Only messages, whose message group is greater than or equal to this value, are determined
MaxGroup (BYTE):	Only messages, whose message group is lower than or equal to this value, are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.11 PendingProViGroupAreaModule

This function block determines whether messages belonging to a specific range of message groups and a specific module are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Module (DWORD):	Module number of the ProVi messages to be searched for
MinGroup (BYTE):	Only messages, whose message group is greater than or equal to this value, are determined
MaxGroup (BYTE):	Only messages, whose message group is lower than or equal to this value, are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.12 PendingProViMessage

This function block determines whether messages with a certain message number are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Message (DWORD):	Only messages with this message number are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.13 PendingProViMessageModule

This function block determines whether messages with a specific message number belonging to a specific module are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Module (DWORD):	Module number of the ProVi messages to be searched for
Message (DWORD):	Only messages with this message number are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.14 PendingProViMessageArea

This function block determines whether messages belonging to a specific range of message numbers are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84	Type of searched ProVi messages
MinMessage (BYTE):	Only messages, whose message number is greater than or equal to this value, are determined
MaxMessage (BYTE):	Only messages, whose message number is lower than or equal to this value, are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.5.15 PendingProViMessageAreaModule

This function block determines whether messages belonging to a specific range of message numbers and a specific module are pending.

Input parameters:

ProViType: (chapter 5.2 "ProViType" on page 84)	Type of searched ProVi messages
Module (DWORD):	Module number of the ProVi messages to be searched for
MinMessage (BYTE):	Only messages, whose message number is greater than or equal to this value, are determined
MaxMessage (BYTE):	Only messages, whose message number is lower than or equal to this value, are determined

Output parameters:

Pending (BOOL):	TRUE	Min. one searched message
	FALSE	No searched message available

5.6 IL_ProViArray

This program function block "IL_ProViArray" consists only of a body with the definition of input and output variables. It is provided for the usage of bit arrays with ProVi messages.

Input parameters:

Enable (BOOL)	Enabling to issue messages
ArrayName (STRING(80)):	Array used to issue messages
ArraySize (DWORD)	Array size and number of messages
ProViType: (chapter 5.2 "ProViType" on page 84)	Type of ProVi messages issued by this function block
Module (BYTE)	Module number used when issuing messages
ProViOffset (DWORD)	Defines the message numbers and the following assignment of the message texts. The message number consists of offset + array bit.

Output parameters:

InOperation (BOOL):	TRUE	The program function block is currently processed
	FALSE	The program function block is not currently processed

6 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany

Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone:	**+49 9352 40 5060**
Fax:	**+49 9352 18 4941**
E-mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com/

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide

Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information

To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

Index

S

T

U

W

X

Notes

Notes

The Drive & Control Company

Rexroth
Bosch Group

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Phone +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics

R911344037

DOK-IWORKS-PROVI2G*V14-AP02-EN-P