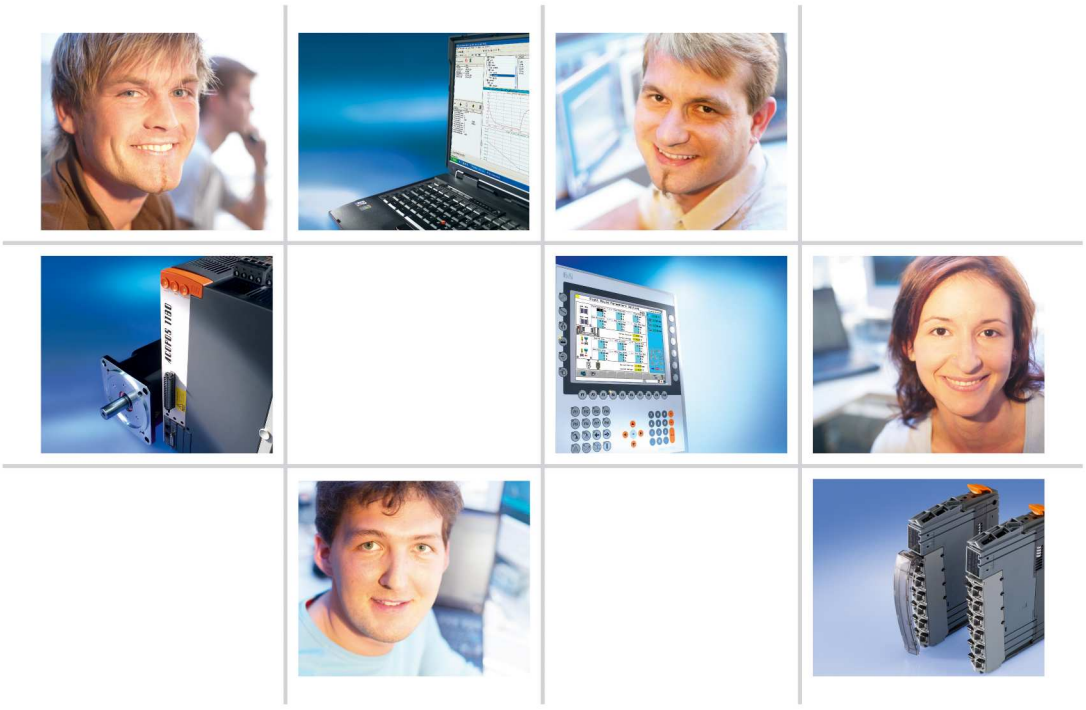
Automation Studio Diagnostics

TM223



Perfection in Automation

www.br-automation.com

Requirements

Training modules:	TM210 – The Basics of Automation Studio
	TM211 – Automation Studio Online Communication
Software:	Automation Studio 3 Automation Runtime 2.90
Hardware:	None

Table of contents

1. INTRODUCTION

As long as a system is running normally and undesired states don't crop up, everything is OK.
However, if this isn't always the case, you need tools and help to get to the bottom of whatever is causing your error.

Automation Studio provides several uniform diagnostics tools for controller systems.
All important data can be read, made and saved into records, and reused for documentation purposes.

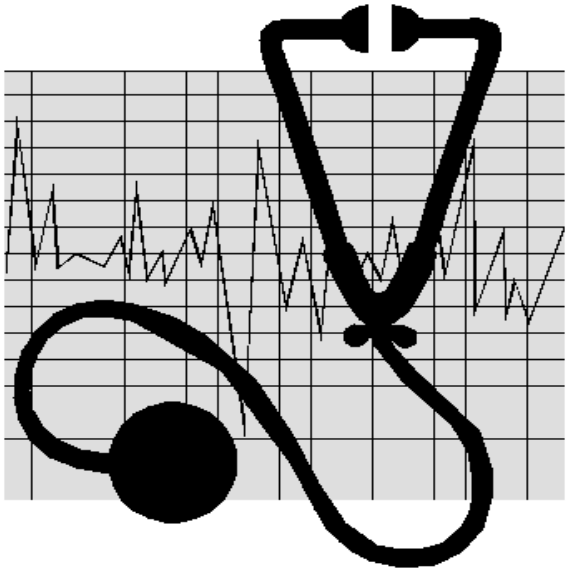


Fig. 1 Diagnostics

The diagnostics tools in Automation Studio provide irreplaceable support not just for commissioning or looking for errors, but during the software development phase as well. Once you've finished this training module, nothing will stand in your way. You will be able to use this documentation at any time as a reference tool while you're working.

A great amount of information is provided in the Automation Studio online help about how the individual tools can be operated. Information about how to use these tools is provided in the following sections.

1.1 Objectives

Participants will become familiar with the use of the various diagnostics tools in Automation Studio for error detection and correction.

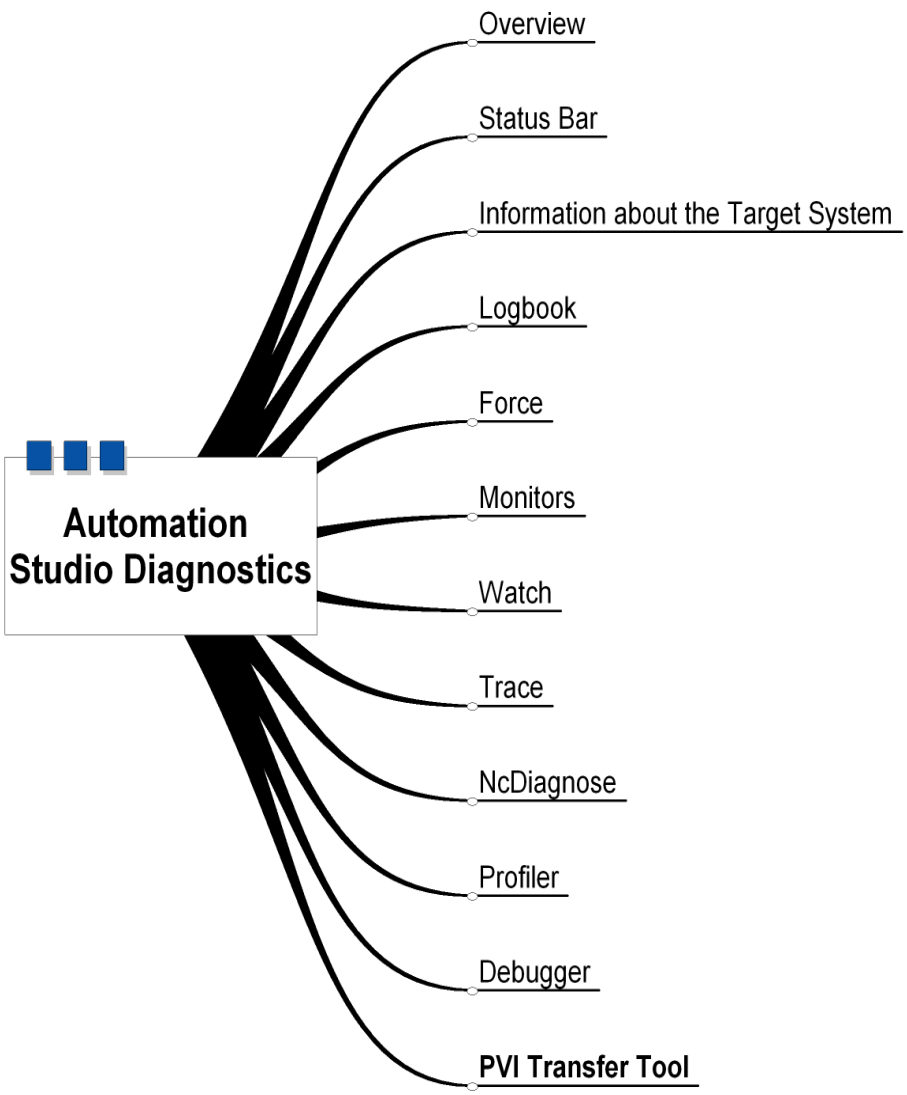


Fig. 2 Overview

2. OVERVIEW

Automation Studio provides numerous different tools for diagnostics. There are tools that just provide an overview of what's going on, tools that allow information to be read, saved, and modified, and tools that can be used to get deep into the system to optimize it.

The following diagnostic tools are available:

- Status bar
- Online info
- Logger
- Monitors (system, hardware, I/O, force, programming languages)
- Watch & archive
- Trace
- NcDiagnose
- Profiler
- Debugger

In engineering, it's always necessary to learn and master a large number of different tools. Choosing the right tool is decisive if you want to receive the information you need within the shortest possible time.

3. STATUS BAR

The status bar is located on the bottom edge of the Automation Studio screen. Important information about the controller status is displayed on the right-hand side:

- Connection
- CPU type, operating system version on the controller
- Status of the controller

Fig. 3 Offline – No connection to the controller

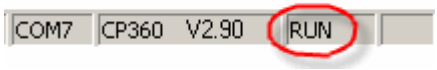


Fig. 4 Online – connection to the controller

Fig. 5 Online – Service mode

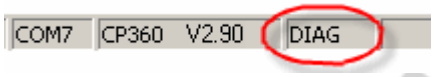


Fig. 6 Online - Diagnostic mode

Fig. 7 Online – Boot mode (default operating system was loaded)

4. INFORMATION ABOUT THE TARGET SYSTEM

The Online Info is an overview window that displays information about the target system. This includes information about available memory, battery status, node numbers on the CAN bus or Ethernet interface, and clock settings on the target system.
Online Info is opened by selecting **Online info...** from the CPU's **shortcut menu**.

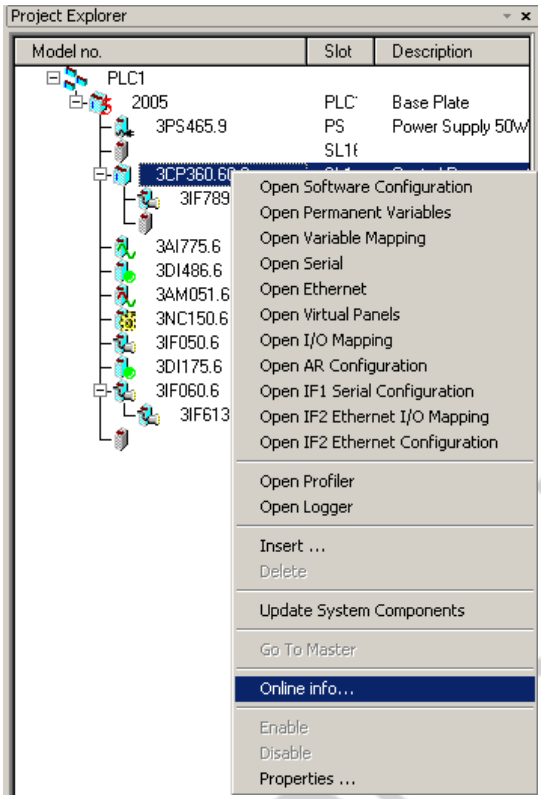


Fig. 8: Opening the Online Info

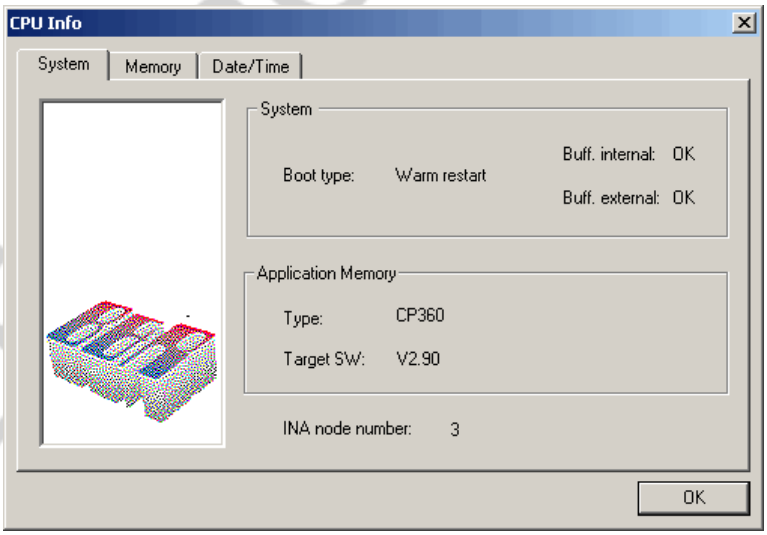


Fig. 9 Online Info

5. LOGGER

All fatal errors, warnings, and information which occur within the framework of the application are recorded by the operating system. This log is stored in a separate system logbook in the controller's memory. It can be displayed using the logger and saved as a file as long as there is an online connection.

The logger can be opened via the main menu **Open:Logger**.

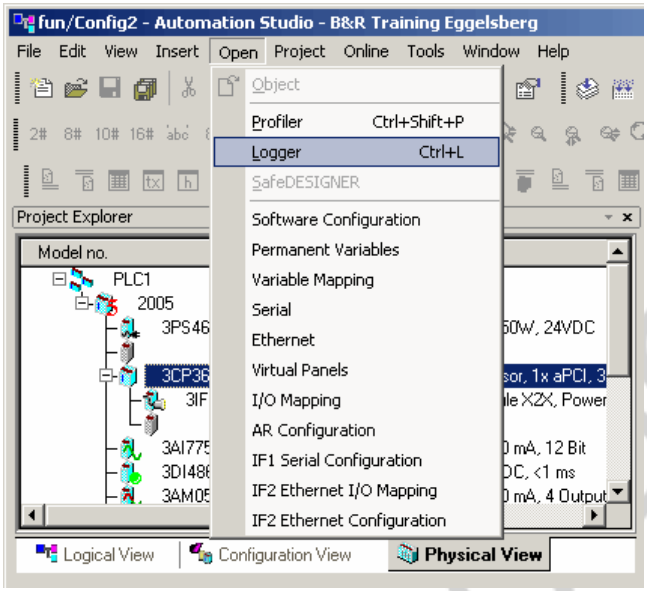


Fig. 10: Opening the Logger

The logger window is divided into five areas:

- Module list (1)
- Table in which the Logger entries are displayed (2)
- Details window (3)
- Backtrace (4)
- Toolbar (5)

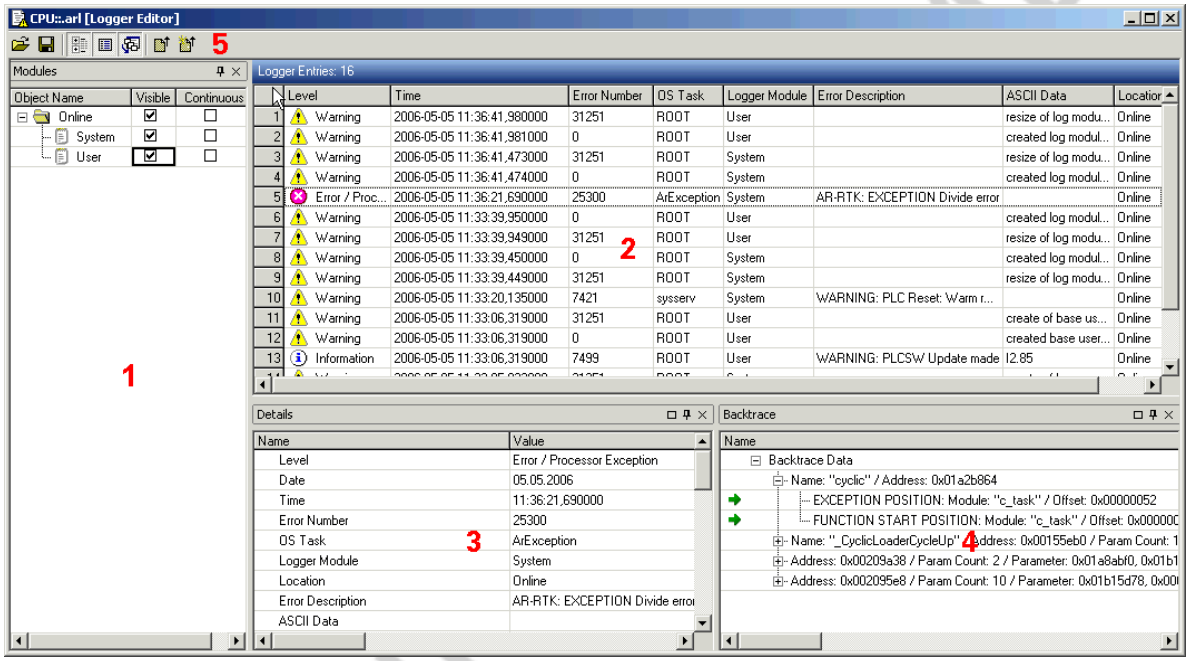


Fig. 11 Logger window

Module list

A list of all Logger modules that can be displayed is shown here. This includes modules on the controller (online) or modules from the PC (stored modules). Logger data is refreshed as frequently as possible if **Continuous** is selected.

Logger entries

A list of the Logger events that have occurred is shown in this window.

Detail window

Detailed information about the selected Logger event is shown in the detail window.

Backtrace

For exceptions that are triggered by the application, the source code that triggered the error can be determined under certain circumstances. If the faulty source code is able to be determined, the information is shown in the Backtrace window.

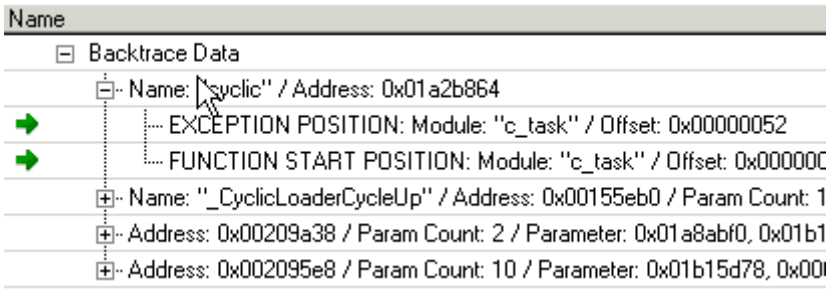


Fig. 12 Backtrace

Double-clicking on the first line, which is marked with a green arrow, brings you to the code line that caused the error.

Toolbar

The icons for opening, saving, refreshing data, re-reading data and activating window sections are shown here.

Note:

It is possible to create and edit one or more **User Logger module(s)** using the functions in the "AsArLog" library.

Task: PLC overview

The status bar, online info, and Logger are tools that can be used to provide a general overview:

- Check the online status (RUN, SERV, DIAG, BOOT) of your CPU.
- Check the date and time to see if they are current.
- Check the available memory.
- Read the number of the onboard interface.
- Analyze the last five Logger entries.

6. FORCE

The FORCE option is used to force a defined I/O value. For example, this could be used for testing purposes:

- How does my program react if this input had a different value?
- Does the relay respond if I set this digital output?

Caution:

Forcing an I/O data point causes entries that you have made to be applied directly in the software and outputs to be set. As a result, the application no longer has any influence over what happens. This may cause "illegal" conditions. For this reason, this tool can be just as dangerous as it is useful (e.g. motor runs although the end switch is active!)

6.1 The Watch window

The Watch window displays I/O variables with red (outputs) and green (inputs) LEDs. When attempting to change the value of such an I/O variable, the force function can be enabled by confirming the message that appears. When forcing is turned on, the value entered into the Watch window is written to the software or hardware regardless of the state of the I/O data point.

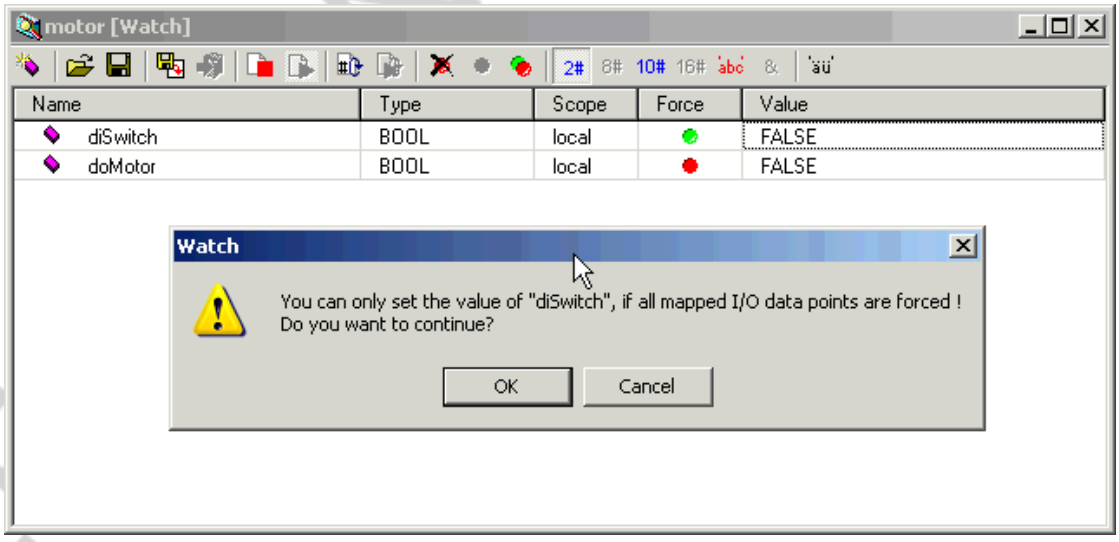


Fig. 13: Forcing in the Watch window

6.2 I/O monitor

When clicking on an I/O card in the hardware tree, its values are displayed and can be forced. Monitor mode must be enable before you can do this.

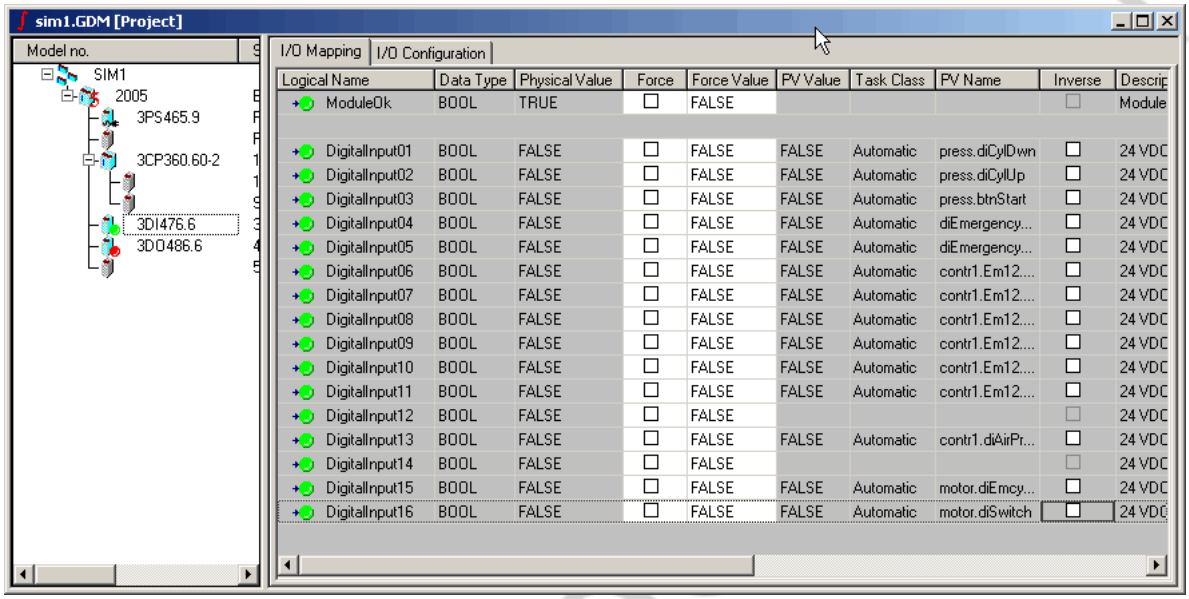


Fig. 14: I/O configuration in monitor mode

The force option can be selected for any of the I/O data points. The set force value is then applied.

Input and output states can also be forced in a task's monitor mode.

Note:

Forcing can be turned off for each individual input/output. The **Online:Force:Force Global Off** menu item can be selected or a restart carried out to turn off all force operations at once.

7. MONITORS

The monitors can be used to get information about the target system. The system monitor shows the version and status of the task, the I/O monitor provides information about the status of the data points and the monitor for programming languages offers extensive possibilities for checking the program flow. All monitors are started with the  icon.

7.1 System monitor

The system monitor provides an overview of the software version installed on the CPU.

You can determine which software version is running on the machine and if any changes to the current version of the project must be planned.

The system monitor is started with the  icon in the software tree / software configuration.

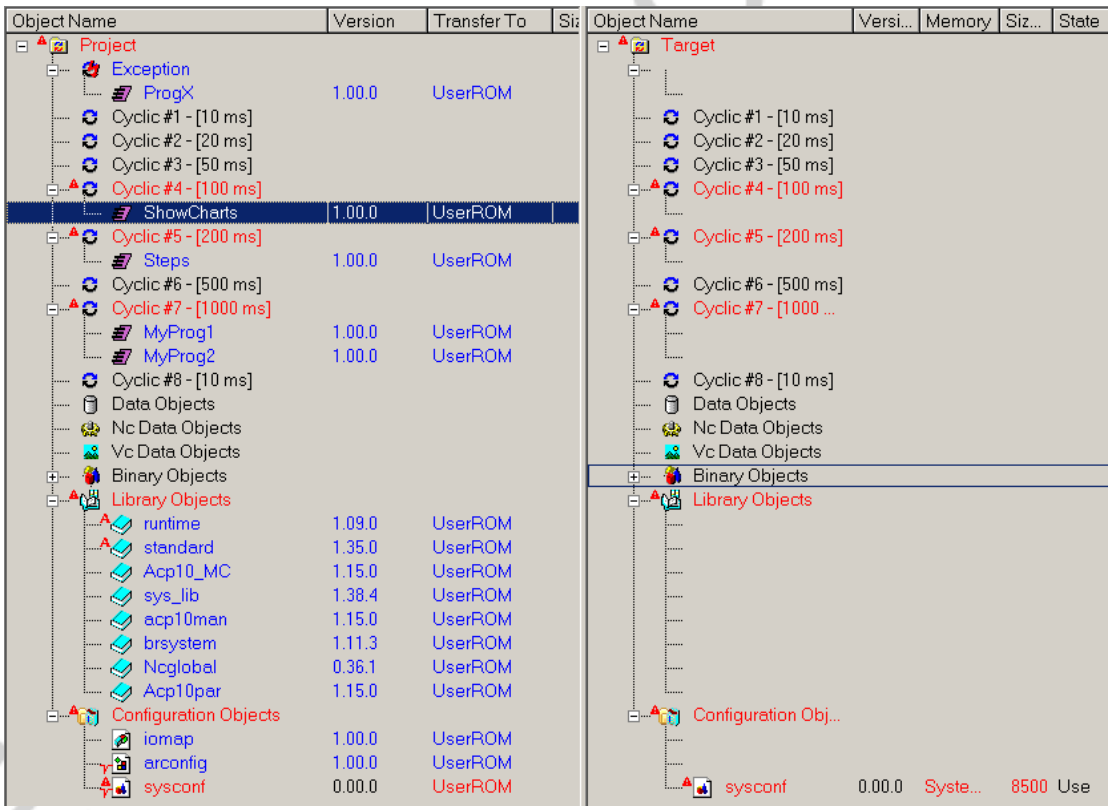


Fig. 15: System Monitor

Information about the difference between project and CPU, target memory and state is displayed next to each object.

Individual software objects can be stopped, started and deleted using the shortcut menu. The Watch window and the Tracer can also be started in the system monitor for each task (even if not present in the project). Stopping particular parts of the software can be used to disable an entire machine function (testing individual software objects).

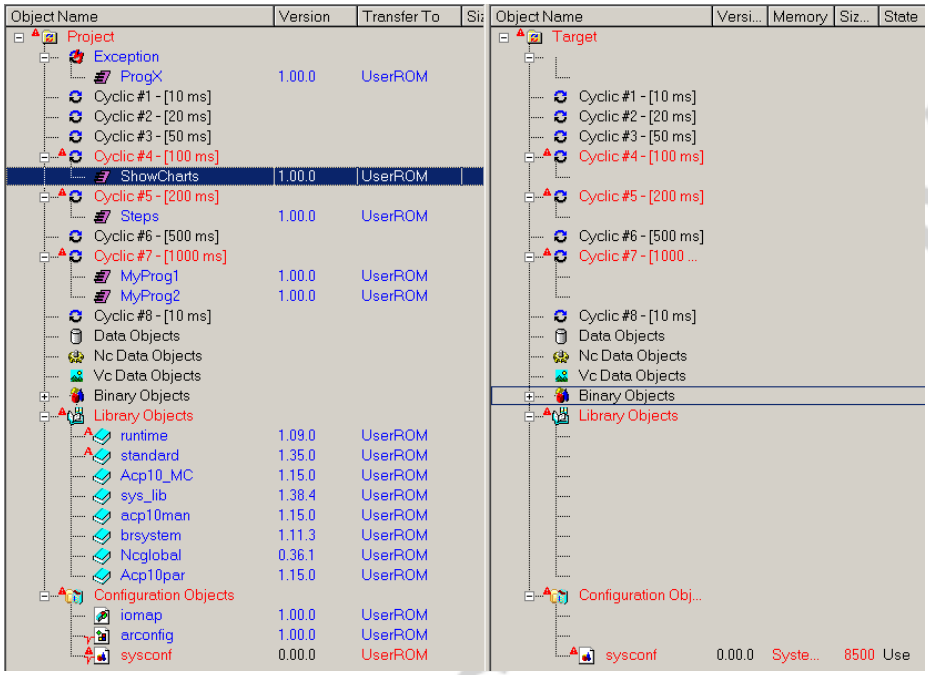


Fig. 16: Stopping a task

Caution:

Stopping and deleting software objects while a machine is running can be dangerous because this task's function is then no longer available.

7.2 Hardware configuration in monitor mode

The hardware configuration in monitor mode allows you to determine whether the hardware configuration defined for your project matches the actual configuration. Unlike with the project, graphical icons are used here.

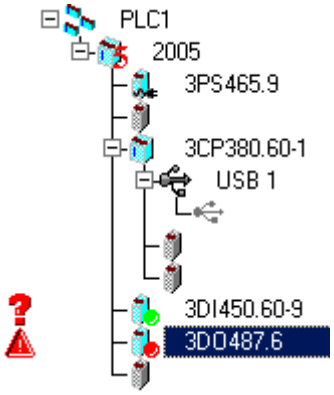


Fig. 17 The hardware configuration in monitor mode

Explanation of icons:

- Question mark: Module not in this slot or not detected.
- Exclamation mark: A different module than the one configured is in this slot.

7.3 I/O assignment in monitor mode

The I/O configuration in monitor mode allows you to check whether the I/O modules configured in the project actually physically exist. This is mapped as the module status. It's also possible to display the physical values of each I/O module channel. Outputs can also be set (see Force).

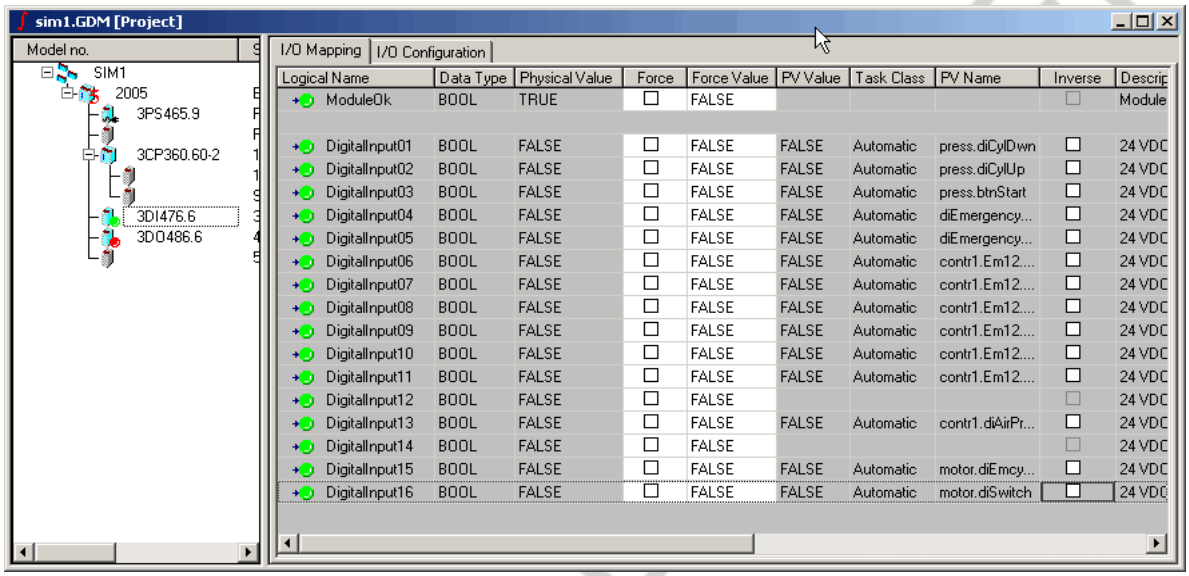


Fig. 18 I/O configuration in monitor mode

7.4 Programming languages in monitor mode

Monitor mode is available for all programming languages. However, it shouldn't be confused with the system monitor. In monitor mode, both the program itself and its variables (and their values) can be viewed. This gives you the advantage of allowing you to draw inferences about both the program flow and its results.

7.4.1 Graphical programming languages

In graphical programming languages, variable values are displayed right next to their symbols. There are also optical aids that show the program flow in a very simple manner.

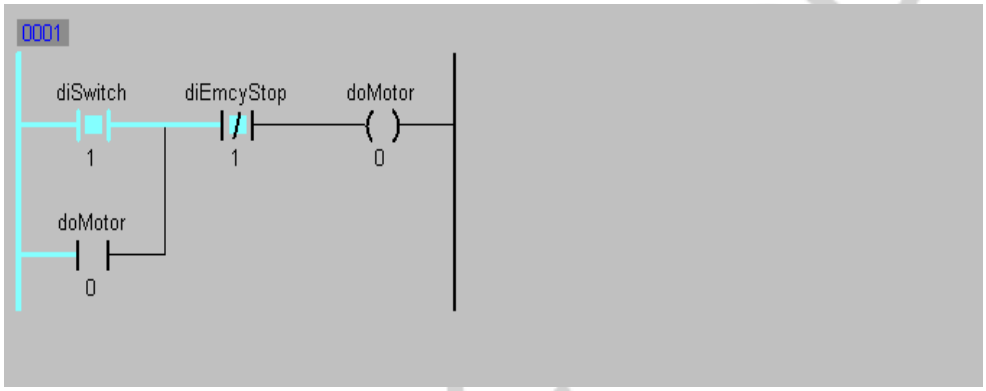


Fig. 19 Ladder diagram – Signal flow display

In the image above, the signal flow is shown by coloring both the lines and the symbols. In addition to the values in the program, this provides another way to carry out diagnostics.

The signal flow display can be turned on with the  icon.

7.4.2 Textual programming languages

In textual programming languages, variable values are shown on the right-hand side of the screen in the Watch window.
The Variables can be inserted into the Watch window by using Drag and Drop.

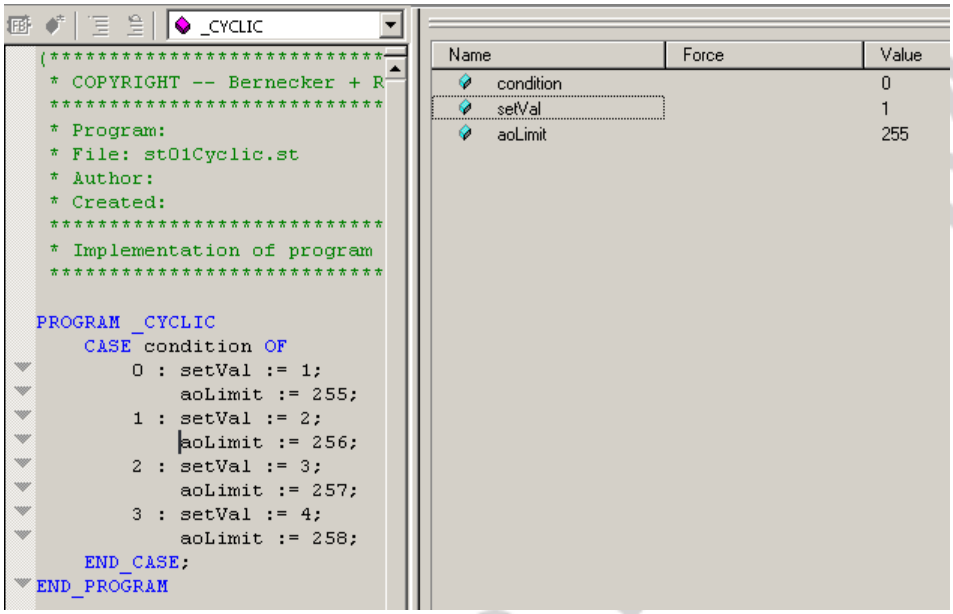


Fig. 20: High-level language monitor (ST)

7.4.3 Line coverage

Line coverage makes it possible to see which program lines and controller systems are being executed at a particular moment. Line coverage can be turned on with the  icon.

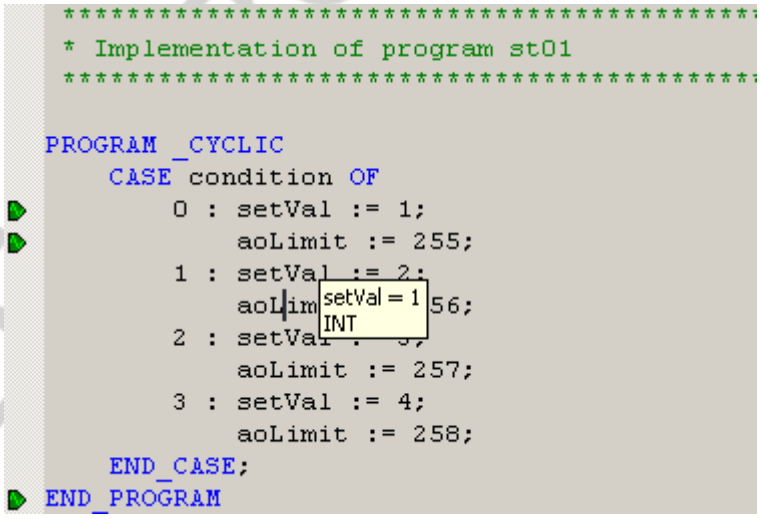


Fig. 21: Line coverage (ST)

Note:

When activating Line Coverage a message occurs which points out that the cycle time monitoring will be disabled. The appropriate message can be confirmed with OK or CANCEL.

Task: Monitors

Apply the following tools:

- System monitor
  Determine which tasks are installed on the CPU and whether they are identical to the tasks in your project. Stop and start a task. Determine a system module's version.
- I/O configuration in monitor mode
  Force an I/O data point using the I/O monitor.
- Programming languages in monitor mode
  Open a task in monitor mode and observe the program flow.

8. WATCH

The Watch window allows variable values that are on the target system to be displayed, monitored, and modified.

8.1 Watch window

All functions in the Watch window relate to the task that was selected in the software tree when it is opened. In other words, only variables from this task can be selected. If the Watch window is opened when the CPU is highlighted in the software tree, then all global variables can be displayed. This window is opened by clicking on the **Watch** item in the **shortcut menu**.

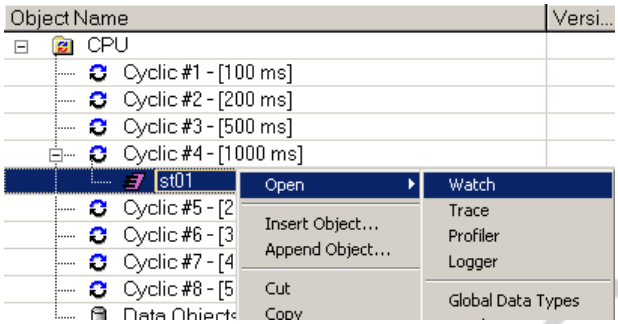


Fig. 22: Opening the Watch window

The primary purpose of the Watch window is to view and modify controller variables. In addition to the values of variable, other important information can be displayed about them (data type, scope, I/O data point, etc.). Variables can be managed in the form of a list, and different configurations (groups of variables) can be stored.

Variables can be chosen for display in the Watch window by selecting **Insert Variable** from the shortcut menu.

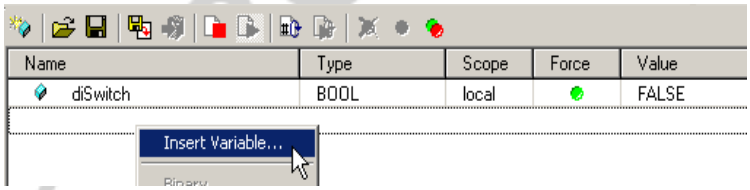


Fig. 23 Adding variables

The current configuration of the Watch window can be saved, opened, and restored at any time. You can also save several Watch configurations under different names.

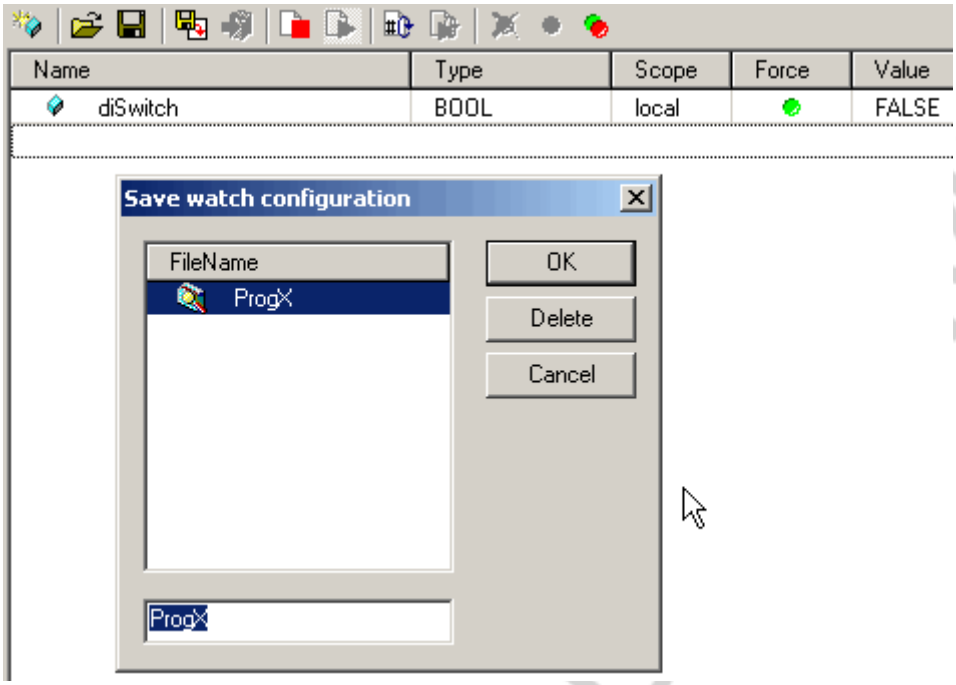


Fig. 24 Saving the configuration

The following actions can be carried out using the icons in the Watch window (functions listed from left to right):

Fig. 25 The Watch window's toolbar

- Insert variable
- Load configuration
- Save configuration
- Archive mode
- Write archive data to PLC
- Stop task
- Start task
- Configure cycle count
- Execute cycle count (only if the task is stopped)
- Turn off forcing
- Turn on forcing
- Turn off forcing globally

8.2 Archive

It is no problem changing ONE variable in the Watch window. The new value is transferred to the target system as soon as the entry has been confirmed.

However, archive mode can also be used to transfer more than one variable at the same time. The variable values are no longer refreshed when archive mode is activated. This allows you to easily change the values of multiple variables and to transfer them to the target system with a single command (write values).

This also allows you to put together small recipes (collections of values that are related to one another) that can be loaded and saved at any time. When a Watch configuration is saved, the values are saved as well. This data is called archive data.

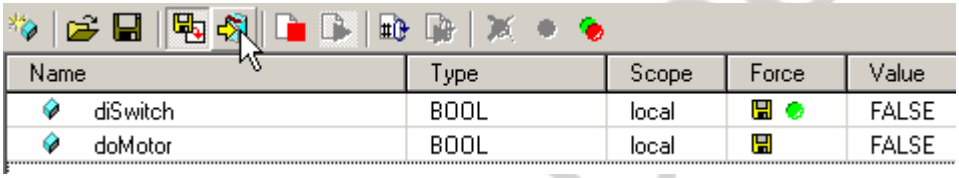


Fig. 26 Archive

The saved configuration can now be used to perform any additional processing that may be necessary.

Archive data is stored in the respective Watch configuration. If you open one of these configurations, you will be advised to enter archive mode.

Task: Archive

Open the Watch window for a task and insert variables. Change the value of the variables. Activate archive mode and save your configuration. Change the value of the variables again. Use archive mode to restore the original value.

Note:

You can use the Watch window and archive mode to save important variables to a file so that they can be restored later if necessary.

9. TRACE

Until now, the different monitors, line coverage, and the Watch window were sufficient for viewing both static and ever-changing variables. However, there are some values that change rapidly and cannot be followed in the Watch window.

The Tracer is able to record how values change over time and display them in a diagram. The diagrams which result can be stored to be viewed or processed later.

The Trace function records the values directly on the controller. An upload function takes the values from the controller and displays them as a diagram. This function makes it possible to record values from different task classes in a precisely defined timeframe.

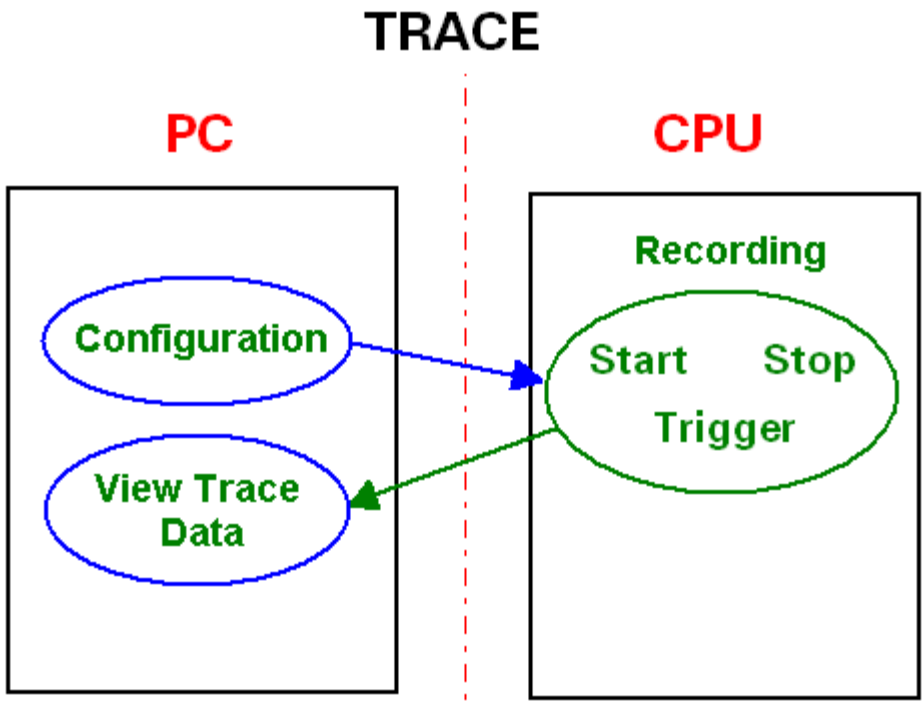


Fig. 27: Trace

The smallest unit of time is the task class cycle time in the task for which the trace is opened. The values can be sampled at the start or end of a task. A maximum of 8 values can be recorded at one time. One trace can be opened for each task.

It is opened by selecting **Trace** from the **shortcut menu**.

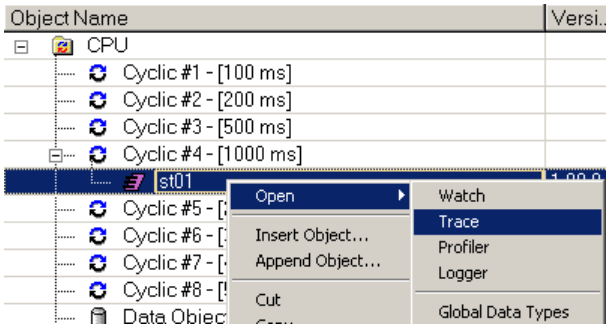


Fig. 28: Opening the Tracer

A new trace configuration added using the icon  . Variables can be added to the trace configuration by clicking the icon  .

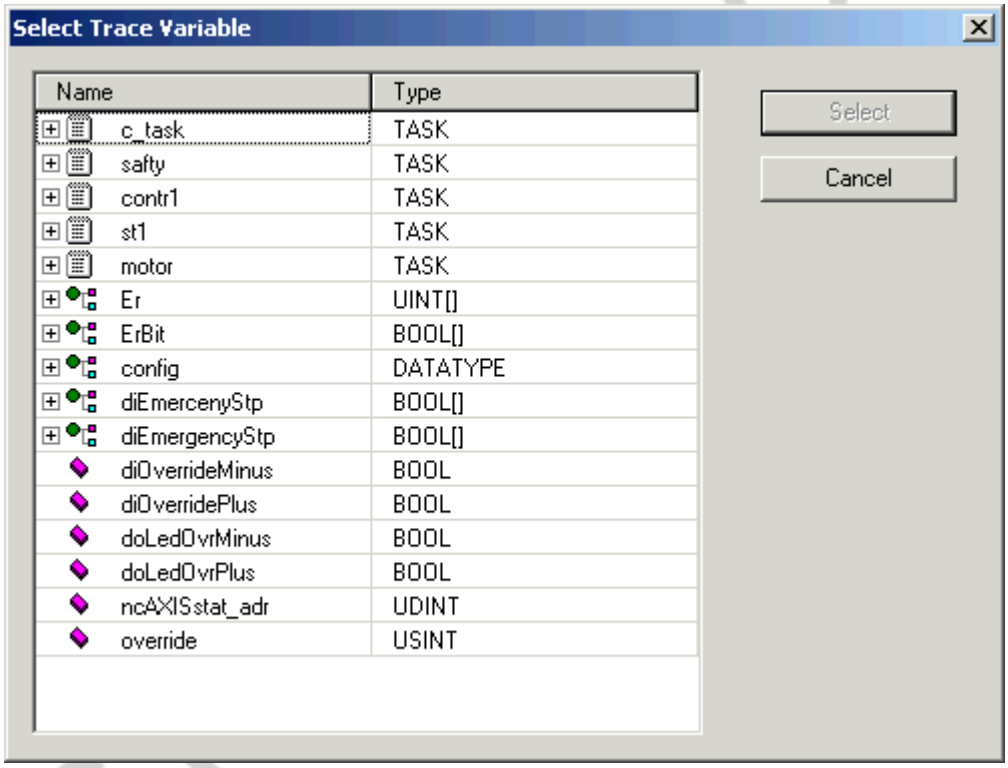


Fig. 29: Adding variables to the trace configuration

The trace can be configured using the shortcut menu from **TARGET_CONFIGURATION:Properties**. The maximum trace duration depends on the buffer size and the time base (task class cycle time and scaling). Furthermore, you can also define whether the trace should record continuously or should be started by a trigger condition.

Once the variables have been selected and the Trace has been configured, it must be installed on the controller using the icon.

Note:

The monitor mode must be active in order for the trace to be installed.

The Tracer can be started with the icon and stopped with the icon.

When the Tracer is stopped, the icon can be used to load the data from the controller.

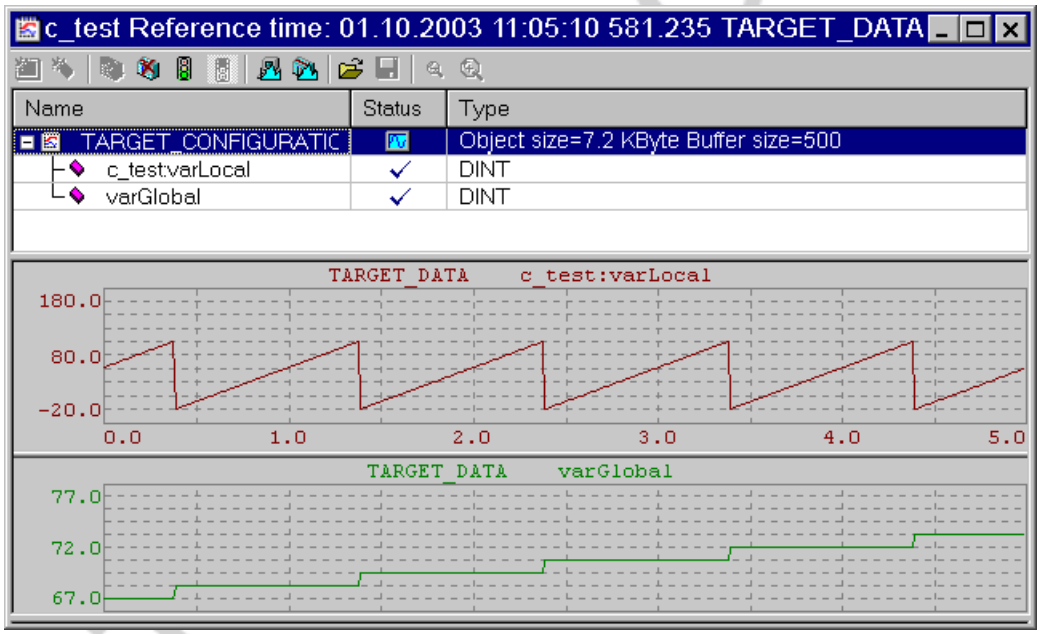


Fig. 30 Trace record

There is also the possibility to install a trigger-steered trace. Here the trace begins with a trigger and stopps when the trace buffer is full.
Negative and positve values can be indecated for the trigger delay.

A mesurement cursor and a reference cursor help with the analysis of the trace data.

In order to process the trace data externally, these can be also exported

Task: Trace

Create a task that makes a saw tooth pattern.

The code could look like this:

```
(* cyclic program *)
gVarUSINT:= gVarUSINT + 1;
VarUINT:= VarUINT + 1;
```

Fig. 33: Code for saw tooth

Name	Type	Scope	Attribute	Value	Owner
VarUINT	UINT	local	memory		
gVarUSINT	USINT	global	memory		

Fig. 33: Declaration for saw tooth

1. Start the Trace and record the values.
2. Start the Trace and record the values when the value of “gVarUSINT” is greater than 200.
3. Use the trigger delay to get at least 5 trace samples before the event.

10. NCDIAGNOSE

NcDiagnose in Automation Studio is an important topic when using drives with ACOPOS in your project. This allows you to perform detailed recordings of drive parameters and to read status information for analyses. Specific sequences, e.g. positioning, can be selectively readjusted.

10.1 NcWatch

NcWatch can be used to display status information, actual values, target values, etc. in a list. The desired parameters can be added and deleted as with the "normal" Watch function.

A drive's error messages can be displayed by selecting a function in the toolbar.

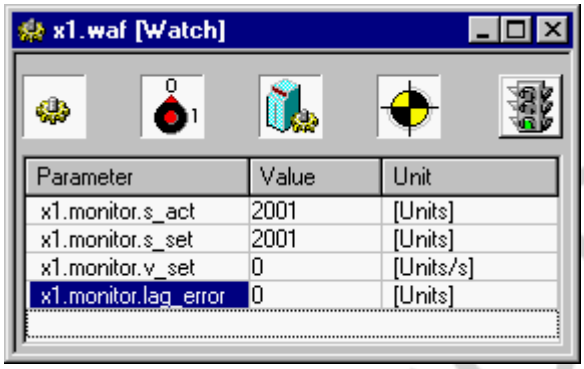


Fig. 34 NcWatch

10.2 NcTrace

The NcTrace allows the user to record specific drive parameters.

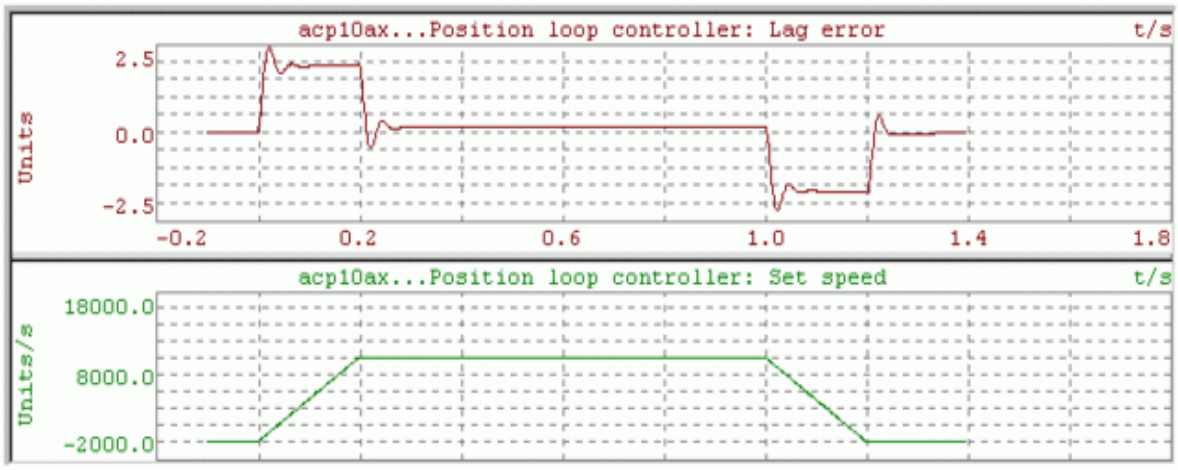


Fig. 35 NcTrace

Note:

NcTrace trace can be opened in the test center or directly on the axis object. Opening directly on the axis object does not interrupt the running application.

The description of the trace configuration can be found in the Automation Studio online help under **Automation Software:NC Software:ACP10:Trace:NC Trace**.

10.3 NcNetwork command and DPRtrace

Communication between the Acopos and the application can be logged in the network command trace. This trace can be an important aid when correcting errors for positioning tasks.

The network command trace can be opened by selecting **Show Network Command Trace** from the shortcut menu in the trace.

10.4 NcTest

All of the functionalities for analyzing and testing drives are combined in the NcTest. NcTrace (cyclic or network trace), NcWatch, the parameter interface, and the command interface are all within the scope of NcTest.

All of the drive's functions can be executed using the NcTest (moving, homing, etc). No additional controller software is necessary on the target system to do this.

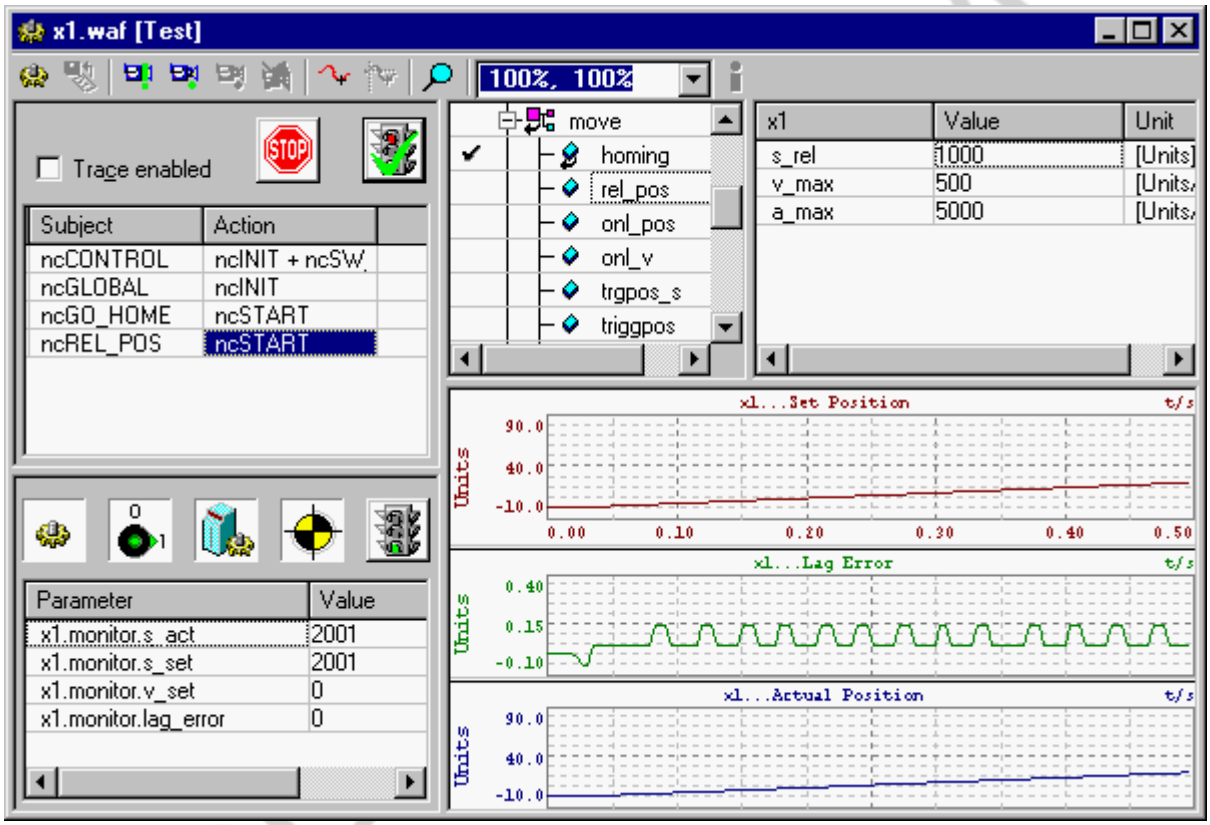


Fig. 36 NcTest

Caution:

The application has no control over the axis as long as the ncTest window is open. All functions can be controlled via the test window.

11. PROFILER

Important system files can be measured and displayed with the Profiler:

- Task runtimes
- Stack usage
- System usage

This allows the runtime system to be analyzed with regard to system usage (load). This information can then be used to optimize how the system uses its resources.

The Profiler is started by selecting **Open:Profiler**.

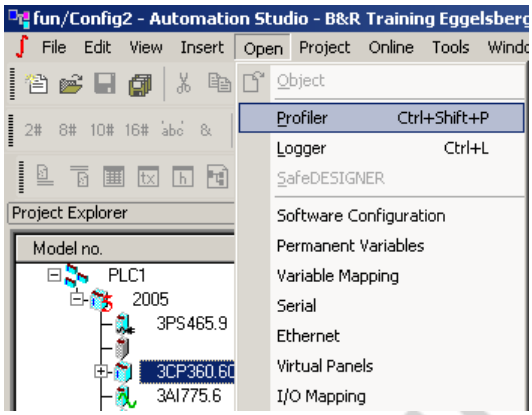


Fig. 37: Opening the Profiler

The Profiler configuration is installed by clicking on the icon. Recording can be started with the icon and stopped with the icon.

When the Profiler is stopped, the icon can be used to load the data from the controller.

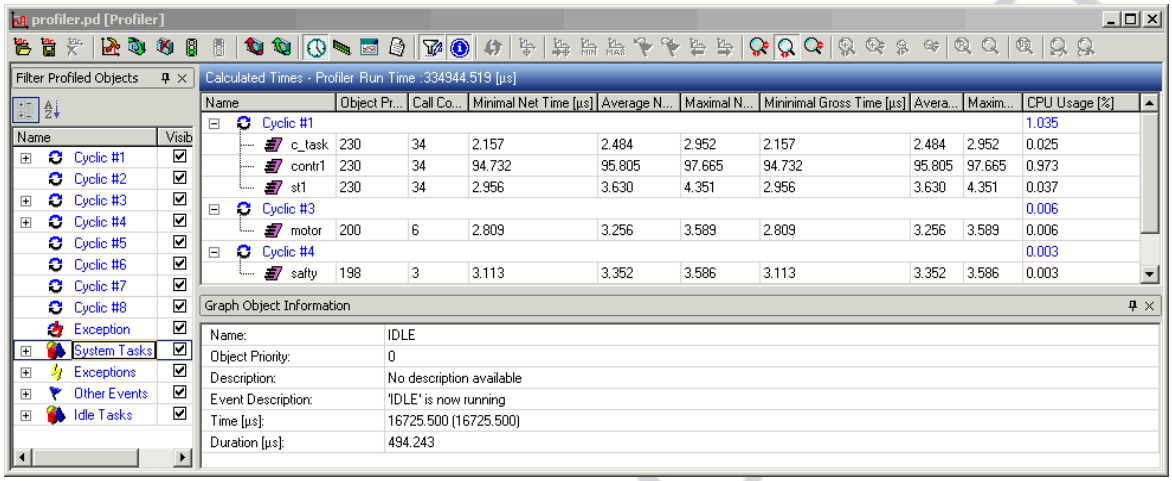


Fig. 38 Profiler logging

Measurement results can be displayed graphically or in table form as minimum, maximum, and average.
Configurations and measurements can be saved and e.g. sent by email if needed.

Note:

An active Profiler configuration on the target system can also be enabled manually from the software using the functions in the "AsArProf" library.

Note:

Interpreting Profiler measurements should be done by experts who know the software in detail. These measurement results are especially important for the software developers.

Note:

The Profiler is always active by default. It can be disabled by using the shortcut menu: **CPU - Properties**.

Task: Profiler

Create a task with a loop whose final value is a variable. Make profiler measurements using different end values. Try this in different task classes.

The code could look like this:

```
(* cyclic program *)
FOR i:=0 TO endValue DO
    value:= value + 1;
END_FOR
```

Fig. 40: Code

Name	Type	Scope	Attribute	Value
endValue	UDINT	local	memory	
i	UDINT	local	memory	
value	UDINT	local	memory	

Fig. 40: Declaration

12. DEBUGGER

The debugger can be used to track down well-hidden software errors in textual programming languages.
By setting **breakpoints** in the source code, you can stop the program execution at the exact code line where the breakpoint is set.

This makes it possible to determine the direct connection of variable values with the program line currently being executed. Using the **Step into** and **Step over** functions allows you to go through the program while getting a detailed look into what it is actually doing.

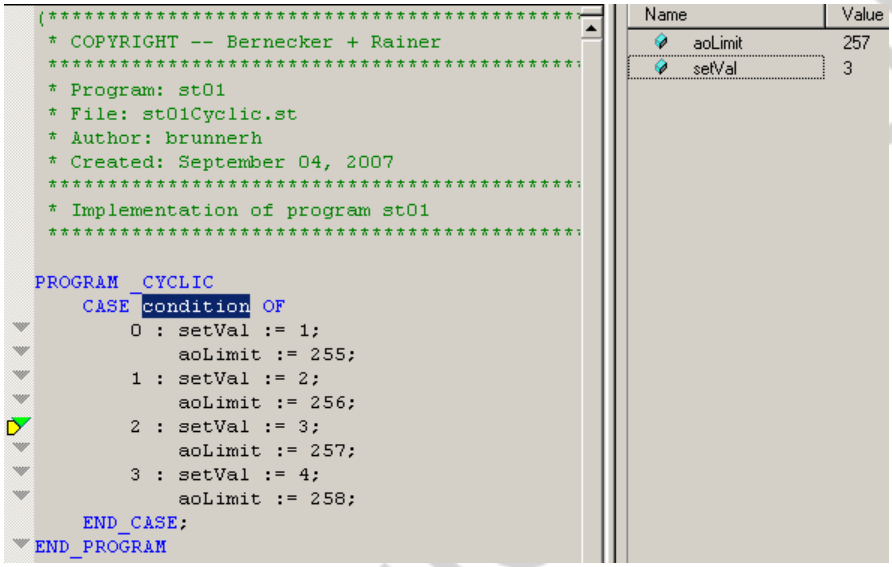


Fig. 41: Breakpoint in the programming language monitor

Step into mode goes through each program line one-by-one. **Step over** treats related constructs (loops, step sequencers) as individual program lines.

Caution:

If the program runs into a breakpoint, the entire application is stopped. It is not recommended to try this on machines that are actually producing.
Breakpoints are deleted automatically during a restart.

Task: Debugger

Create a program for testing parts:

- Start the test (doStartCheck) when a part is present (diPart).
- Evaluate the result (diPartOK) once the test has finished (diCheckReady).
- If you are dealing with a bad part, activate the bad part cylinder (doBadPart) and wait until it has reached its end position (diBadPartEnd).
- If it is a good part, then activate the good part cylinder (doPartOK) and wait until it has reached its end position (diPartOKEnd).

Resolve the task using a CASE statement.

The code could look like this:

```
(* cyclic program *)
CASE testStep OF
    (*wait for a part*)
    0:      IF (diPart = 1) THEN
                doStartCheck:= 1;               (*start check*)
                testStep:= testStep + 1;
            END_IF
    (*wait for check is ready*)
    1:      IF (diCheckReady = 1) THEN
                doStartCheck:= 0;
                IF (diPartOK = 0) THEN
                    testStep:= 2;               (*bad Part*)
                ELSE
                    testStep:= 3;               (*Part OK*)
                END_IF
            END_IF
    (*it is a bad Part*)
    2:      doBadPart:= 1;
            IF (diBadPartEnd = 1) THEN
                doBadPart:= 0;
                testStep:= 0;
            END_IF
    (*it is a good Part*)
    3:      doPartOK:= 1;
            IF (diPartOKEnd = 1) THEN
                doPartOK:= 0;
                testStep:= 0;
            END_IF
END_CASE
```

Set the breakpoint and process the code in step into mode.

13. PVI TRANSFER TOOL

The PVI Transfer tool is an extensive **diagnostics and service tool** that combines several different diagnostics functions with one another and which is not limited to just **transferring projects**.

The PVI Transfer tool can be started in Automation Studio by selecting the **PVI Transfer tool** entry in the **Tools** menu. If you don't have Automation Studio installed, it can be started directly from the **Windows Start** menu.

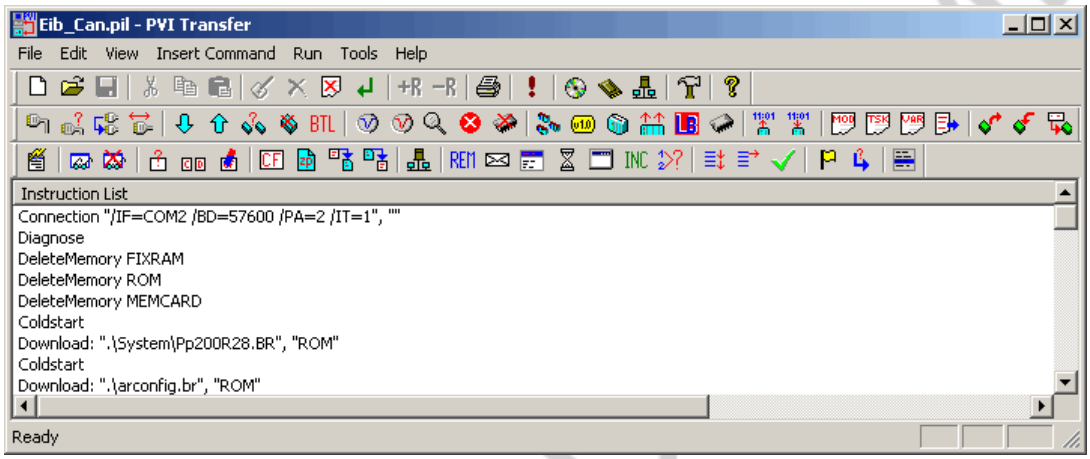


Fig. 42 The PVI Transfer tool menu

The PVI Transfer tool provides the following functions:

- Downloading, uploading, and deleting modules
- Checking modules
- Transferring the operating system and reading the version
- Cold restart, warm restart, diagnostics and service mode
- Reading the operating status
- Clear memory...
- Reading the CPU type
- Reading hardware and memory information
- Reads logbook
- Reading and setting the time
- Loading module, task, and variable lists
- Read/Write variables
- Reading CompactFlash info
- Partitioning and formatting CompactFlash
- Creating and restoring CompactFlash images

14. SUMMARY

There are several different tools available to localize errors. And it makes good sense to use them. You will be required to use your analytical thinking and common sense to choose the right way to go.

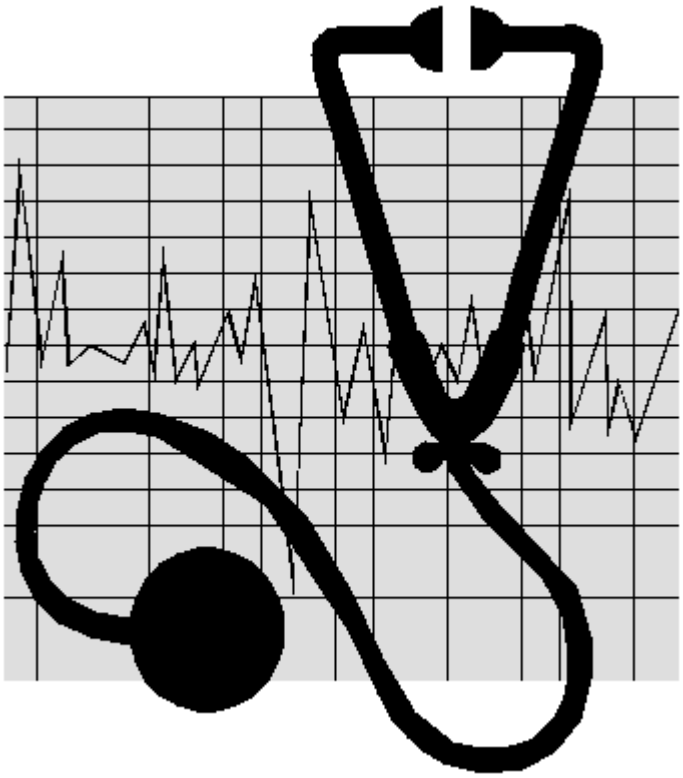


Fig. 43 Diagnostics

To be able to use these diagnostic tools effectively, it's necessary to get an overview of the situation, establish the general conditions, and view things from a certain distance.
The circumstances can then be clarified and analyzed in detail. A comprehensive overview of potential errors can be achieved by excluding and reducing the number of possible error sources. This will make it considerably easier to correct any errors that occur.

Notes

Overview of training modules

TM200 – B&R Company Presentation **
TM201 – B&R Product Spectrum **
TM210 – The Basics of Automation Studio
TM211 – Automation Studio Online Communication
TM212 – Automation Target **
TM213 – Automation Runtime
TM220 – The Service Technician on the Job
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Generation
TM240 – Ladder Diagram (LAD)
TM241 – Function Block Diagram (FBD)
TM246 – Structured Text (ST)
TM247 – Automation Basic (AB)
TM248 – ANSI C
TM250 – Memory Management and Data Storage
TM260 – Automation Studio Libraries I
TM261 – Closed Loop Control with LOOPCONR

TM400 – The Basics of Motion Control
TM410 – The Basics of ASiM
TM440 – ASiM Basic Functions
TM441 – ASiM Multi-Axis Functions
TM445 – ACOPOS ACP10 Software
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Starting up Motors

TM500 – The Basics of Integrated Safety Technology
TM510 – ASiST SafeDESIGNER

TM600 – The Basics of Visualization
TM610 – The Basics of ASiV
TM630 – Visualization Programming Guide
TM640 – ASiV Alarm System
TM650 – ASiV Internationalization
TM660 – ASiV Remote
TM670 – ASiV Advanced

TM700 – Automation Net PVI
TM710 – PVI Communication
TM711 – PVI DLL Programming
TM712 – PVIServices
TM730 – PVI OPC

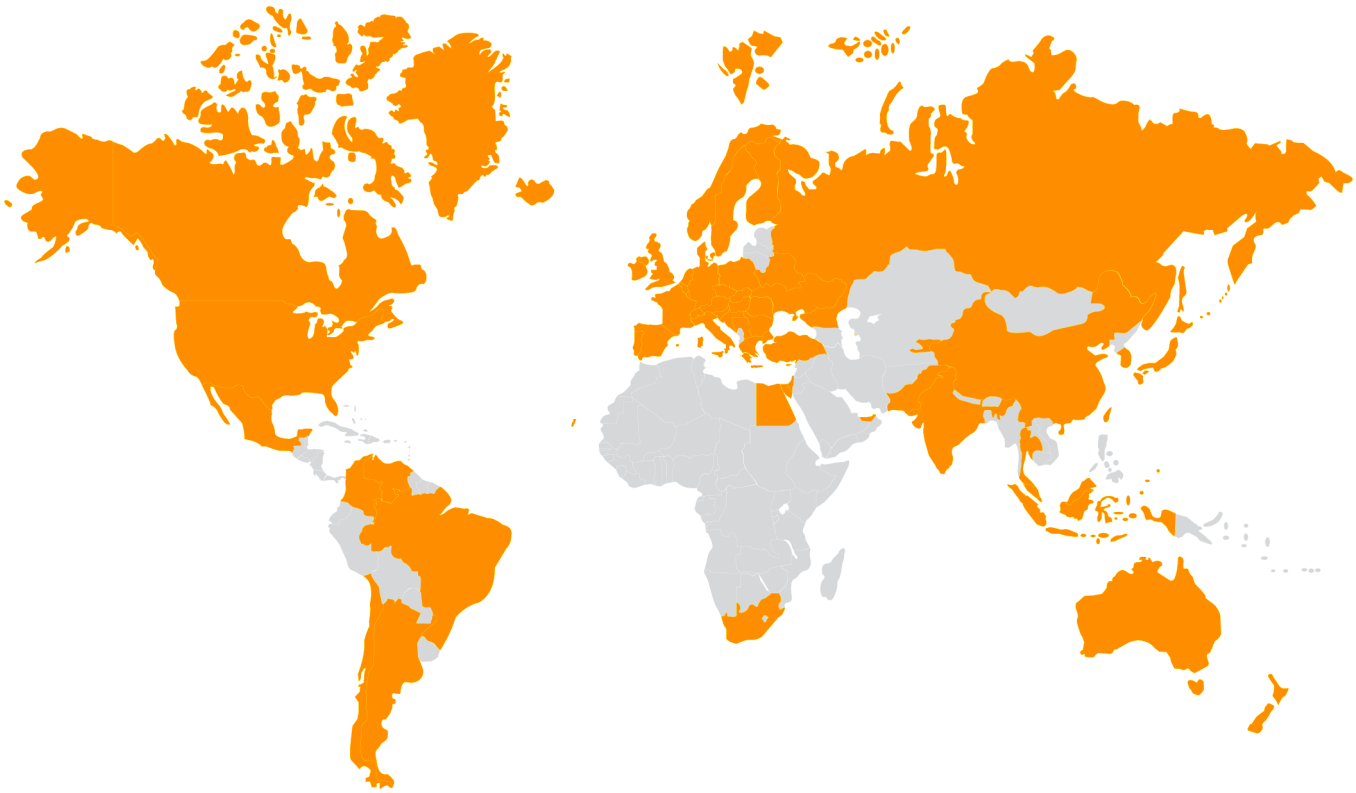
TM800 – APROL System Concept
TM810 – APROL Setup, Configuration and Recovery
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM840 – APROL Parameter Management and Recipes
TM850 – APROL Controller Configuration and INA
TM860 – APROL Library Engineering
TM865 – APROL Library Guide Book
TM870 – APROL Python Programming
TM890 – The Basics of LINUX

**) see Product Catalog

CORPORATE HEADQUARTERS

Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.
B&R Strasse 1
5142 Eggelsberg
Austria
Tel.: +43 (0) 77 48 / 65 86 - 0
Fax: +43 (0) 77 48 / 65 86 - 26
info@br-automation.com
www.br-automation.com

140 offices in more than 55 countries - **www.br-automation.com/contact**



Australia • Argentina • Austria • Belarus • Belgium • Brazil • Bulgaria • Canada • Chile • China • Colombia • Croatia • Cyprus
Czech Republic • Denmark • Egypt • Emirates • Finland • France • Germany • Greece • Hungary • India • Indonesia
Ireland • Israel • Italy • Japan • Korea • Luxemburg • Kyrgyzstan • Malaysia • Mexico • The Netherlands • New Zealand
Norway • Pakistan • Poland • Portugal • Romania • Russia • Serbia • Singapore • Slovakia • Slovenia • South Africa
Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • USA • Venezuela • Vietnam