TM223

Automation Studio diagnostics

PERFECTION IN AUTOMATION
A MEMBER OF THE ABB GROUP

Requirements		
Training modules	TM210 – Working with Automation Studio TM213 - Automation Runtime	
Software	Automation Studio 4.4.4 Automation Runtime 4.44	
Hardware	X20 controller and X20 I/O modules ETA210 or ETAL210 + ETAL690 **www.br-automation.com/eta-system**	

Table of contents

1 Introduction

Automation Studio offers a wide range of tools to support you during development and with any diagnostics tasks that occur later on. The integrated diagnostic tools help to create a system overview including the detailed recording of system behavior. Automation Studio is therefore the ideal tool for programming, commissioning and diagnostics.

The diagnosis begins with selecting the right tool. This training module shows how the diagnostic tools can be used individually or in combination. Numerous tasks contribute to better understanding and provide examples for practical use.

Automation Runtime provides access to diagnostic information. For this reason, a lot of diagnostic data is available directly in the System Diagnostics Manager. This can be called using a web browser. Data is also available in the control application using library functions.

Figure 1: Automation Studio diagnostics

The diagnostics tools for integrated motion control are described in the training module "TM410 – Working with Integrated Motion Control". The diagnostics tools for integrated safety are dealt with in the training module "TM510 – Working with SafeDESIGNER".

1.1 Learning objectives

This training module uses selected examples illustrating different diagnostic possibilities during programming, commissioning and servicing to help participants learn how to work with the various diagnostic tools.

- Participants will learn the criteria for selecting the correct diagnostic tool.
- Participants will learn how to analyze and log general system information.
- Participants will learn how to monitor and record process values.
- Participants will get to know the options for system and application diagnostics.
- Participants will learn which Automation Runtime configuration options are relevant to which diagnostic tools.

1.2 Symbols and safety notices

Unless otherwise specified, the descriptions of symbols and safety notices listed in "TM210 – Working with Automation Studio" apply.

2 The correct diagnostic tool

Selecting the correct diagnostic tool makes it possible to quickly and effectively localize a problem.

Analyzing irrelevant data or sending it to someone else for examination can lead to substantial delays in finding a solution to the problem.

The Logger can be used to recognize a cycle time violation. However, the cause of the cycle time violation would not be best diagnosed by using the Logger.

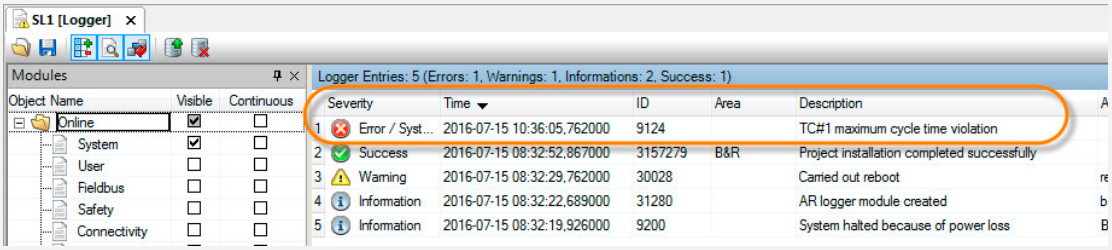


Figure 2: Cycle time violation in the Logger window

The cause of the error in the Logger will be given as a cycle time violation in Task Class #1. The backtrace data refers to a task where the cycle time violation occurred.

Situation 1

In the multitasking system, a task is interrupted by a higher priority task. This can be the cause of the cycle time violation. If the higher priority task has a longer execution time, the task entered in the logger is no longer able to finish its configured cycle time + tolerance.

Situation 2

Several tasks are executed one after another cyclically in the same task class. If one of the previous tasks takes longer to complete, then the task shown in the Logger will likewise not be the cause of the cycle time violation.

In either case, the logger alone does not help in determining the cause of the error. This problem can only be investigated in detail using the Profiler ("Perform runtime measurements in the Profiler" on page 32). The profiler graphically displays the time sequence of the execution time of individual tasks.

2.1 Debugging procedure

There are a number of different ways to analyze a problem. Combining different localization and analysis strategies can considerably increase effectiveness when trying to locate errors.

The methodology of locating errors

The methodology used when searching for errors is extremely important as it allows the available tools to be applied. The environment of the machine up to the control is precisely dealt with through targeted questions. The following methods are used:

- Analysis of environment and framework conditions
- Description of the error type
- Eliminating other possible errors
- Measuring and recording data

Only with a complete overview can specific areas can be isolated and considered in greater detail.

Environment and general conditions

An analysis begins in the environment of the machine as well as during the control of the framework conditions under which it is operated.

By looking at the basic conditions for machine operation, an error can often already be defined. Framework conditions include:

Shift and batch change, time changeover, hall climate, exchange of sensors, operator actions, equipment etc.

If error sources in the environment of the machine, with the sensors and actuators have been excluded, you can deal with the control system and the application software in detail.

One-time or recurring errors

If errors are reproducible in certain actions, they can be investigated in a targeted manner.

Errors that do not occur with certain actions or with particular regularity are extremely difficult to find. Here you have to try to reproduce the errors again in order to investigate them.

The analysis of sporadic errors is simplified by preparing the automation software.

Program or operation errors

For example, runtime errors occur if certain general conditions are not taken into account during the process sequence. Here a few examples of this:

- Division by zero
- Missing evaluation of return values from functions
- Overflow when accessing array elements (e.g. loop counters)
- Access to non-initialized pointer

The results of the methods used are documented. Checklists help to record the data and provide support so that important information is not lost.

2.2 Checklists

Checklists do not just help when trying to analyze a problem during servicing; they are also very useful during configuration.

The information collected here can help those called in later to troubleshoot errors to solve problems more quickly by providing the actual data.

Figure 3: Checklist

Information on relaying information

If help is required in analyzing a problem, it is necessary to provide a detailed description of what it is.

- Detailed checklists filled in
- What actions have already been taken?
- What environmental conditions can be ruled out?
- Can the problem be reproduced in an office environment?
- Version of hardware and software
- Logger files
- Profiler recordings
- System Diagnostics Manager - System Dump

The more detailed the actions taken have been documented and information collected, the better the chances that the problem can be clearly identified and remedied. Further information on the error search in the control environment can be found in the training module "TM 920 - Diagnostics and Service".

Checklist for relaying information

Checklist: Software versions (also include any installed upgrades)

Software[1]	Version	Description, remark
• Automation Studio	•	•
• Automation Runtime	•	•
• mapp Services	•	•
•	•	•

Table 1: Checklist: Software versions

[1] **Automation Studio, Automation Runtime, mapp Technology, Safety Release, Visual components, etc.**

Hardware used (also include installed hardware and firmware version)

Model number	Serial number Revision Hardware/Firmware	Description, remark
•	•	•
•	•	•

Table 2: Hardware used

Checklist for describing the type of problem

- How are the surroundings and environmental conditions?
- What topology is used?
- Can the problem be reproduced, or did it occur only once?
- What actions need to be taken to reproduce the problem?
- When did the problem first occur?
- Have there been any changes in the software and/or hardware configuration or machine environment since then?
- In what status of the controller, and what is the LED status of the installed components?
- What information was loaded onto the controller?
  e.g. Logger files, Profiler data, system dump, etc.

2.3 Overview of diagnostic tools

Automation Studio provides appropriate tools that can handle diagnostics during programming, commissioning and servicing.

Only by selecting the right diagnostic tool is it possible to accurately and quickly access the necessary information.

Automation Help is a constant companion when working with Automation Studio and provides detailed information about the various diagnostic tools.

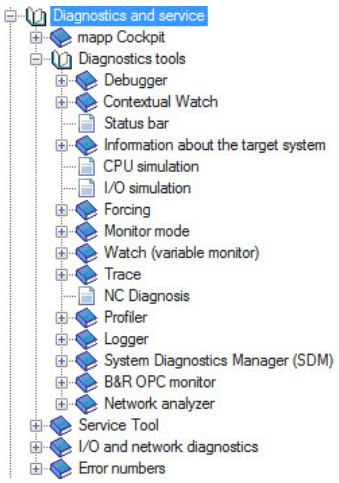


Figure 4: Category "Diagnostics and service" in Automation Help

Exercise: Open Automation Help section for the diagnostic tools

The diagnostic tools are covered in section "Diagnostics and service". Get an overview of the structure of Automation Help in this area.

Diagnostics and service

Requirements for completing exercises in this training module

The descriptions and images in this chapter refer to the project designed in training module TM210 (Working with Automation Studio) or TM213 (Automation Runtime). The following exercises can be done with any Automation Studio project.

Figure 5: X20 controller with Compact Flash

Recording data with the Profiler and the Contextual Watch must be carried out on real hardware because no diagnostically conclusive measurement results arise in the Automation Runtime Simulation (ARsim).

3 Reading system information

System information can be read from the target system in Automation Studio as well as with the aid of a web browser.

Reading system information	
"Status bar" on page 10	Information about the status of the connection, Automation Runtime version and the operating state of the controller
"Information about the target system" on page 11	Display of memory information, battery status and date/time configuration options for the controller
"Comparative functions " on page 11	Comparison possibility between the software versions in the project and on the control system; the modules configured in the project and actually recognized on the controller can be compared. A comparison of configuration files is also available.
"Error analysis in Logger" on page 15	Displays events that occur on the target system at runtime.
"System Diagnostics Manager (SDM)" on page 55	System Diagnostics Manager (SDM) is integrated directly into Automation Runtime. A web browser can be used to read important target system information.
"Network Analyzer" on page 20	The Network Analyzer calculates the cycle times of the individual POWERLINK and X2X networks and is used to control the system configuration.

Table 3: Reading system information

3.1 Controller operating status

A number of options are available in Automation Studio for evaluating the operating status of a controller.

This information can also be displayed or dumped as a System Dump using System Diagnostics Manager ("System Diagnostics Manager (SDM)" on page 55).

3.1.1 Status bar

The status bar is located at the bottom of the Automation Studio window.

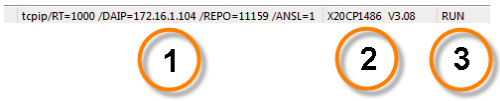


Figure 6: The status bar

The status bar includes the following information:

1. Connection settings
2. CPU type and Automation Runtime version
3. Operating state of the controller

In addition to the data shown, the status bar also indicates license violations on the target system. Additional information can be found in Automation Help.

Project management \ The workspace \ Status bar

Real-time operating system \ Method of operation \ Operating status

3.1.2 Information about the target system

With an active online connection you can query information about the target system using **<Online> / <Info>** in the main menu or using **<Online Information...>** in the Physical View control shortcut menu.

The information dialog box contains the following elements:

- Test the status of the internal backup battery
- Target system type and Automation Runtime version
- Hardware node number
- Available memory on the target system
- Date and time of the target system.
- Setting the date and time

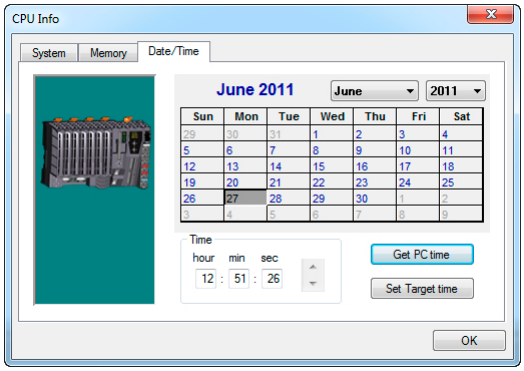


Figure 7: Setting the date and time on the target system

In order to get diagnostically conclusive data, it is necessary for the time and date on the controller to be correct. Date and time are set in the online information dialog box or by using the library functions. In addition, the controller has a configuration option for time synchronization with a time server.

Diagnostics and service \ Diagnostic tools \ Information about the target system

Exercise: Set the time and date

In order to get diagnostically conclusive data, it is necessary for the time and date on the controller to be correct.

1) Open the "Information about the target system" dialog box
2) Get PC time, set target system time
3) Check the logger entry in the "System" category

See: "Error analysis in Logger" on page 15

3.2 Comparative functions

To get an overview, it is necessary to check whether the hardware and software configurations in the opened project match those in the target system. This check is supported by the comparison functions in Automation Studio.

3.2.1 Online software comparison

The online software comparison is used to compare the status and versions of tasks on the target system and compare them with the software configuration in the project.

The online software comparison is opened by selecting **<Online> / <Compare> / <Software>** from the main menu.

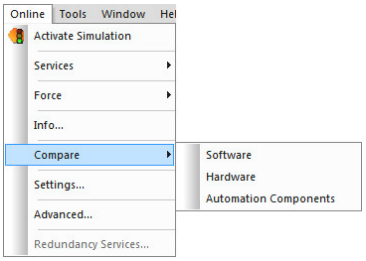


Figure 8: Open online software comparison

The following information is analyzed:

- Comparison of software objects contained in the project with those on the target system
- Target memory of software objects
- Operating status of individual tasks
- Versions and timestamp of the last build

The online software comparison is displayed in two columns. The left side shows the elements configured in the software configuration, while the right side shows the configuration active on the target system.

In the example, the task "**LampTest**" on the target system has been stopped, whereas the task "**Loop1**" is not yet present on the target system.

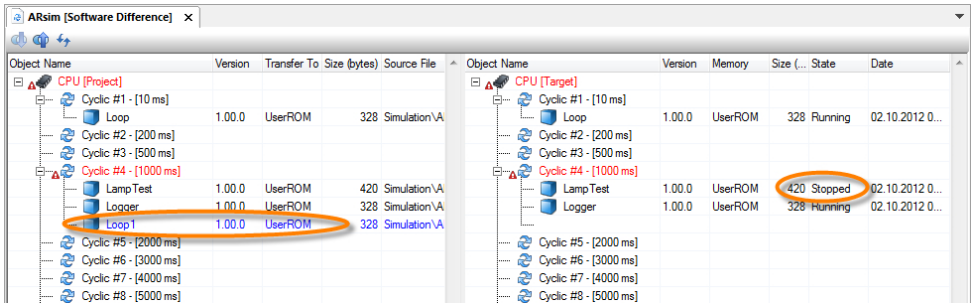


Figure 9: Online software comparison

With the online software comparison, configuration differences between the project and the target system can be recognized. If a comparison is required at a source code level, further tools are made available.

- See: "Source file comparison" on page 51
- See: "Source file comparison on the target system" on page 53

Diagnostics and service \ Diagnostic tools \ Monitor mode \ Online software comparison

Software differences during online installation

During online installation, the differences between the software configuration in the project and the target system are checked. In the transfer dialog box, the differences are displayed by clicking on the corresponding symbol.

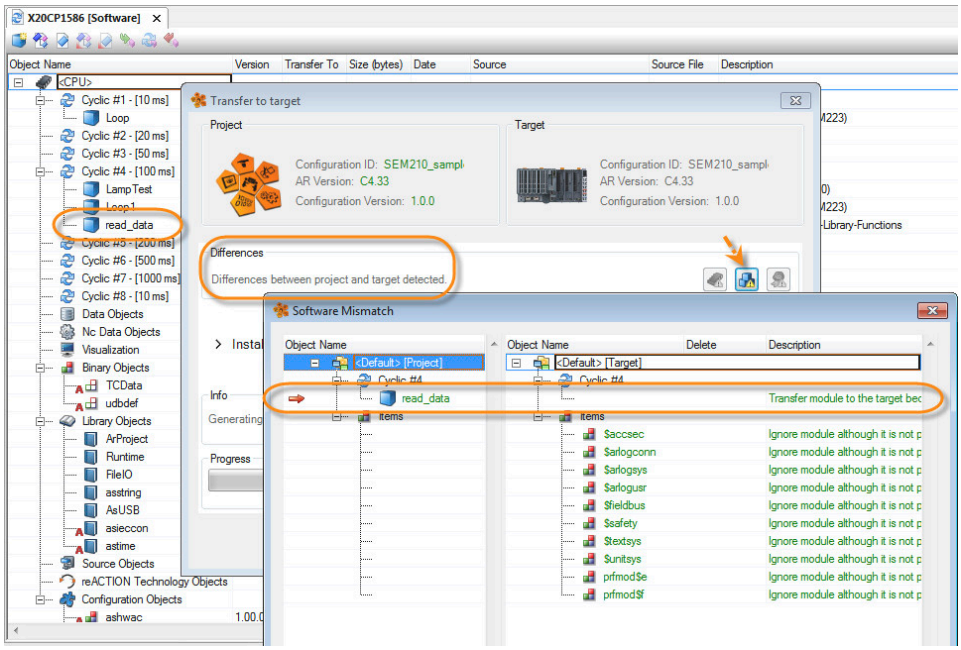


Figure 10: Display of software differences in the transfer dialog box

Project management \ Project installation \ Execute project installation \ Transfer to target \ Differences dialog box

3.2.2 Online hardware comparison

The online hardware comparison can be used to compare the hardware configuration in the project with the one actually being used at runtime. The online hardware comparison can be opened by selecting **<Online> / <Compare> / <Hardware>** from the main menu.

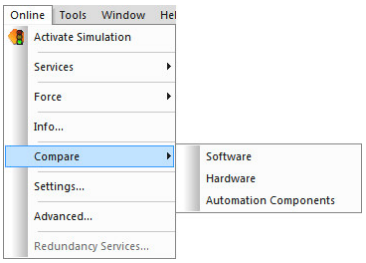
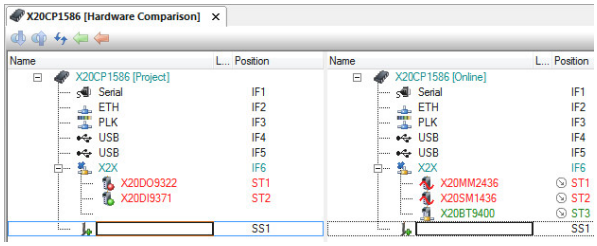


Figure 11: Opening the online hardware comparison window

The window is divided into two halves. The left side shows the hardware configuration in the project. The right side shows the hardware configuration which is used at runtime. Conflicts between the project and the target system are highlighted in red. Additional entries on the project or target system page are highlighted in green.



In this configuration, two digital modules were configured on the X2X Link bus, but other module types are being used on the target system. Another identified module was not configured in the project.

Figure 12: Online hardware comparison

Diagnostics and service \ Diagnostic tools \ Monitor mode \ Online hardware comparison

3.2.3 Online comparison of automation components

Using the comparison for automation components, the configuration objects in the active configuration are compared with the configuration objects on the controller. An overview shows the differences on an object level and the comparison of the configuration objects in detail. Then the parameter values are applied to the Automation Studio project.

The online comparison for automation components can be opened by selecting **<Open> / <Compare> / <Automation Components>** from the main menu.

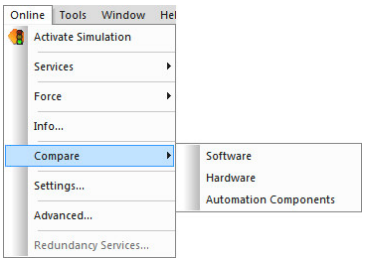
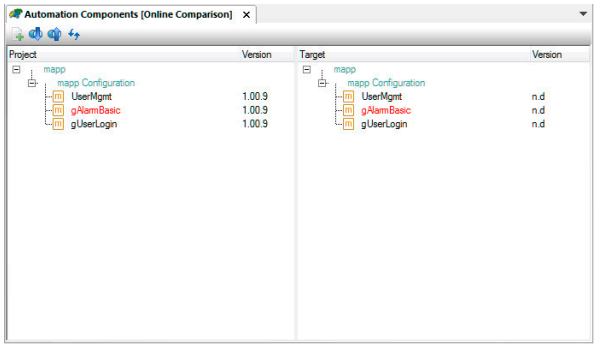


Figure 13: Opening the comparison for automation components



The window is divided into two halves. The left-hand column shows the automation components of the project and the right-hand column shows the configuration objects loaded at runtime. Differences are signaled using different text colors. The possibility of uploading complete configuration objects is offered.

Figure 14: Online comparison of automation components

Applying parameter values

The detailed comparison shows the values of the configuration object in the Automation Studio project alongside the current values on the controller. The values for selected parameters are applied to the Automation Studio project by selecting a parameter or an entire parameter group (1) and then uploading via the toolbar (2).

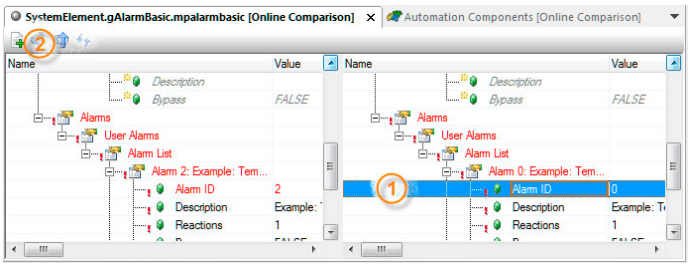


Figure 15: Applying parameter values in the detailed comparison

Diagnostics and service \ Diagnostic tools \ Monitor mode \ Online comparison of automation components

- Overview comparison
- Detailed comparison

3.3 Error analysis in Logger

Automation Runtime logs all fatal errors (e.g. cycle time violations), warnings and information messages (e.g. warm restarts) that take place when the application is executed.

This log is stored in the controller's memory and is available after restarting.

Diagnostics and service \ Diagnostic tools \ Logger window

- Opening the Logger window
- Operating the Logger \ Storing / Loading Logger data
- User interface description \ Backtrace

3.3.1 Logger with an active online connection

The Logger can be opened from the Physical View using the **<Open> / <Logger>** menu item, **<Open Logger>** in the controller's shortcut menu, or using the keyboard shortcut **<CTRL> + <L>**.

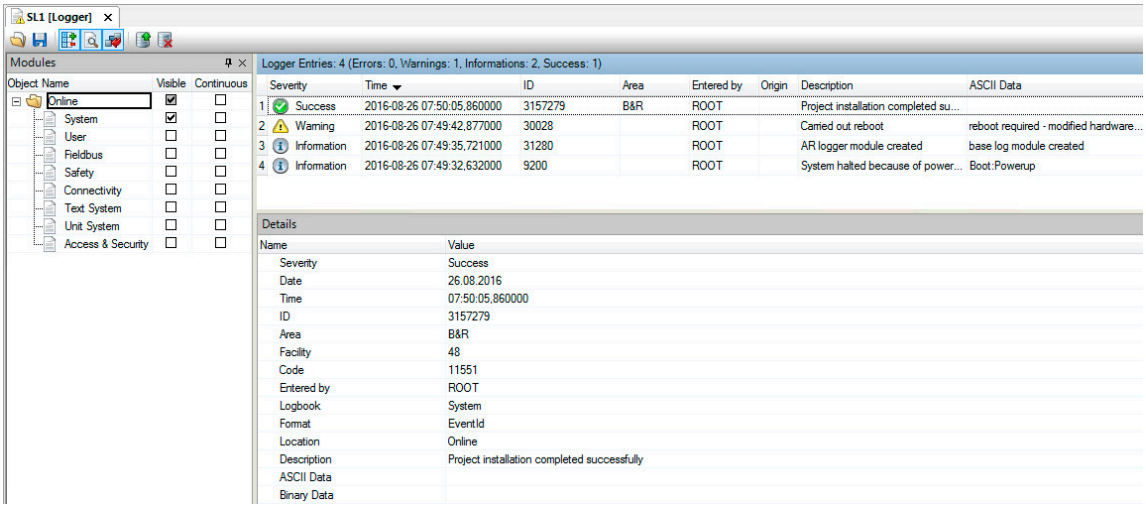


Figure 16: Log window

The entries displayed in this image show the events logged by Automation Runtime after creating an offline installation and the subsequent startup of the controller.

Exercise: Cause a cycle time violation and check the entries in the Logger

By incrementally increasing the variable "**udEndValue**" in the variable monitor in the program "**Loop**", a cycle time violation is achieved.

Once an online connection has been reestablished between Automation Studio and the target system after restarting, open the Logger window and look for the cause of the boot into SERVICE mode.

1) Open the Watch window in the "**Loop**" task's software configuration.

   See "Monitoring and modifying variables" on page 23.

2) Increment the "**udEndValue**" variable until a cycle time violation occurs (loss of connection and restart in SERVICE mode).
3) Open the Logger from the Physical View.
4) Look for the cause of booting in service mode.
5) Select the entry and press **F1**.

Once opened, the Logger indicates the cause of booting in SERVICE mode. Using backtrace, the program that is causing the problem can be quickly identified in this case.

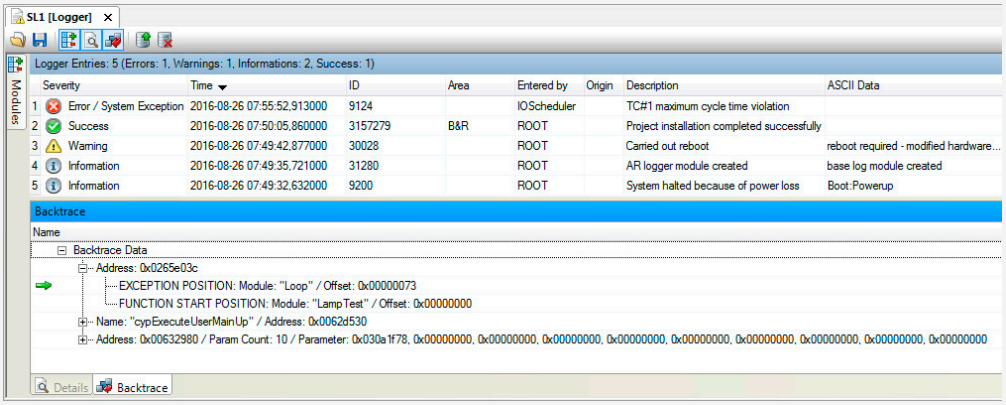


Figure 17: Cycle time violation in the Logger

Once an entry is selected in the Logger, pressing **<F1>** displays a detailed error description in Automation Help. Additional information about error entry is available by displaying the backtrace.

This opens a description of the selected error number in Automation Help.

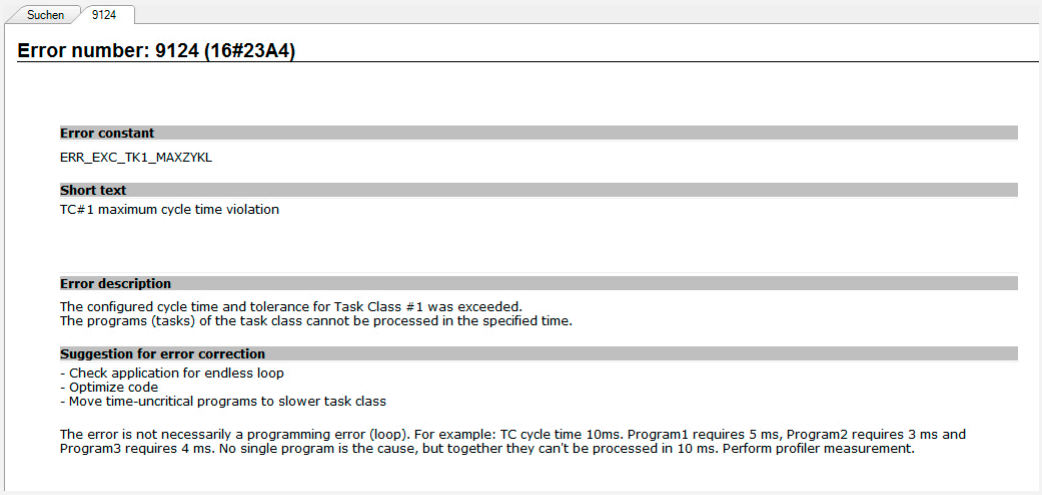


Figure 18: Context-sensitive help for Automation Runtime errors

3.3.2 Offline evaluation of Logger data

Logger records can also be evaluated without a connection to the controller.

Nonetheless, the data itself must always be uploaded by means of an existing online connection to Automation Studio or with the aid of System Diagnostics Manager.

- See "System Diagnostics Manager (SDM)" on page 55.

System dump use case

The Logger data is saved by the service engineer individually using System Diagnostics Manager or as a system dump. The transferred data can now be opened in Automation Studio and analyzed.

In Automation Studio, Logger entries can be saved and reloaded from the Logger's toolbar.

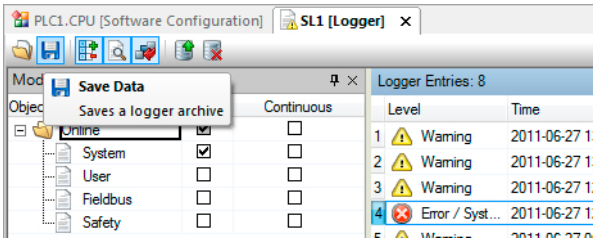


Figure 19: Saving Logger entries

3.3.3 Generating user log data

Logger functions can also be used by the application program to log certain events.

This is handled using the functions in the **ArEventLog** library. The functions contained in this library allow 32-bit event IDs to be entered. They are distinct within the system. Errors, warnings, information and successes are differentiated. Event IDs from the user and from the system can be easily differentiated.

The user cannot enter any event IDs used by the system in the Logger. This is prevented by the library.

The 32-bit event IDs are put together as follows:

Bits 31-30	Bit 29	Bit 28	Bits 27-16	Bits 15-0
Severity	1 .. Customer	Reserve	Facility	Code

User event IDs are generated according to the represented circuit diagram or by using the ArEventLogMakeEventID() function.

This facility makes 12 bits available to clearly identify ranges. For user event IDs, the facility is divided up as follows:

- Values 0 .. 15: Customer applications
- Values 16 .. 4096: Range for device manufacturers and special cases

Programming \ Libraries \ Configuration, system information, runtime control \ ArEventLog

Applications:

- Logging of service actions (e.g. battery exchanged)
- Logging user actions (e.g. forbidden entries)
- Retrieving events of an exception task and entering them in the Logger
- Logging events with deactivated module monitoring (e.g. moduleOK = FALSE)

Exercise: Generate user Logger file

Create a user log file in the existing Automation Studio project. The message **"This is a user warning"** is entered in this Logger file. Concerning message type, it should be a warning (severity = warning). The library example of the ArEventLog library is used for this purpose. After importing and transferring the sample program, the individual functions are enabled via the Watch window.

1) Import "LibArEventLog_ST" library example to the Logical View
2) Performing project installation
3) Generate user Logger file using Watch window
4) Enter warning in the user Logger file using the Watch window

After generating the user Logger file and the Logger entry, the "UsrEvLog" Logger file with the respective entry in the Logger becomes visible.

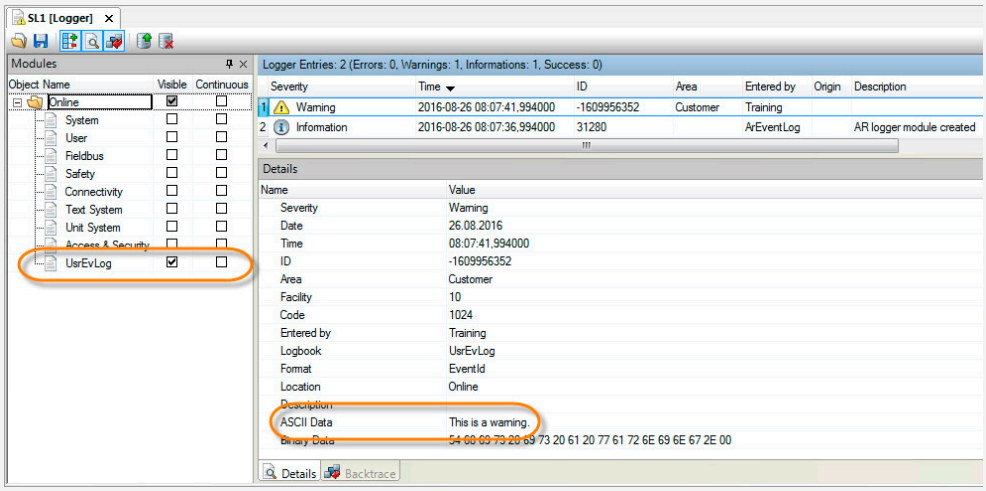


Figure 20: The Logger entry generated by the ArEventLog library

Basically, the sequence when creating a separate program with the ArEventLog library would look like the following:

- Generate user Logger file "UsrEvLog" with ArEventLogCreate()
- Read ident of user logbook with ArEventLogGetIdent()
- Generate 32-bit event ID with the ArEventLogMakeEventID() function
- Write Logger entry in the user logbook with ArEventLogWrite()

Programming \ Libraries \ Configuration, system information, runtime control \ ArEventLog

Programming \ Examples \ Adding examples

Programming \ Examples \ Examples - Libraries \ Configuration, system information, runtime control \ Create and evaluate user logbook

3.4 Network Analyzer

The Network Analyzer calculates cycle times for POWERLINK and X2X networks. It is verified whether it is possible to adhere to the set cycle times with the configured hardware configuration. To this end, the calculated run times are compared with the configured cycle times. Misconfigurations can be detected in this way. The number of hub levels as well as the configured cable lengths are also included in the calculation. The Network Analyzer is opened by selecting **<Open> / <Network Analyzer>** from the main menu.

A POWERLINK bus coupler was inserted in the hardware configuration. The X2X interface of the bus controller was configured at a cycle time of 200µs. The Network Analyzer shows that the calculated runtime is 230µs and therefore this configuration is faulty.

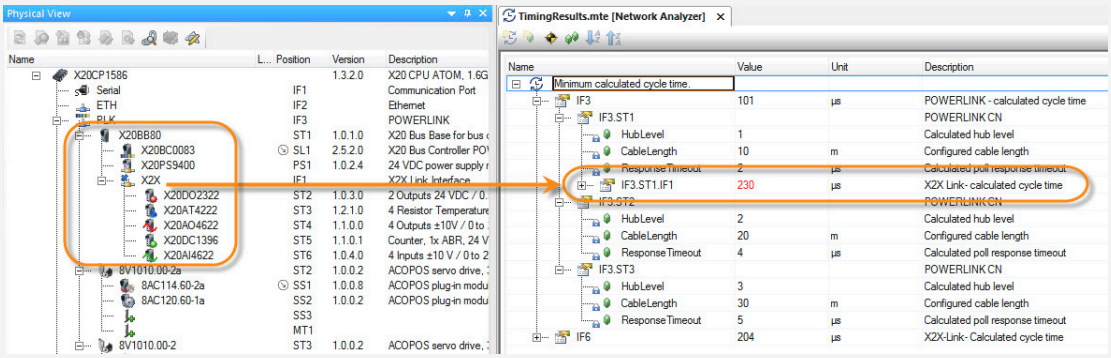


Figure 21: Tabular view of the data in the Network Analyzer

A summary of the results of the Network Analyzer is also shown in the output window. The "Position" column also shows which configuration entry must be checked and corrected.

#	Category	Date/Time	Description	Position
20	Error	24.04.2017 10:20:57,8500	Analyze networks finished. 1 cycle time violation detected. For detailed results see the entries below.	
21	Info	24.04.2017 10:20:57,8812	X20CP1586.IF3: Calculated POWERLINK cycle time (101 µs) is lower or equal than configured time (2000 µs).	Module: X20CP1586.IF3, Property: CycleTime
22	Error	24.04.2017 10:20:57,8812	X20BB80.IF1: Calculated X2X Link cycle time (230 µs) is higher than configured time (200 µs).	Module: X20BB80.IF1, Property: CycleTime

Figure 22: Cycle time violation in X2X cycle time in the output window

In addition to the tabular representation of the calculated results as well as the summary in the output window, configuration problems are marked with a red triangle symbol in the System Designer.

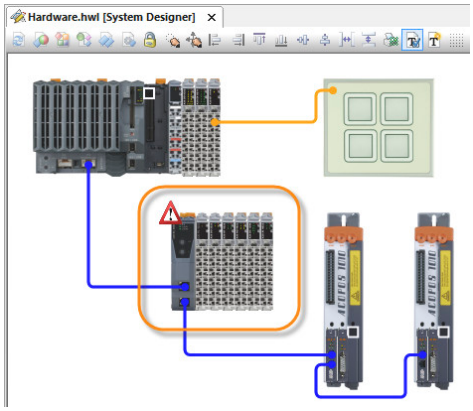


Figure 23: Display of a configuration problem in System Designer

Diagnostics and service \ Diagnostic tools \ Network Analyzer

Detailed information about diagnostics for I/O modules and networks is documented in Automation Help.

Diagnostics and service \ I/O and network diagnostics

Exercise: Check cycle time configuration using Network Analyzer

It is useful o make statements about the entire system load at any time. The cycle time configuration for X2X and POWERLINK networks should now be checked. Set the calculated load for the X2X and POWERLINK interfaces configured in the project using the Network Analyzer.

1) Open Network Analyzer
2) Analyzing the entries in the output window as well as in the tabular view
3) Checking the calculated cycle time in comparison with the configured cycle time

4 Monitoring and analyzing process values

Process values can be monitored, analyzed and modified in many different ways in Automation Studio.

Monitoring and analyzing process values	
"Monitoring and modifying variables" on page 23	The Watch window allows variable values to be monitored and modified.
"Recording variables in real time" on page 26	The trace function makes it possible to record several values in real time over a defined period of time. This data can be uploaded using Automation Studio and displayed in the form of a curve.
"Monitor and force I/O" on page 30	The I/O monitor makes it possible to read the values of I/O variables and status information of I/O modules.

Table 4: Monitoring and analyzing process values

Requirements for the exercises in this section

First, a small program will be added to the existing project. The program contains all necessary process variables needed to complete the exercises in this section.

Exercise: Create the "signal" program

For the other exercises in this training module, some process variables that can be changed are required. For tis purpose, a new program in the Structured Text programming language is now added. Then the variable are implemented and declared.

1) Adding a new Structured Text program in Logical View
2) Rename the program to "signal".
3) Assign the program to task class #1
4) Implementing the program function

   If a digital input is set to TRUE, a number is incremented in each cycle. The number has data type USINT.

   At the end of the program, a total sum is increased in each cycle by the value of the number. The data type of the total sum is INT.

```
PROGRAM _CYCLIC
    (*add 1 each cylce when input is TRUE*)
    IF diEnableRamp = TRUE THEN
        rampSignal := rampSignal + 1;
    ELSE
        rampSignal := 0;
    END_IF

    (*add value to the sum each cycle*)
    sumOverTime := sumOverTime + rampSignal;
END_PROGRAM
```

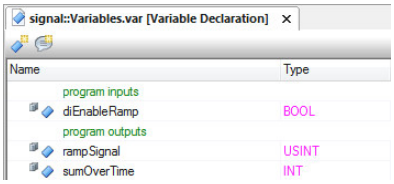


Figure 24: Variable declaration for the "signal" program

4.1 Monitoring and modifying variables

The Watch window allows the values of variables on the target system to be displayed, monitored and modified.

Variable lists are saved in the Watch window for diagnostic and function tests with use and are reused at a later time.

Exercise: Operate and diagnose the "signal" program

The previously created "signal" program should be operated and monitored using the Watch window in Automation Studio.

First, the Watch window is opened via the shortcut menu for the "signal" program. Then all available process variables are added to the Watch window.

Overwriting of process variables during operation may only be carried out by authorized personnel.

4.1.1 Adding the process variables to the Watch window

It is necessary to check if the current project status has been transferred to the controller. This can be done, for example, via the online software configuration. Then the variable monitor is opened and the process variables are added and monitored.

Open the variable monitor (Watch) from the shortcut menu for task "signal" in the software configuration.

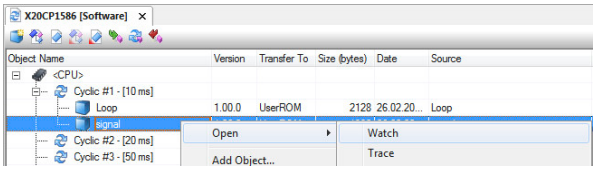


Figure 25: Opening the variable monitor (Watch)

Add all available process variables using the toolbar or by pressing the **<Ins>** key.

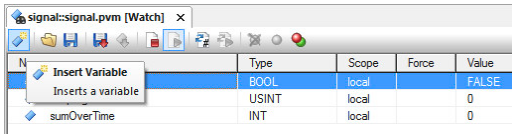


Figure 26: Inserting variables

Once the variables have been inserted in the variable monitor, the process sequence of the application can be simulated.

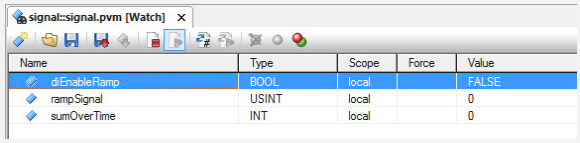


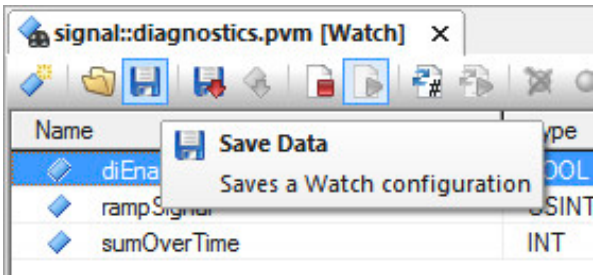
Figure 27: Display the variables in the Watch window

4.1.2 Running the "signal" program using the Watch window

1) Start the process
   Setting variable "diEnableRamp" to TRUE starts the process.
2) Check the status of the process
   Variable "rampSignal" counts up to overflow and then starts again with the value 0.
3) Stop the process
   Setting variable "diEnableRamp" to FALSE stops the process.
   Variable "rampSignal" is set to 0. Variable "sumOverTime" stops at the last value that was calculated.

Variables on the controller can be monitored and modified using the Watch window.

In addition to their values, additional information about the variables such as their scope, data type and I/O type, is displayed.



The variable list in the variable monitor should be saved for later use. This way the used variable list can be restored at any time. The variable list is saved using the toolbar.

Figure 28: Saving the configuration using the toolbar

The name of the list is entered when saving. Several lists can be managed depending on the application.

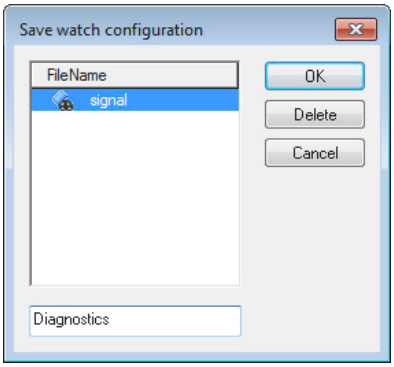


Figure 29: Saving the variable list

Diagnostics and service \ Diagnostics tool \ Variable watch

4.1.3 Writing to variable values simultaneously

If a value is changed in the Watch window, it will be transferred to the controller immediately after **<Enter>** is pressed. The controller will then apply the new value in the next cycle.

In order to enter several values in the Watch window without immediately transferring data to the controller, archive mode must be turned be on.

Archive mode can be started or ended using the "**Archive mode**" icon in the toolbar.

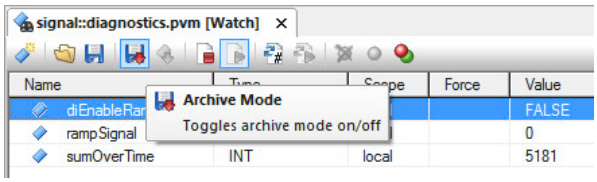


Figure 30: Enabling archive mode

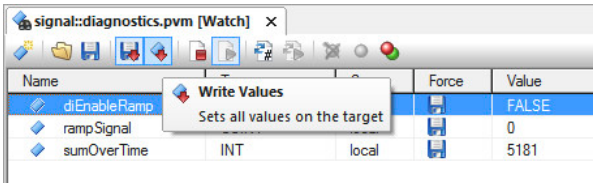


Figure 31: Changing all values in archive mode

After the values for the variables that need modification have been entered in the Watch window, they will be sent to the controller by clicking the "**Write values**" icon in the toolbar.

Diagnostics and service \ Diagnostic tools \ Watch window \ Archive mode

4.2 Recording variables in real time

When using the Watch window, the variables on the controller are polled by Automation Studio.

However, this type of asynchronous accessing of the actual value changes in the Automation Runtime task class system leads to the following limitations:

- Value displayed asynchronously to the task class
- Unable to determine series of value changes and their dependencies

The "**Trace**" function can be used to record changes in values on the target system in real time and synchronous to the task class.

By analyzing trace recordings, processes in the application can be optimized and errors detected.

The following example shows how another process is started when the state of a particular variable is changed. The measurement cursor can be used to establish the time difference between the corresponding value changes of both curves.

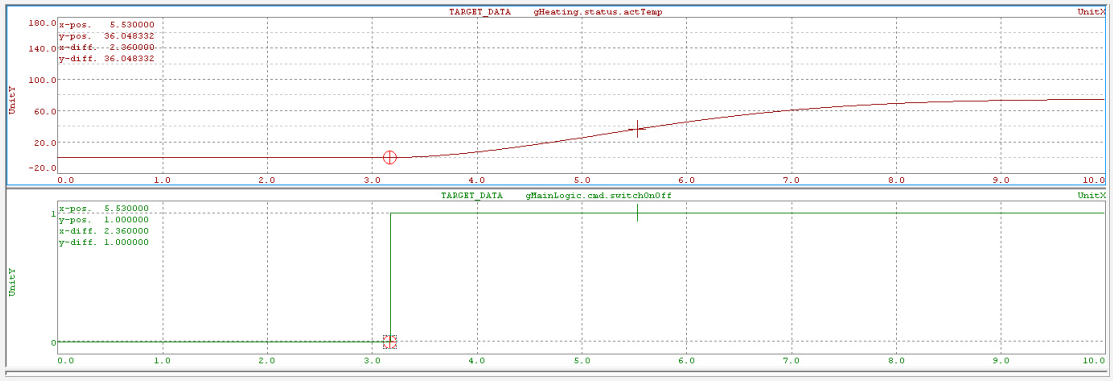


Figure 32: Example of a Trace recording

Exercise: Record a curve that depends on other variables

When running the "signal" program, the variables are interdependent. When the process is activated by setting variable "diEnableRamp" to TRUE, all other variables are affected. The dependencies are recorded using the trace.

1) Opening the Trace window and adding variables

   See "Opening the Trace window and adding variables" on page 27.

2) Configuring the "diEnableRamp" trigger condition

   See "Edit ing theTrace configuration" on page 28.

3) Starting the process and analyzing Trace data

   See "Starting the process and analyzing Trace data" on page 29.

4.2.1 Opening the Trace window and adding variables

The Trace dialog box is started in the software configuration using the shortcut menu for the corresponding task **<Open> / <Trace>**.

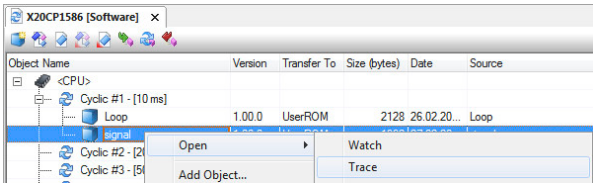


Figure 33: Opening the Trace window in the software configuration

A new Trace configuration must be inserted in the Trace dialog box using the "**Insert Trace Configuration**" button.

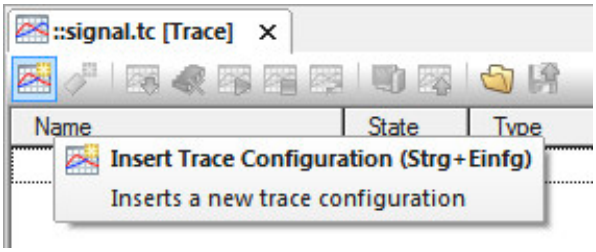


Figure 34: Inserting a new Trace configuration

The variables to be recorded are added to this Trace configuration by clicking the **"Insert New Variable"** pushbutton. For this example, all available variables from the "signal" program are added.

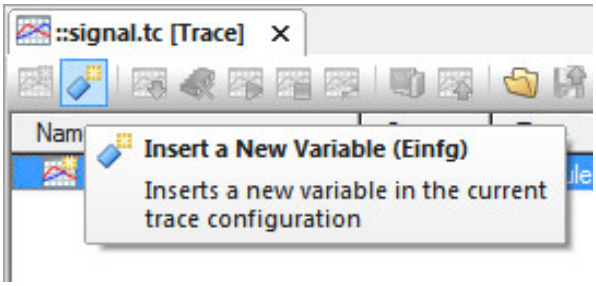


Figure 35: Adding variables to the Trace configuration

The Trace configuration looks like this:

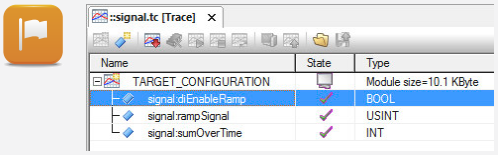


Figure 36: Finished Trace configuration

The Trace configuration is now installed on the target system. Values are recorded cyclically in the context of the task class. The period and start condition of the recording can be configured in the Trace configuration's properties.

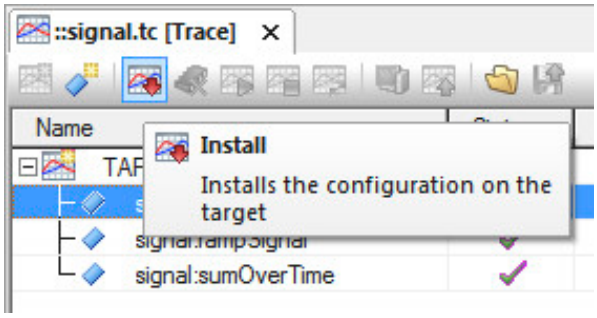


Figure 37: Installing the Trace configuration on the target system

For the application to be able to control the start time of Trace recording on the target system, it is necessary to modify the Trace configuration.

See "Edit ing theTrace configuration" on page 28.

4.2.2 Edit ing theTrace configuration

Trace recording should not be started until the process has been enabled in the application program by setting variable "diEnableRamp".

A trigger condition is configured for this. The "**Properties**" entry in the shortcut menu for the Trace configuration opens the "Properties" dialog box.

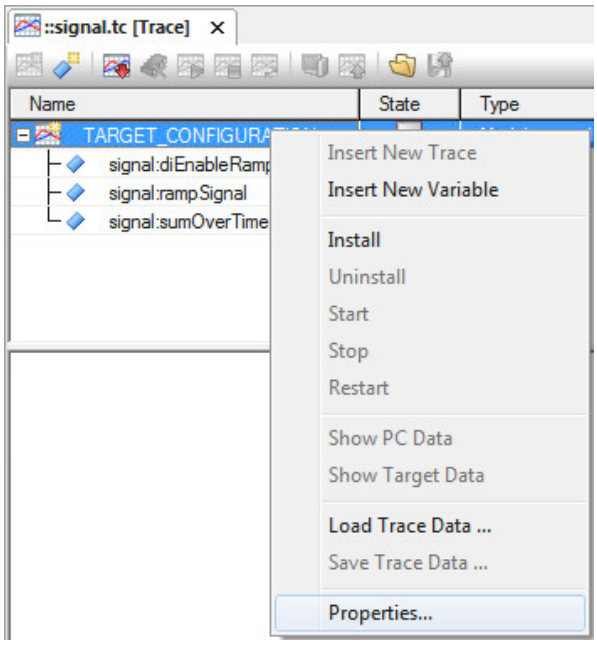


Figure 38: Trace properties

The recording buffer is configured for a longer recording duration. The size of the recording buffer can be set to 3,000 entries on the "**General**" property page.

The Trace mode is changed to set a trigger condition. A trigger condition for starting the recording can be configured on the "**Mode**" property page (**diEnableRamp = 1**).

The dialog box for selecting the variables for the trigger condition is opened by pressing the **<Space bar>**.

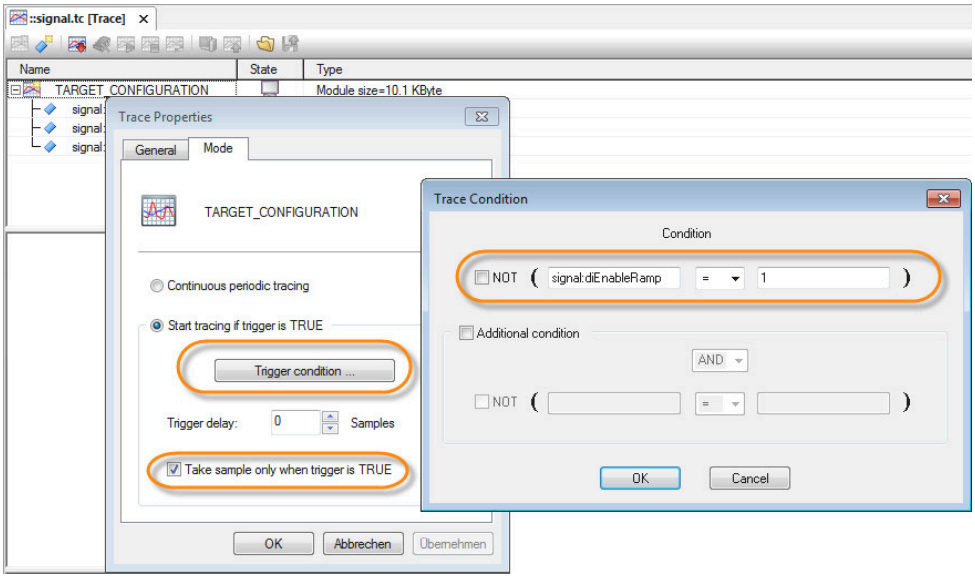


Figure 39: Configuring the trigger condition

Once the recording itself has been configured, it will be transferred to the target system by pressing "**Install**".

Recording now takes place automatically as soon as the trigger condition is met.

4.2.3 Starting the process and analyzing Trace data

Now the Trace data is recorded depending on if the trigger condition has been met.

1) Open the Watch window
   After opening the Watch window, all variables from the "signal" program are added.

   If the Watch configuration for the task was saved in "Monitoring and modifying variables" on page 23, it will reopen automatically.

2) Start the process
   The process is started by setting variable "diEnableRamp". This results in the previously configured trigger condition being met and the Trace data is recorded.

Recording can be paused at any time by clicking the "**Stop**" icon in the toolbar. The results are displayed by clicking the "**Show target data**" icon after the upload has taken place.

Data is recorded when the trigger condition has been met. Values can be modified as needed in the variable monitor. After uploading, the recorded variables are displayed as individual curves. The curves can also be superimposed, resulting in the following view:

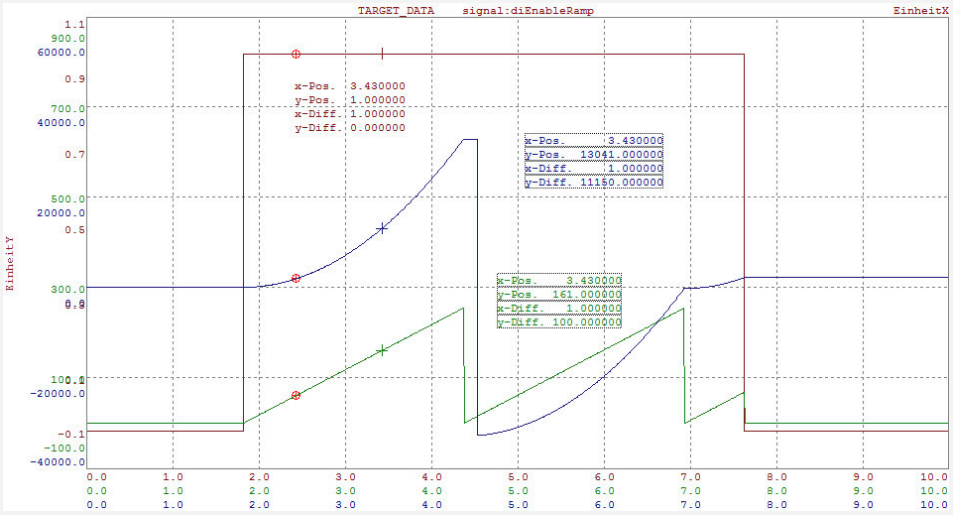


Figure 40: Superimposed Trace data for multiple process variables

Using the measurement cursor, value changes and differences are determined exactly.

Diagnostics and service \ Diagnostic tools \ Trace window

4.3 Monitor and force I/O

Double-clicking on a module in the Physical View opens the I/O mapping window. The physical status of the I/O channels is displayed when there is an active online connection and the active monitor mode is selected.

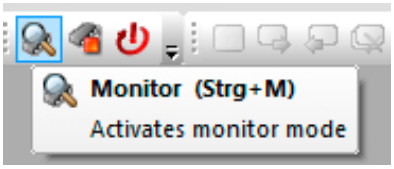


Figure 41: Switching on monitor mode

Channel status and status data points

Via the I/O mapping, the channels of an I/O module are directly assigned to process variables. In addition to the channel values, further information is also available. The control application can be used to evaluate whether the module has been detected and configured at runtime, via the channel "ModuleOk". Additional status inputs allow a diagnosis of the I/O channels in the application.

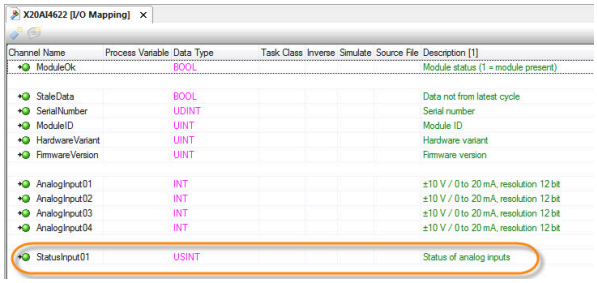


Figure 42: I/O mapping of an analog input module; ModuleOk, StatusInput01

The channel "StatusInput01" is shown in the figure. This contains bit-by-bit information about the individual channels of the module. This makes it possible to detect short circuits, wire breaks and other conditions. A detailed breakdown of the diagnostic information is documented in the register description in the data sheet of the I/O module used. Depending on the type of module, further status information can be activated in the I/O configuration.

Forcing

The option "**Force**" makes it possible to assign any of the I/O data points a value, regardless of their actual physical value. The function "Force" is enabled in the I/O mapping and in the Watch window for variables that is linked to the I/O data points.

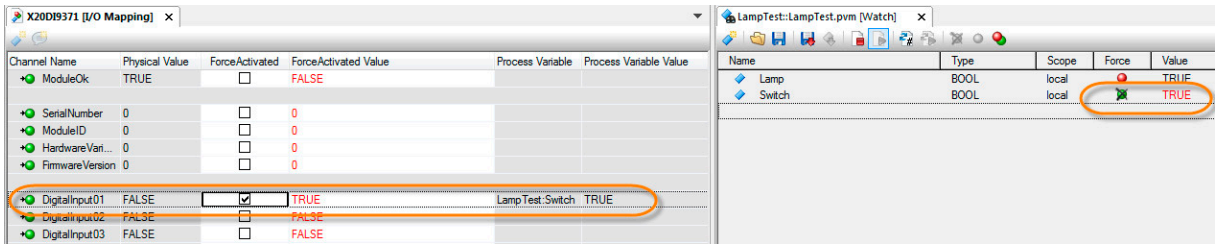


Figure 43: I/O mapping in monitor mode; forcing I/O channels in the I/O mapping and in the Watch window

When forcing inputs of an input module (for example, X20DI9371), the user program operates with the "force" value rather than the actual input state.

When forcing outputs of an output module (for example, X20DO9322) it is written directly to the output of the corresponding hardware. This is independent of the value calculated by the user program.

Before leaving a machine, it must be ensured that there are no force operations still in effect. This is disabled automatically by **restarting** the control system or using the **<Online> / <Force> / <Global force off>** menu item.

Diagnostics and service \ Diagnostic tools \ Monitor Mode \ Mapping I/O channels in monitor mode

Diagnostics and service \ Diagnostic tools \ Force

Diagnostics and service \ I/O and network diagnostics

5 Software analysis during programming

There are several different diagnostic tools available in Automation Studio that provide support when designing the application software and for the error search at run time.

Software analysis during programming	
"Perform runtime measurements in the Profiler" on page 32	The Profiler can be used to measure and display important system data such as task runtimes, system and stack loads, etc.
"Line coverage" on page 42	Line coverage indicates the lines of the source code that are currently being executed.
"Contextual Watch" on page 43	Contextual watch is a tool for displaying the value of variables and parameters at exactly defined source code positions.
"Debugging the source code" on page 45	The debugger makes it easier to search for errors in the source code of a program or library.
"Evaluating event IDs, status variables and return values" on page 48	Status variables are used to identify the status when calling functions and function blocks in the user program
"Using variables in the programs" on page 49	The output window is used to display information about ongoing processes, e.g. building, downloading, generating the cross-reference list, displaying search results, etc.
"Source file comparison" on page 51	With the Automation Studio source file comparison, it is possible to compare individual files (programs, libraries, packages and data objects) or entire projects.

5.1 Perform runtime measurements in the Profiler

Automation Runtime continually records all runtime behavior. Using the Profiler, the runtimes of individual user tasks, the CPU load and different system events can be recorded. The Profiler is opened by selecting **<Open> / <Profiler>** from the main menu.

In the Profiler, different views for displaying the recorded data are offered. These are switched in the toolbar of the Profiler. A table view, a graphic view and a raw data view are available. The most important functions of the Profiler, such as opening the configuration, and starting and stopping the recording, are also controlled via the toolbar.

Figure 44: Toolbar in the Profiler window

For the following tasks, it's recommended to leave open the **Watch window** for the tasks "Loop" and "Loop1", the **software configuration** and the **Profiler**.

Diagnostics and service \ Diagnostics tools \ Profiler

- Recording Profiler data
- FAQs

5.1.1 Configuring the Profiler and carrying out recording

In order to get diagnostically conclusive measurement results, the recording duration and the event types must be configured. For the following tasks, only the execution times of the user tasks, task classes and exceptions are recorded. Short instructions follow, explaining how to edit the configuration and record data.

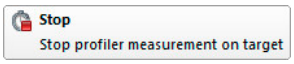
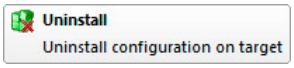
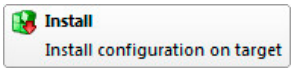
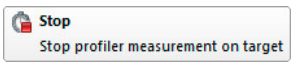
1	In order to edit the configuration, you must stop the current recording.	
2	The existing Profiler configuration is uninstalled from the target system.	
3	To open the dialog box for making configurations, click the "Configuration" icon in the Profiler toolbar.	
4	Transfer your changes to the target system using the "Install" icon in the toolbar. Recording begins again immediately with the new configuration.	
5	The current recording is stopped with the "stop" symbol.	
6	The Profiler data is loaded to and displayed in Automation Studio via the "upload data object" symbol.	

Table 5: Overview of the required steps for changing the Profiler configuration and uploading the Profiler data

Make the configurations shown in the following images:

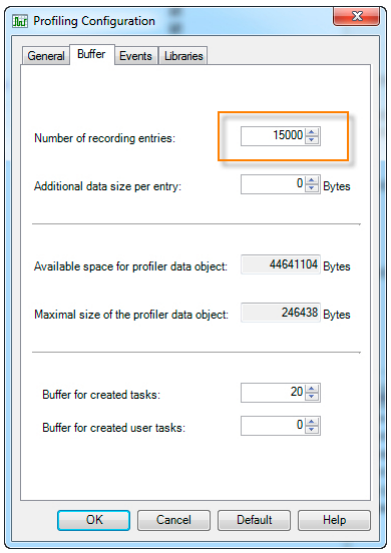


Figure 45: Profiler configuration: recording buffer

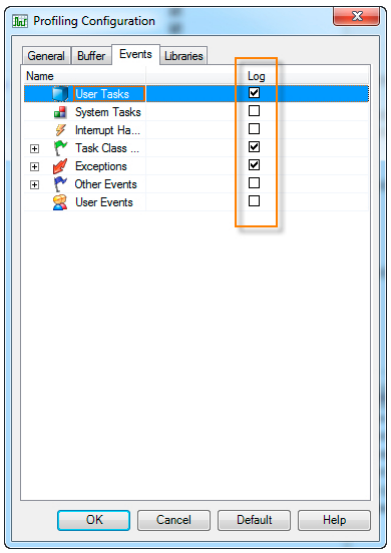


Figure 46: Profiler configuration: event type

If the Profiler data is relayed to third parties, it is recommended to log all events in the "**Events**" tab. Filtering can be performed later.

Diagnostics and service \ Diagnostic tools \ Profiler \ Preparing the Profiler

Exercise: Measure the execution time of the "Loop" program

In the "**Loop**" task, which runs in task class #1, an increased runtime must be triggered by changing the value of the "**udEndValue**" variable in the Watch window.

Use the **Profiler** to monitor the runtime behavior of the task and the available remaining time.

1) Set the "udEndValue" variable to 50000

   The first step is to set the value of the "udEndValue" variable to 50,000.

2) Increase "udEndValue" variable in steps

Results should be analyzed in the Profiler between each of these steps.

5.1.2 Profiler table and graphic views

Different views are available for analyzing the Profiler data. The data uploaded in the preceding task is now displayed in table or graphic form.

Table display of Profiler data

A table view of the Profiler data shows – with the appropriate filtering – the execution time and CPU load of each task.
This view is opened with the "**Table**" icon in the toolbar.

Name	CPU Usage [%]	Tolerance Count	Object Priority	Call Count	Minimal Net Time [µs]	Average Net Time [µs]	Maximal Net Time [µs]	Minimal Gross Time [µs]
Cyclic #1	2.145		230		217.356	218.539	219.832	965.138
loop	1.943		230	15	201.461	202.663	203.198	201.461
Cyclic #4	0.142		198		216.306	216.306	216.306	216.306
System Tasks	6.059							
Interrupt Ha...	1.061							
Idle Tasks	90.593							

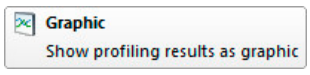
Figure 47: Analyzing the CPU load with the Profiler

In order to get informative Profiler graphs, the call count of the slowest task class in the system must be >3. The recording duration of the Profiler is set via the buffer size in the Profiler configuration.

Diagnostics and service \ Diagnostic tools \ Profiler \ Recording Profiler data \ Analyzing Profiler data \ Analyzing Profiler data in table form

Visual display of Profiler data

Switch to the graphic display of the Profiler data via the "visual" symbol in the toolbar. The representation can be shown as in the following image. Comprehensive analysis options are available via filter, zoom and measurement cursor functions.



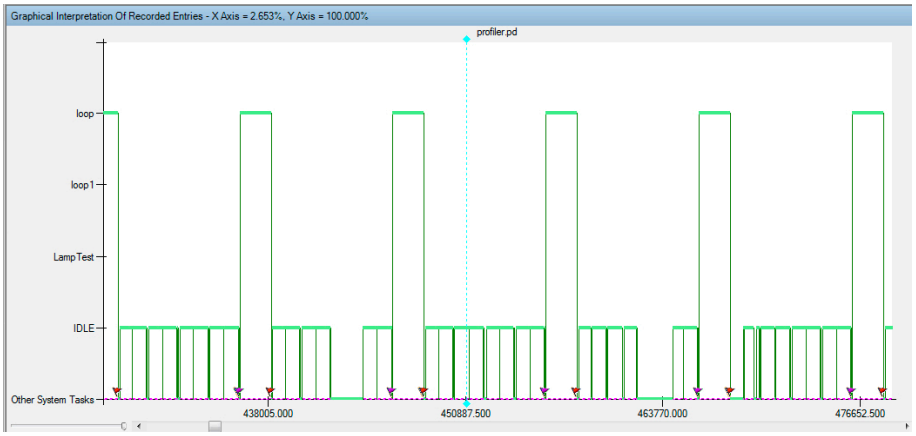


Figure 48: Result of the first Profiler measurement

Diagnostics and service \ Diagnostic tools \ Profiler \ Recording Profiler data \ Analyzing Profiler data \ Show Profiler data as graph

- Navigating in the graph view
- Zooming in the graph view

5.1.3 Measuring program runtime in the Profiler

To determine the exact execution time, you can set a reference cursor at the beginning of **"Loop"** using the icon in the Profiler toolbar and then set the cursor at the end of **"Loop"** to see the time.

Exercise: Increase the "udEndValue" variable in steps

The Profiler is restarted in this step. Increase the value of the variable "**udEndValue**" in steps (e.g. 10000). Use the Profiler to monitor the execution time of the "**Loop**" task. In addition, when a value is changed, the recording is stopped and uploaded.

The **"Loop"** task operates in a 10 millisecond task class. Looking more closely at the image, a cycle time violation must have occurred since the execution time has already reached 16.5 milliseconds.

The tolerance – defined in the properties of task class #1 as 10 milliseconds – now takes effect.

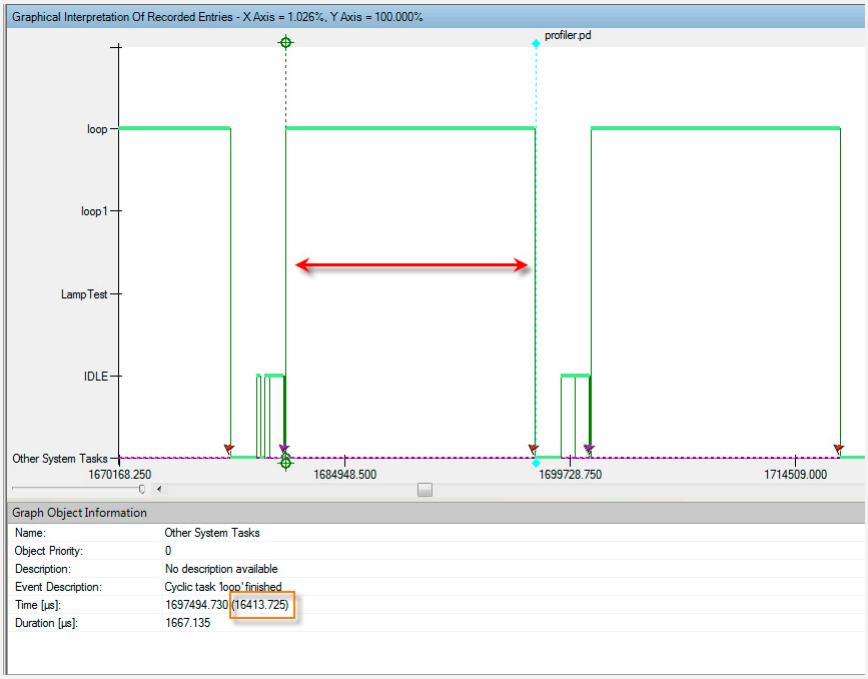


Figure 49: Exceeding the cycle time, effect of tolerance

With the value "tolerance count", it can be determined in the table view if the tolerance time is already active in a task class. In the following image, the middle runtime of task class #1 is 10.7 ms. The tolerance count shows that the configured execution time was already exceeded 271 times.

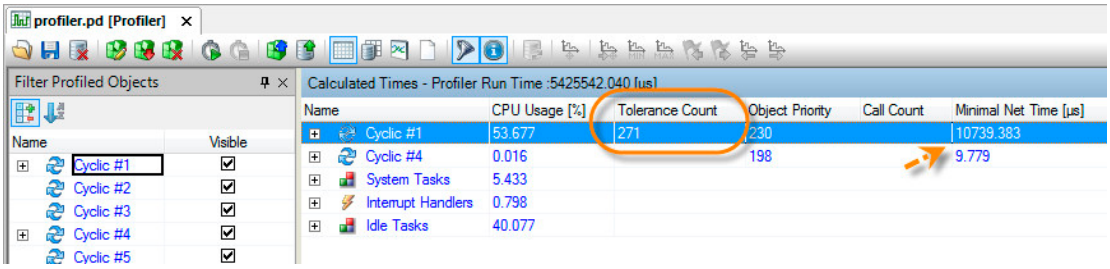


Figure 50: Parameter tolerance count in the Profiler

The example shows the effect of using the tolerance on the average CPU load. Task class #1 causes a CPU load of around 54% even though the configured cycle time of 10 ms was exceeded. In this case, when the tolerance becomes effective, the task class #1 is only called again every 20 ms.

Exercise: Record task classes with different priorities

Task class priority makes it possible for a task in a higher priority task class to interrupt a task in a lower priority task class that takes longer.

Based on the tasks "**Loop**" and "**Loop1**" use the Watch window to change the final value of the variable "**udiEndValue**" so that the task "**Loop1**" is interrupted exactly **twice** by the task "**Loop**". Set the starting value for the **"udEndValue"** variable in the **"Loop"** task to 2,000. A Profiler measurement could look like the following:

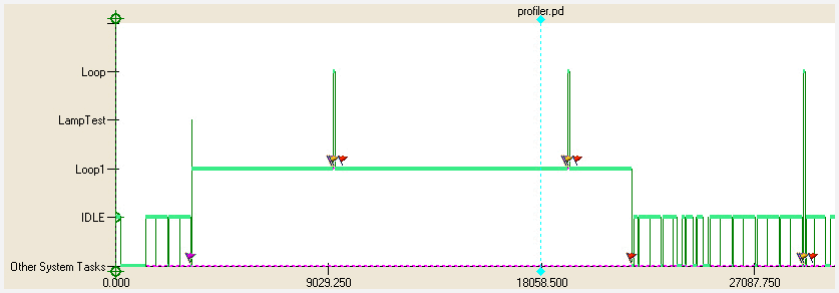


Figure 51: "Loop" task interrupts "Loop1" task - Display in the graphic view of the Profiler

When the "Loop1" task is executed in task class #4, the input image is available until the task has been completely executed.

5.1.4 Reading Profiler data after an error

The Profiler data can be uploaded by the target system at any time after stopping the recording. In case of an error, the Profiler data is also retained. It can then be determined which event, for example, led to the system being restarted.

The Profiler configuration should be adjusted so that the Profiler data is kept after restarting. The target memory for Profiler data and the Profiler configuration is set to "USERROM".

Diagnostics and service \ Diagnostic tools \ Profiler \ Preparing the Profiler \ General settings

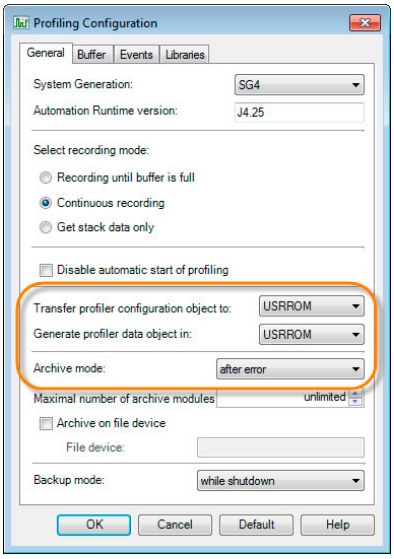


Figure 52: Settings for the target memory of Profiler data and the Profiler configuration

Exercise: Cause a cycle time violation and evaluate the Profiler data

A cycle time violation can occur by increasing the value of the "**udEndValue**" variable in the "**Loop**" task.

After restarting the target system in SERVICE mode, open the Profiler and load the Profiler data from the target system.

1) Cause a cycle time violation by setting the "**udEndValue**" variable to the value 500000 in the Watch window
2) After restarting into SERVICE mode, open the Profiler in the software configuration by selecting the <Open> / <**Profiler**> menu item.

If the configured **cycle time + tolerance** was exceeded during runtime, Automation Runtime triggers an exception. If the application program is not configured to handle this exception, the target system will restart in SERVICE mode.

In the Profiler, data is uploaded by clicking on the **"Upload data object"** icon in the toolbar. If there is an error, a new Profiler file is generated upon restart, which is given a timestamp. The corresponding file can be selected from a list during the uploading process.

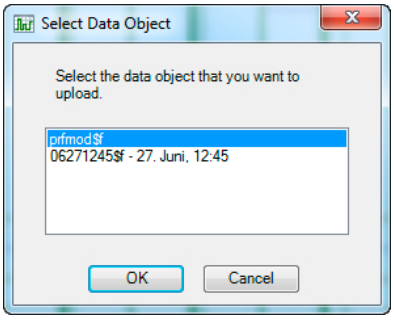


Figure 53: Selection of the profiler data

Profiler data can be filtered to limit the events being displayed. Which events should be displayed depends on the situation itself.

Diagnostics and service \ Diagnostic tools \ Profiler \ Recording Profiler data \ Analyzing Profiler data

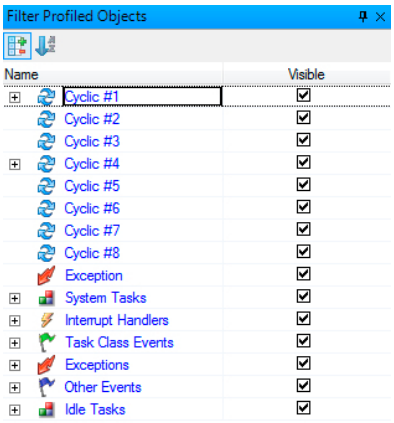


Figure 54: Filtering Profiler data

At a certain point in time, the time it takes to complete the task exceeds the configured cycle time and tolerance. This event (exception) is indicated by an appropriate icon.

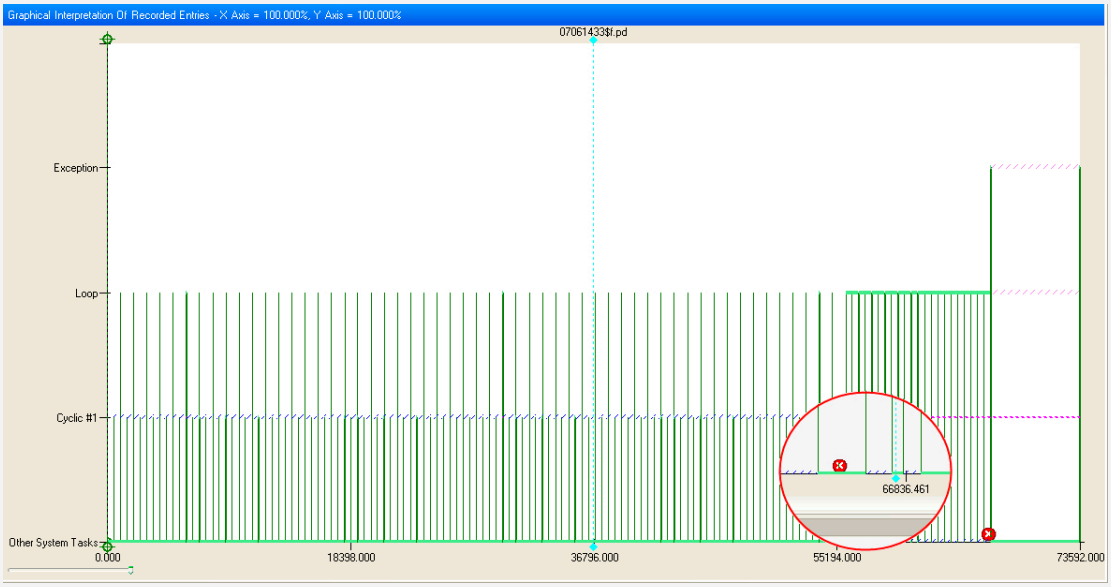


Figure 55: Exception in the profiler data

To analyze the cause, the data that comes before this point in time must be observed.

Using the measurement cursor and zooming in as necessary on the Profiler data are two ways that the data can be analyzed.

"Loop" task execution time

The image shows that the task "Loop" is normally carried out in less µs (blue arrow). In the case of cycle time violation, the configured cycle time + tolerance is exceeded (red arrow).

This image shows how a simple application is recorded in the Profiler. The cause of a problem is generally harder to detect in real applications since there are usually several tasks / task classes running. By setting filters, specific processes can be targeted.

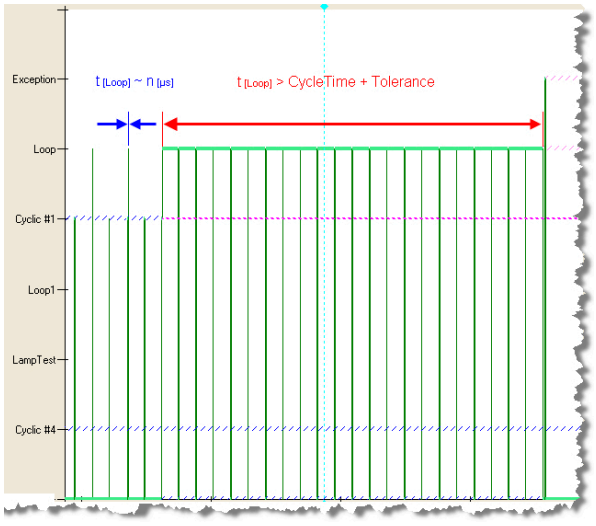


Figure 56: Determining the cycle time violation

Example:

Two tasks are running in task class #1 that usually finish executing within the configured task class cycle.

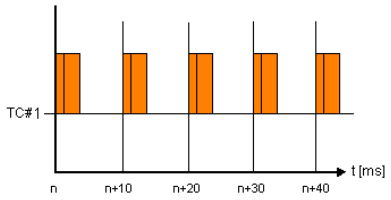


Figure 57: Timing diagram - Execution times in task class #1

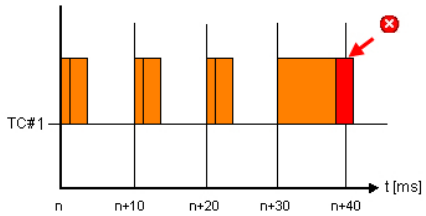


Figure 58: Timing diagram - Execution times in task class #1 exceed the configured cycle time

If it takes longer to complete the first task (beyond n+30 ms in the diagram) and the completion time for both tasks together exceeds the configured cycle time plus tolerance, then it will be the second task that is entered as the cause of the error although it is not really the main reason for the cycle time violation.

The sequence of events can be analyzed chronologically by evaluating the raw data ("**Output data**" icon in the toolbar).

The start and end of the call of the "Loop" task is entered in this list. If the time sequence is continued, it can be determined that the task has been started but has not ended.

Raw Data Of Recorded Entries

Nr.	Name	Event	Event Description
465	TickGen	0x000...	'TickGen' is now running - 'IOScheduler' was waiting
466	Cyclic #1	0x001...	'Cyclic #1' is now running - 'TickGen' was waiting
467	Cyclic #1	0x1e0...	Task class 'Cyclic #1' started
468	Cyclic #1	0x1e0...	Input scheduler of task class 'Cyclic #1' finished
469	Loop	0x020...	Cyclic task 'Loop' started
470	Loop	0x020...	Cyclic task 'Loop' finished
471	Cyclic #1	0x1e0...	End of cyclic programs in task class 'Cyclic #1'
472	Cyclic #1	0x1e0...	Output scheduler of task class 'Cyclic #1' started
473	DdX2XAcc.IF6	0x007f...	'DdX2XAcc.IF6' is now running - 'Cyclic #1' was waiting
474	tEplV2If.IF3	0x007f...	'tEplV2If.IF3' is now running - 'DdX2XAcc.IF6' was ready
475	DdX2XAcc.IF6	0x007f...	'DdX2XAcc.IF6' is now running - 'tEplV2If.IF3' was delayed and waiting

Figure 59: Raw data for a Profiler recording

5.2 Searching for errors in the source code

When it comes to software, statistics have shown that there are usually around two to three errors contained in every 1,000 lines of code.

Automation Studio provides extensive diagnostic tools for locating the source of program errors.

5.2.1 Monitor mode in the program editor

Monitor mode is started by selecting the "**Monitor**" icon in the programming editor toolbar.

Monitor mode is available for each programming language and allows variables to be observed in several ways:

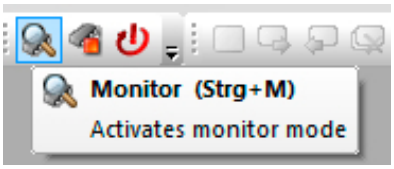


Figure 60: Enable monitor mode

Tooltips in text-based programming languages

The value is shown as a tooltip of a variable in both text-based and visual programming languages.

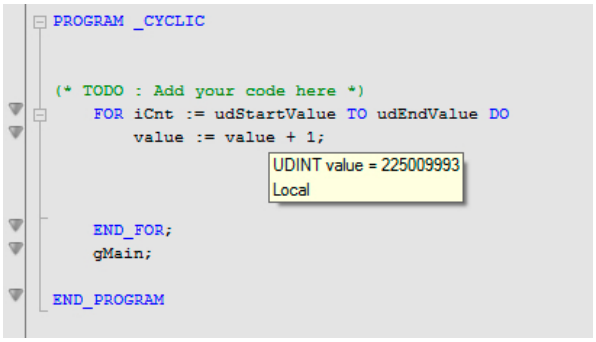


Figure 61: Tool tip in source code

Value display in visual programming languages

The value is shown right beside the variable name in visual programming languages.

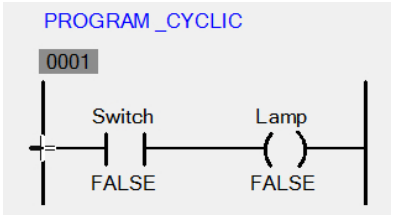


Figure 62: Graphical programming language in monitor mode

Watch window

The Watch window is shown next to the source code when monitor mode is enabled. It is also possible to open the Watch window from a task's shortcut menu in the software configuration or from the online software comparison.

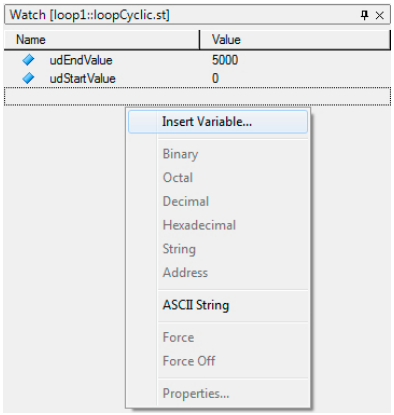


Figure 63: Watch window view

5.2.2 Powerflow

The path of a signal can be displayed in visual programming languages such as Ladder Diagram. This signal path is enabled by selecting the "**Powerflow**" icon in the toolbar. When activating the signal flow display, the message for deactivating the cycle time monitoring must be confirmed.

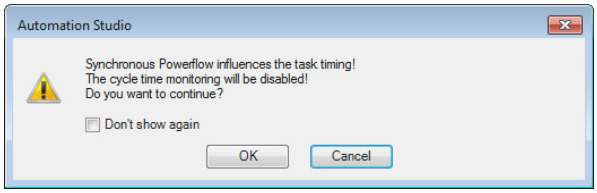


Figure 64: Message for deactivating the cycle time monitoring

Exercise: Power Flow in Ladder Diagram

The objective of this exercise is to enable powerflow in the "**LampTest**" program.

The path of the signal can be changed in the Watch window by changing the value of the "**Switch**" variable.

1) Open the "**LampTest**" program when there is an online connection
2) Enable monitor mode
3) Add the "**Switch**" and "**Lamp**" variables to the Watch window
4) Set the "**Switch**" variable
5) Observing the results

If the contact condition for the variable "Switch" is fulfilled, the coil "Lamp" is set and the signal path is colored.

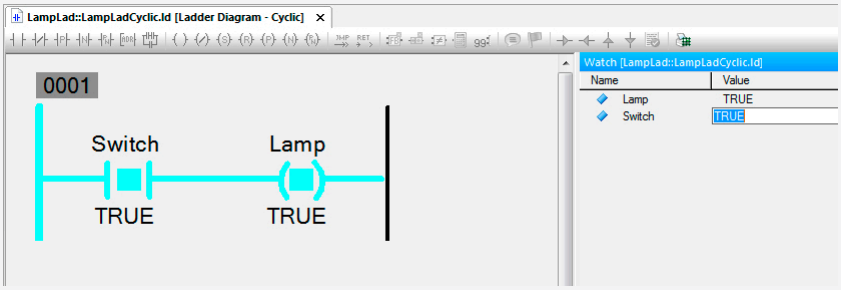
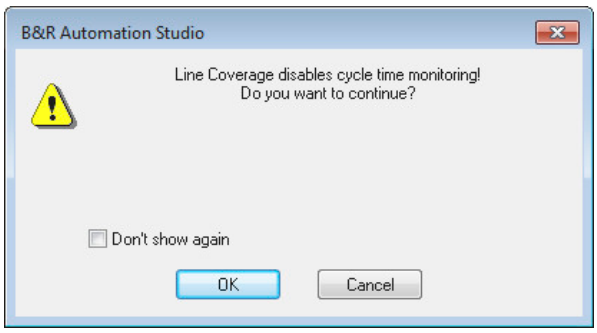


Figure 65: Power Flow enabled in Ladder Diagram

Diagnostics and Service \ Diagnostic tools \ Monitors mode \ Programming languages in monitor mode \ Powerflow

5.2.3 Line coverage

Line coverage is enabled via the "Line coverage" icon in the toolbar.



When activating the line coverage, the message for deactivating the cycle time monitoring must be confirmed.

Figure 66: Message for deactivating the cycle time monitoring

Activated line coverage makes it possible to see exactly which lines are being run at which time. If line coverage is enabled for text-based programming languages, a marker indicates the lines of code that are currently being executed.

```
PROGRAM _CYCLIC


(* TODO : Add your code here *)
    FOR iCnt := udStartValue TO udEndValue DO
        value := value + 1;



    END_FOR;
    gMain;

END_PROGRAM
```

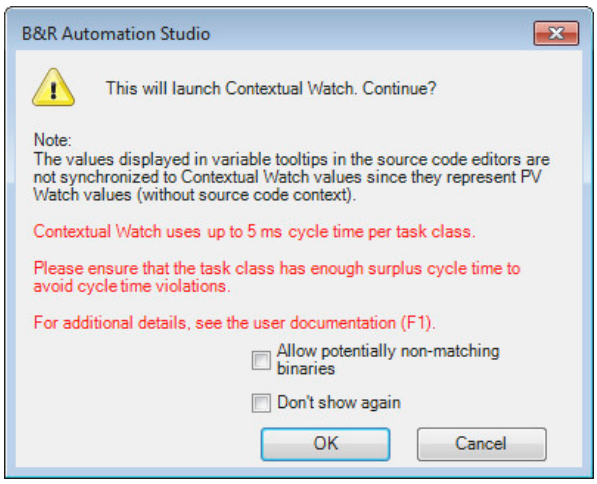
Figure 67: Line coverage in Structured Text

Diagnostics and service \ Diagnostics tool \ Monitors \ Programming languages in monitor mode \ Line coverage

5.2.4 Contextual Watch

Contextual watch is a tool for displaying the value of variables at defined source code positions. The real-time capability of the target system is not affected.

The Contextual Watch is activated with the activated monitor mode via the Debugger toolbar. The corresponding display of the Contextual Watch is part of the output window.



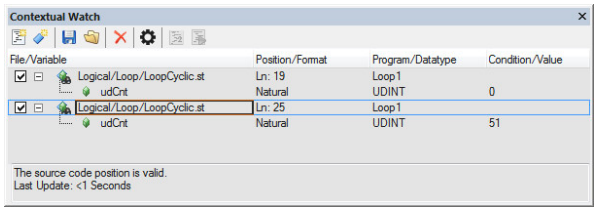
When the Contextal Watch is activated, a message appears, which must be confirmed. It should be noted that the values in Tooltips are not synchronized with the values in the Contextual Watch. Furthermore, Contextual Watch requires an additional runtime for each task class. This is important in order to ensure that there are no cycle time violations during the diagnosis with Contextual Watch.

Figure 68: Message when activating Contextual Watch

```
16  PROGRAM _CYCLIC
17
18      (* implementation of program Loop *)
19      FOR udCnt := udStartValue TO udEndValue DO
20          udValue := udValue + 1;
21      END_FOR
22
23      gMain := gMain + 1;
24
25      udCnt;
26
27  END_PROGRAM
```

As soon as Contextual Watch is active, drag and drop variables can be dragged into the output window.

Figure 69: Source code with variable "udCnt";color-marked source code lines were inserted into Contextual Watch



In the example, the variable "udCnt", which is used at several points in the program, has been inserted into Contextual Watch. It can be seen that the variable value is different depending on the source code position.

Figure 70: Representation of both values of "udCnt" in ContextualWatch

Diagnostics and service \ Diagnostics tools \ Contextual Watch

Exercise: Observe "Loop" program in Contextual Watch

When observing the program "Loop" in the variable monitor, it was observed that the value of the variable is displayed at the end of the program. The objective of this task is to display variables that are used several times with contextual watch and their values in a context-dependent manner.

1) Program "Loop" at the end of the program to expand the following line

```
udCnt;
```

2) Transfer the program to the target system
3) Activate Monitor Mode
4) Enabling Contextual Watch
5) Insert the "udCnt" variable into the Contextual Watch
6) Verify the results

5.2.5 Debugging the source code

The debugger is an important tool for programmers that makes it easier to find errors in the source code of a program or in a library.

Debugging possibilities in Automation Studio

- Line-by-line execution of a program and the parallel monitoring of variables in the auto-watch window
- Stopping the application using breakpoints
- Stepping into called functions (e.g. in library functions / function blocks as long as the source code is available).

Exercise: Find errors in a Structured Text program using the debugger

Create a Structured Text program called "**dbgTest**".

Add a USINT array called "**AlarmBuffer**" with a length of 10 and a UINT variable named "**index**" to the "**dbgTest.var**" file.

In the cyclic part of the program, the array initializes with any value (e.g. 112).

The following – faulty – program code contains one of the most commonly made errors.

Program code

```
PROGRAM _CYCLIC
    FOR index :=0 TO 10 DO
        AlarmBuffer[index] := 112;
    END_FOR
END_PROGRAM
```

Table 6: Faulty program subroutine

When the program is started, the value 112 is written to each of the 10 elements of the array; the program seems to be working.

Error overview: there is write access to index 0 to 10. The array only has 10 elements (index 0 to 9). This type of error is often difficult to detect at first glance and causes the program to overwrite the following memory locations.

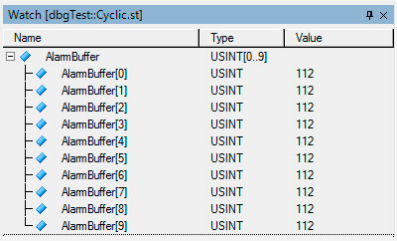


Figure 71: Watch window in monitor mode

The information in the debugger as well as the variables (and their values) in the Watch window will be used to try to analyze the error situation.

Monitor mode and debugger in the toolbar

Monitor mode can only be enabled if a program editor is open.

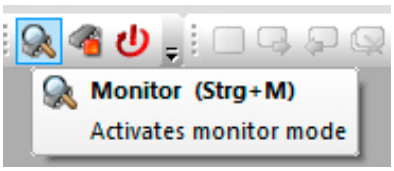


Figure 72: Enabling monitor mode

The debugger can be turned on and off from the menu bar.

Figure 73: Turning the debugger on/off

Adding variables to the Watch window

After activating monitor mode, the "AlarmBuffer" variable is added to the Watch window.

The left side of the window shows the program code, whereas the Watch window is shown on the right side.

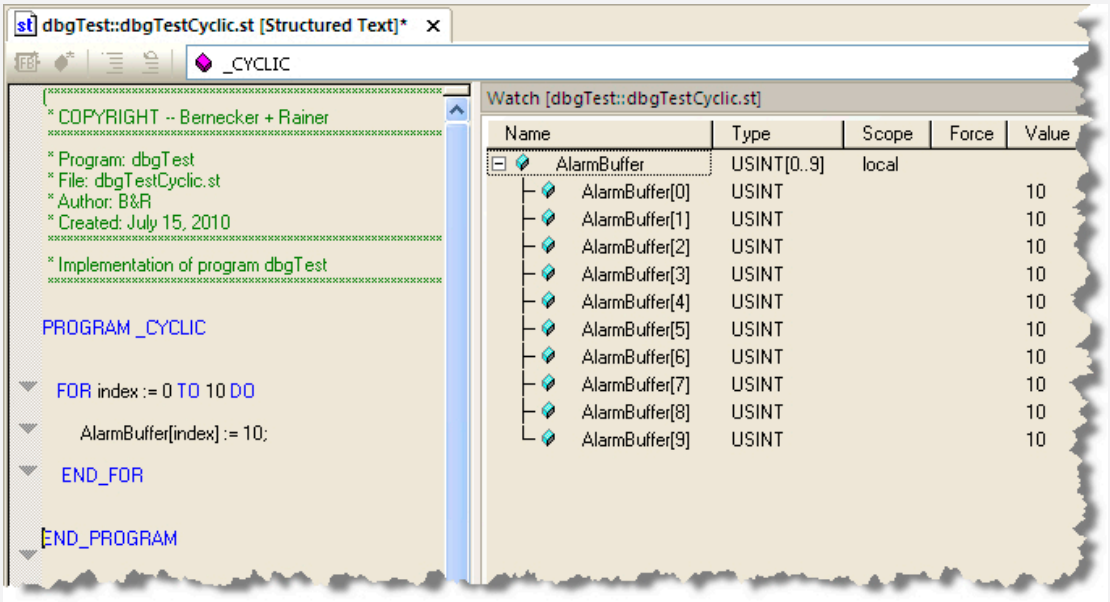


Figure 74: Monitor mode in the program editor

Set the breakpoints

- Move the cursor to the first line in the FOR loop.
- Set a breakpoint using the <**Debug**> / <**Toggle Breakpoint**> menu item or by pressing the <**F9**> key.

Enable the debugger

The debugger and any set breakpoints must be enabled from the menu.

If the debugger hits a breakpoint, then the active line is indicated by a yellow marker.

Reaching a breakpoint stops the entire application running on the target system.

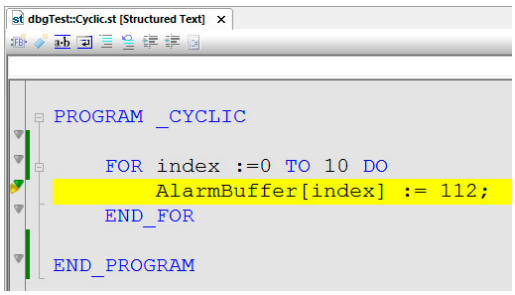


Figure 75: Active line in the debugger

"Step into" debugging

First, the elements of the "**AlarmBuffer**" array are manually changed to the value 0 in the Watch window.

The "**Step Into**" command or <**F11**> can be used to execute the program code one line at a time. The active line is displayed with the yellow arrow.

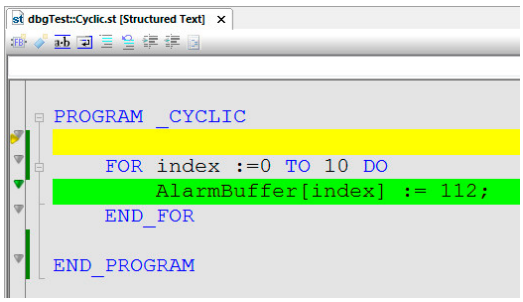


Figure 76: Active step marked yellow

If the <**F11**> key is pressed several times, each iteration of the loop causes the value of an element of the array to be changed. Continue through each step with the <**F11**> key until the new value has been assigned to the last element of the array.

Name	Type	Value
AlarmBuffer	USINT[0..9]	
AlarmBuffer[0]	USINT	112
AlarmBuffer[1]	USINT	112
AlarmBuffer[2]	USINT	112
AlarmBuffer[3]	USINT	112
AlarmBuffer[4]	USINT	112
AlarmBuffer[5]	USINT	0
AlarmBuffer[6]	USINT	0
AlarmBuffer[7]	USINT	0
AlarmBuffer[8]	USINT	0
AlarmBuffer[9]	USINT	0

Figure 77: Step-by-step writing to the variables

In this case, the "index" variable receives the value 9, which also corresponds to the upper limit of the array ([0..9]).

If continuing step by step with <**F11**>, the loop will iterate once more with a index outside of the array.

This type of error can be detected by the IEC Check library.

- See "IEC Check library" on page 48.

Diagnostics and service \ Diagnostics tool \ Debugger

5.2.6 IEC Check library

The IEC Check library contains functions for checking division operations, range violations, proper array access as well as reading from or writing to memory locations.

The corresponding checking function is called by the program (supported IEC 61131-3 languages or Automation Basic) before each of these operations is carried out.

With the IEC Check library, the user can can use a dynamic variable to determine what should happen when divided by zero, out of range errors or illegal memory access occurs.

Programming \ Libraries \ IEC Check library

5.2.7 Evaluating event IDs, status variables and return values

Return values and status values of functions and function blocks must be evaluated in the user program. The terms "status" and "event IDs" should be understood as synonyms in this context.

The following example shows a function block being called. This function returns a status that can be used to determine whether an error has occurred during the call. A list of the status values or the event IDs are documented in the description of the library used.

```
2:  (*Read information from an AR logger user module*)
    Logger.AsARLogGetInfo_0.enable := 1;
    Logger.AsARLogGetInfo_0.pName :=  ADR('usrlog');
    Logger.AsARLogGetInfo_0;  (*Call the Functionblock*)

    (*AsArLogGetInfo successful*)
    IF Logger.AsARLogGetInfo_0.status = 0 THEN
        Logger.Step := 0;
    ELSIF Logger.AsARLogGetInfo_0.status = ERR_FUB_BUSY THEN
        (*Busy*)
    ELSE  (*Go to Error Step*)
        Logger.Step := 10;
    END_IF
```

Figure 78: Status evaluation of function blocks

5.3 Using variables in the programs

The proper usage of variables in the different programs that are in the Logical View can be checked by creating a cross-reference list or explicitly searching for a known variable name.

5.3.1 Generating cross-reference list

The cross-reference list indicates which process variables, functions and function blocks can be used at which point in the project.

The cross-reference list is optional and can be generated when the project is compiled (built); the results are then displayed in the output window under the "**Cross Reference**" tab.

To generate a cross-reference list, you must activate it using the following menu option: **<Project> / <Settings>**. Alternatively, the cross-reference list is generated via the menu item **<Project> / <Build cross-reference>**.

Frequent search tasks can be handled easily with the help of the cross reference list.

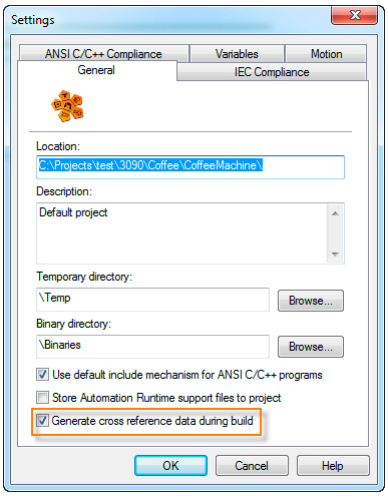


Figure 79: Enabling the creation of cross-reference list during build

Exercise: Generate a cross reference list

Create a cross-reference list for the open project.

1) Enable the cross-reference list in the project settings.
2) Building the project
3) Check the cross-reference list in the output window.

You can analyze the cross-references for variables and their attributes in the output window.

If a variable is selected on the left side, its usage and the type of access will be displayed in the source code or in the I/O allocation on the right side.

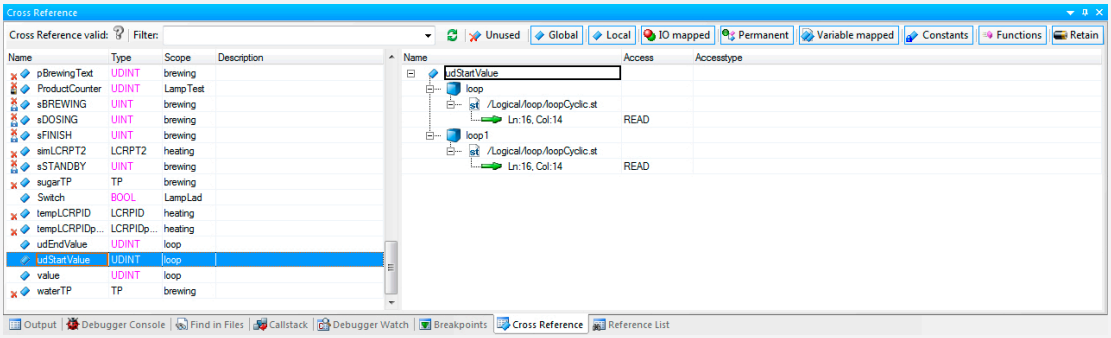


Figure 80: Display in the cross reference list

Project management \ Workspace

- General project settings
- Output window \ Cross reference
- Main menu \ Menu - Project

5.3.2 Searching in files

If you are looking for matches of names, product IDs or comments, you can search in the project files.

To search for a variable, use **<Edit> / <Find and Replace – Find in Files>** or press **<CTRL> + <Shift> + <F>**.

The search term is entered in the dialog box. The result of the search is displayed in the output window in the "Find in Files" tab.

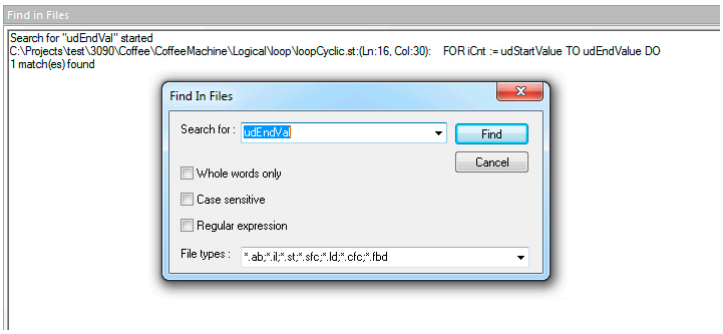


Figure 81: Searching in files

Double-clicking on a result in the output window opens the respective source file and places the cursor at the corresponding position.

Project management \ Workspace \ Find in files

5.4 Source file comparison

With the Automation Studio source file comparison, it is possible to compare individual files, folder elements (programs, libraries, packages and data objects) or entire projects.

During local source file comparison, an element from the currently open project or the entire project is always compared with another element or project on a data storage device.

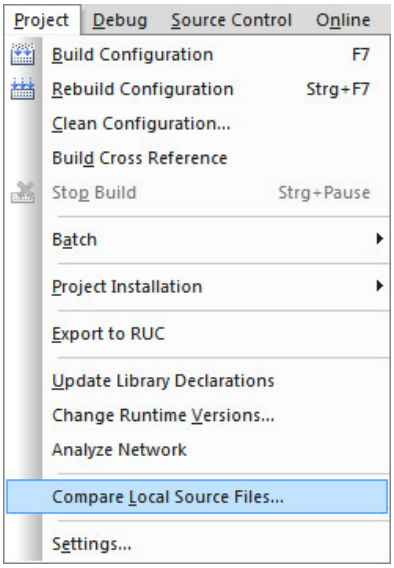


Figure 82: Opening the source file comparison

Overview comparison

After opening the source file comparison, an overview comparison is performed first of all. The structure of both projects is compared. For example, only different objects are displayed via filters.

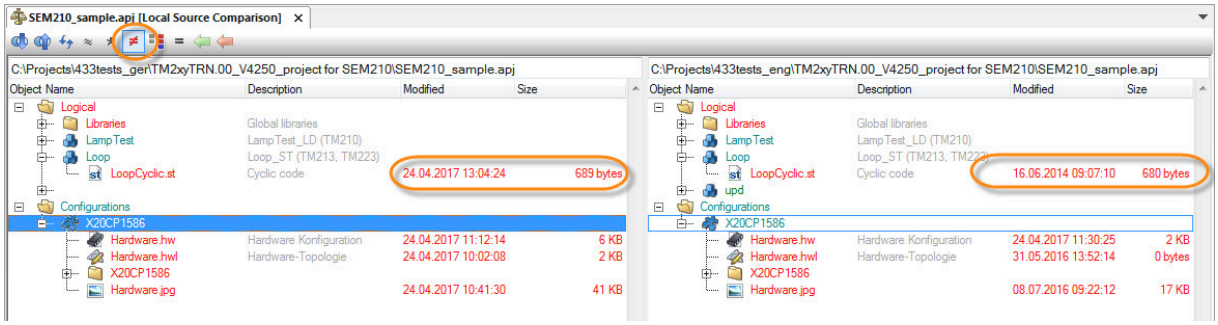


Figure 83: Overview comparison of two projects - only the differences are displayed.

Detailed comparison

By double-clicking on an object, both source files are opened in a detailed comparison. The editor highlights the differences in the corresponding text-based or tabular representation.

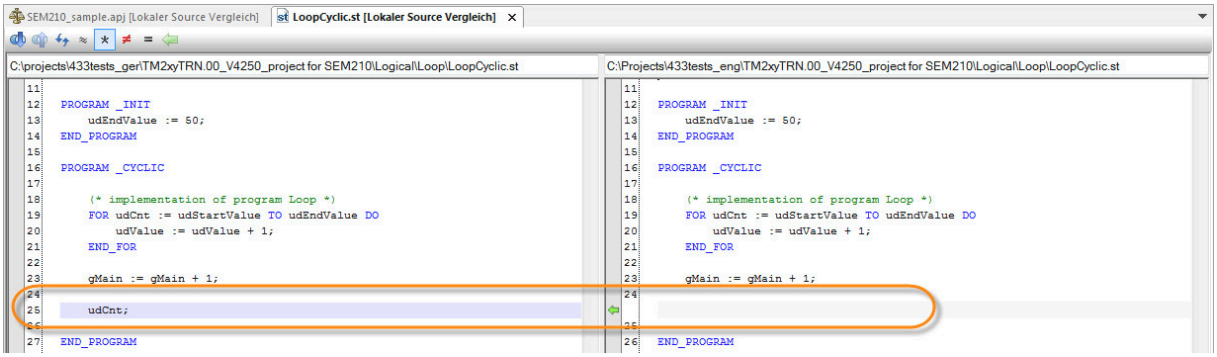


Figure 84: Structured text program in detailed comparison

Project management \ Automation Studio source file comparison

- Overview comparison
- Detailed comparison

5.5 Source file comparison on the target system

With source file comparison on the target system, it is possible to compare the source files of a local project with the files on a target system. In order to do this, source file comparison on the target system must have been enabled the last time the configuration was transferred.

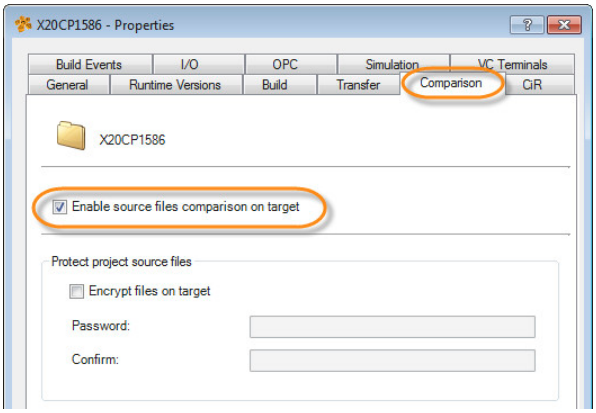


Figure 85: Properties dialog box is opened using the shortcut menu in the Configuration View

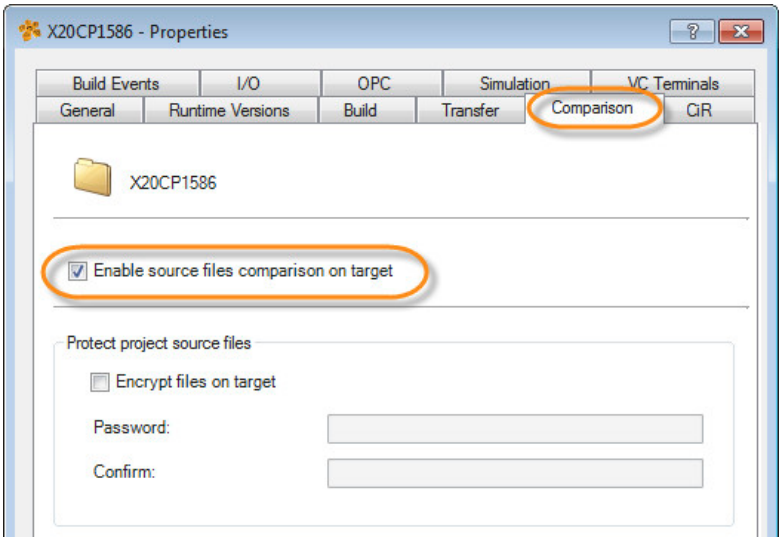


Figure 86: Enabling source file comparison for the target system in the "Comparison" tab

This function is enabled in the properties of the respective configuration in the "Comparison" tab.

As an alternative, the source files can also be encrypted on the target system.

After transferring the project to the controller, source file comparison is available for the target system

A comparison of source files in the Automation Studio project with the target system is opened via the shortcut menu for a source file.

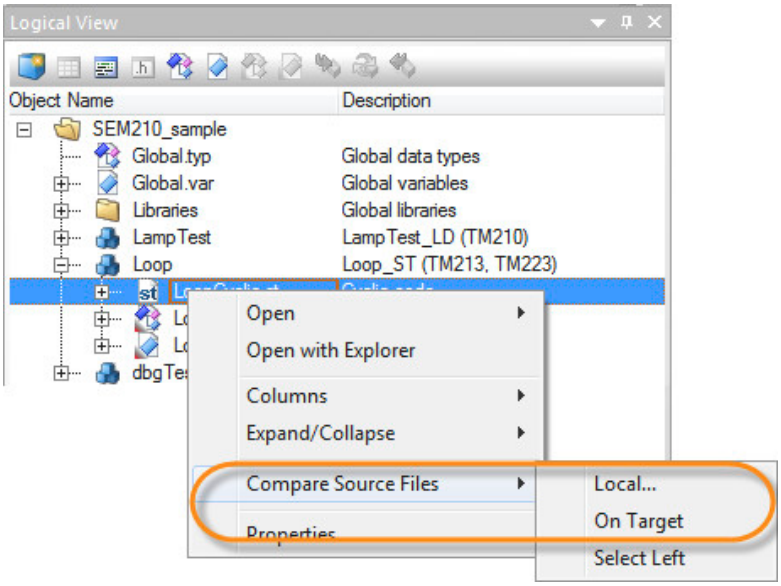


Figure 87: Opening the source file comparison for the target system

The source files on the target system and in the project are displayed parallel to each other. Differences are marked.

Changes can be filtered, for example, using the toolbar and transferred to the project.

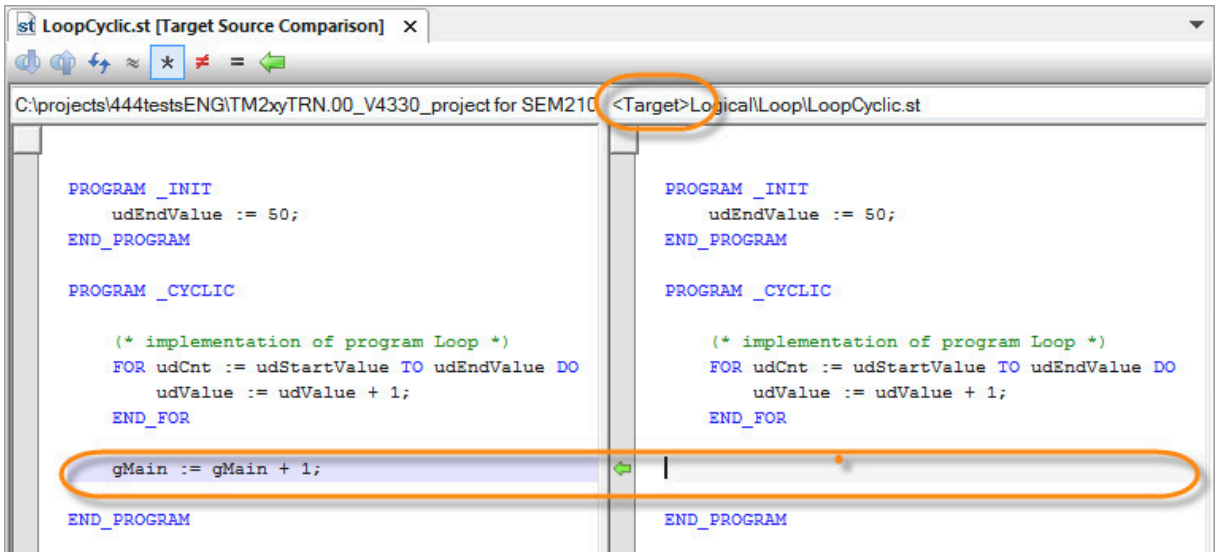


Figure 88: Displaying source files in the project and on the target system

Project management \ Automation Studio source code file comparison \ Source code file comparison on the target system

6 Making preparations for servicing

It is necessary during the configuration, commissioning and testing of the application to prepare the machine or system for service activities that may occur later.

6.1 System Diagnostics Manager (SDM)

With the help of the System Diagnostics Manager (SDM), a diagnosis of the control can be carried out via a standard web browser.

The only requirement for these diagnostics is an Ethernet connection to the controller.

The following functions are offered, among others:

- General system overview
- Showing and saving logger files
- Overview of installed software objects
- Hardware modules and I/O status
- Motion control diagnostics
- Creating system dumps

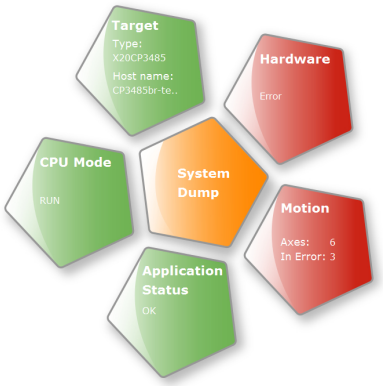


Figure 89: SDM startup screen

Diagnostics and service \ Diagnostics tools \ System Diagnostics Manager

6.1.1 Enabling the SDM

SDM is automatically activated when creating a new project.

The SDM configuration is opened in Physical View using the **<Configuration>** option in the controller's shortcut menu.

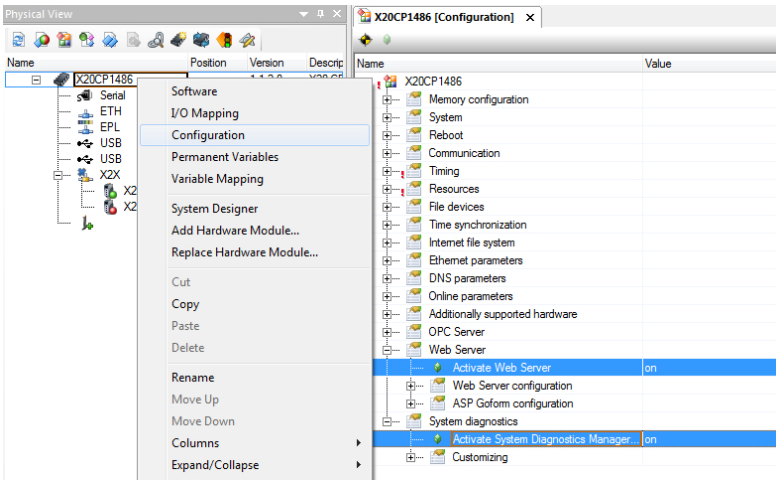


Figure 90: Opening the configuration, SDM and web server settings

System Diagnostics Manager requires the web server service, which is also a component of Automation Runtime.

Exercise: Check the System Diagnostics Manager configuration

Check whether the web server and System Diagnostics Manager are enabled in the Automation Runtime configuration.

1) Open the Automation Runtime configuration from the controller's shortcut menu in the Physical View.
2) Enable the Web Server and System Diagnostics components

If a new Automation Studio project or a new configuration is created in the Configuration View, the System Diagnostics Manager is automatically activated.

6.1.2 Accessing the SDM

The connection to the SDM is made via the controller's IP address or hostname. The SDM is accessed using a web browser via the link "**http://ip-address/SDM**".

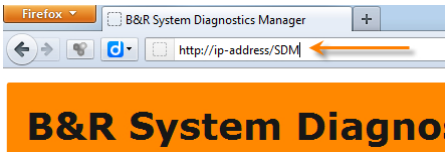
Step 1 - Connect PC to controller	Step 2 - View SDM in web browser
	

Figure 91: Connecting PC to controller via network cable

Figure 92: Accessing the SDM via web browser

Table 7: Overview – "Establishing a connection to the SDM"

The target system IP address is checked in the controller's Ethernet configuration. If there is already an active connection to the controller, SDM can be opened using the **<Extras> / <System Diagnostics Manager>** menu option.

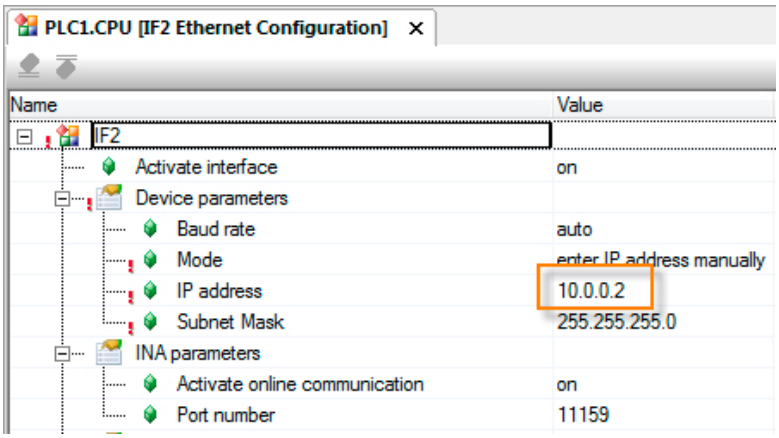


Figure 93: Check the network settings in the controller's Ethernet configuration

Diagnostics and service \ Diagnostics tools \ System Diagnostics Manager

Exercise: Access SDM using a web browser

Enter the URL for accessing the SDM: e.g. http://10.0.0.2/SDM

1) Start the web browser
2) Enter the URL for accessing the SDM.

Exercise: Evaluate Logger with SDM and in Automation Studio

Assumption: the system has booted into SERVICE mode for no apparent reason. Unfortunately, Automation Studio is not available on site. The logger file "**$arlogsys**" should be read out using the SDM and then opened in the Automation Studio Logger.

1) Establish a connection to the SDM and change to the "Logger" page
2) Upload the "$arlogsys" module.
3) Save the file using the .br file extension.
4) Open the Logger in Automation Studio.
5) Load the Logger file with the .br file extension.
6) Highlight the error entry in the Logger and press the **<F1>** key to receive detailed information

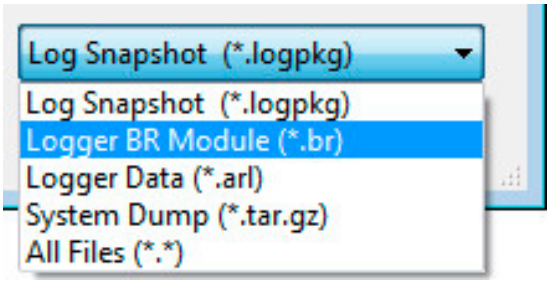


Figure 94: Select the file type with the .br file extension in Automation Studio

If a system dump was generated in the System Diagnostics Manager with the option "Parameters + Data Files", then it can be opened and displayed directly with the Automation Studio Logger.

Entries are shown in the SDM Logger without additional supplementary information. This can only be displayed in Automation Studio.

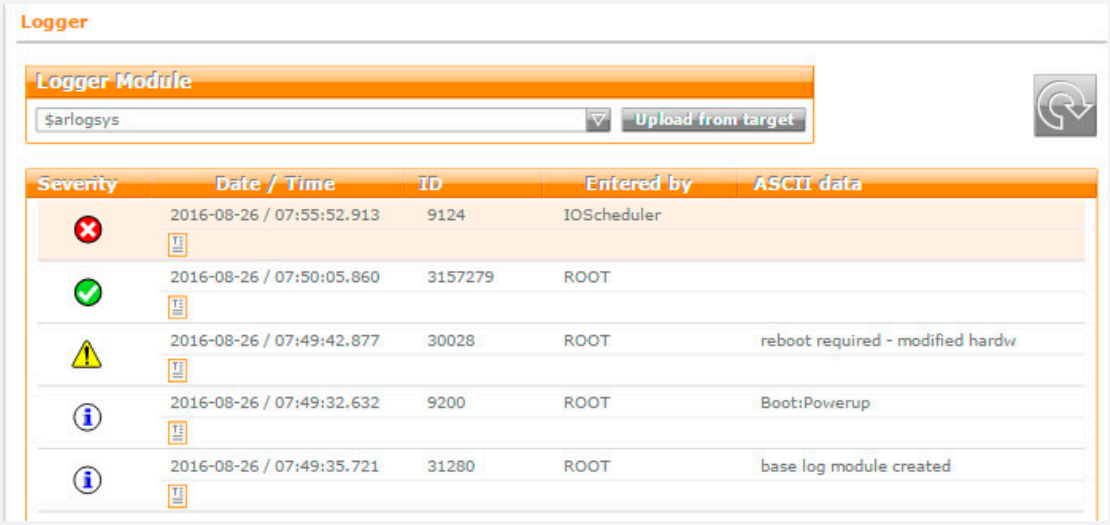


Figure 95: Logger entries in System Diagnostics Manager

The online data must be deselected in the Automation Studio Logger; otherwise, the online entries are displayed with the entries loaded from the file.

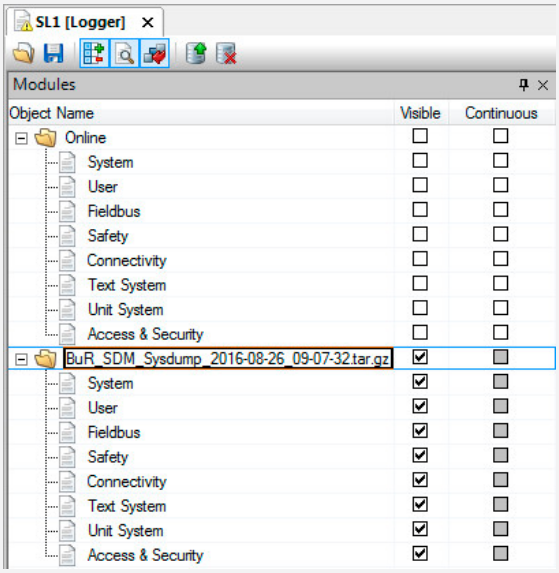


Figure 96: Disabling the online Logger entries

6.2 Query the status of the battery

Depending on the used target system, a battery is used for data buffering. Further details about this are available in the data sheet of the used controller.

The backup battery for the real-time clock and the nonvolatile variables (retain, permanent) being used in the application can be monitored from the application itself.

Figure 97: "4A0006.00-000" lithium battery

Note the replacement of the battery in the service instructions for the machine / system. The life of the battery is specified in the documentation for the controller being used.

The battery status can be queried in a number of ways:

- AsHW library in the application
- Control I/O mapping
- Online information dialog box
- System Diagnostics Manager

The I/O allocation is opened using the **<I/O Allocation>** option in the Physical View shortcut menu for the controller. The variable attached to the "BatteryStatusCPU" can be evaluated in the application program as needed.

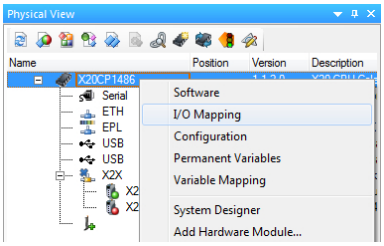


Figure 98: Opening the controller's I/O mapping

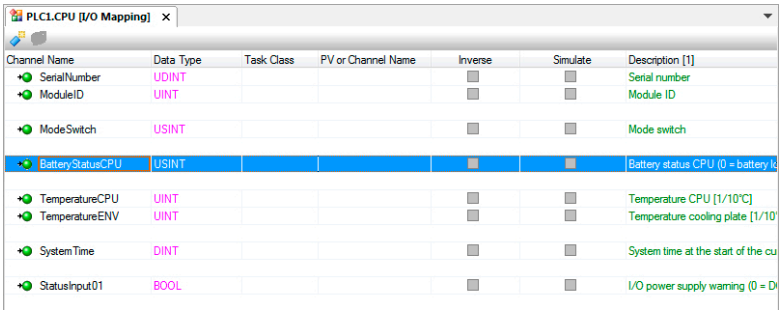


Figure 99: Querying the battery status in the controller's I/O mapping.

The values of the status data points in the I/O mapping of the controller can be monitored using monitor mode.

See: "Monitor and force I/O" on page 30

6.3 Runtime Utility Center service tool

Runtime Utility Center is a system tool that provides a range of utilities for diagnostics and service on B&R controllers. The installation program for Runtime Utility Center is included in the Automation Studio installation, or it can be downloaded separately from the B&R website.

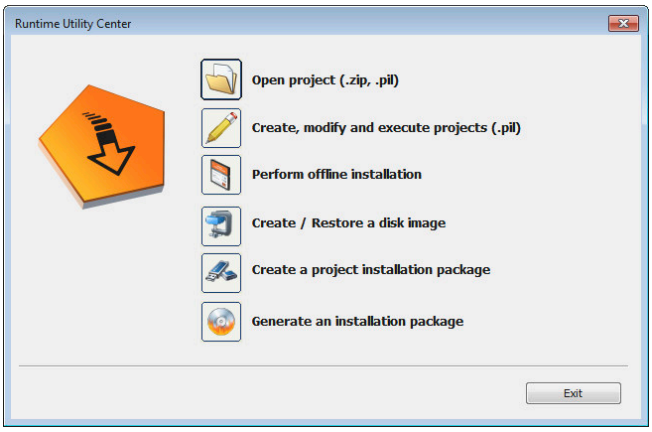


Figure 100: Runtime Utility Center - Start page

The most important functions are:

- Performing service functions via an online connection to the controller
- Variable functions for backing up and restoring process variables
- Creating individual instruction lists for testing and installation procedures
- Backing up and restoring a CompactFlash/CFast card
- Offline installation of a control project on a CompactFlash/ CFast card
- Creating project installation packages for USB installation
- Custom mode allows the creation of a user-defined user interface

The Runtime Utility Center includes a complete help documentation. The help documentation can be started via the Windows start menu by selecting "Runtime Utility Center Help". The following entries provide additional important information about using the Runtime Utility Center.

Runtime Utility Center \ Start page

Runtime Utility Center \ Operation \ The Workspace

Runtime Utility Center \ Operation \ Commands \ Establish connection, etc.

Runtime Utility Center \ Operation \ Commands \ PLC Info \ Logger

Downloading the Runtime Utility Center

The Runtime Utility Center is part of the "PVI Development Setup" and can be downloaded from the B&R website: **www.br-automation.com**→ Downloads → Download "PVI Development Setup"

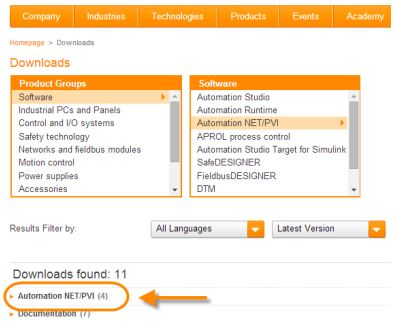


Figure 101: Downloads section: Filter "Software→Automation NET/PVI"

6.3.1 Runtime Utility Center packet

Create Runtime Utility Center export

The Runtime Utility Center export is started from the Project menu in Automation Studio. After the destination folder is selected and confirmed, the necessary data is exported as a *.zip file.

The export file can then be processed with the Runtime Utility Center.

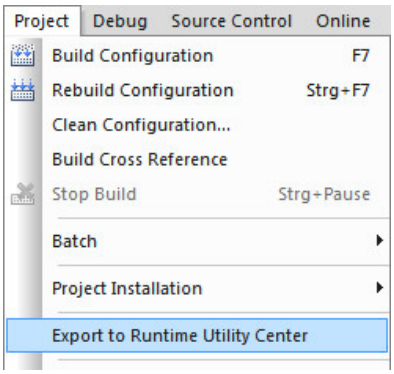


Figure 102: Runtime Utility Center (RUC) export in Automation Studio

Project management \ Project installation \ Performing project installation \ Export RUC

Loading Runtime Utility Center export package

Select "Open project (*.zip, *.pil)" to load the Runtime Utility Center export package. Then the following functions are available:

Figure 103: Runtime Utility Center start page with RUC export file already loaded

- Perform offline installation
  This function can be used to perform an initial transfer to a CompactFlash/CFast card.
- Generate project installation package
  This function can be used to create a project installation package, e.g. for USB installation.

Runtime Utility Center \ Creating a list / data medium \ Project installation

- Offline installation
- Create a project installation package

Exercise: Reinstallation of the controller using a USB flash drive

The controller should be reinstalled without using Automation Studio. After a build of the project, a Runtime Utility Center export is created in Automation Studio.

The export file is opened in the Runtime Utility Center. A project installation package is then created for the USB flash drive.

1) Create Runtime Utility Center export
2) Start Runtime Utility Center
3) Open export file (*.zip)
4) Generate project installation package

6.3.2 Backing up and restoring variable values

One function of Runtime Utility Center is to load variable values from the controller and to restore them at a later point in time.

Exercise: Save variable values

Due to mechanical damage to the system, the CPU must be replaced. To prevent e.g. recipe variables or other process data from being lost, the necessary information on the CPU can be uploaded using Runtime Utility Center and then transferred later to the new CPU.

Create a Runtime Utility Center list that saves the values of variables in the "Loop" task.

1) Open Runtime Utility Center from Automation Studio.
2) Execute menu option "**Create, edit and execute projects (*.pil)**"

   A new project is created.
3) Insert the command for establishing a connection by selecting **<Command> / <Connection>**

   The IP address of the target system needs to be entered in the connection settings.
4) Insert the command for loading the variable list by selecting **<Command> / <Process variable functions> / <Variable list>**

   Only the variables in the "**Loop**" task are being backed up in this example. The list can be stored to any directory.
5) Execute the functions with **<F5>**.

Diagnostics and service \ Service tools \ Runtime Utility Center \ Operation \ Commands \ List functions

Diagnostics and service \ Service tools \ Runtime Utility Center \ Operation \ Commands \ Establish connection, wait for reconnection

Diagnostics and service \ Service tools \ Runtime Utility Center \ Operation \ Menus \ Start

The variable values from the "Loop" task are backed up in the specified file.

Exercise: Restore variable values

The variable values backed up in the last task now need to be transferred to the controller using Runtime Utility Center.

A Runtime Utility Center list is to be created that restores the values of variables.

1) Open Runtime Utility Center in Automation Studio Create a new list by selecting <**File**> / <**New**> from the menu
2) Insert the command for establishing a connection by selecting <**Commands**> / <**Connection**>
3) The IP address of the target system needs to be entered in the connection settings.
4) Insert the command for writing the variable list by selecting <**Command**> / <**Process variable functions**> / <**Transfer variable list to PLC**>

   The variable list saved in the last task can be selected in the <**Browse**> dialog box.

5) Execute the functions with <**F5**>.

The variable values saved in the file are written to the corresponding variables in the "**Loop**" task.

If every variable from every task is backed up and then transferred back to the controller, be aware that writing to variables sequentially can cause unexpected behavior in the process.

If this situation is still necessary, it is recommended to put the controller in SERVICE mode first.

7 Summary

Automation Studio offers several different tools for localizing problems and errors.

They need to be used sensibly in combination with analytical thinking. To be able to use these diagnostic tools effectively, it is necessary to get an overview of the situation, clarify the general conditions and examine these conditions from a certain distance.

Figure 104: Diagnostics

Only then can the circumstances be cleared up and analyzed in detail. A comprehensive overview of potential errors can be achieved by excluding and reducing the number of possible error sources, making it considerably easier to correct any errors that may still occur.

Offered by the Automation Academy

The Automation Academy provides targeted training courses for our customers as well as our own employees.
At the Automation Academy, you'll develop the skills you need in no time!
Our seminars make it possible for you to improve your knowledge in the field of automation engineering. Once completed, you will be in a position to implement efficient automation solutions using B&R technology. This will make it possible for you to secure a decisive competitive edge by allowing you and your company to react faster to constantly changing market demands.

Seminars

Quality and relevance are essential components of our seminars. The pace of a specific seminar is based strictly on the experience that course participants bring with them and tailored to the requirements they face. A combination of group work and self-study provides the high level of flexibility needed to maximize the learning experience.
Each seminar is taught by one of our highly skilled and experienced trainers.

Training modules

Our training modules provide the basis for learning both at seminars as well as for self-study. These compact modules rely on a consistent didactic concept. Their bottom-up structure allows complex, interrelated topics to be learned efficiently and effectively. They serve as the best possible companion to our extensive help system. The training modules are available as downloads and can be ordered as printed versions.

Topic categories:

- ⇨ Control technology
- ⇨ Motion control
- ⇨ Safety technology
- ⇨ HMI
- ⇨ Process control
- ⇨ Diagnostics and service
- ⇨ POWERLINK and openSAFETY

ETA system

The ETA system provides realistic constructions for training, education and laboratory use. Two different basic mechanical constructions can be selected. The ETA light system offers a high degree of mobility, saves space and is well-suited for lab work. The ETA standard system has a sturdy mechanical structure and includes pre-wired sensors and actuators.

Find out more!

Would you like additional training? Are you interested in finding out what the B&R Automation Academy has to offer? You've come to the right place.
Detailed information can be found under the following link:
www.br-automation.com/academy