TM213

Automation Runtime

Requirements		
Training modules	TM210 - Working with Automation Studio	
Software	Automation Studio 4.6 Automation Runtime 4.61 and later	
Hardware	X20 controller and X20 I/O modules ETA210 or ETAL210 + ETAL690 www.br-automation.com/eta-system	

Table of contents

1 Introduction....4
1.1 Learning objectives....4
1.2 Symbols and safety notices....5
2 Automation Runtime....6
2.1 Automation Runtime structure....7
2.2 Automation Runtime diagnostics capability....9
2.3 Connectivity, user management and security....10
3 Installation and commissioning....11
3.1 Changing and updating Automation Runtime....11
3.2 Configuration ID and partitioning....12
3.3 Project installation....13
4 Memory management....16
4.1 Logical partitioning of the flash memory....16
4.2 Used RAM....17
4.3 Tools for determining memory information....17
4.4 Global and local variables....20
5 Runtime performance....24
5.1 Start Automation Runtime....24
5.2 Program initialization....28
5.3 Executing cyclical programs....29
5.4 Cycle time and tolerance....33
5.5 Settings when transferring programs....36
6 I/O handling....39
6.1 I/O configuration and I/O mapping....40
6.2 Handling I/O images....41
6.3 Error handling for I/O modules....46
6.4 reACTION Technology....47
7 Summary....49

1 Introduction

Automation Runtime is a deterministic real-time operating system that serves as the software platform for B&R's entire product range. Automation Runtime provides services for freely configuring and troubleshooting the target system and for executing programs.

In addition, Automation Runtime features a modular structure, configurability and the ability to quickly execute applications repeatedly within a precise time frame. This makes it possible to achieve optimal product quantities while guaranteeing quality and precision at runtime.

Figure 1: Automation Runtime: A software platform for the entire B&R product range

This training module provides an overview of Automation Runtime and its features and configuration options.

1.1 Learning objectives

This training module uses selected examples illustrating typical applications to help participants learn how to configure Automation Runtime for their own applications in Automation Studio.

- Participants will learn what demands are placed on a real-time operating system in the field of automation.
- Participants will learn about the features and functionalities available in Automation Runtime.
- Participants will get an overview of integrated server and client functions and diagnostic options.
- Participants will learn how a uniform runtime system can benefit integrated automation solutions.
- Participants will learn how Automation Runtime handles memory management and variable scope and how variables are stored in memory.
- Participants will learn about the startup and runtime behavior of B&R controllers.
- Participants will learn how Automation Runtime I/O management works.
- Participants will learn about the interrelationships involved in multitasking.
- Participants will learn about Automation Runtime's configuration options.

1.2 Symbols and safety notices

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

2 Automation Runtime

The fundamental idea of integrated automation and the free scalability of automation solutions results in challenges for the configuration tool as well as the runtime system it's based on.

Requirements for Automation Studio and Automation Runtime

Automation Studio is the configuration environment used for B&R automation components. This includes controllers, motion control components, safety modules and HMI applications. Clearly organized project structuring options and the ability to manage multiple configurations ensure that teams can work together efficiently and all machine variants can be represented in a single project.
Users can choose from a wide range of programming languages, diagnostic tools and editors to assist them at every stage of engineering. Efficient work is made possible by using the standard libraries supplied by B&R and the IEC programming languages integrated in the system.

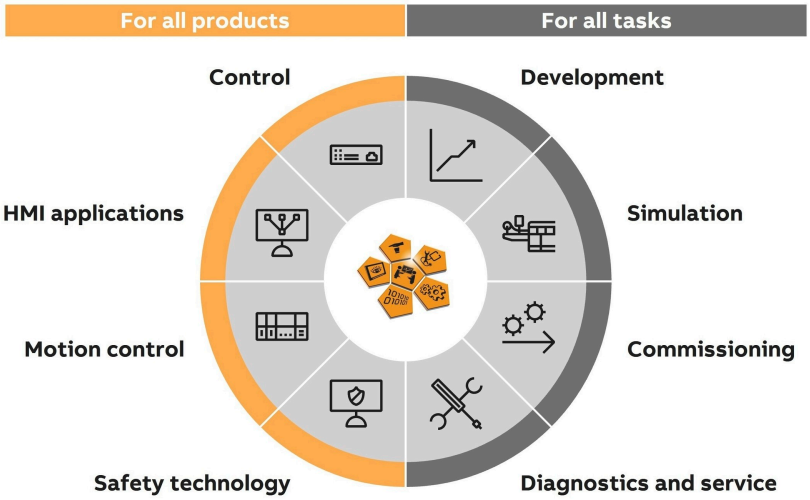


Figure 2: Automation Studio: One engineering tool for the machine's entire lifecycle

Automation Runtime characteristics

The characteristics of Automation Runtime result from the challenges faced by the configuration tool and the required runtime system functions. Automation Runtime is fully integrated into B&R target systems. It supports all hardware platforms and makes application creation hardware-independent.

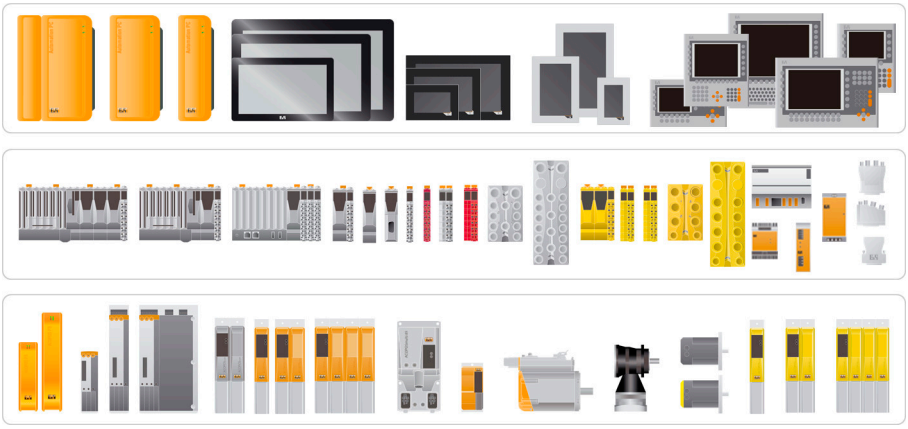


Figure 3: Automation Runtime - The platform for a scalable system

Automation Runtime allows application programs to access I/O systems, interfaces, fieldbuses, networks and storage devices. Comprehensive configuration options for timing as well as client and server applications offer the flexibility required for modern control systems.

Automation Runtime provides a number of important features:

- Runs on all B&R target systems
- Makes the application hardware-independent
- Guarantees deterministic behavior with a cyclic runtime system
- Allows the configuration of 8 different task classes and cycle times
- Guarantees a response to timing violations
- Provides configurable tolerance limits for all task classes
- Offers extensive function libraries in accordance with IEC 61131-3
- Provides access to networks and bus systems
- Contains integrated client and service interfaces
- Offers an OPC UA client and server interface as well as user management
- Provides comprehensive diagnostic functions

Real-time operating system \ Method of operation

2.1 Automation Runtime structure

Automation Runtime provides the user with a deterministic, hardware-independent, multitasking tool for creating applications. It manages hardware and software resources and offers complete diagnostics.
The IEC library functions in Automation Runtime make programming faster and easier while helping to avoid errors.

Automation Runtime meets the highest demands in terms of deterministic behavior and speed.
To take advantage of this performance advantage in the application, an abstraction layer is put over the real-time OS. This ensures the user that no adjustments will need to be made to the application if a different operating system is used. The uniform programming interface always remains the same.

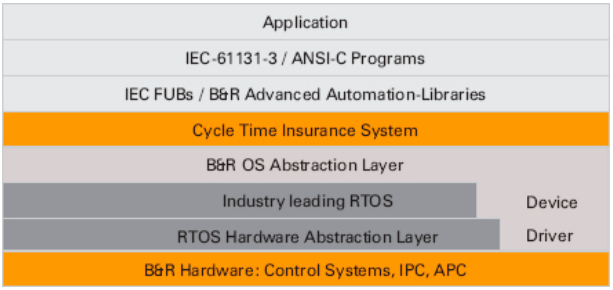


Figure 4: Automation Runtime architecture

Programs, tasks and task class

A program that is assigned to a task class and executed on the target system is also called a task.
Tasks are executed cyclically in a task class.
Cycle time and priority are determined via the task class.

Real-time operating system

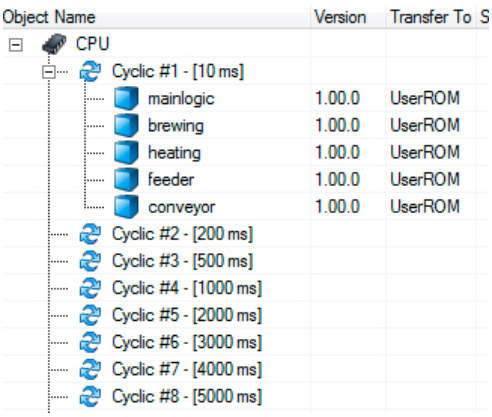


Figure 5: Task class system with cyclic tasks in task class #1

2.1.1 Automation Runtime target systems

Automation Runtime can run on many different hardware platforms. For X20 controllers, Power Panels and Automation PCs, PC-based hardware platforms are used.

Automation Runtime Embedded

Automation Runtime Embedded can be implemented on all target systems where Automation Runtime is run as an independent operating system. The X20 operating system, Power Panels and PC-based hardware platforms are suitable for Automation Runtime Embedded.

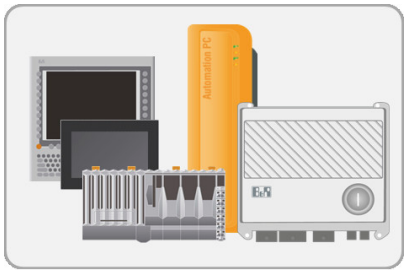


Figure 6: Automation Runtime on all types of controllers

Automation Runtime Simulation - ARsim

ARsim is a Windows-32-based Automation Runtime system that, as a demo PLC, is not real-time capable, but essentially corresponds to the functionality of all other target systems. ARsim can be used to fully simulate all controllers, HMI applications and drives.

Figure 7: Automation Runtime Simulation on the PC

In Automation Studio, you can switch between real hardware configuration and simulation. You can switch to simulation under "Online" \ "Activate simulation". Automation Runtime Simulation is started automatically and a connection is established. Tests on the PC are made possible quickly and easily due to seamless integration of the simulation.

Figure 8: "Activate Simulation" button

Project management \ Simulation \ ARsim

Automation Runtime Windows - ARwin

ARwin is a target system capable of running on a Windows operating system. A real-time operating system imposes itself over the Windows operating system, which assumes complete control of computer resources. The host operating system itself is handled as a low-priority task on the real-time operating system.

Figure 9: Automation Runtime - real-time applications on Windows

B&R Hypervisor

With B&R Hypervisor, it is possible to run Automation Runtime on a target system parallel to Linux or Windows. With B&R Hypervisor, it is possible to partition the hardware resources on a system. The CPUs, memory areas and peripheral devices can then be assigned to the running operating systems. Each operating system is logically separated from the others at runtime.

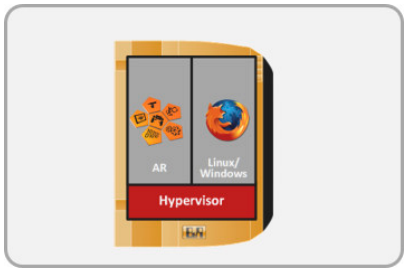


Figure 10: B&R Hypervisor - Multiple operating systems on one Automation PC

Real-time operating system \ Target systems \ Target systems - SG4

- ARsim
- ARwin
- B&R Hypervisor

2.1.2 Server and client functions

Numerous client and server functions are available in Automation Runtime. Different network services and protocols are supported.

Open the configuration entries from the shortcut menu for the CPU and the Ethernet interface. Every protocol has a standalone configuration entry that is used to enable and configure the corresponding protocol.
Many of the functions offered can be diagnosed and configured at runtime using libraries.

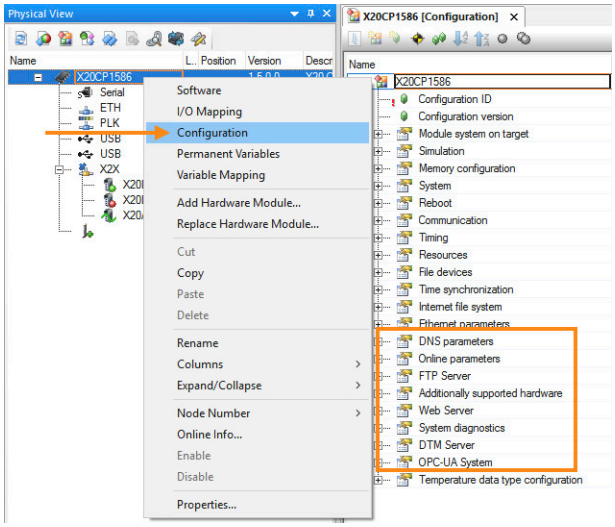


Figure 11: Open CPU and Ethernet configuration via the shortcut menu

Communication \ Fieldbus Systems
Communication \ Ethernet \ AR configuration \ Interface configuration (SG4)
Programming \ Editors \ Configuration editors \ Hardware configuration \ CPU configuration \ SG4
Programming \ Libraries

- Configuration, system information and runtime control
- Communication

2.2 Automation Runtime diagnostics capability

Independent of the implemented hardware platform, Automation Runtime supports many diagnostic tools that provide information about the current system state.
The use of the integrated diagnostic tools is documented in Automation Help. For a list of diagnostic tools, see the help box below. For further explanations and exercises, see training module "TM223 - Automation Studio Diagnostics".

Diagnostics and service \ Diagnostic tool

- Status bar
- Information about the target system
- System Diagnostics Manager
- Monitor mode \ Online comparison
- Logger
- Profiler

Diagnostics and service \ I/O and network diagnostics
Programming \ Libraries \ Configuration, system information, runtime control \ ArProject

2.3 Connectivity, user management and security

The B&R solution is completely open for **seamless integration in existing networks** and allows direct integration in devices from other manufacturers. B&R components can be implemented as either fieldbus masters or slaves.
An OPC UA server and client enable the publication and exchange of process data. The user roles system helps with the limitation of read and write access. An encrypted connection is enabled using Transport Layer Security (TLS).

Support of devices from other manufacturers

Automation Studio allows fieldbuses and devices from other manufacturers to be directly integrated. Master and slave devices are added to Automation Studio in the Physical View or System Designer.

Communication \ Fieldbuses
Project management \ Hardware management \ 3rd-party devices / 3rd-party device manager

User management and security

Automation Studio supports configuration of user and role systems as well as and a certificate management system and management of SSL/TLS configurations. These configurations are managed in the "AccessANDSecurity" package in the Configuration View.
A **firewall** can be added to the "AccessAndSecurity" package via a Toolbox. It is used to restrict network access based on the sender or destination and the services used.
With the ANSL protocol, it is also possible to limit access to authenticated connections and thus secure the target system against unwanted access via ANSL. To configure a system for ANSL authentication, the ANSL connection must be assigned to a defined role.

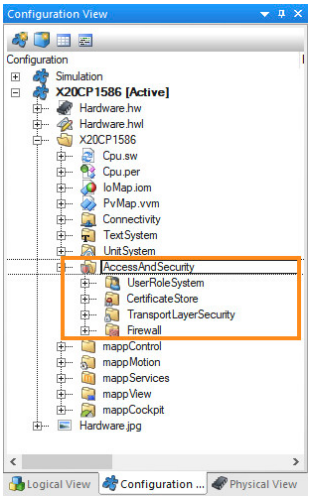


Figure 12: User management system in the Configuration View

Programming \ Access & Security \
- Transport Layer Security (TLS)
- User role system
- Firewall

Programming \ Editors \ Configuration editors \ Hardware configuration \ CPU configuration \ SG4 \ CPU properties - Online parameters

OPC UA server and client

Automation Studio supports the configuration of an OPC UA server. Read/write access for the individual OPC UA codes is managed using the user role system. OPC UA client functions are offered via a library.

Communication \ OPC UA
Programming \ Libraries \ Communication \ AsOpcUac
Automation software \ Sample projects \ Communication with OPC UA client function blocks

3 Installation and commissioning

Every target system starts Automation Runtime and loads the Automation Studio project from flash memory. As storage medium, a CompactFlash, CFast card or the integrated flash memory is used depending on which target system is used.
There are 3 ways to install Automation Runtime: Online installation, offline installation or USB installation.
Regardless of which method is selected, Automation Studio provides support during the installation process.
During initial installation, the flash memory of the controller is partitioned according to the requirements of the application.
When installing Automation Runtime, settings like the I/O configuration and the interface configuration are included.

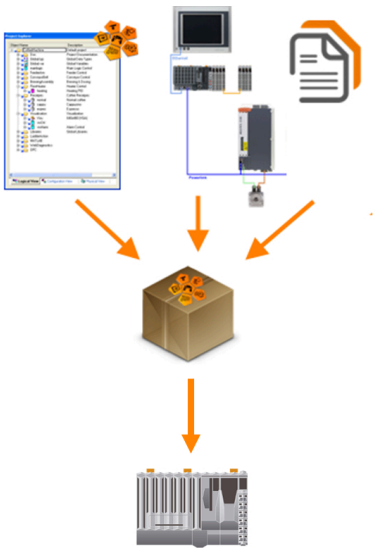


Figure 13: Installation and commissioning

Hardware \ X20 system \ X20 modules \ CPUs

- X20(c)CP1301, X20CP1381 and X20CP1382 \ Operating and connection elements \ Programming the system flash memory
- X20(c)CP158x and X20(c)CP358x \ Programming the system flash

Project management

- Project installation
- Simulation \ ARsim \ Creating an executable project structure

3.1 Changing and updating Automation Runtime

If new software versions are available for the runtime environment, then these software versions can be added in Automation Studio via the "Upgrade" dialog box.
The "Upgrade" dialog box is opened by selecting <Tools> / <Upgrades> from the menu.

The Automation Runtime version can be changed for the active configuration of the Automation Studio project. Individual software versions for the runtime environment can be modified by selecting <Project> / <Change Runtime Versions...> from the main menu.

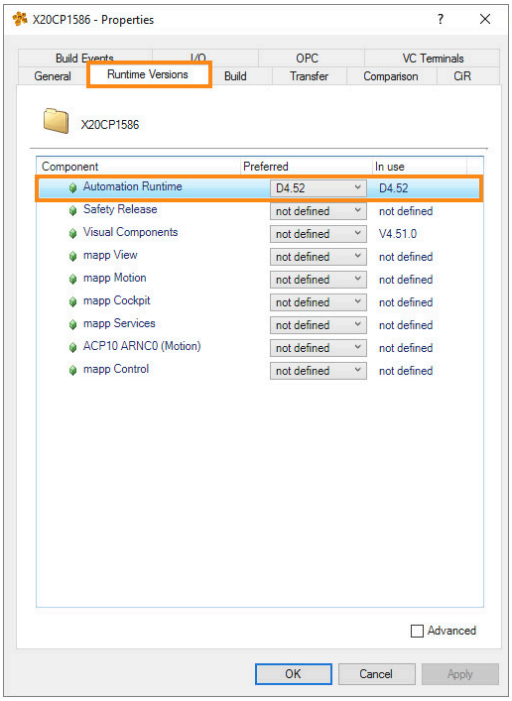


Figure 14: Changing runtime versions

> Changing the Technology Package version (mapp Motion, mapp Services, etc.) affects all configurations in the project, since the libraries linked with them are managed in the Logical View.

> Project management
> - Workspace \ Upgrades
> - Changing runtime version
>
> Real-time operating system \ Version overview

Upgrading from Automation Studio

If Automation Runtime is already installed on the target system, the project can be transferred along with the new Automation Runtime version. Alternatively, the creation of installation media, such as a CompactFlash, CFast and USB flash drive, is supported.

Further information:

- 3.3 "Project installation" on page 13

When transferring the project to the target system, any Automation Runtime version conflicts are detected and displayed in a dialog box.

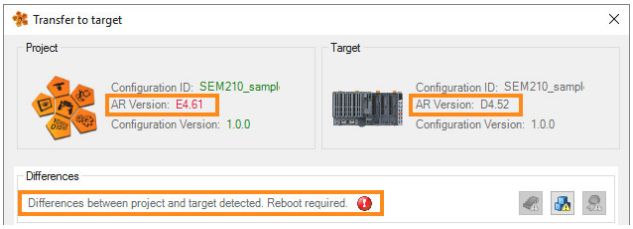


Figure 15: Display of version differences in the transfer dialog box; restart required

Upgrading without Automation Studio

If an upgrade with Automation Studio is not possible, a complete installation image can be created with Runtime Utility Center. For example, it can then be transferred to a USB flash drive.

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

3.2 Configuration ID and partitioning

Before project installation, the settings for the configuration ID and the partitioning of the flash memory are checked in the project. In addition, a configuration version is defined. The settings are opened via the shortcut menu for the CPU.

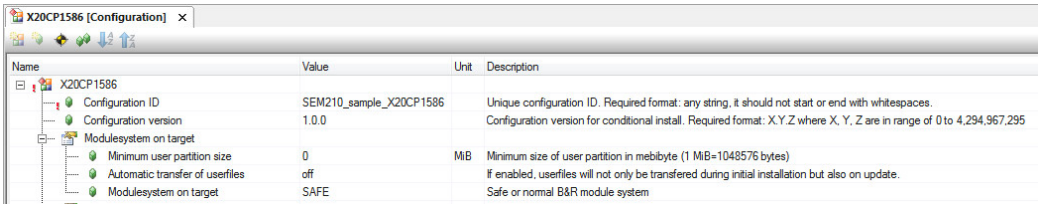


Figure 16: CPU settings: Settings for configuration ID and partitioning

Configuration ID

With Automation Runtime 4.25 and later, a unique configuration ID is assigned to each configuration in the Automation Studio project. The configuration ID serves as a unique identifier of the project and is preset to "<AS Project name>_<Configuration name>". It is necessary to assign a unique configuration ID in the project. This allows the system to differentiate between an update transfer (same ID) or initial transfer (different ID) during project installation.

Figure 17: Comparison of the configuration ID in the transfer dialog box

Partitioning

The flash memory of a controller is organized as a file system. Depending on the selected partition option, a distinction is made between the normal and secure file system.
For the **normal file system**, a partition is created in the flash memory, on which Automation Runtime and the application are saved.
For the **secure file system** on the other hand, Automation Runtime and the application are stored on different partitions. The partition sizes for Automation Runtime and the application are calculated automatically.[1]
A **user partition** can be added to the normal and secure file system. On this partition, the user can save things like recipe data at runtime. The memory size is manually determined for the user partition.

Programming \ Editors \ Configuration editors \ Hardware configuration \ CPU configuration \ SG4
Project management \ Project installation

- Performing project installation \ Settings \ System configuration settings
- FAQ

3.3 Project installation

It is possible to transfer Automation Runtime over an online connection. In order to do this, the online interface must be configured correctly and be in the enabled operating state, see 5.1 "Start Automation Runtime" on page 24. Alternatively, the Automation Studio project, including Automation Runtime, can be transferred via offline installation and the project installation package.

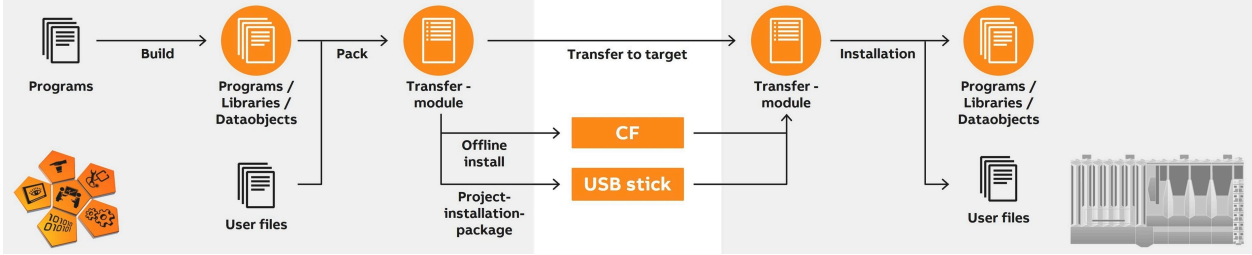


Figure 18: Schematic diagram of project installation options

1 The partition sizes for Data1 and Data2 are set at a ratio of 2:1. This is necessary so that sufficient storage space is available for the transfer module during project installation.

Establishing a connection

A connection to the controller is established using the "Browse for target system" feature in Automation Studio. This function searches the network for B&R controllers. The connection settings of the controllers can be temporarily changed in the search dialog box.

Programming \ Building and transferring projects \ Establishing a connection to the target system \ Ethernet connections \ Browse for targets

Online installation

Automation Runtime can be transferred or onlineinstallation performed once the connection has been established.

Programming \ Building and transferring projects \ Online services \ Transfer Automation Runtime \ Transferring to SGx target systems \ Installing via an online connection
Project management \ Project installation \ Scenarios \ Online commissioning

Offline installation

For offline installation, Automation Studio is used to generate an installation medium (CompactFlash or CFast card). This is then added to the target system. When the supply voltage is switched on, Automation Runtime and the application are started.

Project management \ Project installation \ Scenarios \ Offline commissioning

Project installation package - USB installation / Network installation

Automation Runtime and the application software can be included in a project installation package and transferred to the target system on a USB flash drive, CompactFlash card or via a DHCP server.
If a USB flash drive containing a project installation package is added to the target system, then the update can be triggered by restarting the system or using the ArProject library.

What is the operating mode for installing with a project installation package?
When using a project installation package to install, it may be necessary to set the device to the "BOOT" operating mode, depending on the device configuration and operating status. In "BOOT" mode, the mechanism for installing a project installation package from a USB flash drive or the network is always active.

Project management \ Project installation

- Scenarios \ Offline commissioning
- Overview \ Transfer \ Project installation package

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

Device-specific installation

A distinction is roughly made between devices with integrated flash memory and devices with CompactFlash. The device-specific Automation Runtime installation, initial startup and a description of the reset button can be found in the respective user's manual.

How is the controller set to "BOOT" mode?
All target systems have the option of deliberately starting in operating mode BOOT. Depending on the target system, this is done using the reset button, mode selector switch or node number selector switch. The reset button can be used to restart the system and change the operating mode. The mode selected with a mode selector switch is applied after restarting. Further information about the possible operating modes and actions can be found in the data sheet of the respective target system.

Programming \ Build & transfer \ Online services \ Transfer Automation Runtime
Hardware \ Power Panel \ Power Panel C70 \

- Installation \ Commissioning
- Device description \ Operating and connection elements \ Reset button / Operating modes

Hardware \ X20 system \ X20 modules \ CPUs \ X20(c)CP1301, X20CP1381 and X20CP1382 \ Operating and connection elements \

- Programming the system flash memory
- Button for reset and operating mode

Exercise: Check and update the Automation Runtime version

The goal of this exercise is to explore the options for updating Automation Runtime on the target system. Automation Runtime upgrades are searched for, the settings of the Runtime versions are checked in the project, Configuration ID and partitioning are determined and the appropriate mechanism for updating the target system is selected.

1) Search for Automation Runtime upgrades

   "Tools" / "Upgrades" menu

2) Check and adjust Runtime versions

   "Project" \ "Change Runtime Versions..." menu

3) Specify configuration ID and partitioning

   CPU configuration (configuration ID, module system on the target system)

4) Select the appropriate online/offline installation or project installation package for the target system

   Notes on this are offered in the data sheet under section "Programming the system flash".

5) Perform installation

   For an online installation: Force initial transfer

4 Memory management

Memory on an Automation Runtime target system is divided into **RAM** and **ROM**.
Parts of this memory are used by Automation Runtime; the rest is available to the application.

4.1 Logical partitioning of the flash memory

During the build process for Automation Runtime, an Automation Studio project generates modules that can be executed. These are transferred to the flash memory during project installation. CompactFlash, CFast and integrated flash memory are used as storage mediums.

A target memory is automatically assigned to every module of the software configuration.
Modules that belong to Automation Runtime are transferred to the **systemROM**.
Modules that belong to the application are transferred to the **user ROM**.
This is a purely logical subdivision. The data transferred is saved on the same data storage device.

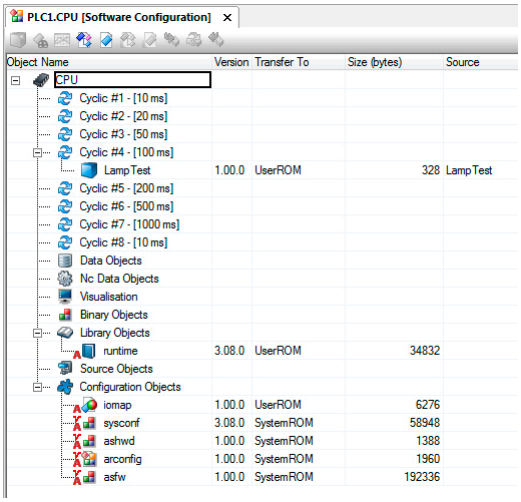


Figure 19: Target memory for software objects in the software configuration

For the **normal B&R file system**, modules from the system ROM and user ROM are stored in different folders on the C partition.
For the **secure B&R file system**, Automation Runtime is stored on the "C" partition. The modules in system ROM and user ROM are stored on the D partitions and a copy of this data is stored on partition E.
System modules are separated from the application by this mechanism.

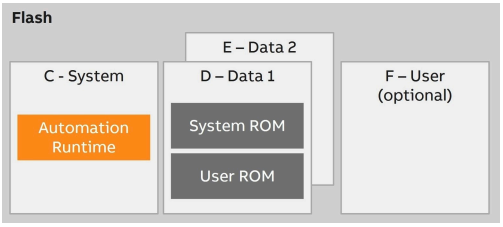


Figure 20: Secure B&R file system

It is possible to configure an additional **user partition** for managing files generated at runtime. During project installation, data is transferred to the user partition using the transfer settings.

Further information:

- 5.5 "Settings when transferring programs" on page 36

The settings for the normal or secure B&R file system are defined in the CPU configuration.

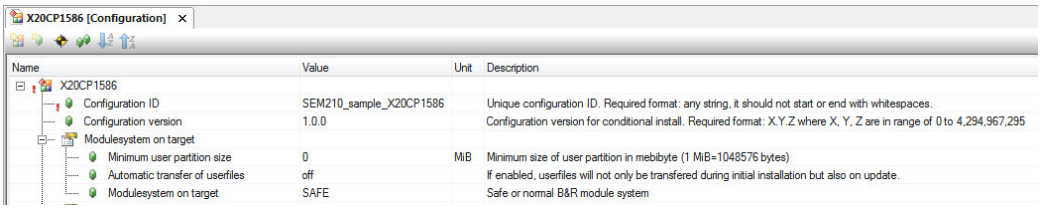


Figure 21: Partitioning configuration for the flash memory in the CPU configuration

Real-time operating system \ Method of operation \ Memory \ Memory types
Project management \ Configuration View \ Software configuration
Project management \ Project installation \ Performing project installation \ Settings

- System configuration settings
- Installation settings

4.2 Used RAM

DRAM is used as a fast read/write memory for the execution of Automation Runtime and the application.
Alternatively, the target systems have battery-backed RAM memory that is used, for example, to retain data after a restart.

What type of memory does this controller have?
The memory types and memory sizes used in the target system are documented in the data sheet of the selected CPU.

Further information:

- 4.3 "Tools for determining memory information" on page 17

DRAM

During startup, Automation Runtime, all configurations and the tasks are copied into DRAM and executed there. This is necessary because access to the DRAM is quicker than flash memory.
In DRAM, a task also requires the configured local or global variable memory. Initialization of this memory is carried out automatically.

SRAM and FRAM

SRAM (static RAM) is battery-backed memory. This allows data to be retained even after a power failure. Of course, the backup battery must be functional.
Newer controller models no longer require a backup battery. FRAM saves data on a nonvolatile electronic memory type, as opposed to SRAM and DRAM.

Real-time clock

The real-time clock is backed up in different ways depending on the device. The duration that it is protected as well as the clock drift in relation to the ambient temperature are described in the respective data sheet of the CPU being used.

Further information:

- 4.4.3 "Initializing variable memory" on page 22
- 5.1.4 "Restart and power failure" on page 26
- 5.1.5 "Using retain variables" on page 27

What to do if the backup battery needs to be replaced?
Information about the service interval of the backup battery and instructions regarding changing it are documented in the data sheet of the respective CPU.

Real-time operating system \ Method of operation \ Memory \ Memory types

4.3 Tools for determining memory information

Automation Studio unites many diagnostic tools that, among other things, provide information about the system state. Diagnostic tools for calculating memory information will be introduced below.

Online info

The online info dialog box is opened using menu item "Online" \ "Info...". The available memory and the status of the backup battery are displayed. In addition, the date and time of the controller can be read out and set. The change of time and date are logged in the "System" Logger module.

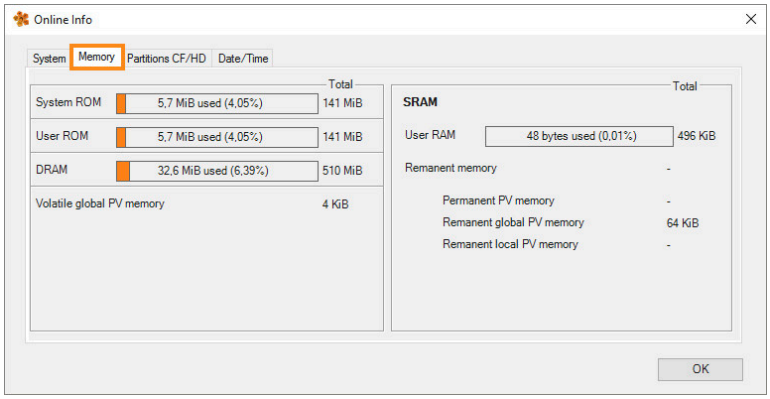


Figure 22: Reading information from the CPU

System Diagnostics Manager (SDM)

System Diagnostics Manager (SDM) is an integral component of Automation Runtime. Selecting "Tools" / "System Diagnostics Manager" from the main menu opens SDM in a browser window.
Information about memory usage is shown in the category "System" \ "Memory". In the "Software" category, you can find information about the modules that have been loaded on the target system.

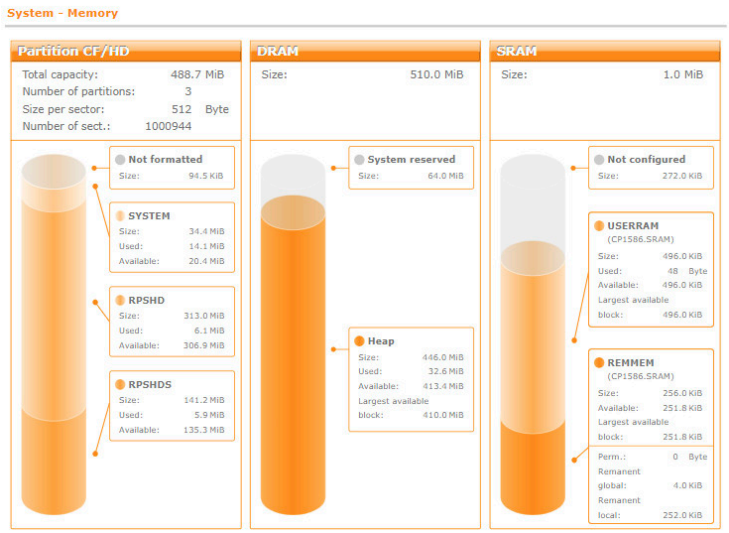


Figure 23: Memory allocation being shown in SDM

Online software comparison

The software comparison compares the modules configured in Automation Studio with the modules installed on the target system.
The software comparison is opened by selecting "Online" \ "Comparison" \ "Software" from the main menu.

The software comparison window displays all of the modules in the project as well as on the controller. The "Size (bytes)" column displays the target memory of the module on the target system. It is possible to compare the version, size, target memory and build date of modules.

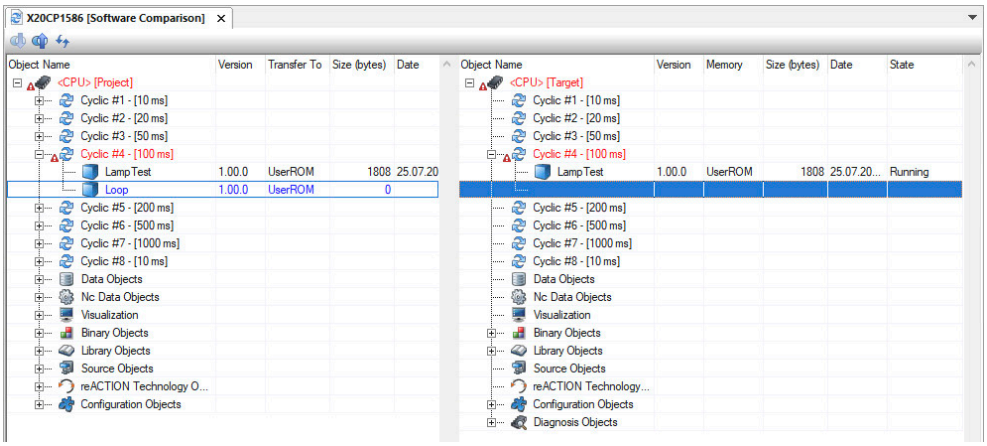


Figure 24: Opened online software comparison, LampTest requires 1808 bytes

Library functions

Using function block MEMxInfo() from library BRSystem, the free memory on the target system can be calculated in the application.
For example, you get direct access to the file system on the flash memory with the FileIO and MpFile libraries.

Diagnostics and service \ Diagnostic tool

- Information about the target system
- System Diagnostics Manager
- Monitor mode \ Online software comparison

Programming \ Libraries

- Configuration, system information and runtime control \ BRSystem
- Data access and data storage \ FileIO

Services \ mapp Services \ MpFile: File management system

Exercise: Set the time and date

It's necessary to set the time and date properly on the controller in order to interpret system events correctly. This can be done in the online info dialog box. The change is logged in the "System" Logger module.

1) Change date and time under "Online \ "Info..."

2) Open Logger, menu option "Open" \ "Logger"

3) Check result in the "System" Logger module

A warning is displayed in the "System" Logger module.

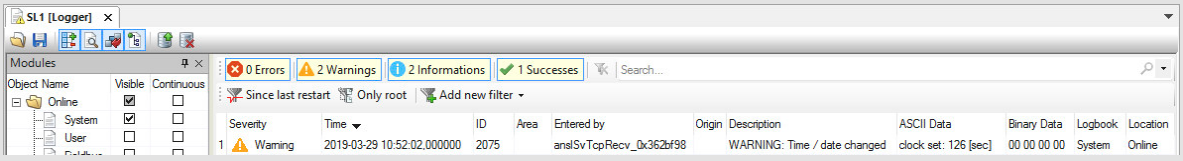


Figure 25: The date and time change is displayed in the "System" Logger module

Exercise: Check free memory on target system

The project, which was created while working with the training module "TM210 – Working with Automation Studio", is already installed on the target system.

Determine the memory allocation on the target system using one of the methods described previously. It should be determined what memory types exist on the used target system and how they are used. You must also check which modules have been loaded on the target system.

1) Determine the free DRAM memory using the online info dialog box
2) Determine free flash memory with System Diagnostics Manager
3) Control which programs are on the target system with System Diagnostics Manager
4) Use the software comparison to check which programs are on the target system

4.4 Global and local variables

Variables are symbolic programming elements whose structure and size is determined by their data type. During the build, a memory location is assigned to the variables by the compiler.
A variable's scope and properties determine its behavior during startup and runtime.

Programming \ Variables and data types

4.4.1 Local variables

Local variables are defined in the scope of a program. Direct access to these variables from other programs is not possible.
A local variable is managed in a .var file at the same level as the program.

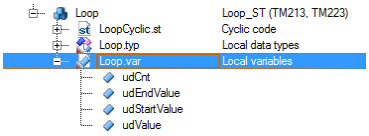


Figure 26: Local variables in the "Loop" program

If local variables from a program must be transferred to local variables of another program, this is carried out using a variable assignment file in the Configuration View.

Exercise: Create "Loop" program with local variables

A new Structured Text program named "Loop" is created.
4 variables of the data type UDINT are declared: "udCnt", "udValue", "udStartValue" and "udEndValue".
Create a loop in the cyclic section of the program. This loop will be explained and used in later exercises.

1) Create a new ST program with the name "Loop"
2) Open the "Loop.var" file and add the 4 variables
3) After saving the "Loop.var" file, the variables in the program "LoopCyclic.st" can be used.

```
PROGRAM _CYCLIC

    FOR udCnt := udStartValue TO udEndValue DO
        udValue := udValue + 1;
    END_FOR

END_PROGRAM
```

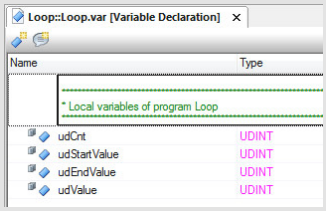


Figure 27: Variable declaration for program "Loop"

Exercise: Multiple assignment of a program in the software configuration

When the previous exercise has been completed, you can start multiple assignment of the program. The "Loop" program is transferred from the Logical View to the software configuration twice using drag-and-drop.

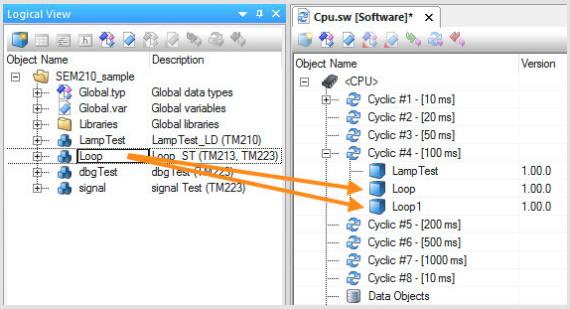


Figure 28: Multiple assignment of programs

1) Open a software configuration.
2) Open the Logical View
3) Add the "Loop" program into the software configuration twice using drag-and-drop.

How is the Watch window opened for "Loop" or "Loop1"?
The Watch window for both instances of the "Loop" program can be opened by selecting it from the shortcut menu in the software configuration or in the Logical View. A selection dialog box appears when the Watch window is opened via the Logical View with regard to which instance the Watch window should be opened.
The Watch window can also be opened by pressing the key combination **<CTRL + <w>**.

All variables are added in the Watch window for tasks "Loop" and "Loop1". Both tasks only have local variables. Changing the variables in the Watch monitor has no effect on the other respective instance of the "Loop" program.

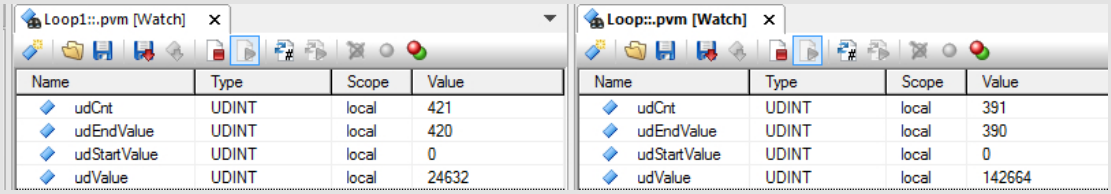


Figure 29: Watch window for tasks "Loop" and "Loop1"

Programming \ Variables and data types \ Scope of declarations
Diagnostics and service \ Diagnostics tools \ Watch window

4.4.2 Global and package global variables

Global variables are displayed at the top level of the Logical View. They can be used throughout the entire Automation Studio project. They can be used in every program. Additional .var files can also be created to provide a better structure for the project.
Variables that were declared within a package are **package-globalvariables**. These are only visible within the respective package and all subordinate packages.

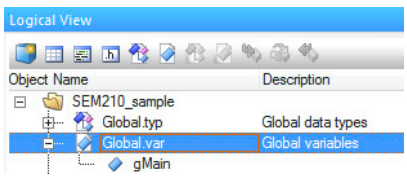


Figure 30: Global variables in the Logical View

When are global variables used?
Global variables should only be used if it is necessary to exchange data between several programs. Another alternative is to connect local variables in different programs together using variable mapping.

Exercise: Create global variable "gMain" and use it in program "Loop"

Enter a new variable in the global variable declaration "Global.var" with the name "gMain" and data type UDINT. This variable should be incremented cyclically in the "Loop" program.

```
gMain := gMain + 1;
```

1) Open file "Global.var"
2) Create variable "gMain" of data type UDINT
3) Once the "Global.var" file has been saved, the variable in "Loop.st" can be used in the program.

If the variable "gMain" is added in the Watch window for tasks "Loop" and "Loop1", the value change in both tasks is visible.

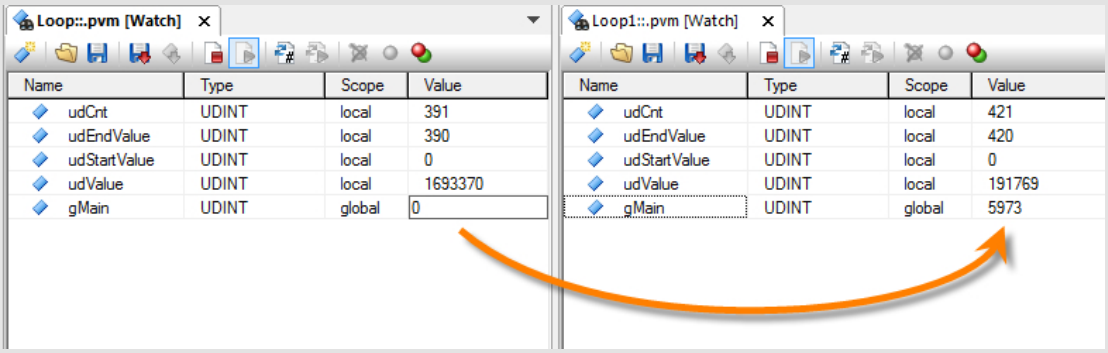


Figure 31: Writing to global variables

Programming \ Variables and data types \ Scope of declarations
Programming \ Editors \ Configuration editors \ Variable mapping editor

4.4.3 Initializing variable memory

During startup, all variables which haven't been given an initial value are automatically initialized with the value "0". An initialization value can be specified in the variable declaration in the "value" column.

Initialization via the variable declaration replaces, for example, the following program lines in the initialization subroutine "**LoopInit.st**":

```
PROGRAM _INIT
    udEndValue := 1000;
END_PROGRAM
```

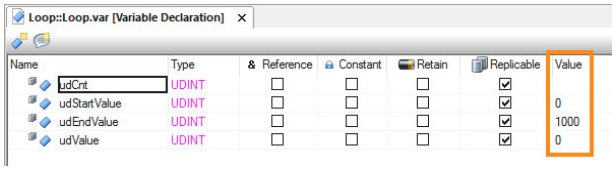


Figure 32: Variable initialization in "Loop.var"

Exercise: Initialize variable "udEndValue"

Configure variable "udEndValue" in the "Loop" program with an initial value of 1000. Monitor the value in the Watch window.

1) Open file "Loop.var"
2) Give the "udEndValue" variable in the "Value" column the initial value 1000

In the Watch window for the "**Loop**" and "**Loop1**" tasks, the "**udEndValue**" variable is initialized with the value 1000. During runtime, this value can be changed to any other value.

Programming \ Editors \ Table editors - General \ Declaration editors \ Table editor for variable declaration

4.4.4 Using constants

Constants are variables whose values never change while the program is running. They are used in the programming instead of fixed numerical values. This makes programs easier to read and maintain.

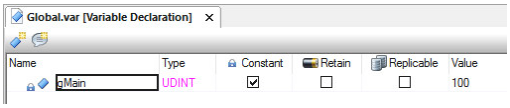


Figure 33: Declaring constants

Exercise: Global variable "gMain" as a constant

Configure the global variable "gMain" as a constant with a value of 100.

1) Open file "Global.var"
2) Give the "gMain" variable in the "Value" column the value 100.
3) Configuring the variable "gMain" as a constant

An error message appears during the build, because there is write access to the "**gMain**" variable.
Write access to constants in programs is not permitted. For the program to operate without error, variable "**gMain**" cannot be defined as a constant.
LoopCyclic.st (Ln: 19, Col: 8): Error 1138: Cannot write to variable "gMain".
It makes more sense to initialize the variable "**udStartValue**" as a constant with the value "0". This variable is defined as read-only in the program.

How and where are the variables used in the project?
The cross-reference list is used to determine read and write access of variables in programs. A cross-reference list is generated by selecting "Project" / "Create cross-reference" from the main menu.

Programming \ Variables and data types \ Variables \ Constants
Project management \ Workspace \ Output window \ Cross reference

5 Runtime performance

This section describes the runtime behavior of the application on the target system.
Startup behavior, program initialization as well as the cyclic program process are explained.

5.1 Start Automation Runtime

Automation Runtime boots when the target system is switched on. Some of the tasks are completed before the cyclic programs are executed.

The following tasks are executed before the cyclic program:

- Hardware check
- Check if hardware firmware upgrade is required
- Check the BR modules
- Copy the BR modules from ROM to DRAM
- Retain variables copied to DRAM
- Set variable memory to its initial values
- Execute program initialization
- Enable cyclic programs

If no errors occur during the boot phase, the target system starts in RUN mode.

Figure 34: Power Panel C70 in RUN mode

Real-time operating system \ Method of operation \ Boot behavior
Real-time operating system \ Method of operation \ Module / Data security

5.1.1 Automation Runtime operating states

After the controller has started up, the following four operating states are possible: BOOT, DIAGNOSIS, SERVICE and RUN.
In the "RUN" operating state, Automation Runtime starts the application created with Automation Studio and executes it cyclically.
Certain events lead to the target system canceling startup and staying in the corresponding operating state for diagnostic purposes. The current operating state is read using the LED status indicators on the CPU and the status bar in Automation Studio. Details are logged in the "System" Logger module.

Operating state	Conditions that lead to this system state
BOOT	• No CompactFlash inserted • No operating system available on the CompactFlash/CFast card[2] • Node number switch in position "00", mode selector switch in position "BOOT", reset button [3]

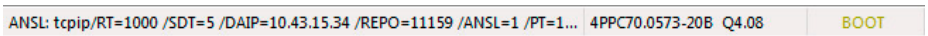


Figure 35: Status bar - "BOOT" mode

Table 1: Overview of Automation Runtime operating states

2 A requirement for this is that the implemented system has Automation Runtime by default. PC-based systems do not have Automation Runtime by default. In this case, a connection to the target system cannot be established. Offline installation must be performed.

3 The setting of the operating mode depends on the device. Depending on the device, node selector switches, mode selector switches or reset buttons can be used for this purpose; see the respective user manual.

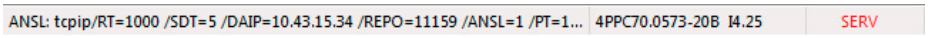
Operating state	Conditions that lead to this system state
DIAGNOSTICS	• Fatal system error • Node number switch to "FF", mode selector switch to "DIAG", reset button  Figure 36: Status bar - "DIAGNOSTICS" mode
SERVICE	• Division by zero[4] • PageFault • Cycle time violation • Missing hardware modules • CPU halted by Automation Studio  Figure 37: Status bar - "SERVICE" mode
RUN	• No error  Figure 38: Status bar - "RUN" mode

Table 1: Overview of Automation Runtime operating states

Real-time operating system \ Method of operation \ Operating states
Hardware \ X20 system \ X20 modules \ CPUs \ X20(c)CP158x and X20(c)CP358x \ Operating and connection elements \ Operating mode switch

5.1.2 Automation Runtime boot phases

The following intermediate states are possible during the startup of the controller: STARTUP, FIRMWARE and INIT. The controller cannot be accessed via the online connection at this time.
These states are also indicated in the Automation Studio status bar. These phases are usually very short.

System state	Description
STARTUP	Initialization procedures relevant to the operating system are performed during this phase.
FIRMWARE	Firmware is updated during this phase, see 5.1.3 "Hardware versions and firmware updates" on page 26.
INIT	The application's initialization subroutines are executed during this phase.

Table 2: Overview of Automation Runtime boot phases

How do you recognize the boot phases on the X20 System?
The phases before the RUN operating mode (STARTUP, FIRMWARE, INIT) are indicated by the blinking green R/E LED on the X20 system.

Real-time operating system \ Method of operation \ Boot phases
Hardware \ X20 system \ X20 modules \ CPUs \ X20(c)CP158x and X20(c)CP358x \ Status LEDs

4 Unlike Intel target systems, division by 0 does not result in a processor exception on ARM target systems, and instead results in 0.

5.1.3 Hardware versions and firmware updates

In Automation Studio, each hardware module has its own device description file, a suitable image for System Designer and firmware (optional). When Automation Runtime starts up, the firmware on the hardware module is compared with the firmware installed on the system and, if necessary, updated.

Automation Studio is provided with one hardware version per hardware module. Additional **hardware versions** can be installed using menu option "Tools" / Upgrades".
In the Physical View, in the "Version" column, you can make selections from the installed hardware versions. When installing new hardware versions, they are not automatically updated and must be selected manually.

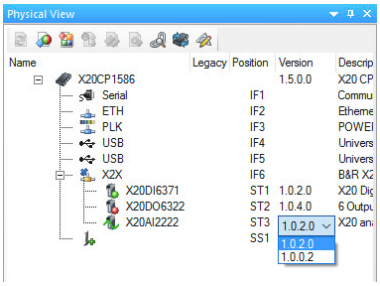


Figure 39: Selecting a hardware version in the Physical View

Project management \ Hardware management \ Hardware versions

5.1.4 Restart and power failure

The restart behavior of target systems when changing operating states and after reset is documented in sections "Operating mode and node number switches", "Operating mode switch" and "Reset" in the respective data sheet for the controller used.
In Automation Studio, the **restart behavior** after a reset or voltage drop can be configured. This setting is defined in the CPU configuration.

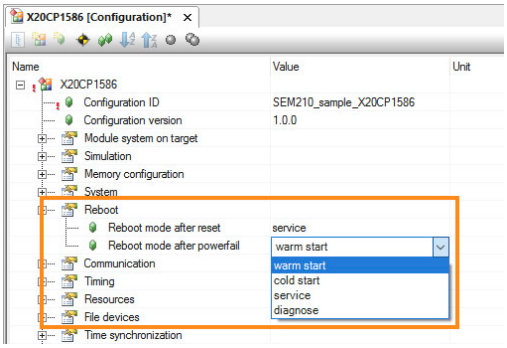


Figure 40: Configuring the restart behavior in the CPU settings

Warm restart

A restart (warm restart) is triggered by the following actions:

- PowerON after power failure
- Pressing the reset button
- Changing a system configuration and transferring it to the target system
- Performing a warm restart with Automation Studio
- Performing a warm restart with SYSreset() function

Remanent variables retain their values during a warm restart.

Programming \ Libraries \ Configuration, system information, runtime control \ SYS_Lib \ Function blocks and functions \ System functions

Power-on handling

SYSROM and USERROM are located on the CompactFlash, CFast card or internal memory when power is not on. The remanent[5] variables and the USERRRAM are located on the buffered RAM (SRAM). Automation Runtime is started when the controller is booted. SYSROM, USERROM and remanent variables are copied from SRAM to DRAM.

Real-time operating system \ Method of operation \ Module / Data security \ Power-on handling

Power-off handling

Many B&R target systems are equipped with a power failure logic. This allows the system to enter a defined state in the event of a power failure. The following tasks are performed in event of a power failure:

- Access to data objects located in USERRAM is locked.
- The remanent variables (RETAIN) are copied into the battery-backed SRAM.

Real-time operating system \ Method of operation \ Module / Data security \ Power-off handling

Exercise: Startup and operating states

First check the configuration for the restart behavior of the implemented target system. Different operating states should be entered using the mode selector switch, reset button and Automation Studio. In the data sheet of the target system used, the functions of mode selector switches and reset buttons are documented. The operating states in the "System" Logger module should be then checked.

5.1.5 Using retain variables

The value of a remanent variable (configured as retain) is retained after a warm restart.
In order to store variables in remanent memory, the following requirements must be met on the target system and in the variable declaration:

- Target system with battery-backed RAM - see data sheet
- Configure the variable by enabling the "Retain" option

Name	Type	& Reference	Constant	Retain	Value
OperatingHours	UINT	☐	☐	☑	0
ProductCounter	UDINT	☐	☐	☑	0

Figure 41: Configure retain variables in the variable declaration

Examples of the use of retain variables:

- Operating hours counters
- Type counters and data for system efficiency
- Condition and status of the machine at the time of a power failure

How large is the remanent memory of the control?
Different target systems have varying amounts of remanent memory available. Information about size and availability can be found in the respective data sheet or in Automation Help.

Real-time operating system \ Method of operation \ Memory
Programming \ Editors \ Table editors - General \ Declaration editors \ Table editor for variable declaration
Hardware \ X20 system \ X20 modules \ CPUs \ X20(c)CP158x and X20(c)CP358x \ Technical data

5 Remanent variables are marked in the declaration as "Retain".

Exercise: Use retain variables

The "udValue" variable is increased with every loop in the "Loop" program. During a warm restart, the last value of the "udValue" variable must be retained.

1) Declare "udValue" variable as retain
2) Transfer program - Check value "udValue" in the Watch window
3) Perform warm restart
4) Checking the result

5.2 Program initialization

An initialization subroutine can be managed for every program. For example, variables can be initialized through calculations in an initialization subroutine.

Running the initialization subroutine

Before the first cyclic program is started, all initialization subroutines are executed once in the configured order. As long as the initialization subroutines are executed, this is indicated in the status bar ("INIT") as well as by the LED status indicators on the CPU.

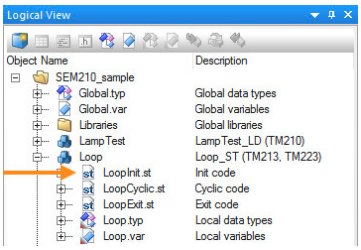


Figure 42: Program initialization

In this example, the initialization subroutines would be executed in the following order:

1) Initialization subroutine for the "LampTest" task
2) Initialization subroutine for the "Loop" task
3) Initialization subroutine for the "Loop1" task

Because initialization subroutines are not subject to cycle time monitoring, a violation will not be triggered by longer initializations.

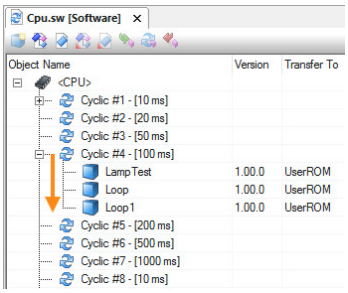


Figure 43: Executing initialization subroutines

Function calls in an initialization subroutine

When using function blocks, make sure that the function returns the result at the first call because the init program is not executed cyclically.

Function blocks that must be called multiple times must be called in the cyclic section of the program. Information about the use of function blocks in the init subroutine can be found in the respective documentation.

Real-time operating system \ Target systems \ Target systems - SG4 \ Runtime behavior - SG4 \ Starting Init SPs - SG4

5.3 Executing cyclical programs

A program is assigned a certain task class, in the software configuration.
The programs assigned to the software configuration are identified as tasks. The programs that are assigned in the software configuration are only executed after the transfer.

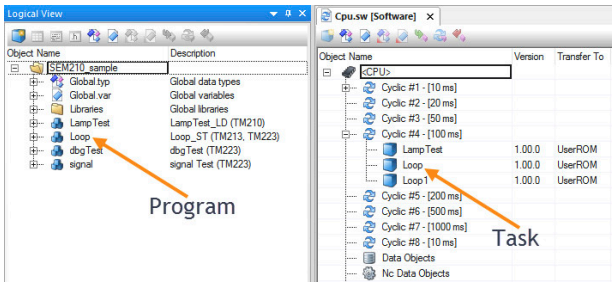


Figure 44: Programs and tasks

The tasks are assigned to configurable task classes and executed one after the other. A program is executed equally "fast" in each **task class (1 to 8)**. The task class only changes the intervals at which the program is executed. In task class 1, for example, the program is executed every 10 ms, and thus more often than a program running in task class 7 with a cycle time of 1000 ms. The times when the tasks start as well as when the I/O system is accessed are deterministic.

5.3.1 Task classes and cycle times

A task is executed cyclically in the time defined by its task class i.e. its cycle time.
Up to eight configurable task classes are provided to help optimize a task for its particular requirements. Every task class contains tasks with the same cycle time, priority and tolerance.

In this example, the task class #4 contains three tasks. A cycle time of 100 ms is configured.

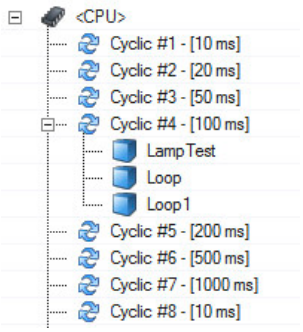


Figure 45: Task class #4 with three tasks

In which task class does the program belong?
Not all tasks need to run within the same task class. Control tasks that must be executed quickly should be assigned to a task class with a smaller cycle time. For slower processes, a task class with a correspondingly higher cycle time is selected.

Ignoring the time it takes to execute the tasks, the program sequence would look like this:

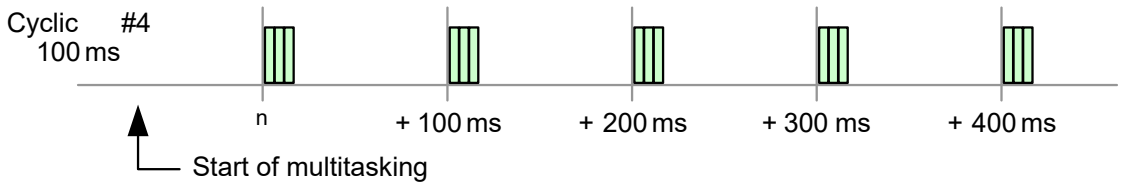


Figure 46: Tasks are called again during each cycle.

This means that the tasks in this task class are executed every 100 milliseconds. The prerequisite for this is that the total processing time of all tasks in this task class does not exceed the cycle time of this task class.

Real-time operating system \ Target systems \ Target systems - SG4 \ Runtime behavior - SG4 \ Start of cyclic tasks

5.3.2 Cycle time and priority

The priority of a task class is determined by the number of the task class.
The lower the number, the higher the priority of the task class.
Moving a task between task classes changes its priority and cycle time.
Task class #8 has the lowest priority in the system, see "Task class with high tolerance" on page 36.

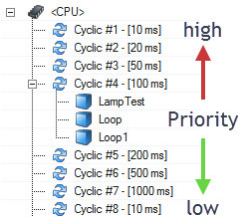


Figure 47: Task class priority

If the "Loop" task is moved from task class #4 to task class #1, it will be executed every 10 milliseconds.

> The execution time of a task is not influenced by moving it to a different task class. It just changes the number of times it is called in a given period of time.

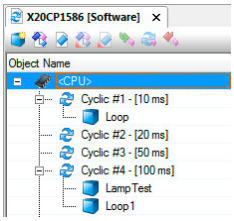


Figure 48: Different task classes

The "**Loop**" task is now executed every 10 ms. The tasks "**LampTest**" and "**Loop1**" in task class #4 are executed every 100 milliseconds, as before.

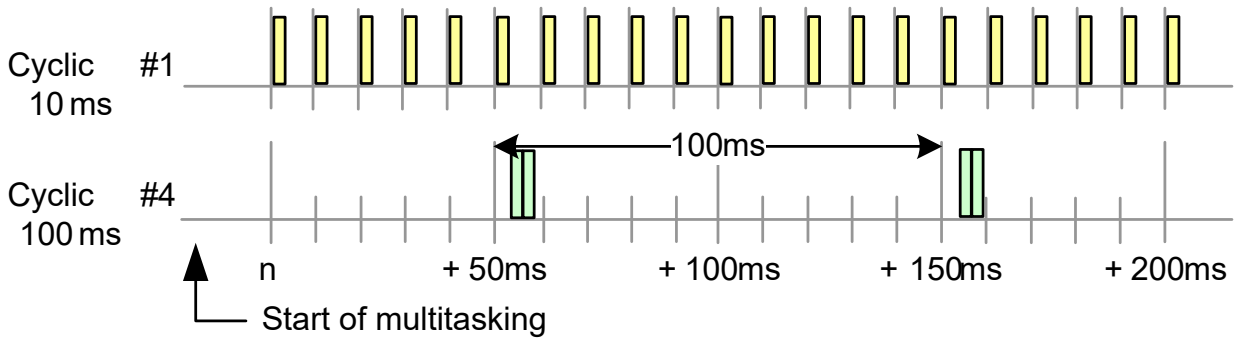


Figure 49: Executing tasks in different task classes

Exercise: Move the "Loop" task to task class #1

The "Loop" task in the software configuration should be moved to task class #1.
In the Watch window, observe the changes to the cycle times of "Loop" and "Loop1", which reference the same program.

1) Open a software configuration.
2) Move the "Loop" task from task class #4 to task class #1.
3) Open the Watch window for both tasks
4) Observe the value of the "udValue" variable

The "Loop" program is executed 10 times as frequently as "Loop1". As a result, the value of the "udValue" variable is increased more often by this factor.

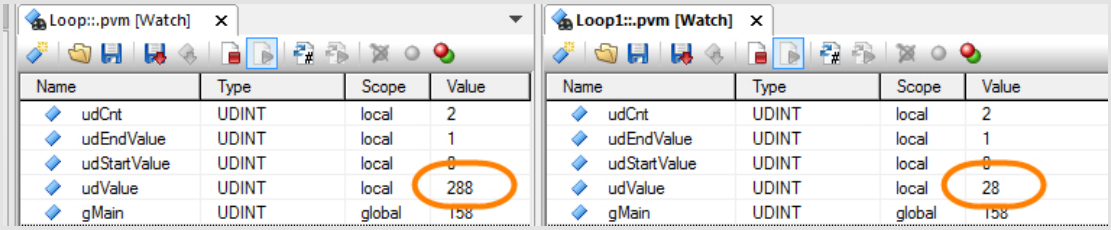


Figure 50: Cycle times in different task classes

Example: Priority task CNC machine
In the program of a CNC machine, it is important to know the exact location of the drill. The fast and precise drilling – and thus the position of the drill – has a very high priority. The position of the drill must be queried very often so that the "gaps without a position" are as small as possible.
If this query were in a high task class, a current position of the drill would not be available in the meantime, since the program is only called again after the cycle time has elapsed.
This task would be created in task class #1.

5.3.3 Idle time

During operation, Automation Runtime performs **other system tasks** in addition to the execution of tasks. They provide the user with useful functions. For example, a constant checksum verification process is performed and online communication is available.
The speed at which these tasks are executed can vary depending on processor performance.
The time during which Automation Runtime does not perform any cyclic tasks is referred to as the **idle time**.

The following tasks are executed during idle time:

- Online communication
- Visual Components application
- Access to the file system

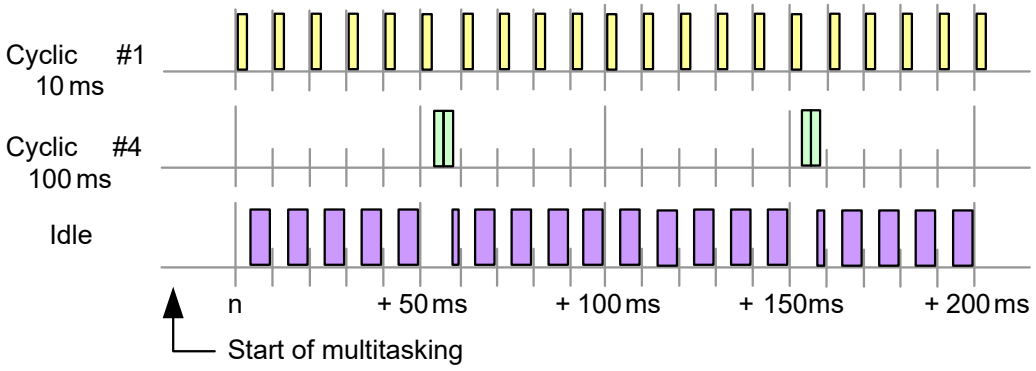


Figure 51: Idle time in the cyclic system

The basic task class for the idle time is established in the CPU configuration. The configured idle time of the system is provided in the context of this task class.
The **Profiler** can be used to determine the idle time. The System Diagnostics Manager can also be used to display the average system load.

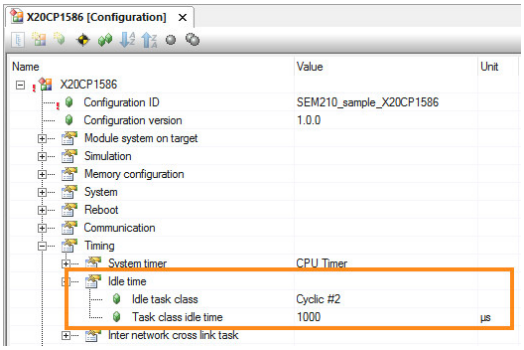


Figure 52: CPU configuration: Basic task class for idle time

What to do if the idle time is insufficient?
If the idle time is insufficient, it can be increased by moving programs to higher task classes or by adjusting the cycle time.

Real-time operating system \ Method of operation \ Runtime performance \ Scheduling \ Idle time

5.3.4 Starting task classes

Not all task classes are started simultaneously after the multitasking system has been started.
The starting point is always half of the task class cycle time.
This means, for example, that the 100 ms task class will be started after 50 ms. Distributing the start time of all task classes makes better use of processor performance and allows, for example, output jitter to be kept to a minimum.

Real-time operating system \ Target systems \ Target systems - SG4 \ Runtime behavior - SG4 \ Start of cyclic tasks

5.3.5 Task interrupted by another task

Task class priority makes it possible for a task of higher priority to interrupt a task of priority task class that takes longer.
Using the Watch window, the limit of the variable "**udEndValue"** can be changed in such a way in tasks "Loop" and "Loop1" that the task "**Loop1**" is interrupted exactly **two times** by the task "**Loop**".

The schematic diagram shows how the timing can look in this case. In order to be complete and promote better understanding, the image was updated with the timing of task class #3.

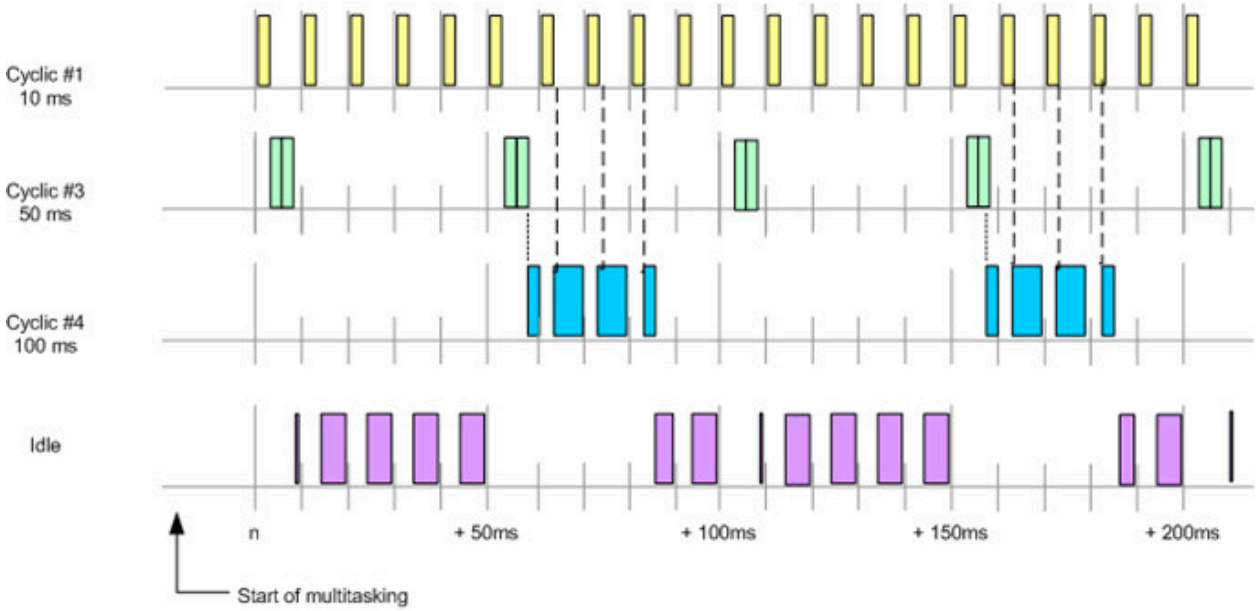


Figure 53: Multitasking with task class #1, #3 and #4; Task class #4 is interrupted by task class #1

For task class #4, a consistent I/O input image is available over the entire execution time. That's why the tasks in this task class are not influenced by the interruption.

5.4 Cycle time and tolerance

Each task class has a defined cycle time. All of the tasks in a task class must be executed within this cycle time.
The cycle times of the task classes are already defined when a project is created. The user can change the **cycle time** of each task class **individually**.
If the total runtime of all tasks of a task class exceeds the cycle time of that task class, a cycle time violation occurs. A configurable tolerance delays the cycle time violation. A restart is performed.
The cycle time and tolerance can be configured in the properties of the task class. Open the Properties window from the shortcut menu for the task class.

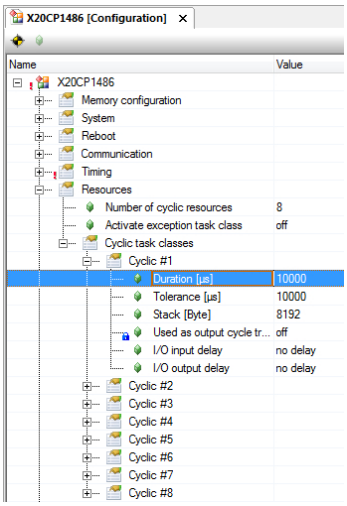


Figure 54: Configuring the cycle time and tolerance

If the configured cycle time of a task class is exceeded, the start of the next cycle is moved backwards by the set tolerance. This can lead to problems in the timing of the application. This behavior is avoided by setting the tolerance to the value "0".

Programming \ Editors \ Configuration editors \ Hardware configuration \ CPU configuration

Exercise: Check the system load using 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.
Using the System Diagnostics Manager, the system load caused by the tasks and the available idle time are monitored.

1) Give the variable "udEndValue" the value 50000
2) Determine the system load using System Diagnostics Manager
3) Increase the value of the variable "udEndValue" step by step

The results in the System Diagnostics Manager must be analyzed between the respective steps.

In the System Diagnostics Manager, the average system load is displayed over a selectable time period. In the graphic, the mean and the maximum can be read.
Details about the runtime behavior of a certain task can only be obtained with the Profiler. Additional exercises are planned in training module "TM223 - Automation Studio Diagnostics".

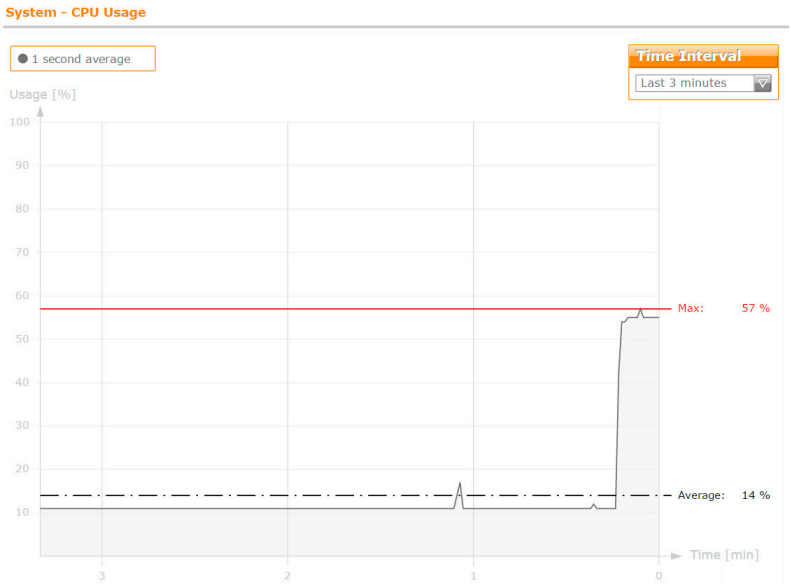


Figure 55: Display of the CPU load in the System Diagnostics Manager

Diagnostics and service \ Diagnostic tool

- System Diagnostics Manager (SDM)
- Profiler

5.4.1 Cycle time violation

Automation Runtime monitors the timing when executing the task classes. A cycle time violation is triggered if the execution time of the tasks exceeds the configured cycle time. If a cycle time violation is detected, the target system starts up in the "SERVICE" operating mode.

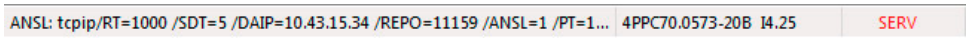


Figure 56: CPU in the "SERVICE" operating mode

In this example, the cycle time violation is the result of changing values in the Watch window. When a cycle time violation occurs, all values in the Watch window are "frozen" and initialized with "0" after restarting in the "SERVICE" mode.
The Logger can be used to analyze the problem when the target system is running in the "SERVICE" operating mode. The Logger window can be opened by selecting "Open" \ "Logger" from the main menu or using the keyboard shortcut **<CTRL> + <L>**.

In the image, a cycle time violation in task class #1 was entered as the cause of error.

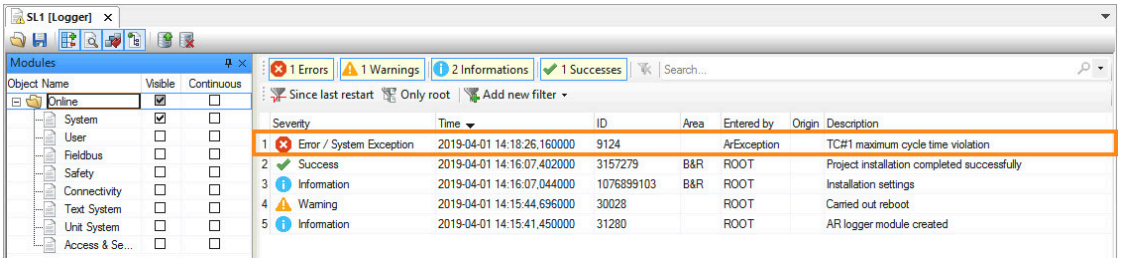


Figure 57: Analyze the cause of the error in the Logger: Cycle time violation in task class #1

As an alternative to the Logger, the Profiler can be used to analyze the problem.

Further information:

- TM223 - Automation Studio diagnostics

Diagnostics and service \ Diagnostics tools \ Logger

Exercise: Increase the value of the "udEndValue" variable until a cycle time violation occurs

The goal of this exercise is to raise the "udEndValue" variable in the "Loop" program incrementally until a cycle time violation occurs. The target system is restarted. The controller starts in the "SERVICE" operating mode. Next an analysis with the Logger has to be performed.

1) Increase the value of the "udEndValue" variable little by little until it reaches the cycle time violation

2) Analyze the Logger entry of the cycle time violation

5.4.2 Reacting to cycle time violations in the application program

In a live production system, changing to the "SERVICE" mode without an option for a response is undesired. Events like a cycle time violation can be reacted to in an exception program.
The exception task class is enabled in the properties of the CPU configuration under the Resources category.

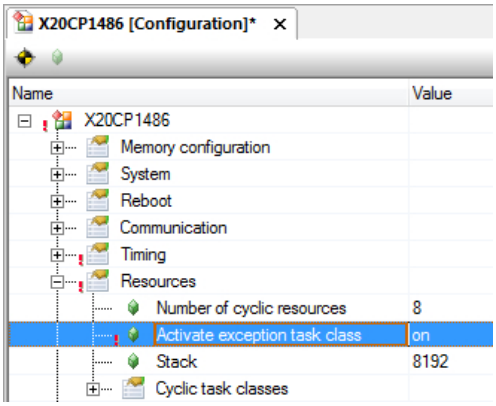


Figure 58: Enabling the exception task class

A program is added from the Logical View into the software configuration in the exception task class.

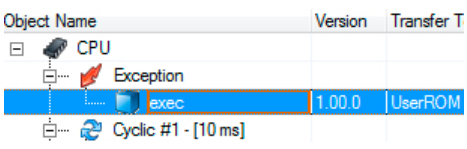


Figure 59: Assign program in exception task class

An **exception number** is specified in the properties of the exception task. The exception number is specified by the system and can be looked up in Automation Help.
A cycle time violation (exception no. 145) occurring at runtime will call this task. Depending on requirement, you can react to the cycle time violation in the exception task.

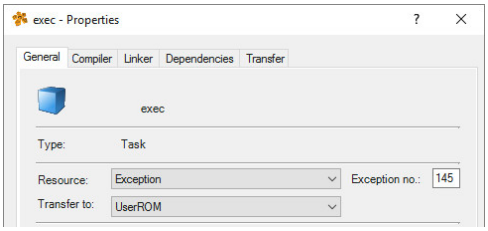


Figure 60: Configuring the exception task

Programming \ Editors \ Configuration editors \ Hardware configuration \ CPU configuration \ SGx \ CPU properties - Resources
Real-time operating system \ Target systems \ Target systems - SG4 \ Runtime behavior - SG4 \ Exception task class - SG4

5.4.3 Task class with high tolerance

Tasks that are executed as quickly as possible but with a low priority should be executed in task class #8.
If possible, this is called every 10 ms, but can be interrupted by any higher priority task class. The configured tolerance determines that task class #8 must be executed to completion at least once every 30 seconds.

Tasks that should be assigned to task class #8:

- File access from application program
- Complex, non-time-critical calculations

5.5 Settings when transferring programs

Online installation

Programs and configurations are transferred after changing to the target system. A restart is required when changing a system configuration.
When transferring program changes, you have to decide whether this occurs during development or on the production system.
Depending on the use case, the **transfer settings** for project installation can be adjusted. The image shows the standard values for project installation.

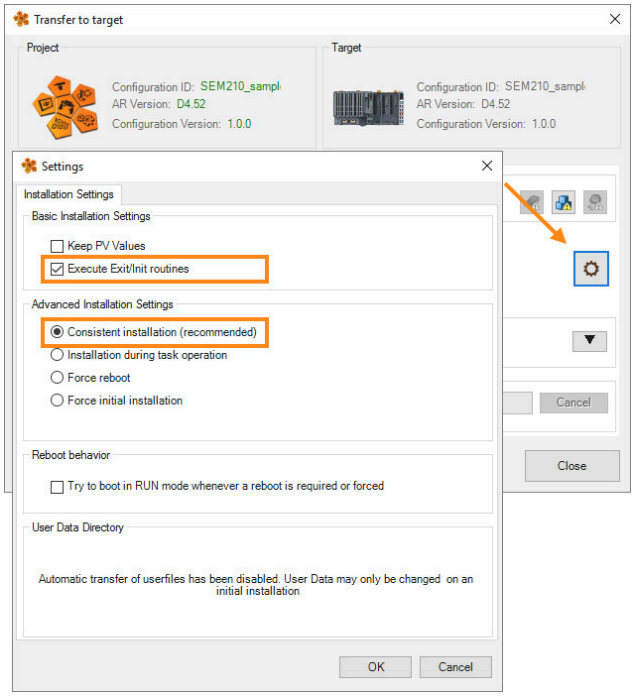


Figure 61: Configuration of the transfer settings for project installation

Keep PV values

This setting defines whether the values of process variables are retained over the installation procedure for existing tasks and tasks updated by the transfer.

Execute Exit/Init routines

If resources (memory, interfaces) were used in the initialization or cyclic subroutine, then these resources can be freed up again using the Exit subroutine.

Consistent installation

All task classes are halted during installation. This guarantees the consistency of all tasks that run on the target system.

Installation during task operation

Only those tasks are halted that are being installed during the installation procedure. The other tasks continue to be executed cyclically and are unaffected by the installation.

Force restart or initial transfer

With the force restart option, the target system is restarted after the project transfer.
An initial transfer is carried out using the option "execute initial transfer".

User directory

If a user partition has been created, this option copies the user data to that partition.

Further information:

- 3.2 "Configuration ID and partitioning" on page 12

Project management \ Project installation
- Scenarios \ Online update
- Performing project installation \ Settings \ Installation settings

Offline installation

During offline installation, the installation settings allow nonvolatile memory to be retained. This means that the values of retain variables are retained, regardless of whether the CPU is performing a warm or cold restart.

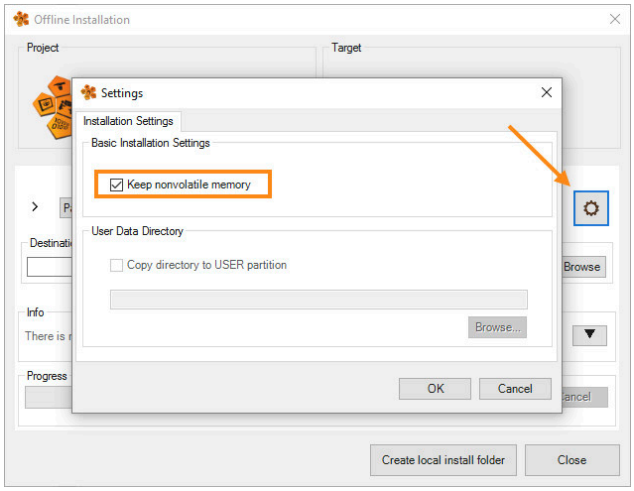


Figure 62: Installation setting for offline installation

Exercise: Adjust transfer settings

The goal of this exercise is to experience effects of the transfer settings.
In the "Loop" task of the Watch window, the value of "udEndValue" is set to 1000.
A change is then made in the source code of "Loop"; this can be a comment. The changes are compiled and the transfer window opens. With the changed transfer settings, the value of "udEndValue" previously set in the Watch window is retained.

1) Set the value of the "udEndValue" variable to 1000
2) Change the source code in the "Loop" program
3) Compile and open the transfer window
4) Adjust transfer settings
5) Check if the value of "udEndValue" is retained

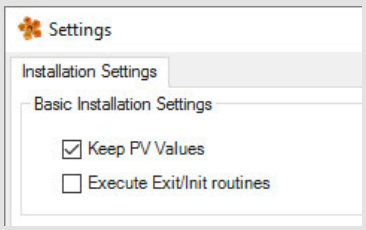


Figure 63: Settings for preserving the values of process variables; the execution of init/exit programs is prevented

6 I/O handling

One of the main functions of the controller is fast and deterministic data transport. The input and output states of the I/O terminals are in constant communication with the controller application.
Automation Runtime I/O Management is designed to meet the highest requirements for **response times** and **configurability**.
The I/O scheduler transfers the input image from the I/O terminal before the beginning of the task class and the output image at the end of the task in a task class.

The image shows the basic sequence of I/O handling in Automation Runtime. The I/O scheduler makes a consistent input image available to the task class. This is indicated with the green arrow. At the end of the last task in the task class, the output data is transferred to the I/O scheduler. This is indicated with the red arrows.

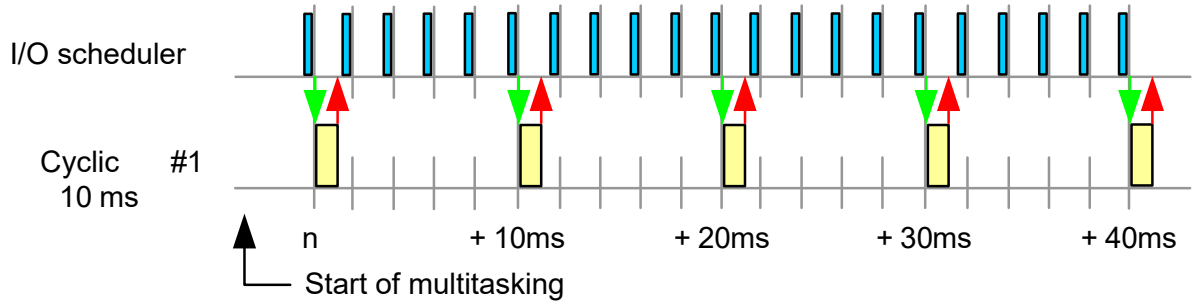


Figure 64: I/O scheduler: Provide input image; write the output image at the end of the execution time of the tasks.

In the I/O mapping, process variables are connected with I/O module channels. The I/O scheduler is controlled by the I/O mapping.

By default, task class #1 is set to 10 milliseconds.
The default configuration for the POWERLINK and X2X interfaces is a cycle time of 2 milliseconds.
The cycle time of the task class can be adapted to the I/O cycle time. This setting can be found in the CPU configuration

Programming \ Editors \ Configuration editors \ Hardware configuration \ CPU configuration

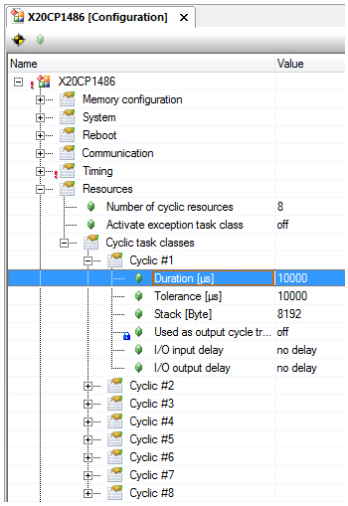


Figure 65: Configuring the cycle time for task class #1

Real-time operating system \ Target systems \Target systems - SG4 \ I/O management - SG4
Real-time operating system \ Target systems \Target systems - SG4 \ I/O management \ Method of operation

6.1 I/O configuration and I/O mapping

6.1.1 I/O configuration

Module-specific properties are configured in the I/O configuration.
The offered functionality of I/O modules continues to open up many implementation options and operating modes in which they can be used. The behavior for module monitoring, individual configurations and function models, among others, are adjusted in the I/O configuration.

The I/O configuration is opened via the shortcut menu for the desired I/O module. The configuration options of an I/O module are documented in the respective data sheet under section "Register description".

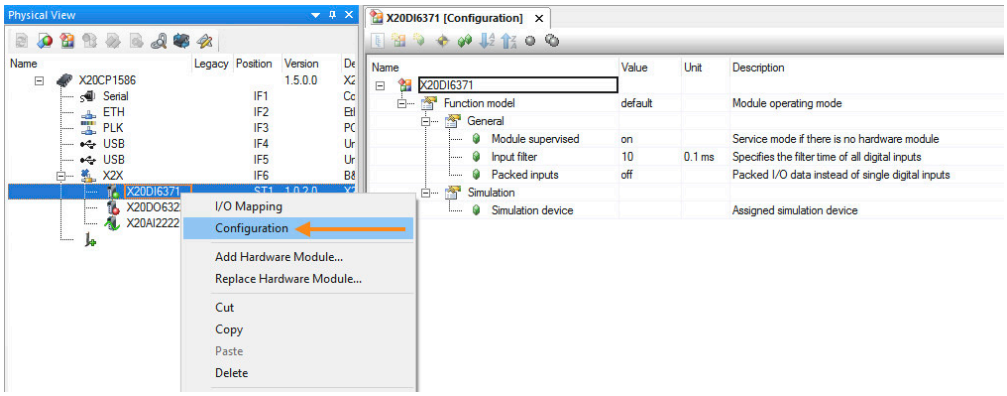


Figure 66: I/O configuration of a digital input module

Programming \ Editors \ Configuration editors \ Hardware configuration \ I/O configuration

6.1.2 I/O mapping

The I/O mapping defines which data from the I/O image is assigned to a process variable. Each variable in a .var file that is used in a program can be assigned to a channel of an I/O module, regardless of its scope.
The I/O mapping is opened via the shortcut menu for an I/O module.

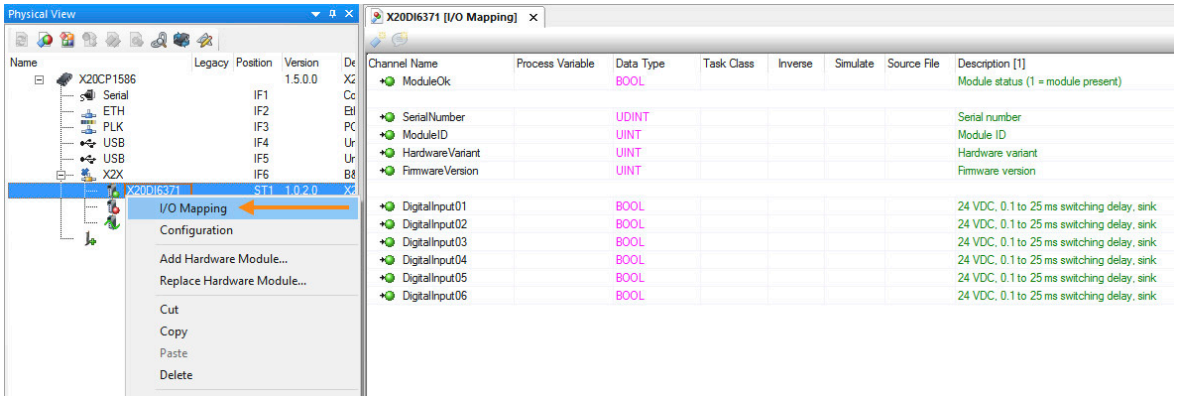


Figure 67: I/O mapping of a digital input card

Which task class is required for the I/O mapping?
For global variables, the channel can be assigned a task class in which the data of the I/O image should be transferred. If "Automatic" is set and the project built, Automation Studio will automatically determine the fastest task class where the variable is being used.

Programming \ Editors \ Configuration editors \ I/O mapping

6.2 Handling I/O images

The I/O scheduler provides a consistent I/O image for the execution of all tasks in a task class and starts the task class at the exactly configured time. This ensures that the inputs at the beginning of the task class and the outputs at the end of the task class are transferred.

Which task class does the I/O scheduler use?
Each task class uses a separate I/O image. The input states remain consistent during the entire task runtime.

Cycle of I/O data

The provision of I/O data occurs in the set I/O cycle time of the bus system used.
The configuration is opened via the shortcut menu for the respective fieldbus.

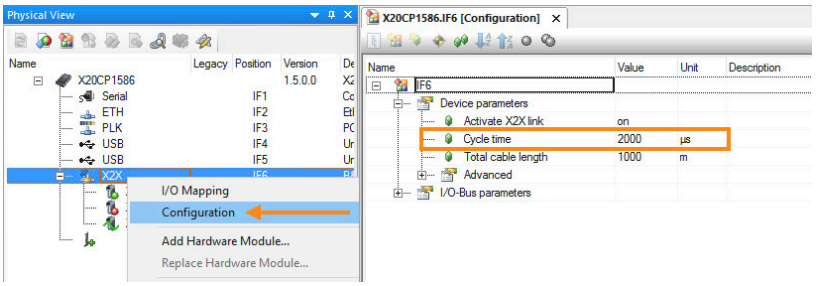


Figure 68: Opening the properties of the X2X Link interface

The I/O cycle time configured in the properties is used as the basis for copying I/O data to the I/O image. This means that the inputs are provided as an I/O image and the output image is described in the set X2X cycle time.

Is the cycle time of the interface sufficient?
Network Analyzer makes it possible to check whether the configured POWERLINK and X2X Link cycle times are permissible. It is opened by selecting "Open" / "Network Analyzer" from the main menu.

Diagnostics and service \ Diagnostic tools \ Network Analyzer

The graphic shows that an input image is provided every 2 milliseconds. For the application, this means that the input data is not older than the configured cycle time when the task class starts; the output data will be written after this time has passed at the latest.

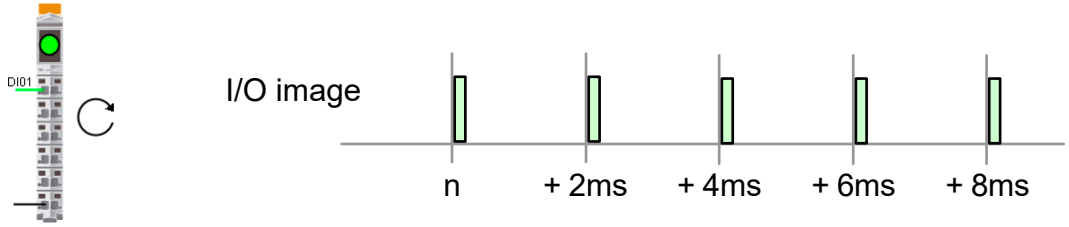


Figure 69: Reading the input image

I/O data in the task class

The I/O scheduler controls when the I/O image for the task classes is provided and the task classes are started.

By default, the I/O scheduler is opened with a system tick of 1000 µs.
The system tick is set in the CPU configuration in the "Timing" category.

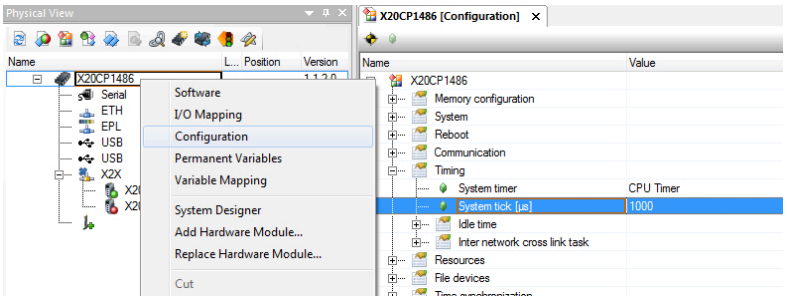


Figure 70: Configurable system tick

The graphic shows that the I/O scheduler is called twice as often as the image is updated.

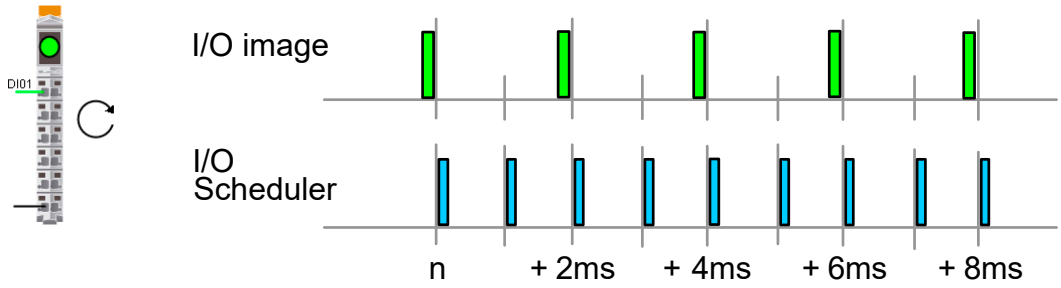


Figure 71: Reading the input image; Calling the I/O scheduler

The cycle time of the fieldbus can be used as a system timer in the timing configuration. By default, the fieldbus cycle time is applied in a 1:1 ratio for the I/O scheduler. I/O cycle and I/O scheduler now work synchronously in the set ratio.

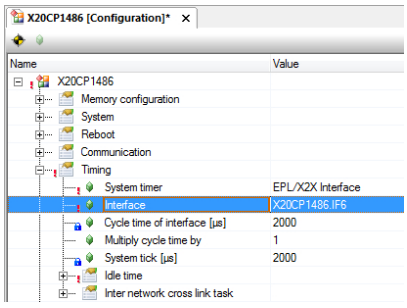


Figure 72: Synchronizing the system tick

By using the fieldbus timer as a system timer and multiplying the cycle time by a factor of 1, the I/O scheduler is called with the configured I/O cycle time.

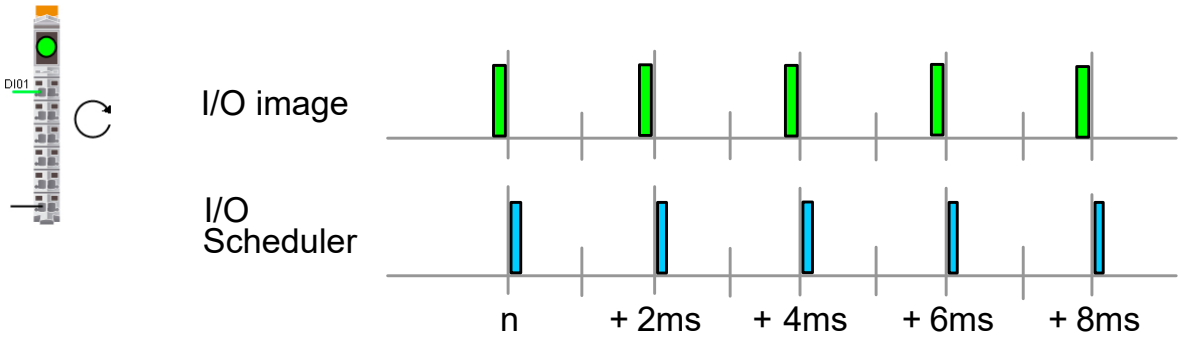


Figure 73: Synchronizing the I/O image and I/O scheduler

As is established in the I/O mapping, the I/O data from the input image is assigned to the configured process variables. For the tasks of a task class, this results in the following when the I/O scheduler is called in a cycle time of 2 milliseconds:

Task class #1

The I/O scheduler starts task class #1 every 10 ms and provides it with the input image for the assigned variables (green arrow).
At the end of the tasks in task class #1, the variables of the assigned outputs are written to the output image (red arrow).

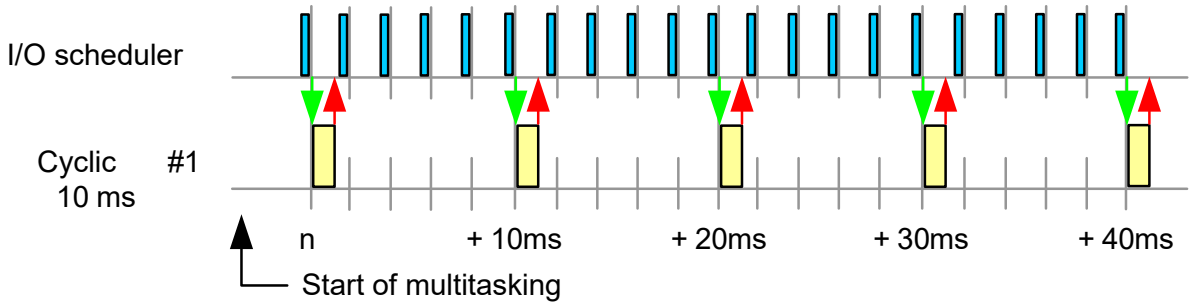


Figure 74: I/O handling for task class #1

Task class #4

The I/O scheduler starts task class #4 at the time, n+50 ms, n+150 ms and provides it with the input image for the assigned variables (green arrow).
At the end of the tasks in task class #4, the variables of the assigned outputs are written to the output image (red arrow).

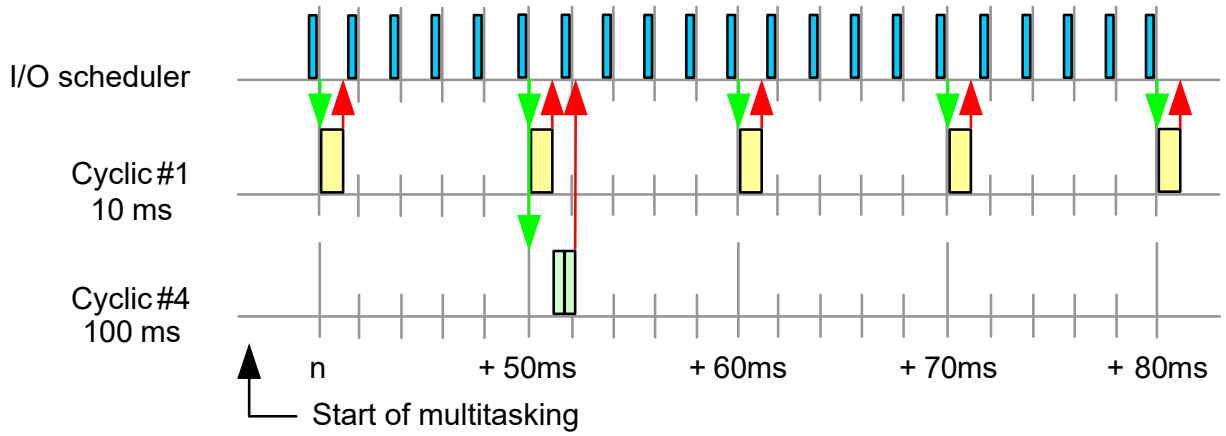


Figure 75: I/O handling for task class #4

Real-time operating system \ Target systems \Target systems - SG4 \ I/O management - SG4

6.2.1 Startup

Booting the controller transfers the I/O configuration to the I/O modules and initializes the interfaces and fieldbus devices. This occurs even before the programs are initialized. This ensures that valid I/O data can be accessed in the initialization subroutine.

6.2.2 Mapping I/O data

The following image displays the relationship between the I/O configuration and the I/O mapping.
The I/O configuration is transferred to the module when the controller boots up or when a configured I/O module is recognized.

In cyclic operation, the input data is provided as a memory block and distributed by the I/O mapping to the configured process variables. In order to write the output data, the process variables are collected and transferred as a memory block to the I/O system.

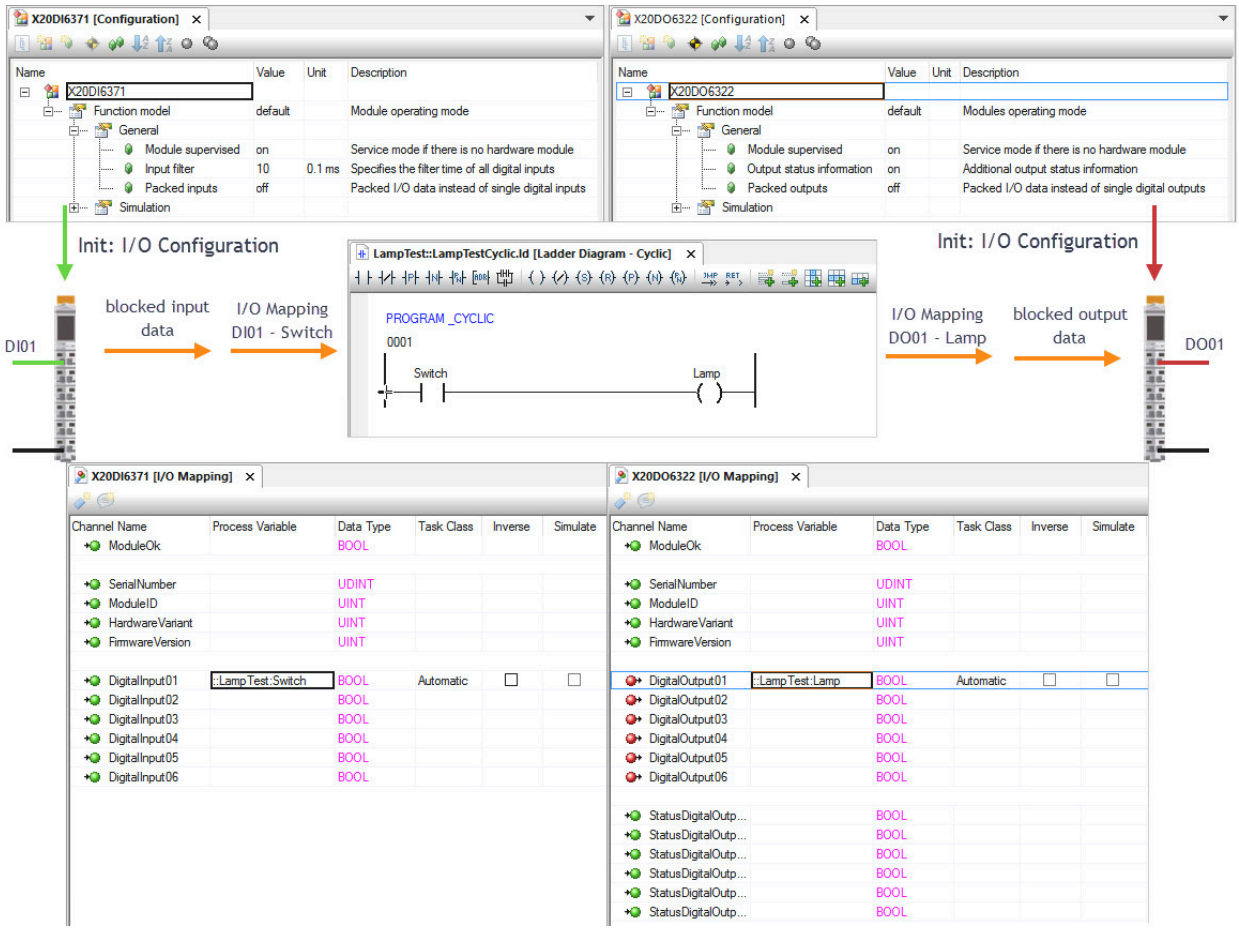


Figure 76: Basic I/O functionality

6.2.3 Settings for the timing characteristics of I/O images

There are configuration options available for the task classes for the chronological handling of I/O images. The standard case is the immediate writing of the outputs when ending the last task in the task class. This causes jitter on the outputs when the task runtime fluctuates.

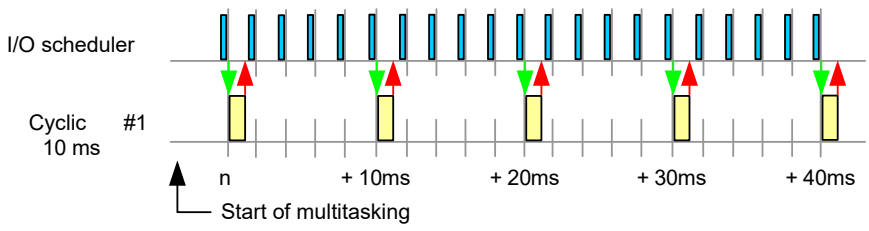


Figure 77: Outputs are transferred immediately to the I/O system after the task has ended.

Alternatively, the time when the input images are read and the output images written can be delayed via the CPU settings for the task class.
If for task class #1 the parameter "I/O output delay" is configured to the value "To end of cycle". This way output data is only transferred to the I/O scheduler at the end of the task class. The effect of this is that even during fluctuating task runtime, the outputs of a task class are written jitter-free.

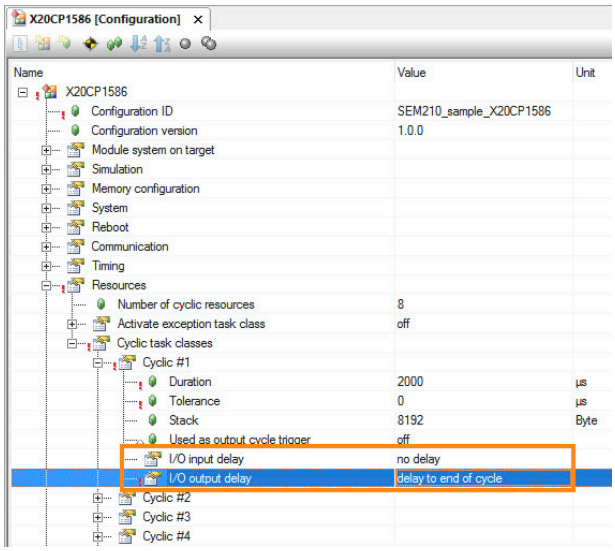


Figure 78: Configuration for the deceleration of the output data to the end of the cycle

The graphic shows the effect of the delay of the output image at the end of the cycle.

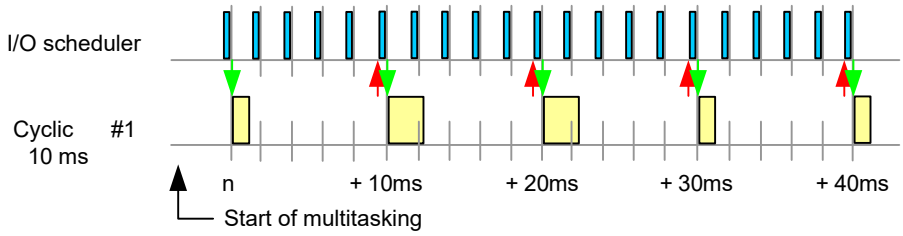


Figure 79: Jitter-free writing of output image at end of cycle

Real-time operating system \ Target systems \Target systems - SG4 \ I/O management - SG4
Real-time operating system \ Target systems \Target systems - SG4 \ I/O management - SG4 \ Method of operation

6.3 Error handling for I/O modules

Every configured I/O module is monitored by the I/O system. The user can configure how the system responds to error situations.

The "**Module monitoring**" property in the I/O Configuration can be used to enable or disable the monitoring of an I/O module.

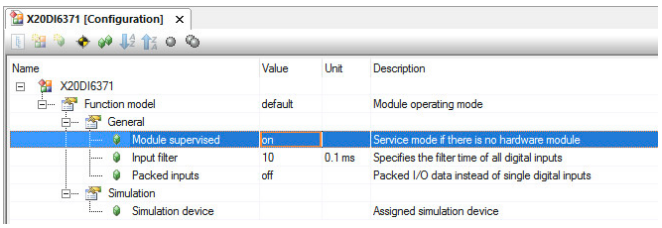


Figure 80: Configuration of module monitoring

Module monitoring enabled

If the module monitoring is enabled, the following states lead to a restart in the "SERVICE" operating mode:

- The module defined for a slot is not present (or not connected).
- The module physically connected in a slot doesn't match the module actually configured for the slot.
- The module is not addressed anymore during operation by the I/O system.

If a missing, incorrectly configured or incorrectly inserted I/O module is recognized on startup or during runtime, then this is logged in the "System" Logger file.

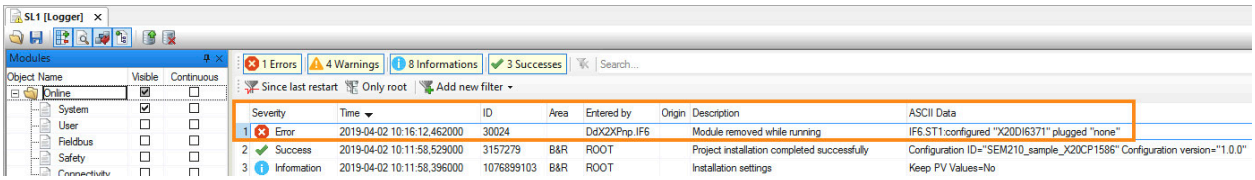


Figure 81: Logger entry "module removed while running"

Which module caused the Logger entry?
The slot of the module can be identified using the ASCII data in the Logger entry. "IF6.ST3" means that the "ST3" module on the IF6 interface, which is the X2X interface in this case, was disconnected. This is the third slot on that bus. The CPU interface names can be viewed in the Physical View as well as in the data sheet of the CPU being used.

Module monitoring disabled

When module monitoring is disabled, the I/O module can be monitored from the application by variable mapping on the "**ModuleOK**" channel. Missing or incorrectly inserted modules don't cause a reboot in the SERVICE operating mode. Only the value of the "ModuleOk" channel becomes "FALSE". The monitoring of the modules is therefore the responsibility of the application.

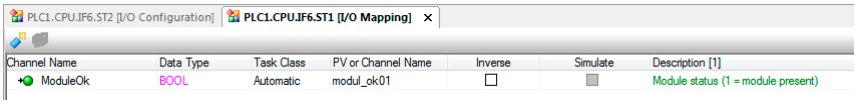


Figure 82: Module monitoring via the application

Real-time operating system \ Target systems \Target systems - SG4 \ I/O management - SG4 \ Method of operation \ Error handling

Exercise: Remove bus connection or I/O module during operation

While the controller is running, either the connection to the X20 bus controller is removed or the I/O module is removed. The controller interrupts the application and restarts in the "SERVICE" operating mode. The Logger file is read and the entries are analyzed.

1) Remove bus station or I/O module during operation

2) Wait for restart and startup in "SERVICE" operating mode

3) Read and analyze the Logger file "System"

6.4 reACTION Technology

The response times that a control system is able to achieve represent a key performance factor. High-end controllers, a fast real-time network and intelligent system architectures can push the limit – often with considerable effort – down to 100 µs or lower. The market still demands even shorter response times, however.

Through the use of reACTION Technology, the execution of small logic programs is passed on from the controller directly to an I/O module. reACTION Technology programs run on the reACTION I/O module with a very fast cycle time, which allows response times in the µs range.
In addition to the short response times made possible by the reACTION I/O module, the load on the controller and fieldbus is also reduced.

reACTION Technology programs are configured in the Logical View. When transferring the project, the reACTION Technology program is installed directly on the I/O module and executed there.
For configuration and diagnosis of reACTION Technology, data points are available in I/O mapping on the reACTION I/O modules.

Programming \ Programs \ reACTION Technology

- Technology description
- Examples for use of reACTION Technology
- reACTION diagram

7 Summary

The operating system serves as an interface between the application and the hardware. The resources of the target systems are managed uniformly by the operating system.
Multitasking in Automation Runtime makes it possible to design an application with a modular structure. Because the application is divided into individual tasks and these tasks run in different task classes, resources are used optimally. Your own requirements can be fully met by adapting the multitasking system to the time response of the application. The result is an optimally configured, powerful application that makes targeted use of existing resources.

Figure 83: Automation Runtime: A software platform for the entire B&R product range

The platform-independent configurable interfaces as well as client and server protocols make Automation Runtime a powerful operating system. The openness of the system makes it possible to directly integrate devices from other manufacturers, establish connections to OPC UA clients and servers and implement the required IT safety mechanisms. The state of Automation Runtime is always transparent thanks to several diagnostic tools and software libraries. This helps with commissioning and the implementation of individual application requirements.

Automation Academy

Your knowledge advantage

The Automation Academy provides **targeted training** courses for our customers as well as for our own employees. Expand your skills in the field of automation technology and learn to independently implement **efficient automation solutions** using B&R systems.

Decide for yourself which **learning concept** you prefer!

Classroom Learning

Virtual Classroom Learning

Online courses

Classroom learning

B&R offers **standard seminars** at all B&R locations. Services include seminar documents, effective communication of training course content by experienced trainers and an Automation Diploma. A combination of group work and self-study provides the high level of flexibility needed to maximize the learning experience.

Virtual classroom

Remote Lectures supplement B&R's continuing education portfolio with a virtual classroom, offering an alternative to our on-site seminars. Selected content from our standard seminars is offered online. In addition to remote learning methods, powerful simulation tools and secure remote maintenance are used.

Online courses

Take control of the content and learn at your own pace. With B&R **online courses**, you can take your first steps in the world of B&R automation at any time. Based on a comprehensive narrative, you will independently work out how to use our products. The mix of different media allows a logical sequence to be followed when learning as well as a targeted choice of information to be used as a reference.

Contact

Would you like additional training? Are you interested in finding out what the B&R Automation Academy has to offer? If so, this is the right place.
Access additional information here:
https://www.br-automation.com/de/academy/

Enjoy your next training course!

AutomationAcademy

B&R
Industrial Automation GmbH
A member of the ABB Group
B&R Straße 1
5142 Eggelsberg, Austria
office@br-automation.com

t +43 7748 6586-0
f +43 7748 6586-26

br-automation.com

TM213TRE.462-ENG