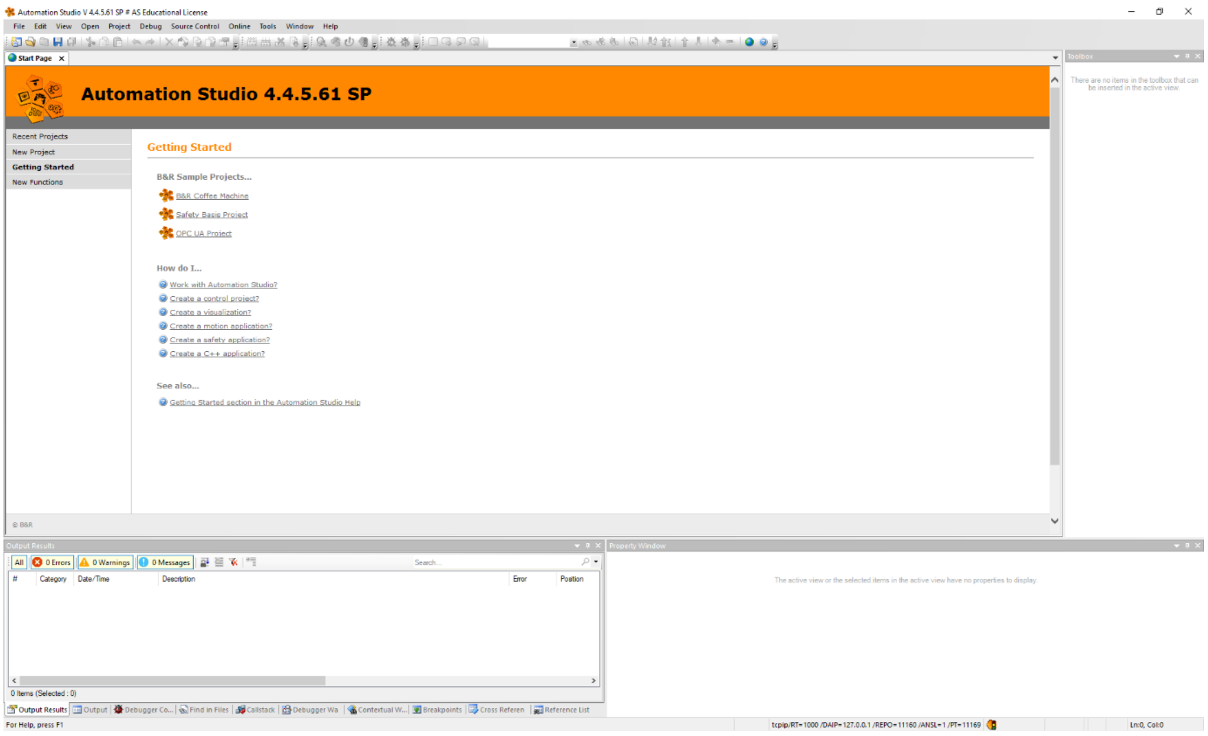
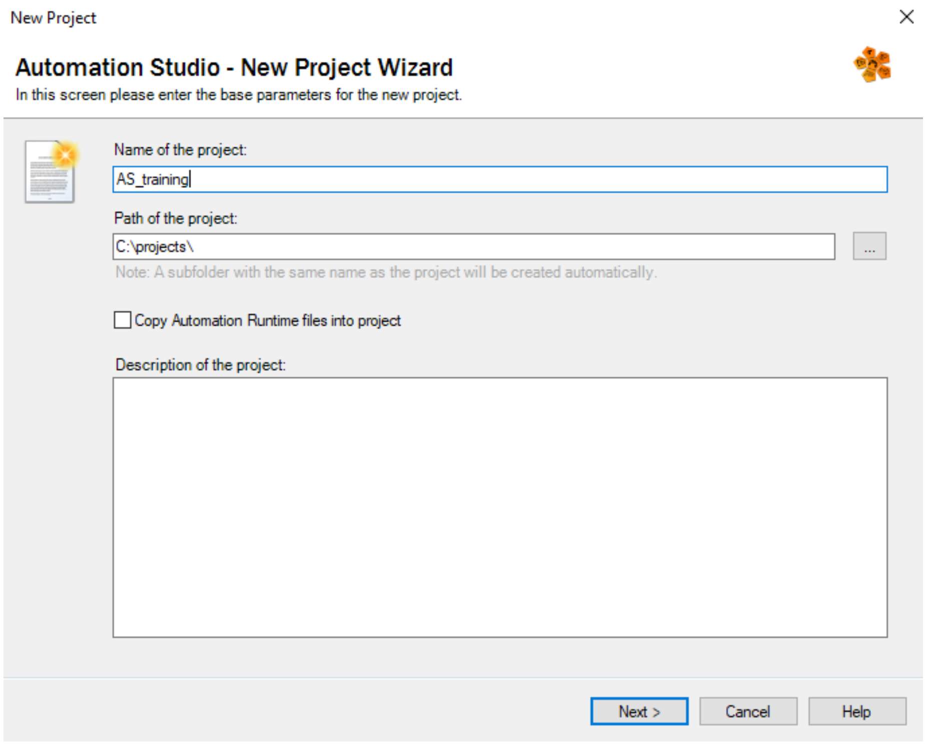
Automation Studio Training – Introduction

1. Open Automation Studio V4


Automation Studio V 4.4.5.61 SP # AS Educational License
File Edit View Open Project Debug Source Control Online Tools Window Help
Start Page
Automation Studio 4.4.5.61 SP
Recent Projects
New Project
Getting Started
New Functions
Getting Started
B&R Sample Projects...
B&R Coffee Machine
Safety Basic Project
OPC UA Project
How do I...
Work with Automation Studio?
Create a control project?
Create a visualization?
Create a motion application?
Create a safety application?
Create a C++ application?
See also...
Getting Started section in the Automation Studio Help
Toolbox
There are no items in the toolbox that can be inserted in the active view.
Output Results
All 0 Errors 0 Warnings 0 Messages
# Category Date/Time Description Error Position
0 Items (Selected : 0)
Property Window
The active view or the selected items in the active view have no properties to display.
For Help, press F1
tcpip/RT=1000 /DAIP=127.0.0.1 /REPO=11160 /ANSL=1 /PT=11169
Ln:0, Col:0

2. Create new project (File - > New Project)



New Project

Automation Studio - New Project Wizard

In this screen please enter the base parameters for the new project.

Name of the project: AS_training

Path of the project: C:\projects\

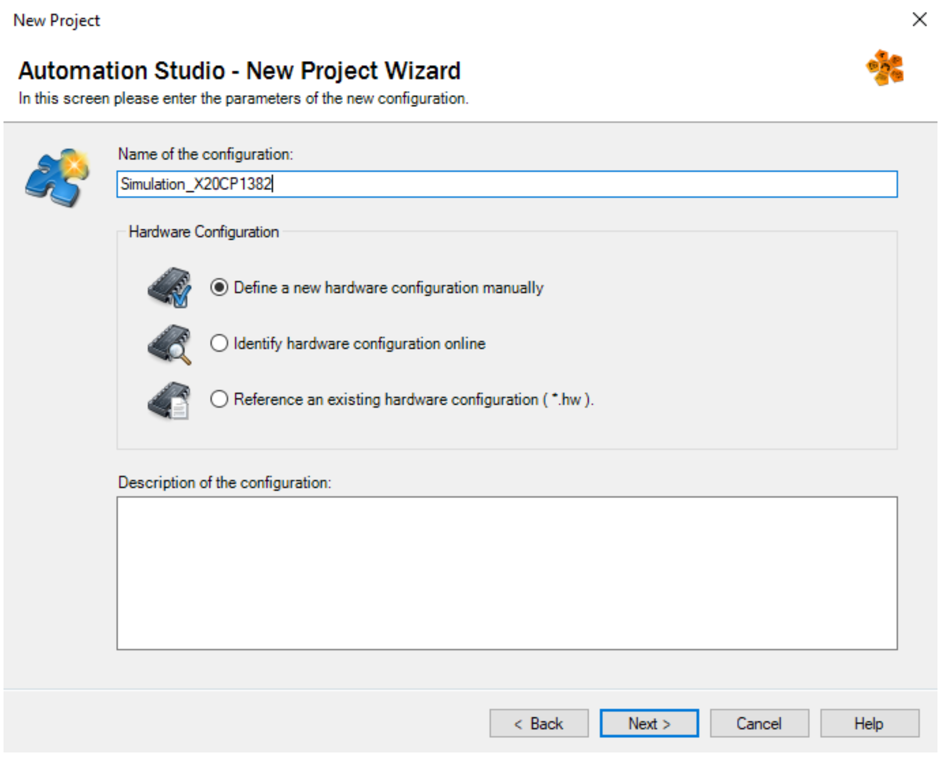
Note: A subfolder with the same name as the project will be created automatically.

☐ Copy Automation Runtime files into project

Description of the project:

Next > Cancel Help

3. Create new Hardware Configuration



New Project

Automation Studio - New Project Wizard

In this screen please enter the parameters of the new configuration.

Name of the configuration: Simulation_X20CP1382

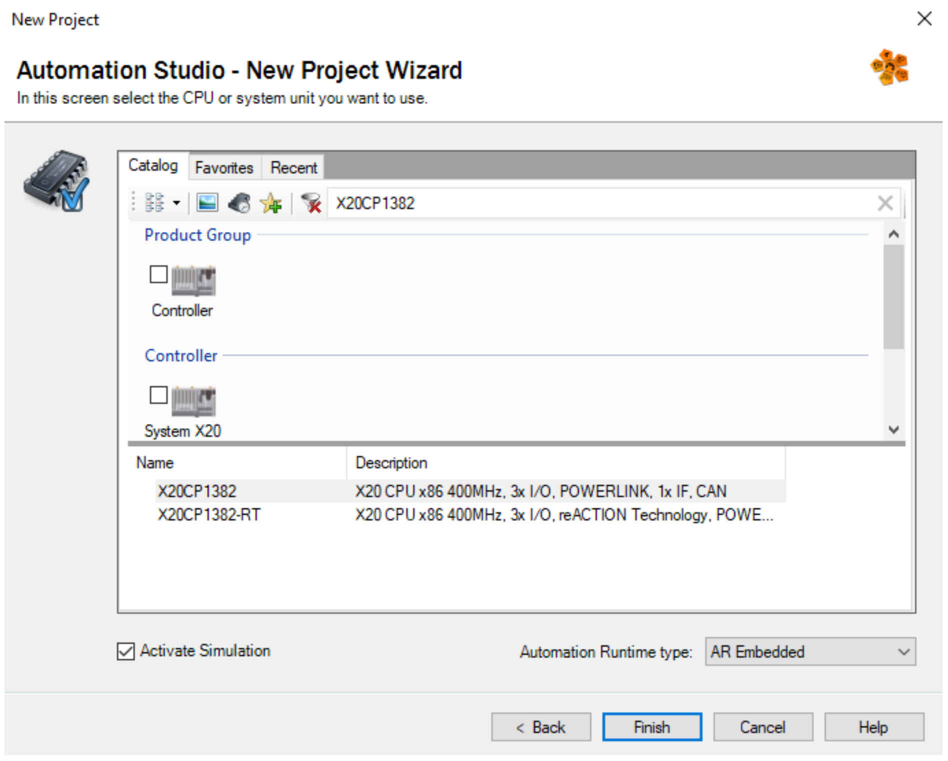
Hardware Configuration

[✓ Define a new hardware configuration manually]

Identify hardware configuration online

Reference an existing hardware configuration (*.hw).

Description of the configuration:



New Project

Automation Studio - New Project Wizard

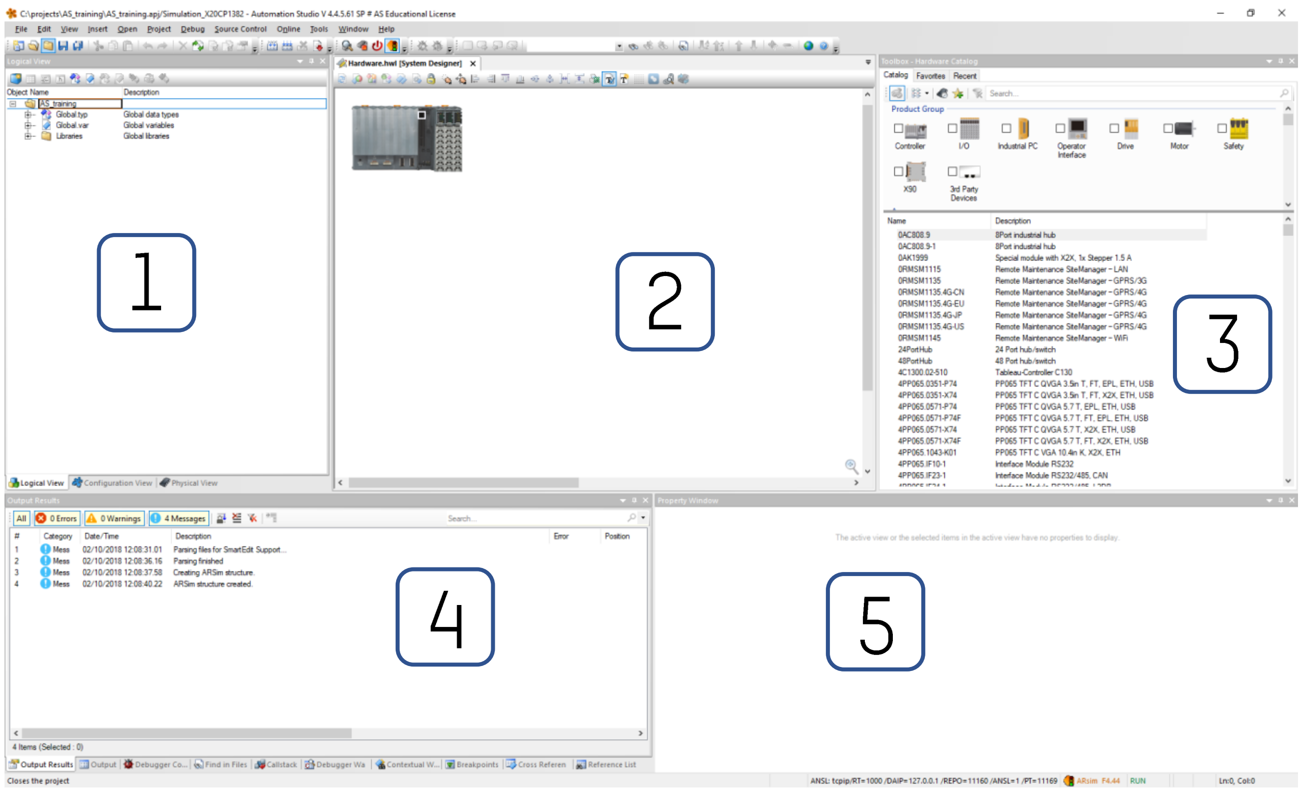
In this screen select the CPU or system unit you want to use.

Name	Description
X20CP1382	X20 CPU x86 400MHz, 3x I/O, POWERLINK, 1x IF, CAN
X20CP1382-RT	X20 CPU x86 400MHz, 3x I/O, reACTION Technology, POWE...

[✓] Activate Simulation

Automation Runtime type: AR Embedded

4. The workspace



Name	Description
0AC808.9	8Port industrial hub
0AC808.9-1	8Port industrial hub
0AK1999	Special module with X2X, 1x Stepper 1.5 A
0RMSM1115	Remote Maintenance SiteManager – LAN
0RMSM1135	Remote Maintenance SiteManager – GPRS/3G
0RMSM1135.4G-CN	Remote Maintenance SiteManager – GPRS/4G
0RMSM1135.4G-EU	Remote Maintenance SiteManager – GPRS/4G
0RMSM1135.4G-JP	Remote Maintenance SiteManager – GPRS/4G
0RMSM1135.4G-US	Remote Maintenance SiteManager – GPRS/4G
0RMSM1145	Remote Maintenance SiteManager – WiFi
24PortHub	24 Port hub/switch
48PortHub	48 Port hub/switch
4C1300.02-510	Tableau-Controller C130
4PP065.0351-P74	PP065 TFT C QVGA 3.5in T, FT, EPL, ETH, USB
4PP065.0351-X74	PP065 TFT C QVGA 3.5in T, FT, X2X, ETH, USB
4PP065.0571-P74	PP065 TFT C QVGA 5.7 T, EPL, ETH, USB
4PP065.0571-P74F	PP065 TFT C QVGA 5.7 T, FT, EPL, ETH, USB
4PP065.0571-X74	PP065 TFT C QVGA 5.7 T, X2X, ETH, USB
4PP065.0571-X74F	PP065 TFT C QVGA 5.7 T, FT, X2X, ETH, USB
4PP065.1043-K01	PP065 TFT C VGA 10.4in K, X2X, ETH
4PP065.IF10-1	Interface Module RS232
4PP065.IF23-1	Interface Module RS232/485, CAN

#	Category	Date/Time	Description	Error	Position
1	Mess	02/10/2018 12:08:31.01	Parsing files for SmartEdit Support...		
2	Mess	02/10/2018 12:08:36.16	Parsing finished		
3	Mess	02/10/2018 12:08:37.58	Creating ARSim structure.		
4	Mess	02/10/2018 12:08:40.22	ARSim structure created.		

1) The Project Explorer is located on the left side of the window. This is used to manage and edit software and configuration objects in a project.

2) The center of the screen is where open documents are worked on. This is where program code is edited, for example.

3) The Toolbox window is located on the right side of the window. Depending on what object you are currently working on, the Toolbox window allows you to select and add hardware modules, program functions or software objects.

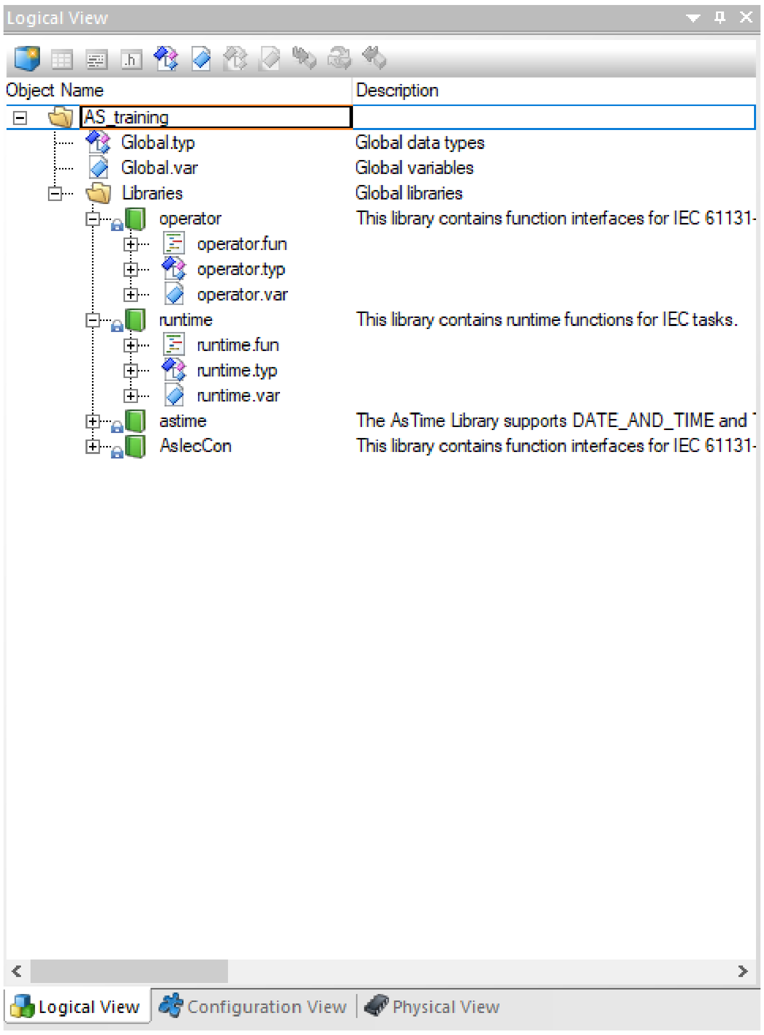
4) The output window is located on the lower left side of the main window. It is used to display information such as messages that are generated when a project is being built.

5) The Properties window is located on the bottom right. This window displays configuration options for whichever object or hardware module is currently selected. It can also be used to edit the properties of selected object.

5. Logical View

The logical view represents the hardware-independent view of the application.

It is located under the first tab of the project explorer. It displays and manages data type declarations, variable declarations, packages, programs, libraries, data objects, and documentation files.



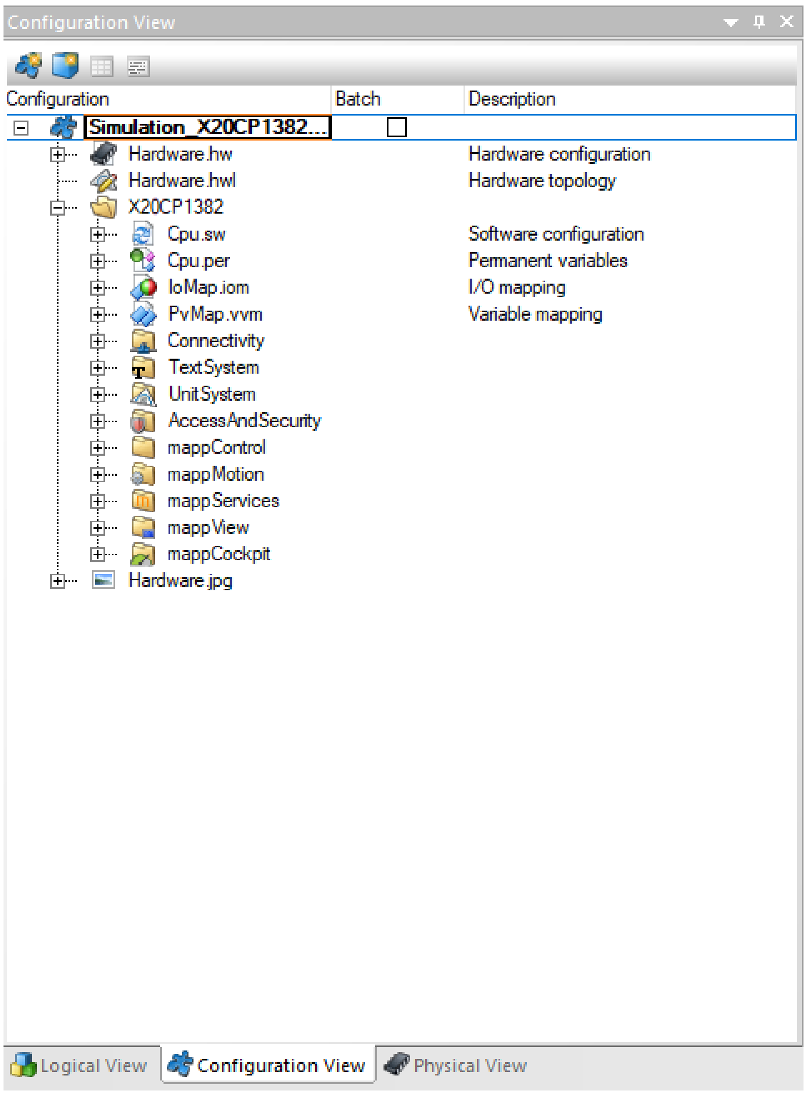
Logical View

Object Name	Description
AS_training	
Global.typ	Global data types
Global.var	Global variables
Libraries	Global libraries
operator	This library contains function interfaces for IEC 61131-
operator.fun	
operator.typ	
operator.var	
runtime	This library contains runtime functions for IEC tasks.
runtime.fun	
runtime.typ	
runtime.var	
astime	The AsTime Library supports DATE_AND_TIME and
AslecCon	This library contains function interfaces for IEC 61131-

6. Configuration View

The configuration view is the hardware-dependent view of the application. It is found on the second tab of the project explorer.

The configuration view allows you to display and manage the hardware configuration, the files for software configuration, declaration of permanent variables, I/O mapping tables, NC configuration, NC mapping tables, VC keyboard format, configuration specific data objects and documentation files.

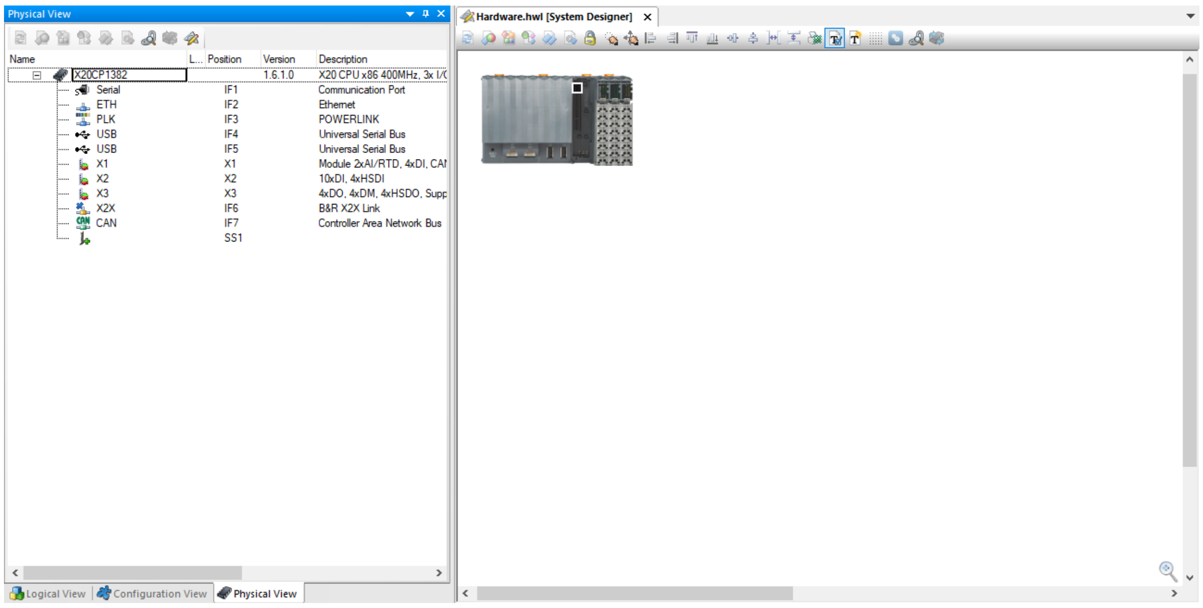


Configuration	Batch	Description
Simulation_X20CP1382...	☐	
Hardware.hw		Hardware configuration
Hardware.hwl		Hardware topology
X20CP1382		
Cpu.sw		Software configuration
Cpu.per		Permanent variables
IoMap.iom		I/O mapping
PvMap.vvm		Variable mapping
Connectivity		
TextSystem		
UnitSystem		
AccessAndSecurity		
mappControl		
mappMotion		
mappServices		
mappView		
mappCockpit		
Hardware.jpg		

7. Physical View

Automation Studio can show configured hardware in either a hierarchical or a graphical view. The hierarchical view is called the Physical View; the graphical view is found in System Designer. Although they both illustrate the same hardware structure, the way it is displayed is vastly different.

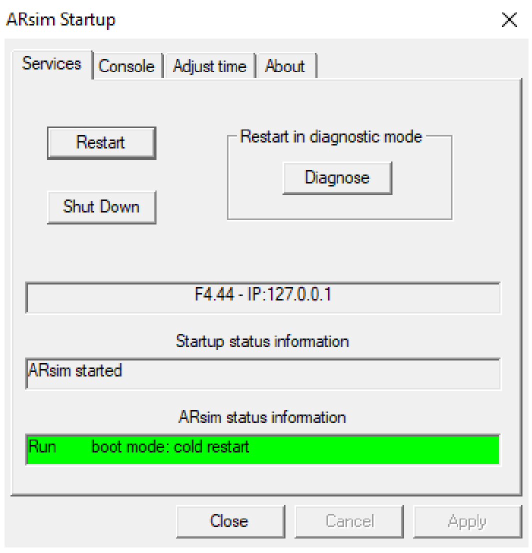
The following images demonstrate an example of a hardware structure based on an X201382 CPU.



Name	L...	Position	Version	Description
X20CP1382			1.6.1.0	X20 CPU x86 400MHz, 3x I/(
Serial		IF1		Communication Port
ETH		IF2		Ethernet
PLK		IF3		POWERLINK
USB		IF4		Universal Serial Bus
USB		IF5		Universal Serial Bus
X1		X1		Module 2xAI/RTD, 4xDI, CAI
X2		X2		10xDI, 4xHSDI
X3		X3		4xDO, 4xDM, 4xHSDO, Supp
X2X		IF6		B&R X2X Link
CAN		IF7		Controller Area Network Bus
		SS1		

8. Automation Runtime

Automation Runtime, the software kernel which allows applications to run on a target system, is an integral component of Automation Studio™



F4.44 - IP:127.0.0.1

Startup status information

ARsim started

ARsim status information

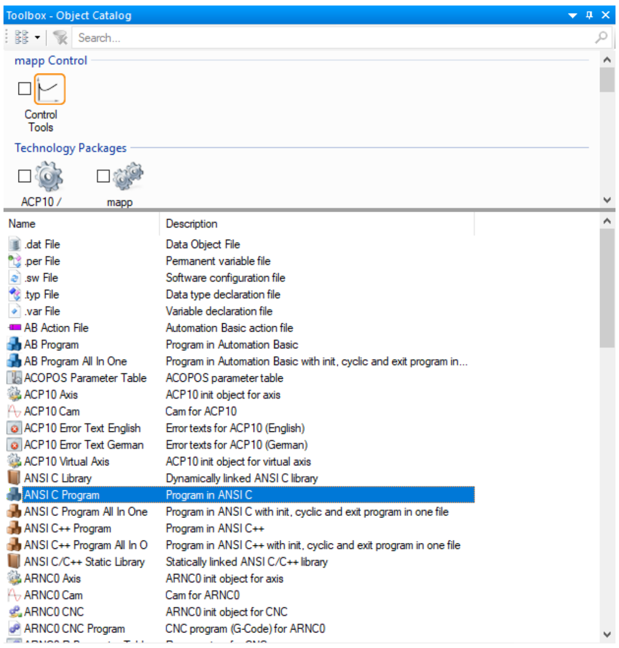
Run boot mode: cold restart

Example project with Automation Runtime Simulation

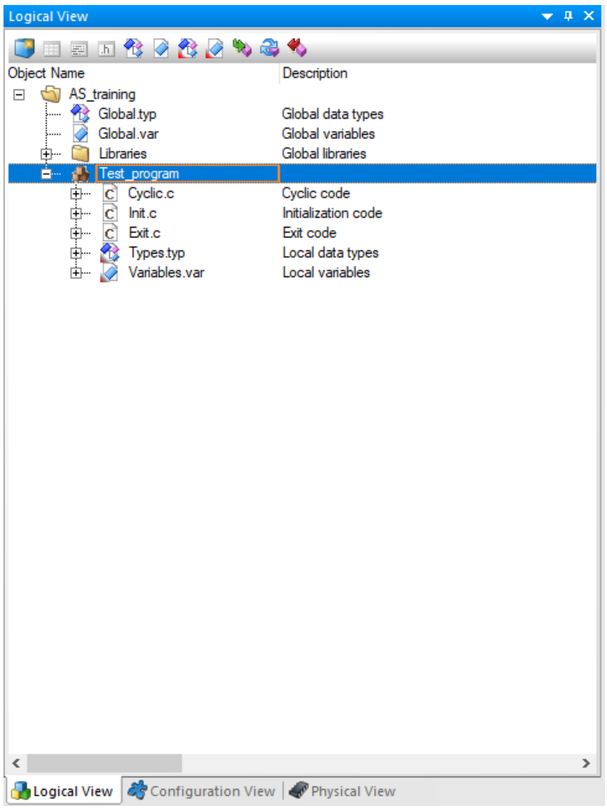
1. Adding a program

After project creation is complete, the first program can now be added.

1. Select **Program** in the Object Catalog.
2. Double-click on **ANSI C program** or use drag-and-drop to add a ANSI C program.
3. The program then appears in the Logical View. Right-clicking on **Program** opens the shortcut menu where the **Rename** menu item can be selected and the name changed to *Test_program*.



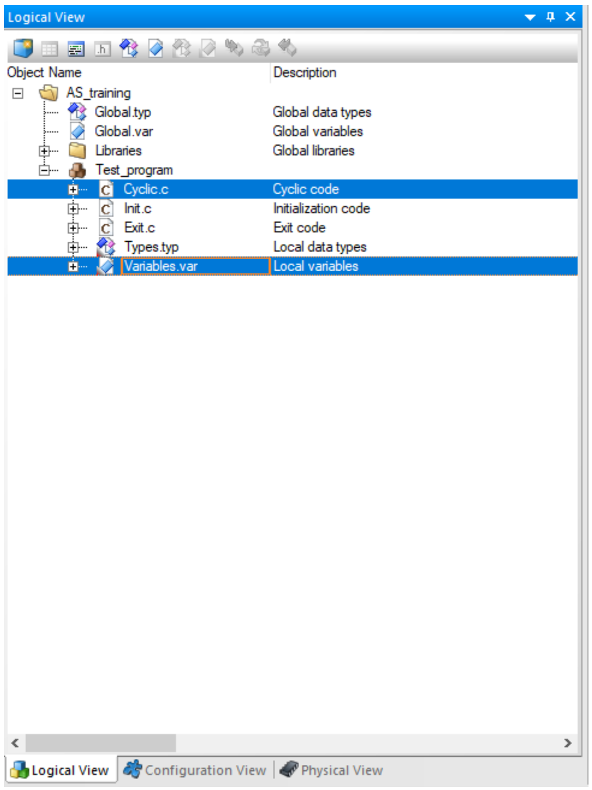
Name	Description
.dat File	Data Object File
.per File	Permanent variable file
.sw File	Software configuration file
.typ File	Data type declaration file
.var File	Variable declaration file
AB Action File	Automation Basic action file
AB Program	Program in Automation Basic
AB Program All In One	Program in Automation Basic with init, cyclic and exit program in...
ACOPOS Parameter Table	ACOPOS parameter table
ACP10 Axis	ACP10 init object for axis
ACP10 Cam	Cam for ACP10
ACP10 Error Text English	Error texts for ACP10 (English)
ACP10 Error Text German	Error texts for ACP10 (German)
ACP10 Virtual Axis	ACP10 init object for virtual axis
ANSI C Library	Dynamically linked ANSI C library
ANSI C Program	Program in ANSI C
ANSI C Program All In One	Program in ANSI C with init, cyclic and exit program in one file
ANSI C++ Program	Program in ANSI C++
ANSI C++ Program All In O	Program in ANSI C++ with init, cyclic and exit program in one file
ANSI C/C++ Static Library	Statically linked ANSI C/C++ library
ARNC0 Axis	ARNC0 init object for axis
ARNC0 Cam	Cam for ARNC0
ARNC0 CNC	ARNC0 init object for CNC
ARNC0 CNC Program	CNC program (G-Code) for ARNC0



Object Name	Description
AS_training	
Global.typ	Global data types
Global.var	Global variables
Libraries	Global libraries
Test_program	
Cyclic.c	Cyclic code
Init.c	Initialization code
Exit.c	Exit code
Types.typ	Local data types
Variables.var	Local variables

2. Program

Now is a program that controls a lamp will be created.
In the Logical View, opening the *Test_program* program makes the variable declaration file and the cyclic portion of the program visible.



Logical View

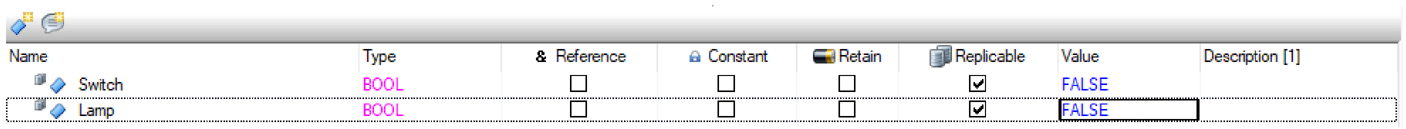
Object Name	Description
AS_training	
Global.typ	Global data types
Global.var	Global variables
Libraries	Global libraries
Test_program	
Cyclic.c	Cyclic code
Init.c	Initialization code
Exit.c	Exit code
Types.typ	Local data types
Variables.var	Local variables

Logical View | Configuration View | Physical View

3. Creating the variables

Double-clicking on *Variables.var* opens the file. Pressing the **INS** key or a clicking on the **Add variable** icon adds a new variable. The variables *Switch* and *Lamp* can now be created with data type *BOOL* and the value set to *FALSE*. After saving (**Ctrl + S**), the file should look like this.

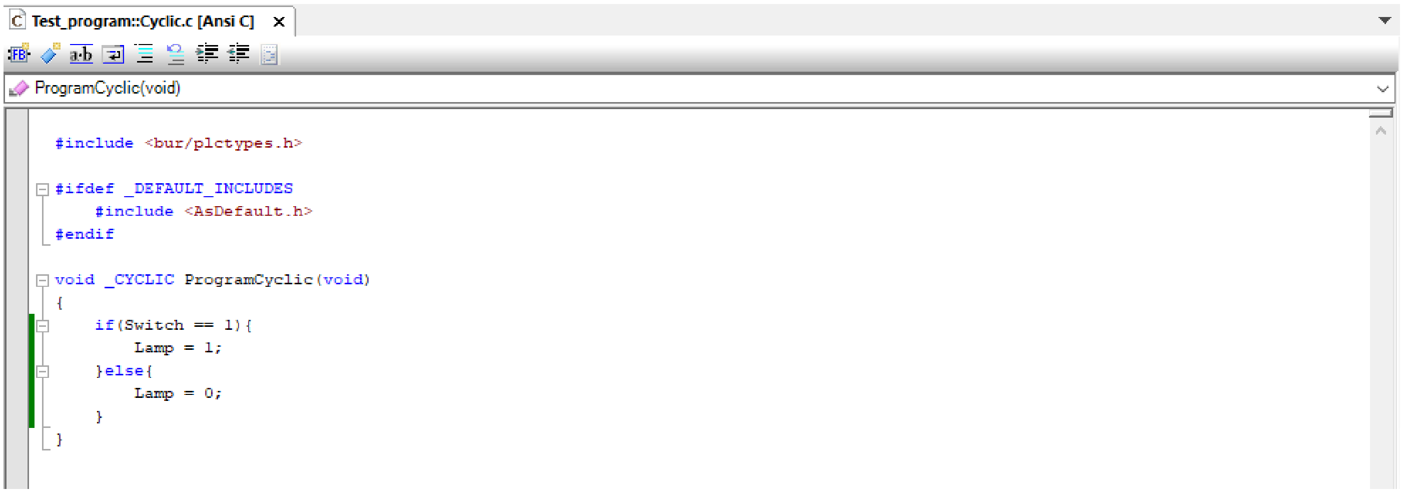
It is necessary to save the file in order to be able to continue to use the variables that have been created.



Name	Type	Reference	Constant	Retain	Replicable	Value	Description [1]
Switch	BOOL	☐	☐	☐	☑	FALSE	
Lamp	BOOL	☐	☐	☐	☑	FALSE	

4. Creating a program

The ANSI C editor is opened by double-clicking on *Cyclic.c*.



```c
#include <bur/plctypes.h>

#ifdef _DEFAULT_INCLUDES
    #include <AsDefault.h>
#endif

void _CYCLIC ProgramCyclic(void)
{
    if(Switch == 1){
        Lamp = 1;
    }else{
        Lamp = 0;
    }
}
```

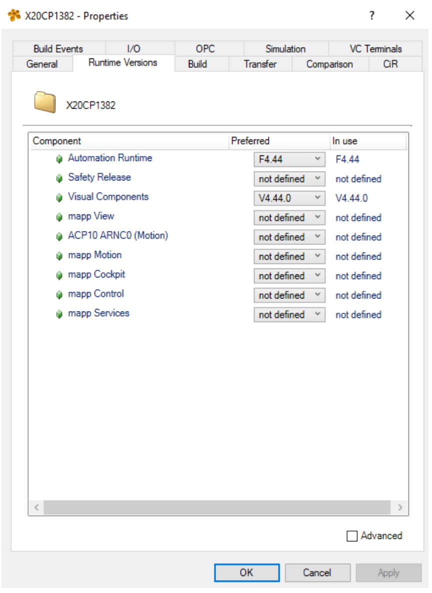
5. Set Automation Runtime

Before the project is compiled, check Automation Runtime version.

To set the Automation Runtime version, open the target system properties via the menu

Project / Change Runtime Version....

1. Settings in the target properties dialog
   - Automation Runtime: *AR versions 4.33 and higher*
2. Save the settings with *OK* and proceed to the next step.



Component	Preferred	In use
Automation Runtime	F4.44	F4.44
Safety Release	not defined	not defined
Visual Components	V4.44.0	V4.44.0
mapp View	not defined	not defined
ACP10 ARNC0 (Motion)	not defined	not defined
mapp Motion	not defined	not defined
mapp Cockpit	not defined	not defined
mapp Control	not defined	not defined
mapp Services	not defined	not defined

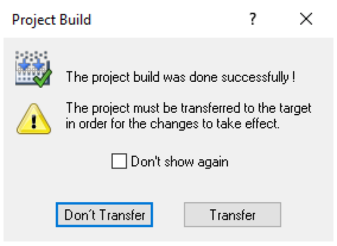
6. Building the project

Before the program is transferred to the target system, the project should be built so that possible errors can be recognized immediately.

Build

The project is built using menu item **Project / Build configuration** or by pressing the **F7 key**. A successful build is indicated in the output window by *Build: 0 error(s), 0 warning(s)*.

After building, the dialog box that initiates project installation appears.


Project Build

The project build was done successfully !

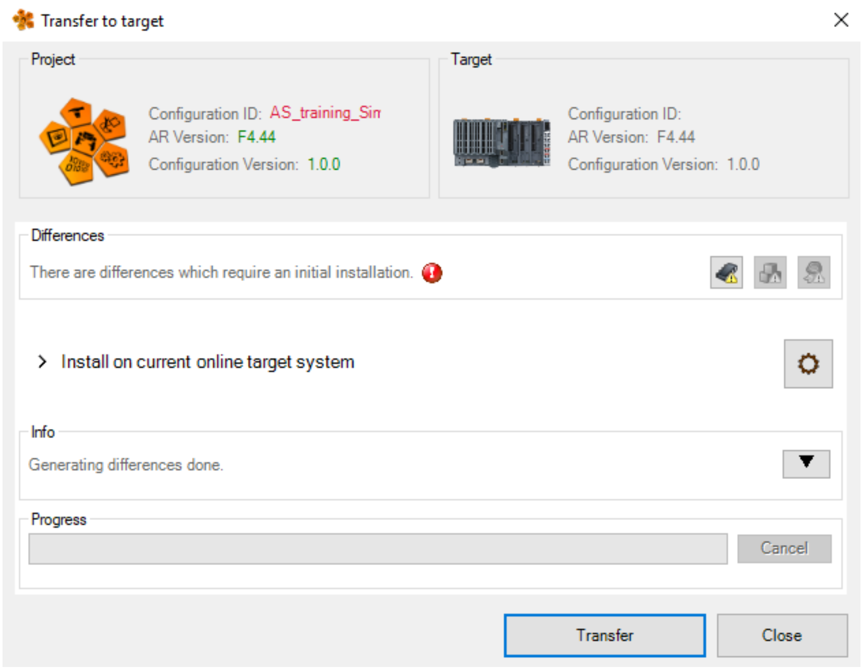
The project must be transferred to the target in order for the changes to take effect.

☐ Don't show again

7. Transferring the project

After the project has been successfully compiled, the next step is to transfer it to the target system. The variant for transferring to the target system during project installation can be used to generate the necessary components and programs (operating system, system components, Automation Studio application, etc.).

1. Transfer to target
   Dialog box **Transfer** appears in the last step. The project installation procedure installs the data on the target system after pressing button **Transfer**.


Transfer to target

Project
Configuration ID: AS_training_Sin
AR Version: F4.44
Configuration Version: 1.0.0

Target
Configuration ID:
AR Version: F4.44
Configuration Version: 1.0.0

Differences
There are differences which require an initial installation.

Install on current online target system

Info
Generating differences done.

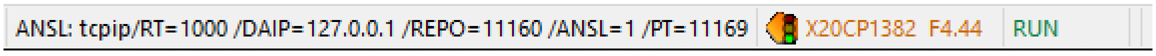
Progress

All stored data is erased when a new image is generated. A warning dialog box will be shown when starting the generation process.

The target system is now updated and will be tested in the next step.

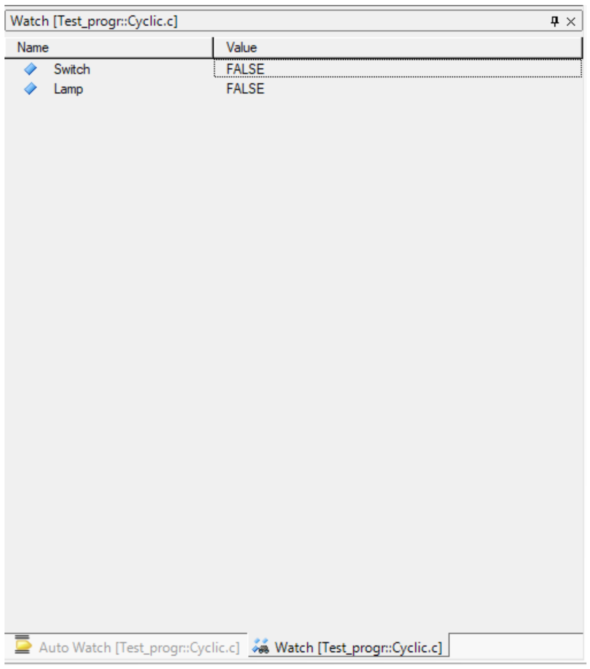
8. Testing the project

Once an Ethernet connection has been established between the workstation and the target system, the status of the online connection is displayed in the status bar.



ANSL: tcpip/RT=1000 /DAIP=127.0.0.1 /REPO=11160 /ANSL=1 /PT=11169 X20CP1382 F4.44 RUN

Open the **Cyclic.c** file in the Logical View to do this. Monitor mode can be started by clicking on the magnifying glass icon in the Online toolbar or using the keyboard shortcut **<CTRL> + M**. Double-clicking on the value of the *Switch* variable allows you to change it from 0 to 1 (FALSE to TRUE). This sets the output (*Lamp*) to the same value.



Watch [Test_progr::Cyclic.c]

Name	Value
Switch	FALSE
Lamp	FALSE

Auto Watch [Test_progr::Cyclic.c]
Watch [Test_progr::Cyclic.c]

9. Cyclic Program

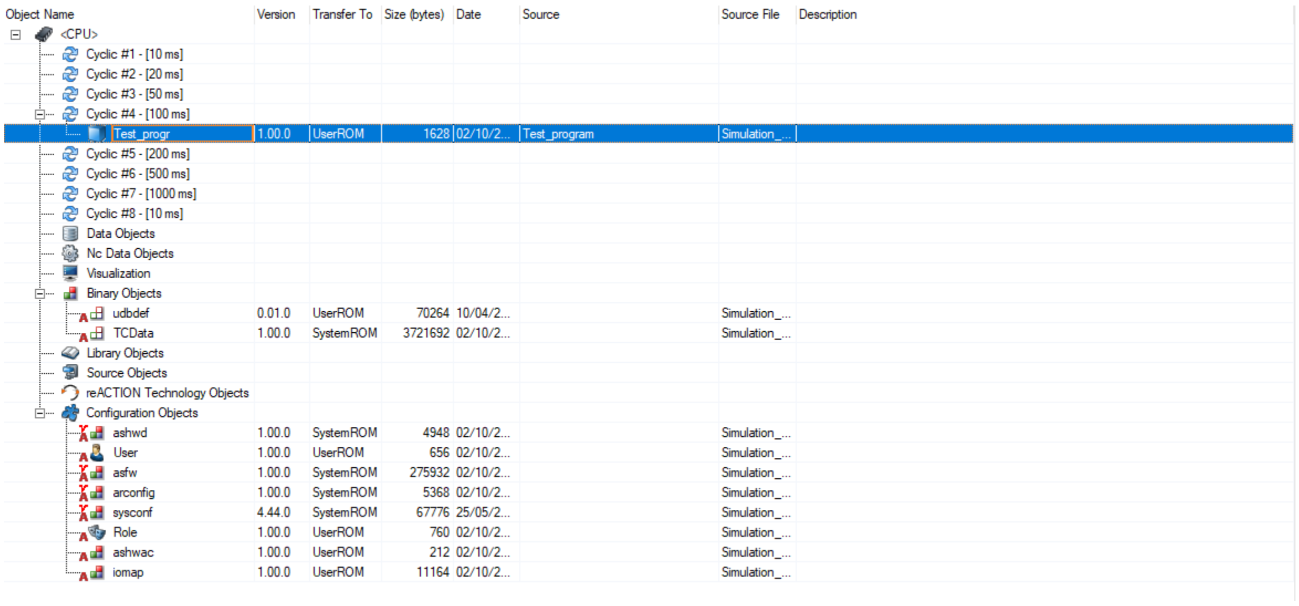
A program is assigned a certain execution time, or task class, in the software configuration. The programs assigned to the software configuration are identified as tasks. The programs that are assigned in the software configuration are only executed after the transfer.

The tasks are assigned to configurable task classes and executed one after the other. The execution time of the tasks may vary. However, the times when the tasks start as well as when the I/O system is accessed are deterministic.

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.

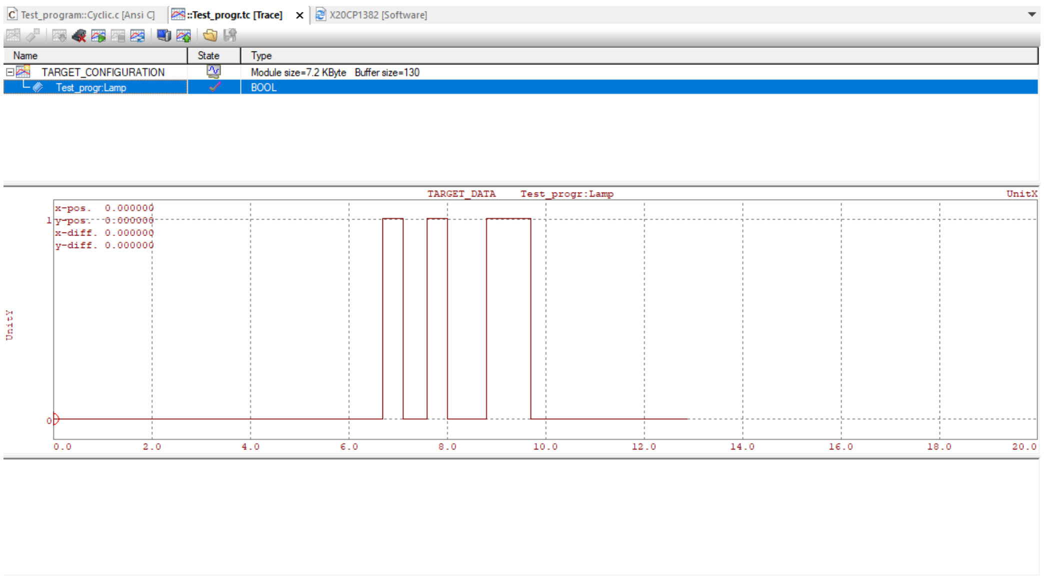
Every task class contains tasks with the same cycle time, priority and tolerance.

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



Object Name	Version	Transfer To	Size (bytes)	Date	Source	Source File	Description
<CPU>							
Cyclic #1 - [10 ms]							
Cyclic #2 - [20 ms]							
Cyclic #3 - [50 ms]							
Cyclic #4 - [100 ms]							
Test_progr	1.00.0	UserROM	1628	02/10/2...	Test_program	Simulation_...	
Cyclic #5 - [200 ms]							
Cyclic #6 - [500 ms]							
Cyclic #7 - [1000 ms]							
Cyclic #8 - [10 ms]							
Data Objects							
Nc Data Objects							
Visualization							
Binary Objects							
udbdef	0.01.0	UserROM	70264	10/04/2...		Simulation_...	
TCData	1.00.0	SystemROM	3721692	02/10/2...		Simulation_...	
Library Objects							
Source Objects							
reACTION Technology Objects							
Configuration Objects							
ashwd	1.00.0	SystemROM	4948	02/10/2...		Simulation_...	
User	1.00.0	UserROM	656	02/10/2...		Simulation_...	
asfw	1.00.0	SystemROM	275932	02/10/2...		Simulation_...	
arconfig	1.00.0	SystemROM	5368	02/10/2...		Simulation_...	
sysconf	4.44.0	SystemROM	67776	25/05/2...		Simulation_...	
Role	1.00.0	UserROM	760	02/10/2...		Simulation_...	
ashwac	1.00.0	UserROM	212	02/10/2...		Simulation_...	
iomap	1.00.0	UserROM	11164	02/10/2...		Simulation_...	

10. Trace



Name	State	Type
TARGET_CONFIGURATION		Module size=7.2 KByte Buffer size=130
Test_progr:Lamp	✓	BOOL