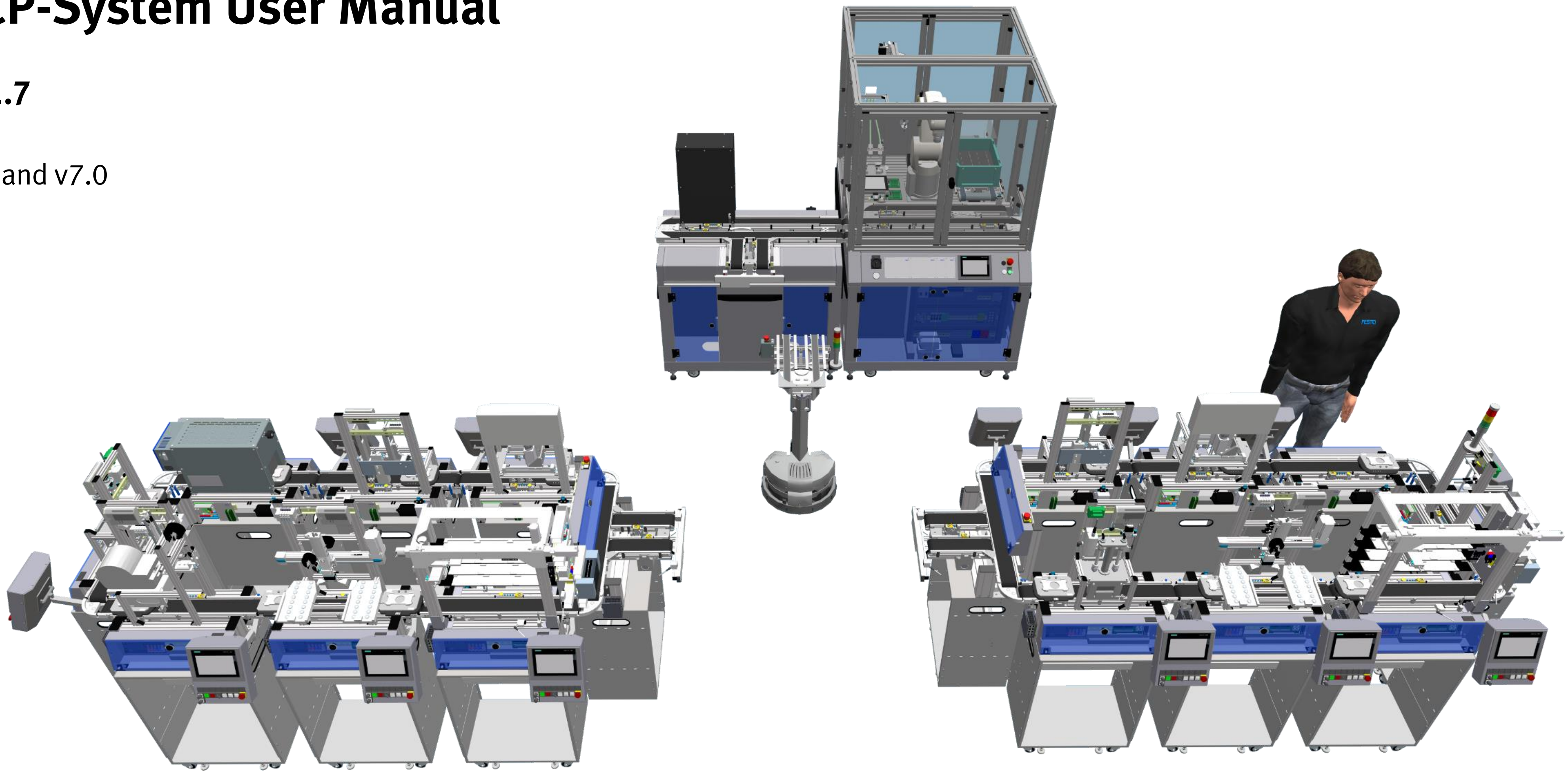
CIROS Festo CP-System User Manual

Based on CIROS v7.1.7

Can be used for v6.4.6 and v7.0
with few variations.



FESTO

Content

Content

FESTO

Content

Content

FESTO

Content

FESTO

Content

11. Robot Programming (2) (→)

- Create / Load Robot Controller Backup
- Upload / Download Robot Programs
- Online Teach-In
- Get Actual Robot Data with Built-In Python Function

12. VR (→)

- Setting Up VR Glasses
- Interact with Model

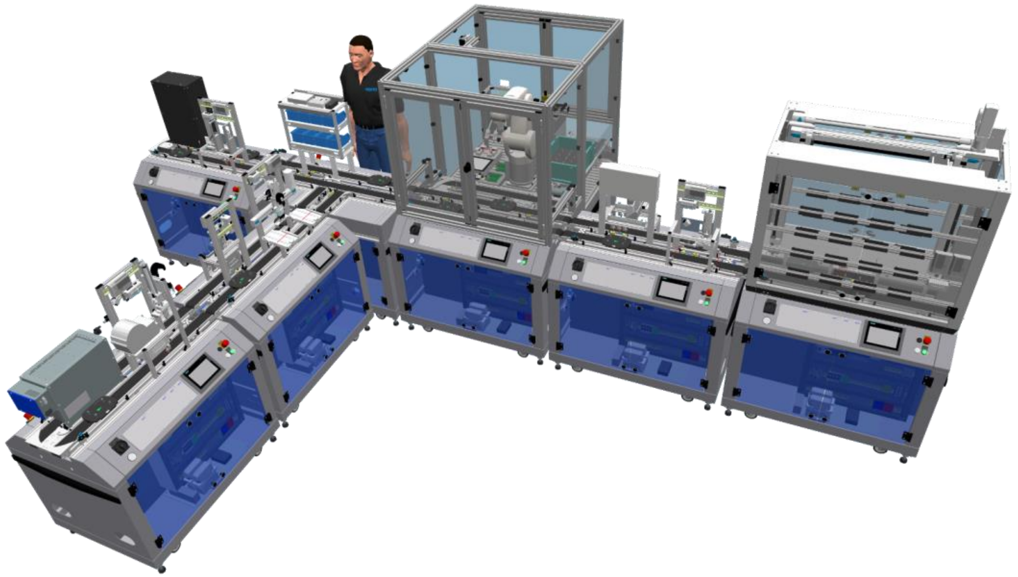
Content

Introduction to CIROS

Overview

CIROS – Computer Integrated RObot Simulation

- Powerful, kinematic real-time 3D simulation
  - Time-discrete simulation kernel (update every 40ms)
  - Large model library: not only CP Lab / Factory, but also >1000 robots from various manufacturers
  - CAD import for user-defined modules / kinematics
  - Several interfaces, e.g. to MES4, Fleet manager, Matlab, Python, VR glasses, Mitsubishi robots, …
  - User interaction during simulation
  - Collision analysis
  - Fault injection / simulation
  - PLC and robot programming
  - Online help with introductory examples



Overview

Studio vs. Education

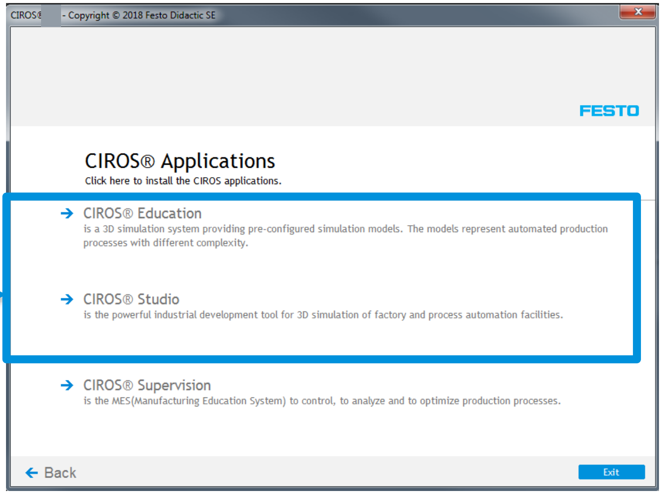
- CIROS Studio
  - Full version including all features
  - Designing and saving models from scratch
  - RCI explorer interface to Mitsubishi robot controllers (download/upload of programs, individual step tracking)
  - Fits perfect for preparation of teaching scenarios to be analyzed by students

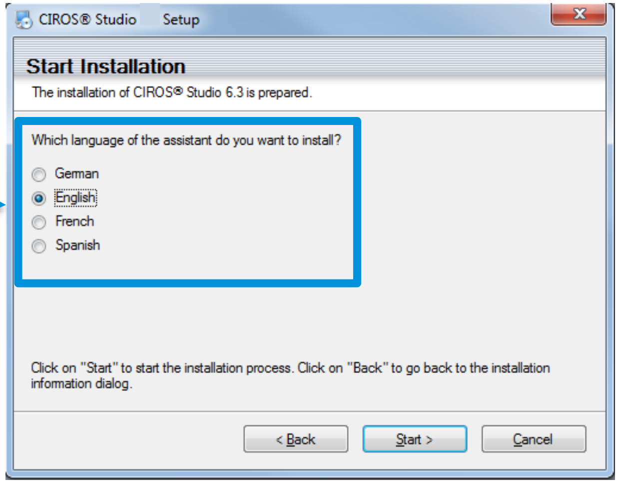
- CIROS Education
  - Limited functionality
  - Already existing models can be analyzed & modified, but not saved
  - No RCI explorer interface to Mitsubishi robot controllers
  - Fits perfect for scenarios, in which CIROS studio models have to be opened, analyzed, and modified only

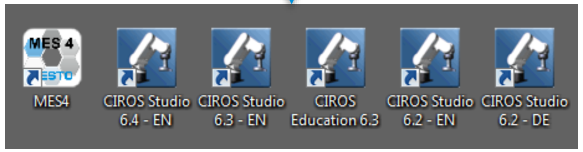
Overview

Language packs & software releases

- CIROS Studio and Education are two different software products.
- Depending on the license key one is allowed to start CIROS Studio and/or Education.
- Installation of different CIROS releases & language packs on a single PC possible.
- Unfortunately, the desired language must be set during installation (there is no option to change the language during runtime) .







FESTO

Overview

Hardware requirements

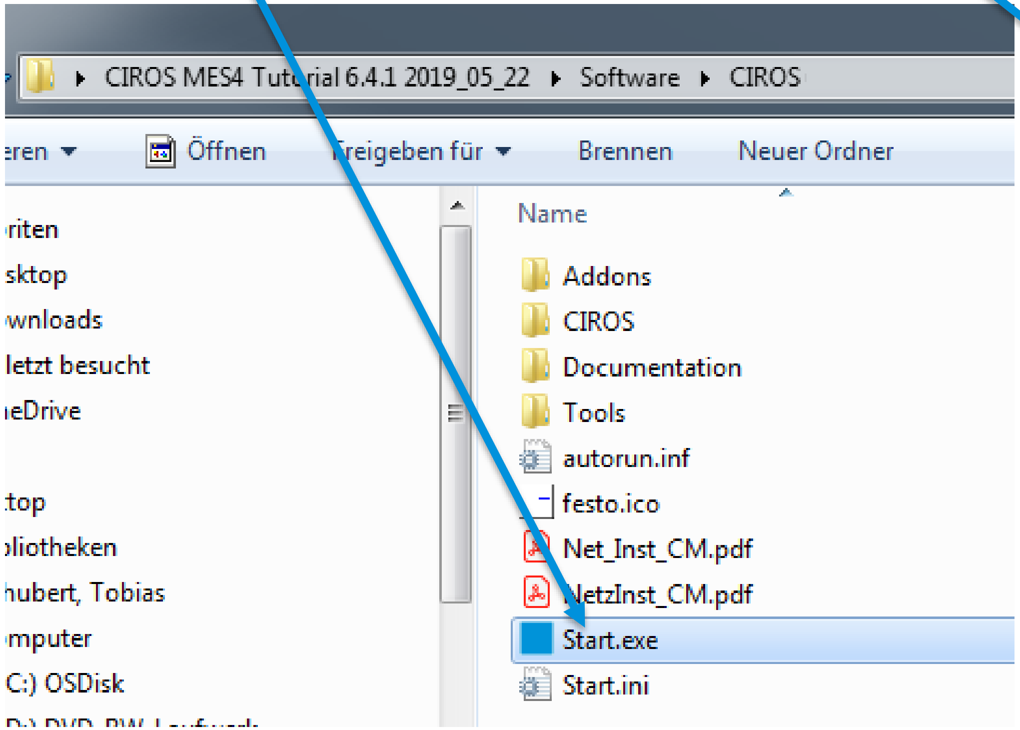
- Either USB port for USB dongle or network access for server-based licensing
- Officially supported operating system: Windows 10

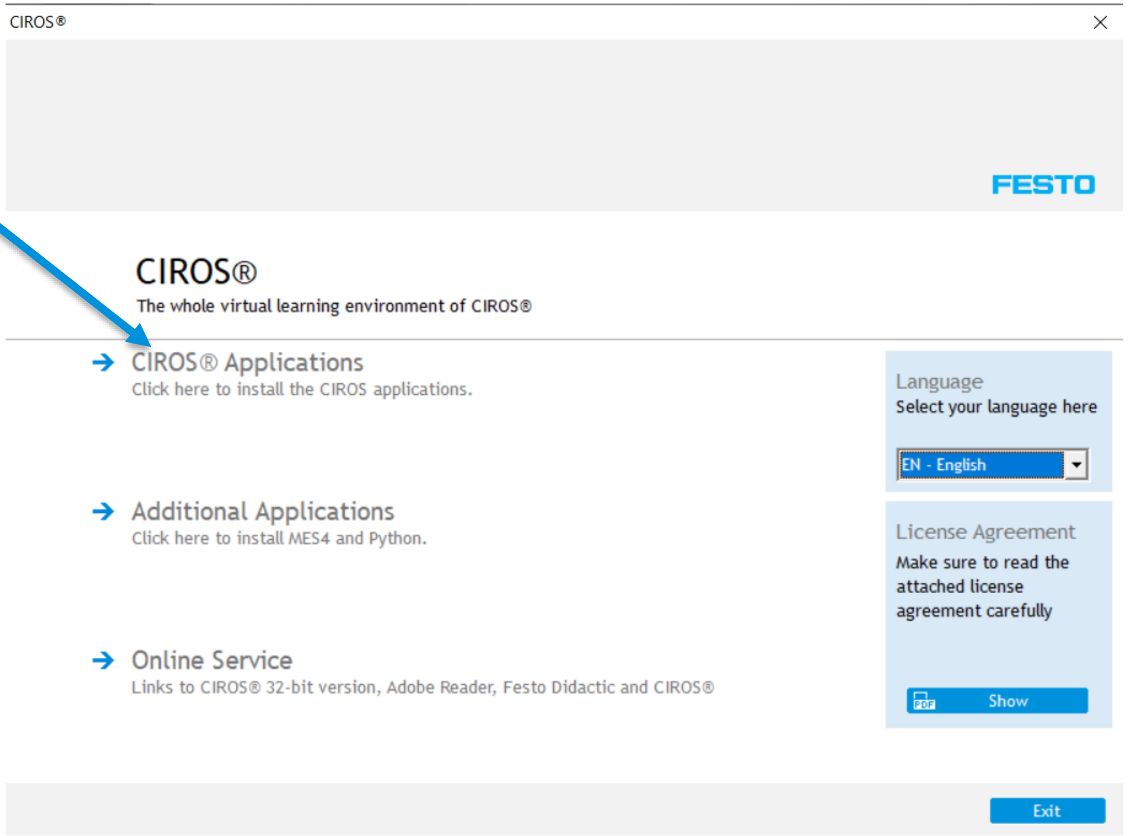
- Hardware requirements
  - High-performance CPU, i.e. Intel i5/i7
  - At least 8 GB main memory
  - At least 4 GB SSD memory
  - NVIDIA graphics card with OpenGL 4.5 support and 4 GB dedicated memory

- When using CIROS in combination with other software (e.g., MES4, PLCSIM Advanced) two screens are highly recommended!
- It is also possible to run CIROS and other software (e.g., MES4, PLCSIM Advanced) on different PCs

Installation

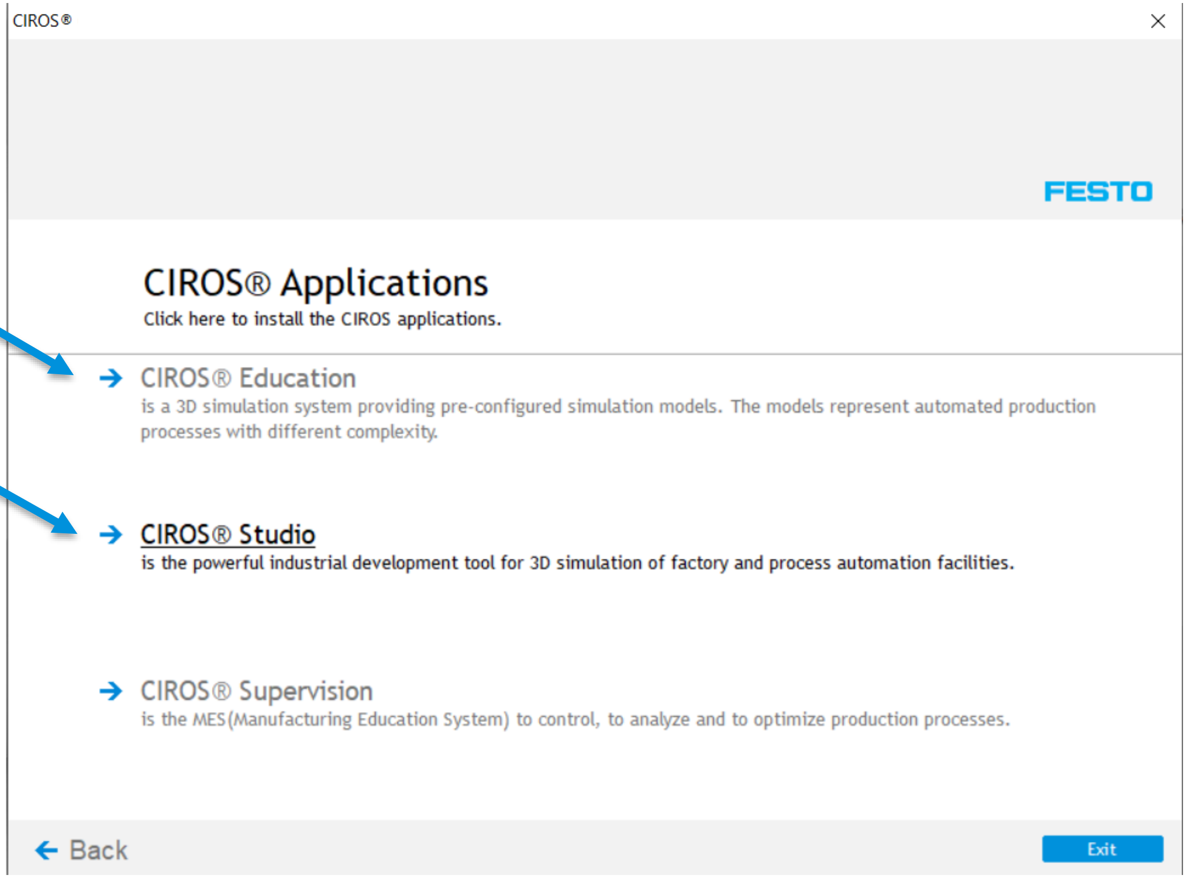
- Open **Start.exe** and select **CIROS Applications**.





Installation

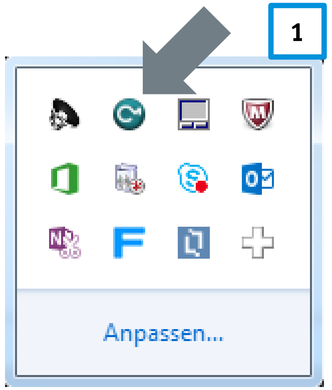
- Select either **CIROS Studio** or **CIROS Education** to be installed.

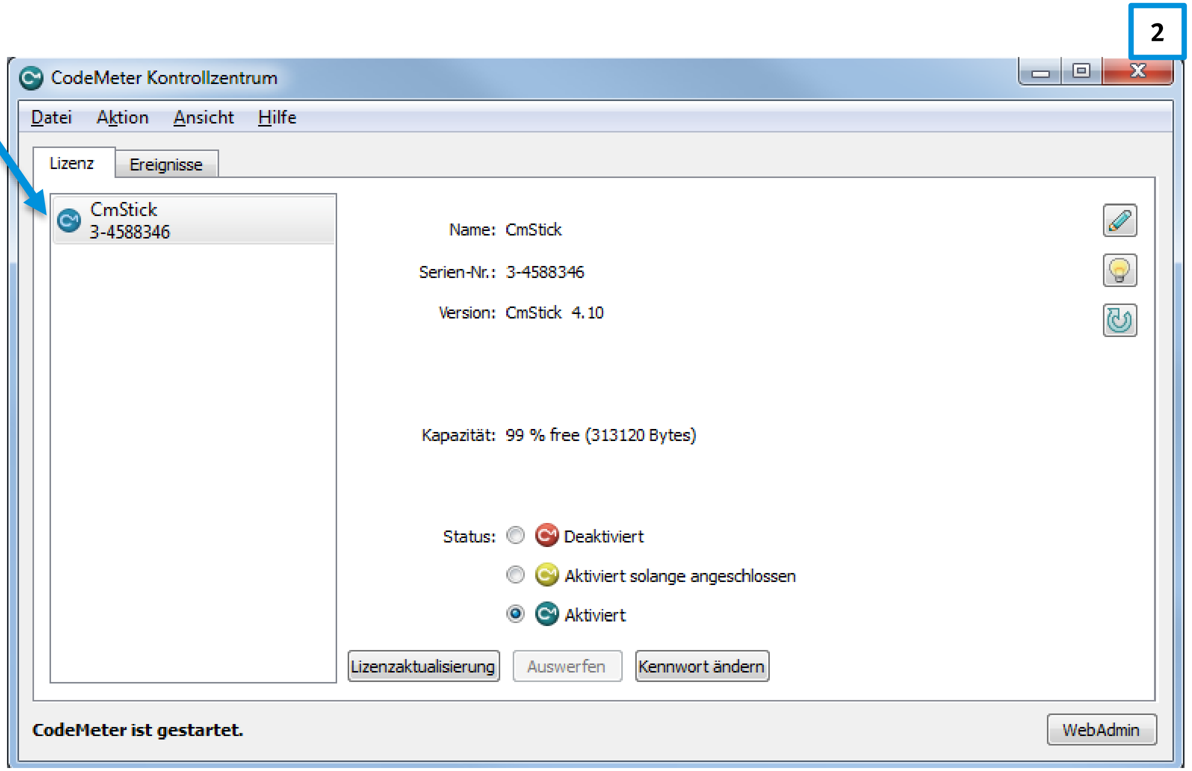


License

License is managed by CodeMeter.

List of USB dongles currently attached to the system.

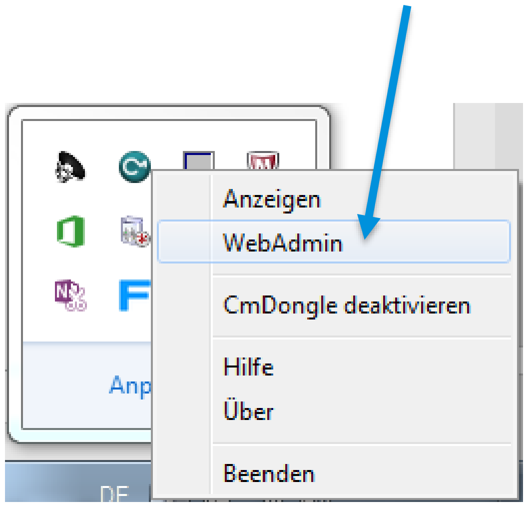




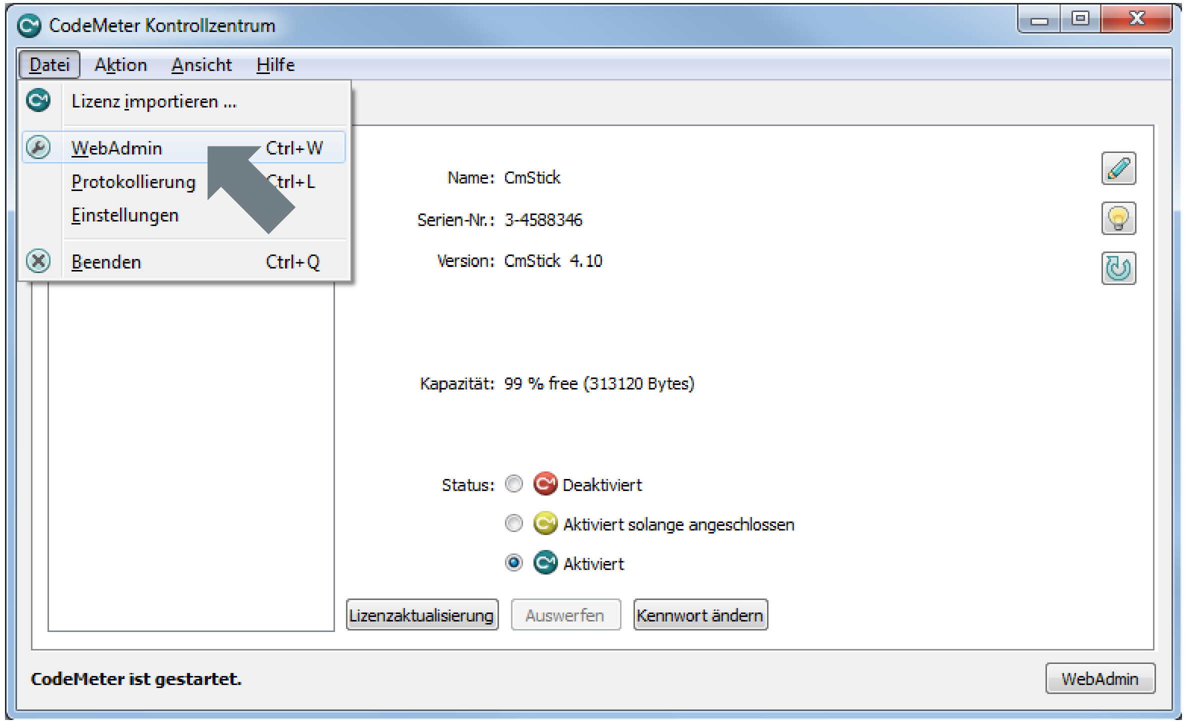
View local licenses in CodeMeter WebAdmin

Option 1

Right mouse button opens context menu.



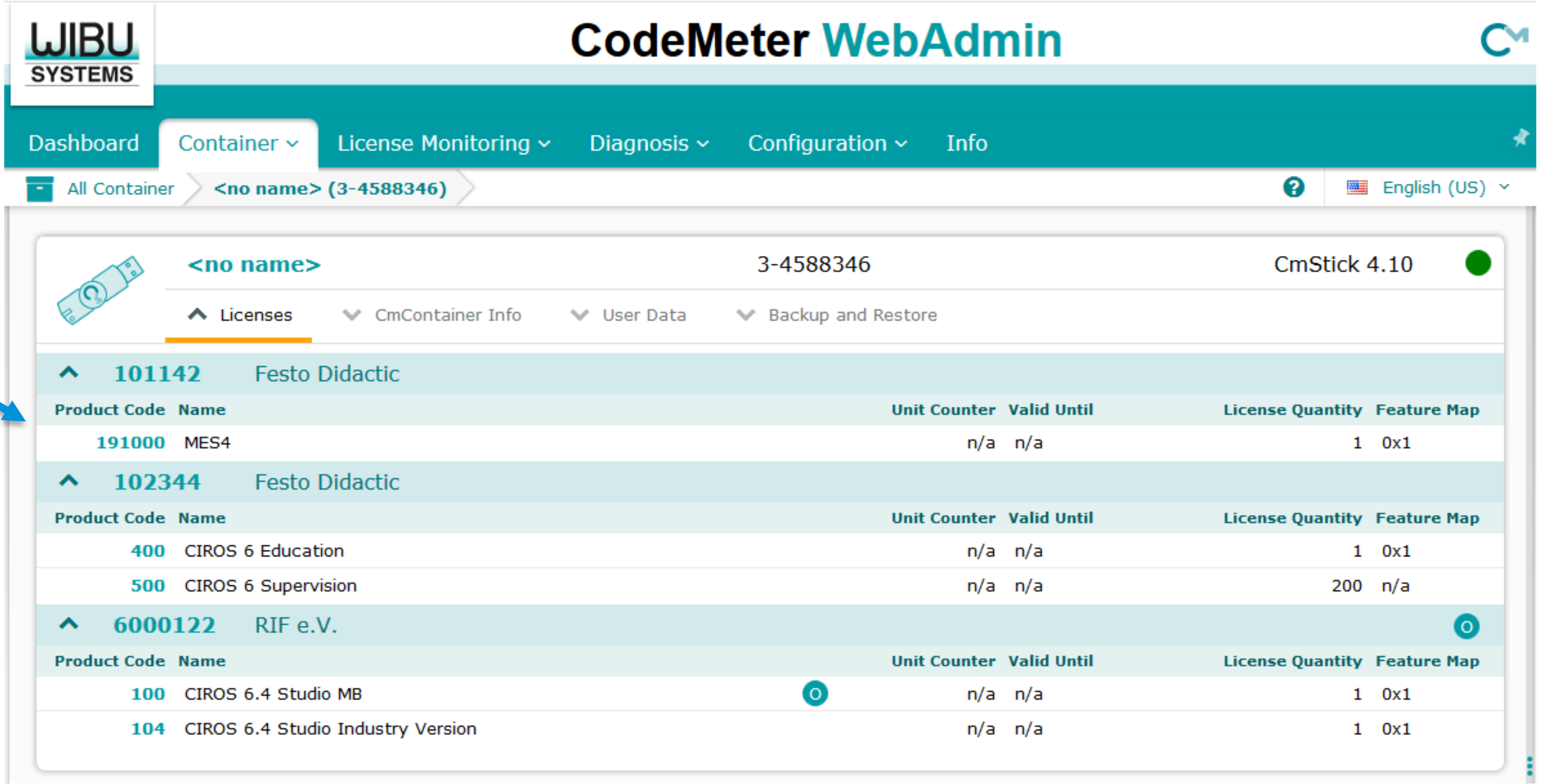
Option 2



CodeMeter WebAdmin

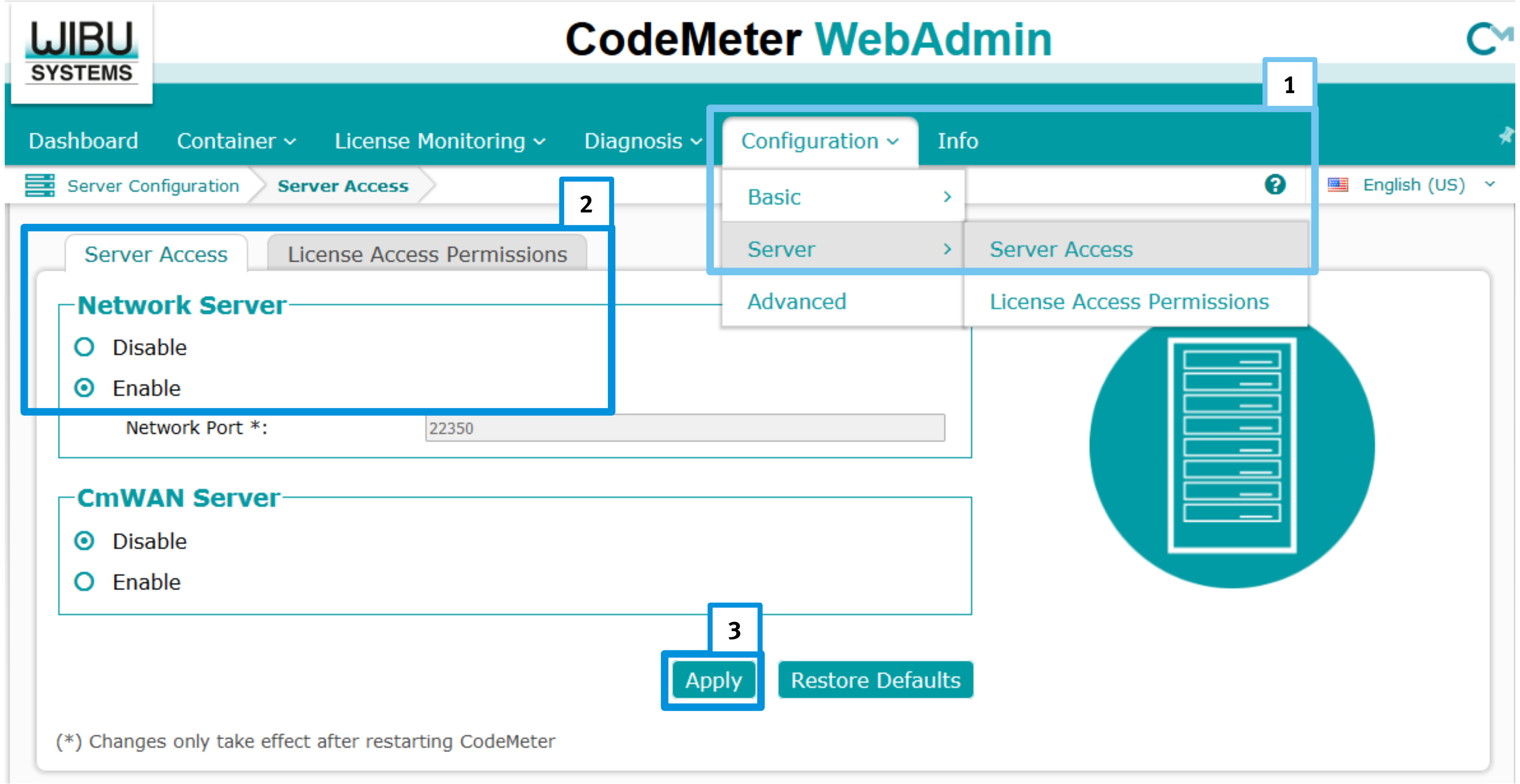
Note: All CIROS related licenses should be placed in the same container.

List of licenses stored on the attached USB dongle.



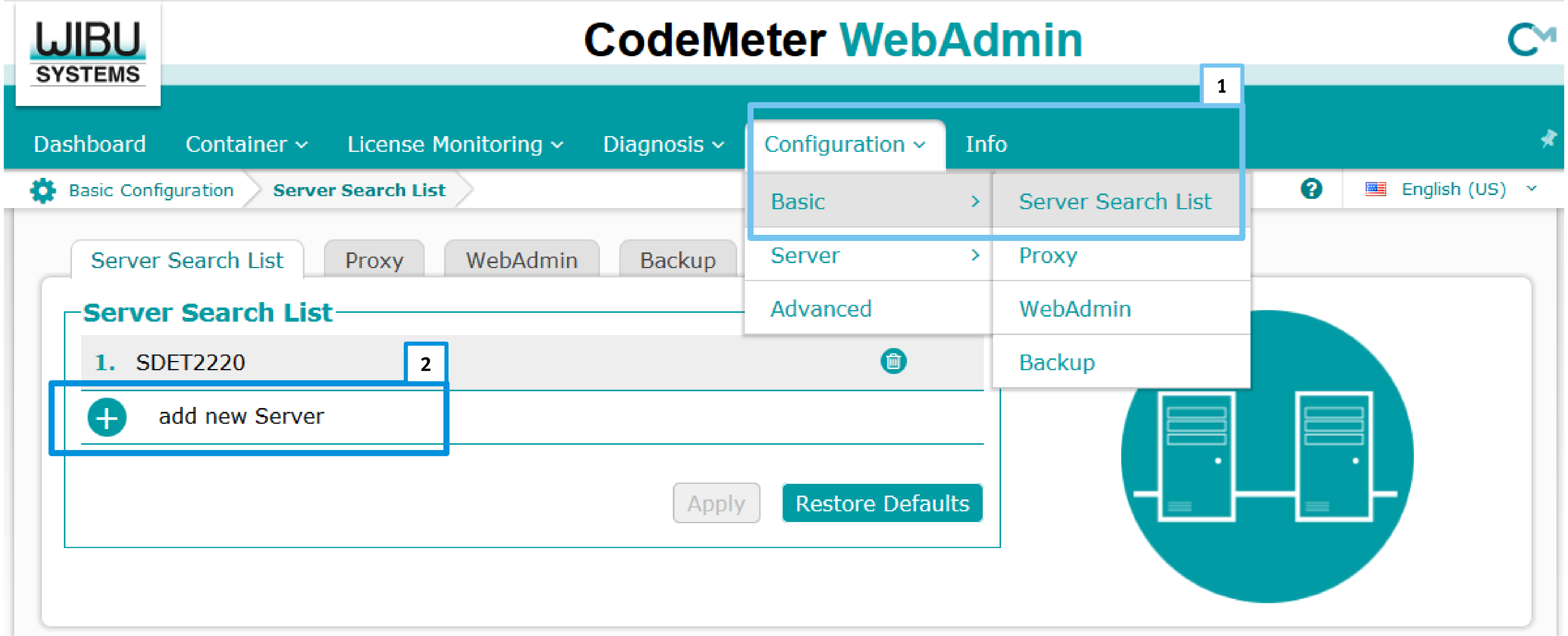
Licenses from Server

Allowing CodeMeter WebAdmin to act as a server.



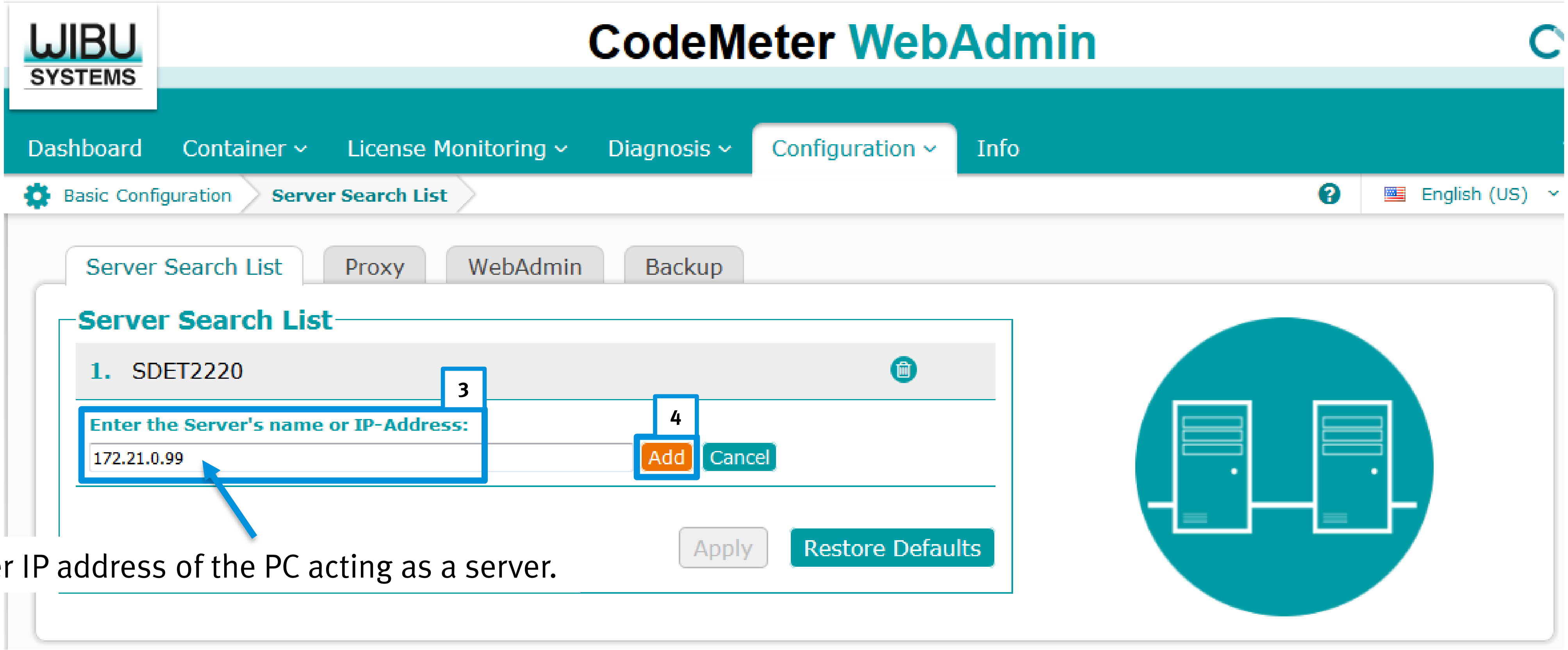
Licenses from Server

Connecting to a CodeMeter WebAdmin server. (1)



Licenses from Server

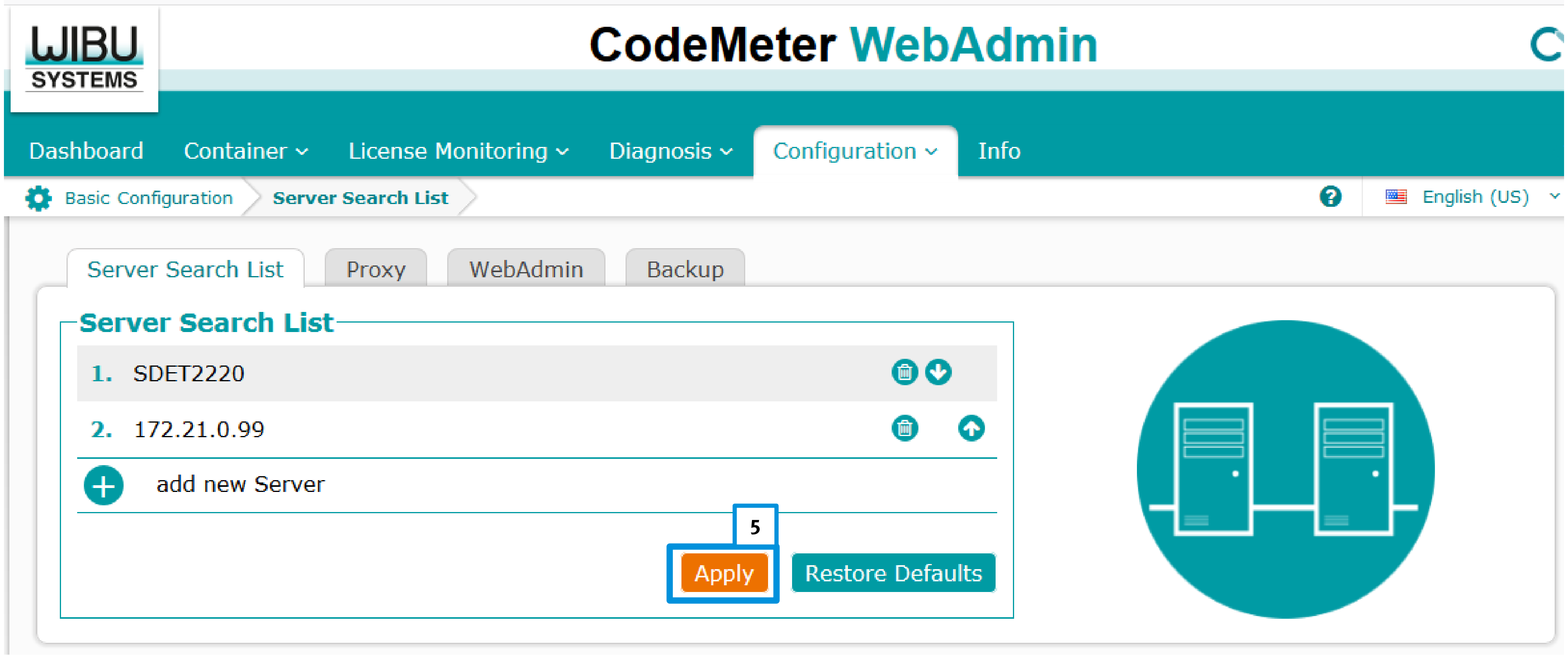
Connecting to a CodeMeter WebAdmin server. (2)



Enter IP address of the PC acting as a server.

Licenses from Server

Connecting to a CodeMeter WebAdmin server. (3)



Licenses from Server

Connecting to a CodeMeter WebAdmin server. (4)

Refresh to see the available servers.

Application Scenarios

OPC UA

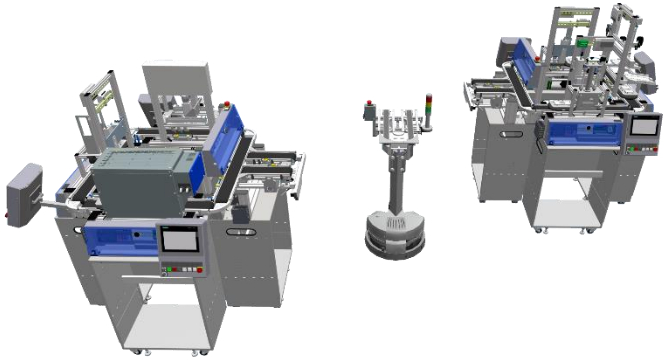
Virtual Commissioning

Process Optimisation

Large Classrooms

PLC Programming

Python



Virtual Reality

Robotics

CAD Import

Data Generator for Data Analytics

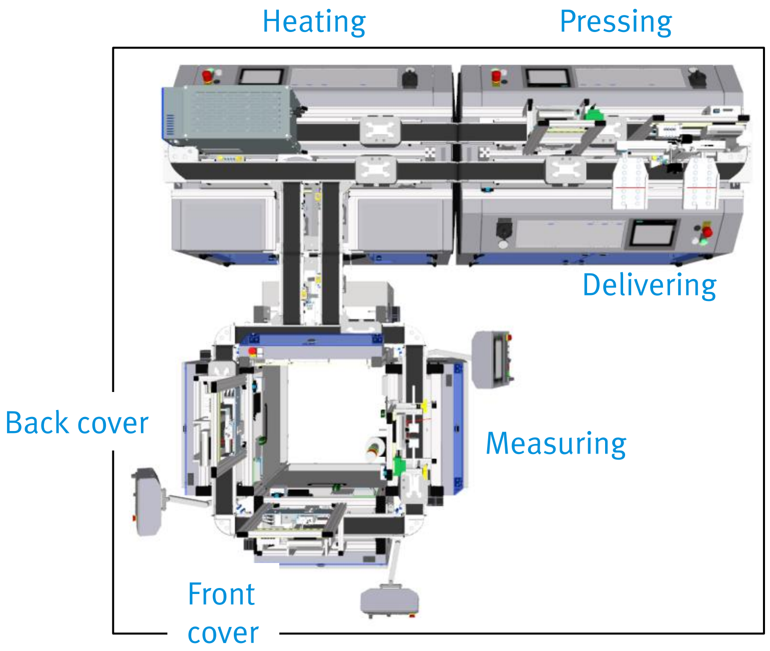
Fault Injection

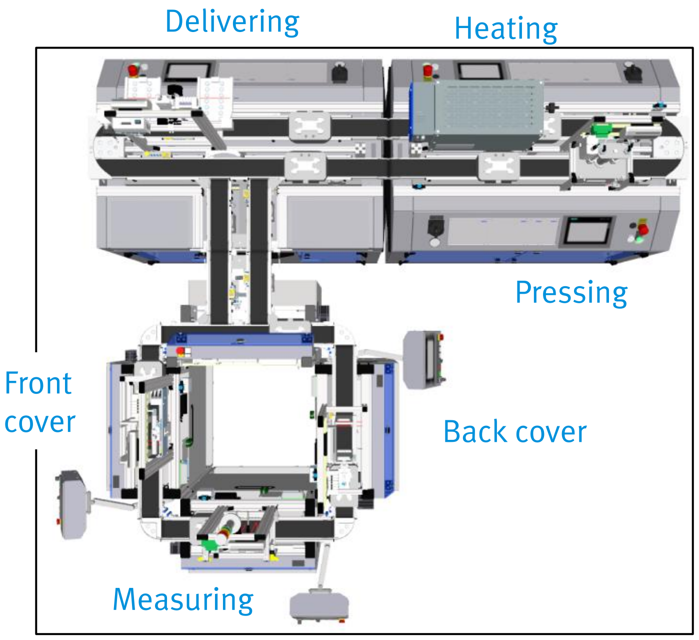
Video Export

Application Scenarios

Process optimisation

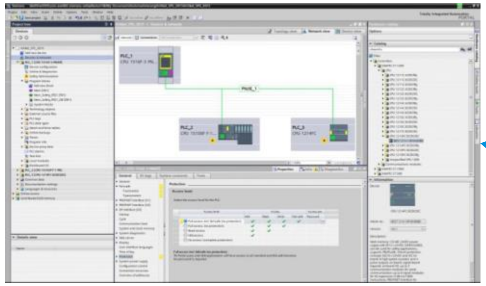
- Assume following process: Front cover → Measuring → Back cover → Pressing → Heating → Delivering
- Which of the two configurations below is more efficient?



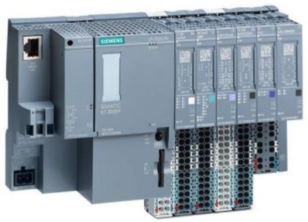


Application Scenarios

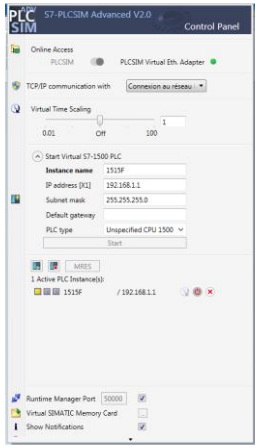
PLC programming (example with Siemens PLC)



PLC programm

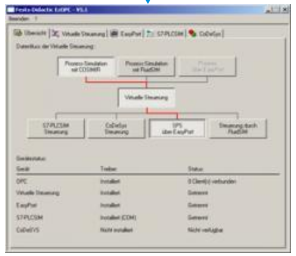


Real PLC



Soft PLC

EasyPort

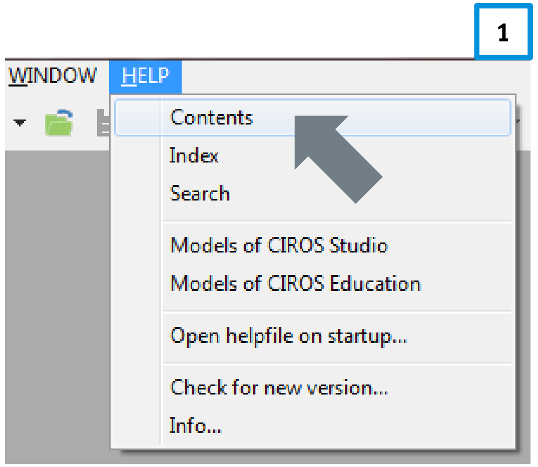


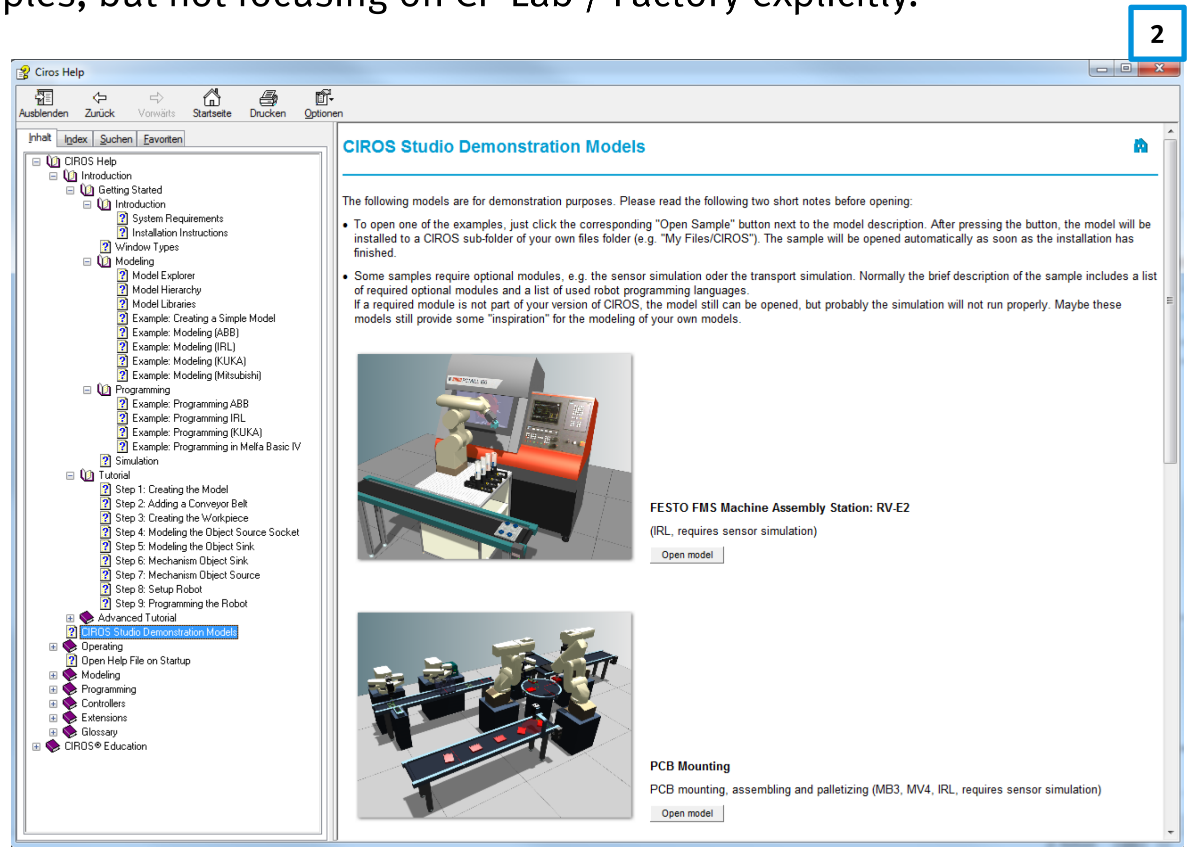
EzOPC

OPC UA

Help Menu

- Detailed help system with a couple of introductory examples, but not focusing on CP Lab / Factory explicitly.





Keyboard Shortcuts

SHIFT + left mouse button	Move user perspective in view mode
CTRL + left/right mouse button	Rotate user perspective in view mode
+ / -	Zoom in / zoom out
V / H	Front / rear view
A	Top view
R	Right view
L	Left view
SHIFT + O	Full screen
F5	Start / stop simulation
CTRL + F5	Reset simulation
CTRL + E	Toggle between view (default cursor) and edit mode (crosshair cursor)
CTRL + . / CTRL + ,	Rotate selected object by +/-90° in edit mode
CTRL + T	Show model explorer
CTRL + F9	Compile robot / PLC programs
CTRL + SHIFT + M	Show model libraries

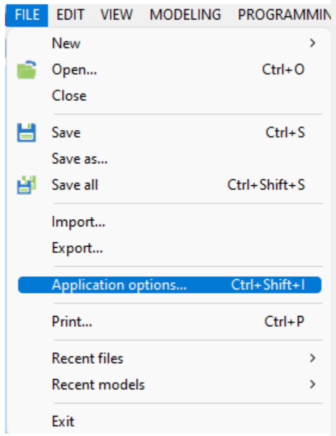
Options Setting

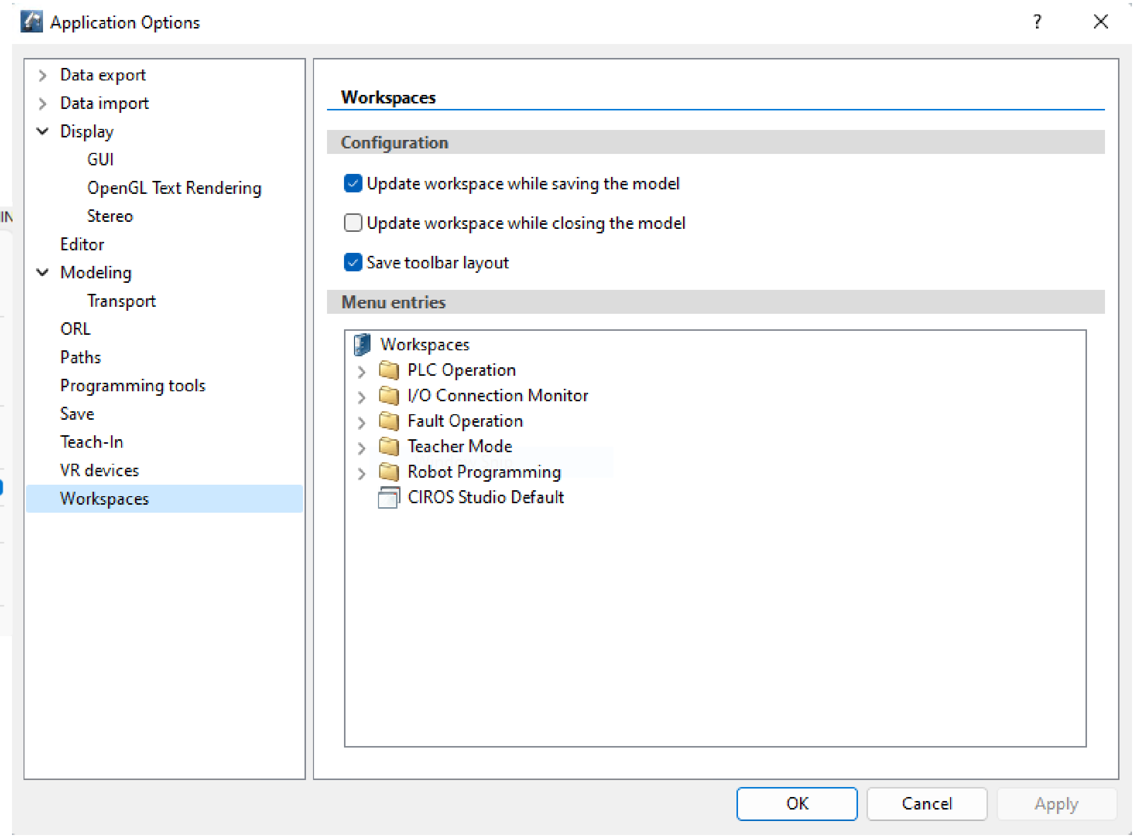
- CIROS has two option settings, application options and model options. There are for different configurations
- Application options configure the whole CIROS application, regardless of the models.
- All changes made in model options are only applied to the active model.

Application Options

File → Application options

- Define data import and export
- General display settings
  - Frequently used to reduce computational load.
- Editor settings
- Warning options for modelling and transport
- ORL
- Position and paths
- Programming tools settings
- VR devices configuration
- Workspaces

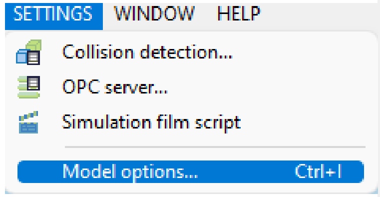


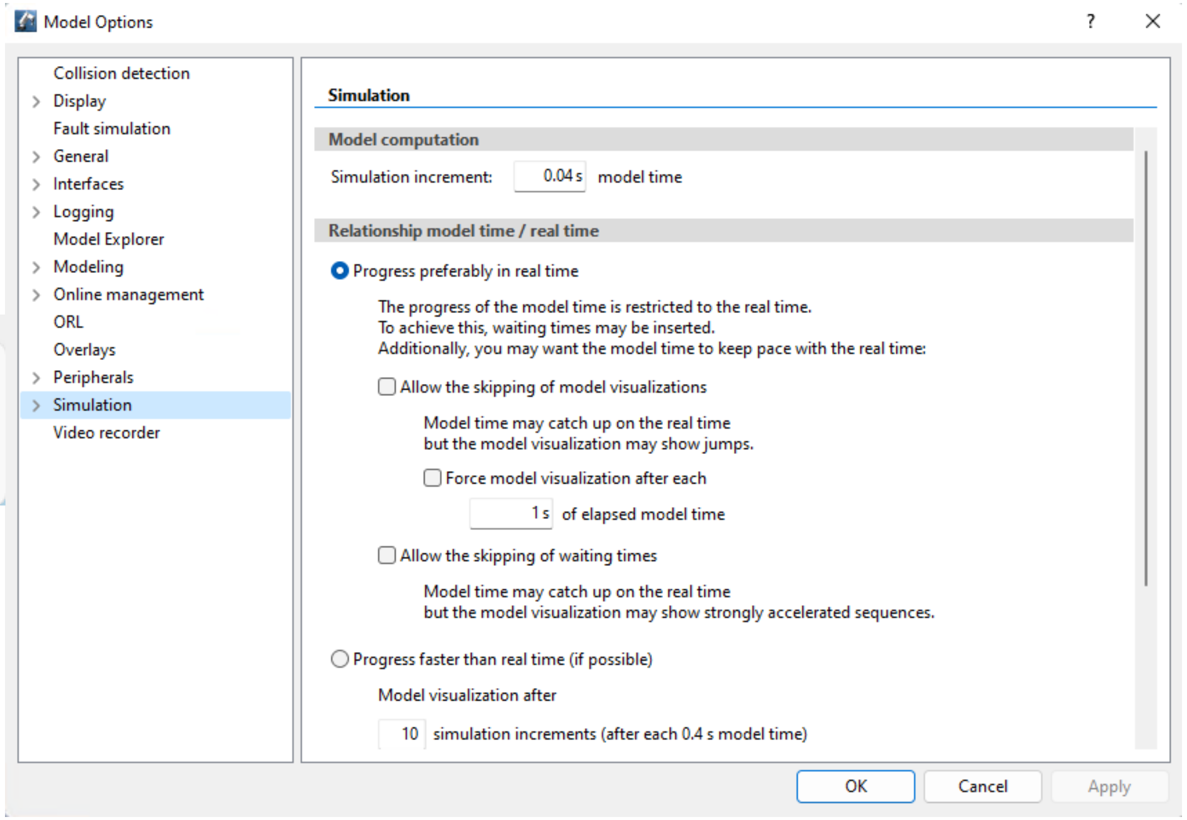


Model Options

Settings → Model options

- Collision detection
- Model display settings
  - Background
  - Floor
  - Sensors
  - Etc.
- Fault simulation in teacher mode
- Interfaces
  - MES4
  - Fleet Manager
- Data logging
- Model explorer settings
- Modelling
- Online management for Mitsubishi robots
- ORL
- Overlays
- Model Simulations
- Video recorders





Where to look for the options?

- Structure and elements of models
  - Modell Explorer: Modelling → Model Explorer or Ctrl+T
  - Often used

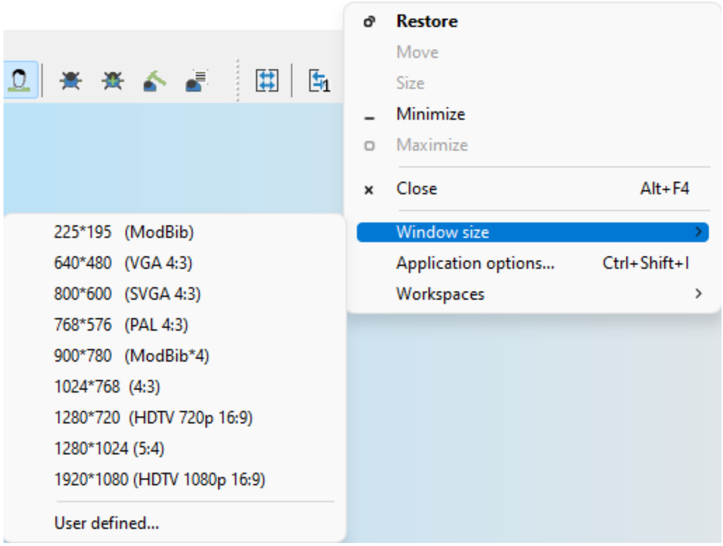
- Properties of elements in the model
  - Properties: Modelling → Properties or Alt+Enter
  - Assistant
    - For example Settings → Collision detection
  - Often used

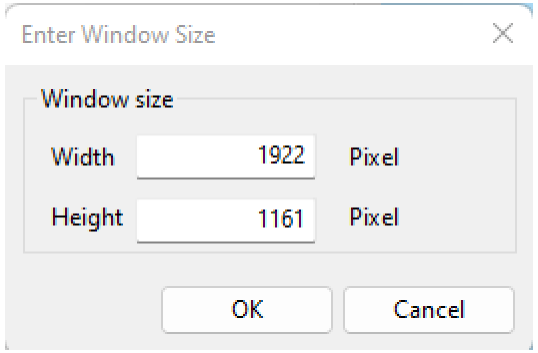
- Properties of model
  - Model options: Settings →Model options or Ctrl+I
  - Used fairly

- Properties of CIROS program
  - Application options: Files → Application options or Ctrl+Shift+I
  - Seldom used

Window’s Size

- Windows size for Application Window and Modell Window can be adjusted.





Introduction to CIROS Model

CIROS Model

- Preliminary remark: Each CIROS model not only consists of the modx/ini files but also the folders CF and Textures, storing the internal PLC programs and textures
- It is highly recommended, to store each CIROS model in a separate folder!
- **Important:** Do not copy the modx/ini files only, but the entire folder containing the subfolders CF and Textures, too!

A CIROS project folder:

Name	Änderungsdatum	Typ	Größe
CF	09.04.2019 10:29	Dateiordner	
Textures	09.04.2019 10:29	Dateiordner	
CIROS.ini	09.04.2019 10:33	Konfigurationseinstellungen	61 KB
CIROS.modx	09.04.2019 10:33	CIROS Model	88.236 KB

Collaborative Working

- Several users can work on the same model at the same time.
- When a user changes the model, other users will receive a notification.
- However, simultaneous changes and changes that crossed over time cannot be merged together.

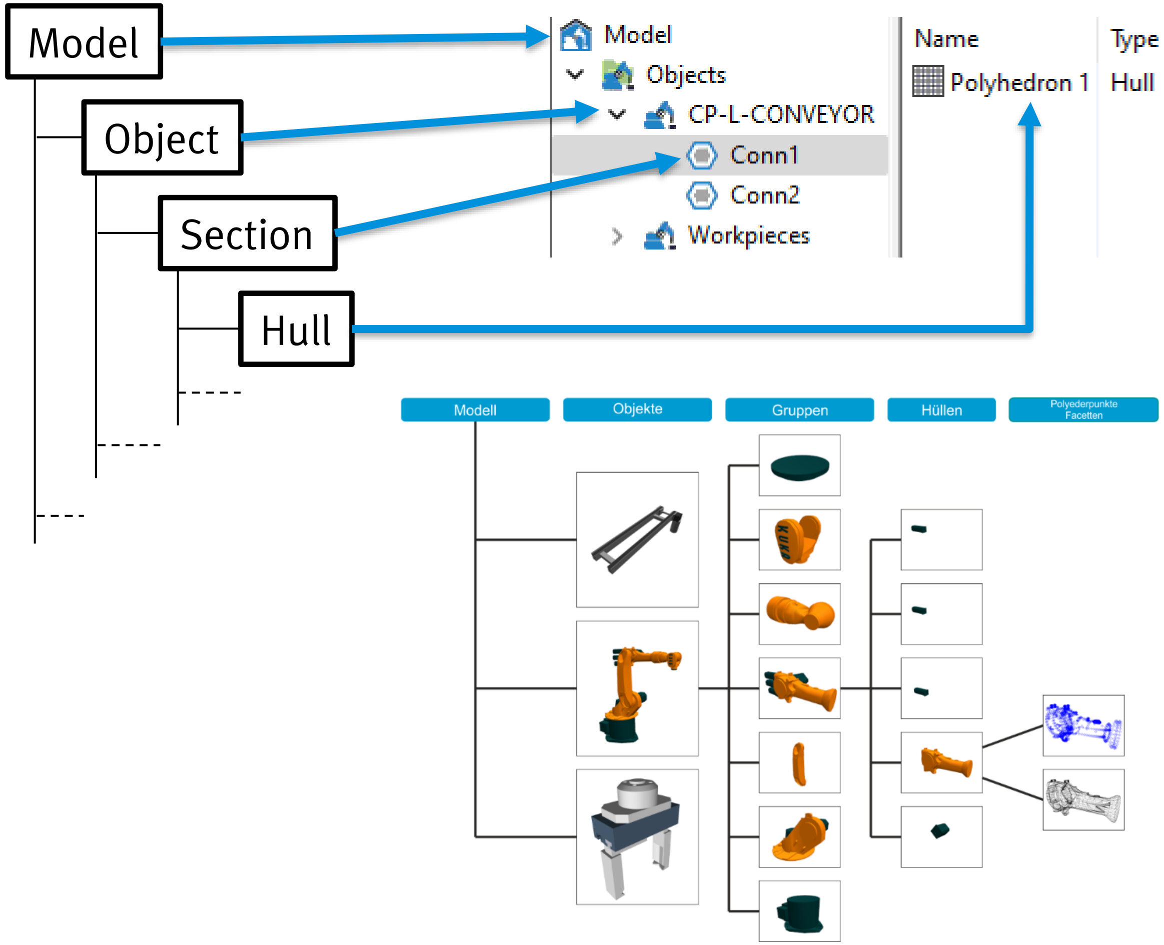
Model’s Structure

Elements in a model

- Structures of elements
  - Model / Environment
  - Objects: Logical unit
  - Sections: Static body
  - Hulls: Geometries

- Positions based on coordinate system

- Hulls
  - Geometric primitives
    - Box,sphere, etc.
  - Polyhedron
    - Vertex, Facet



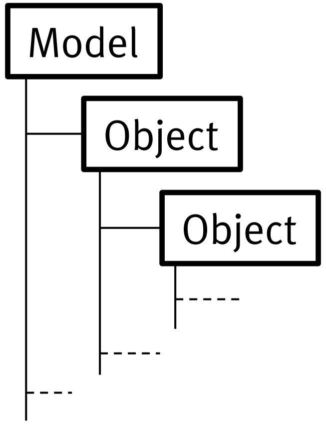
Model's Structure

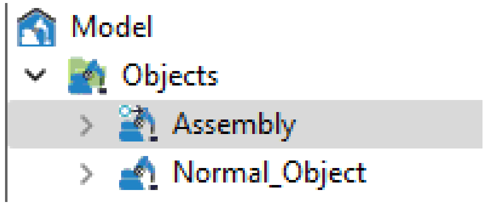
- Objects in the structure tree
  - Placed on the highest hierarchy in the model or
  - Placed under an object

- Object's nomenclature
  - Parent object: superordinate object
  - Child object: child of a parent object

- Usage
  - For a clear structure
  - Definition of static assembly

- Moving child objects
  - During modelling: always
  - During simulation: only when the object is an object assembly
    - Select object → right click → Edit → Assembly

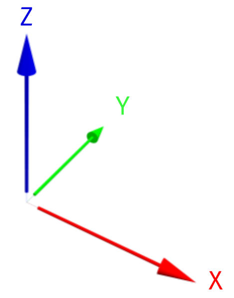




Elements and Coordinate Systems

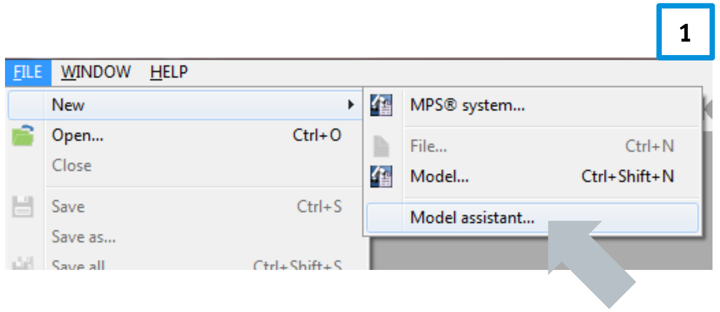
- Different coordinate systems, each might has same or different origins
  - World : based on world coordinate system
  - Object : based on coordinate system of the parent object
  - Section : based on coordinate system of the section it belongs to
  - Hull : based on coordinate system of the hull

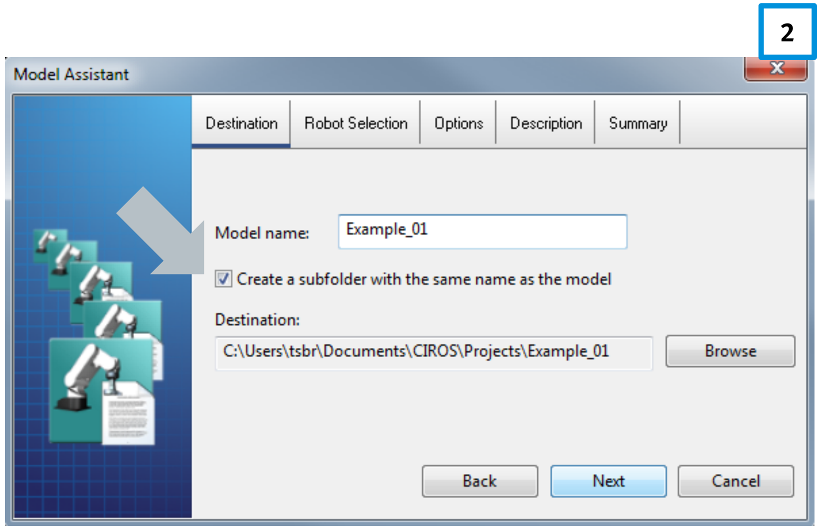
- In model window, origin of three axes of coordinate system are shown in different colours.

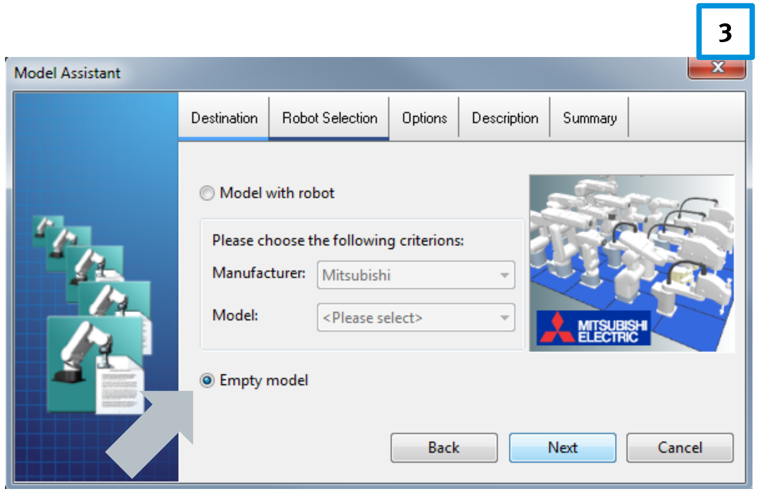


Create a CIROS Model

- Recommended way of defining a new CIROS Model
  1. Choose FILE → New → Model assistant
  2. Specify the model’s name and enable Create a subfolder with the same name as the model.
  3. **Important:** Do not select a robot, these ones are not the ones integrated within CP Lab / Factory! Choose Empty model instead!



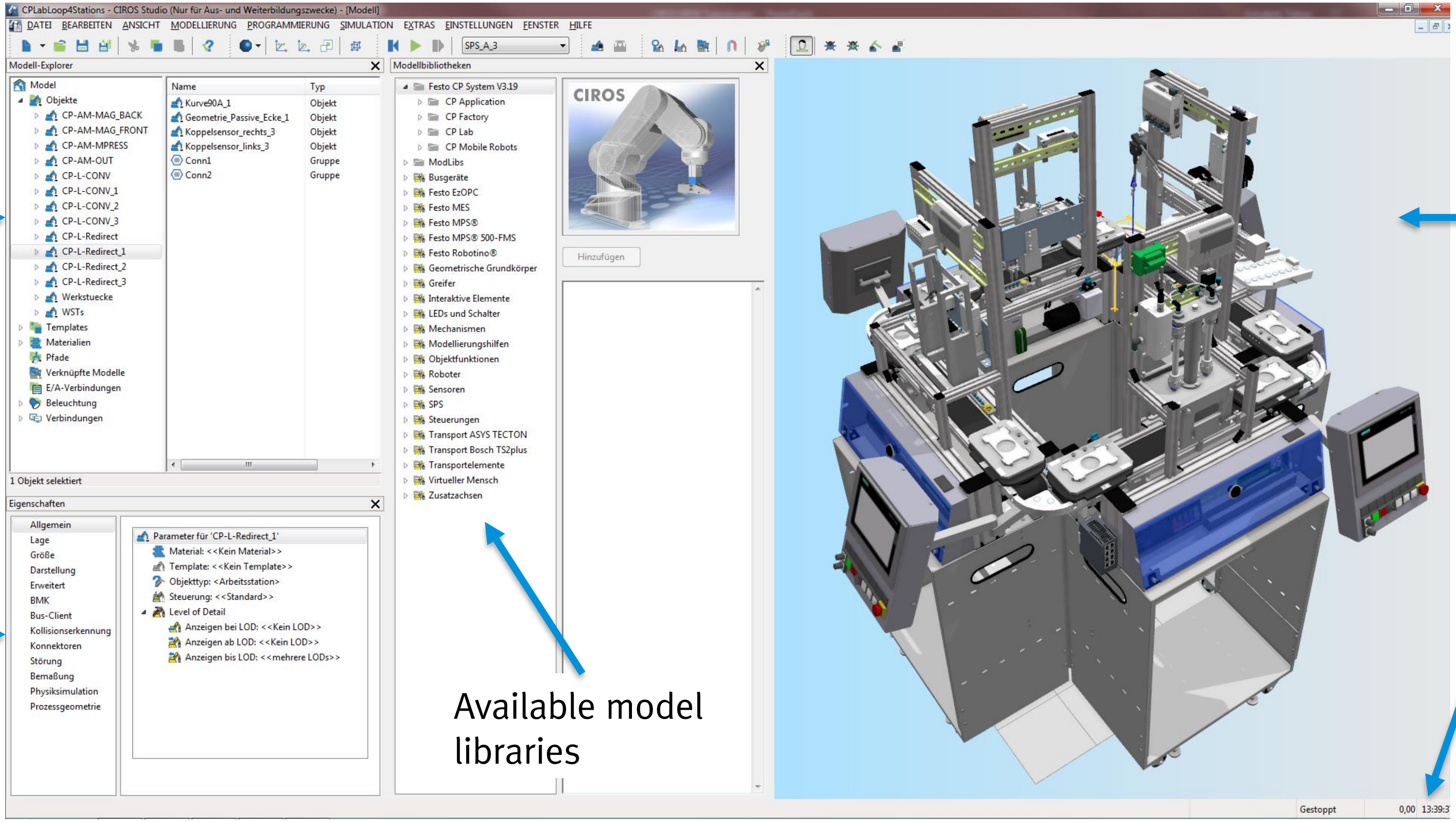




Window’s Layout

Model explorer

Properties window



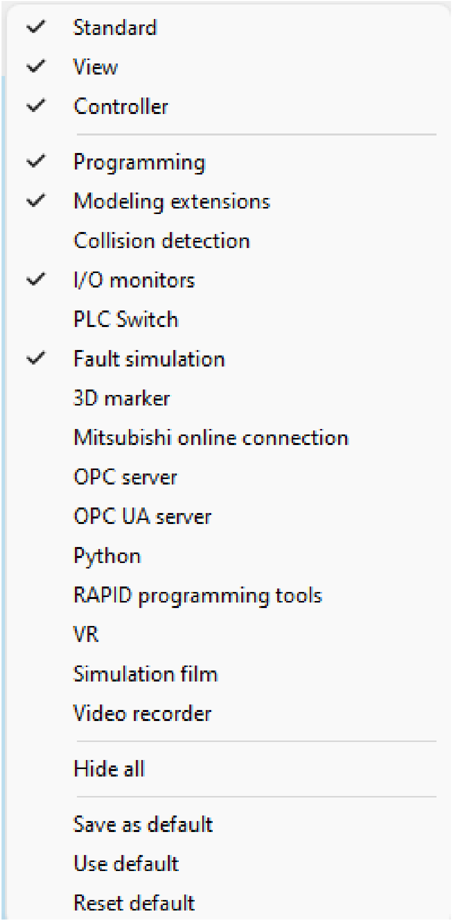
Available model libraries

View window

Model vs. real time

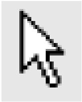
If the model time is printed in red, the system is not able to perform the simulation in real time.

Toolbar can be configured

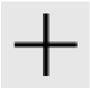


View and Edit Mode

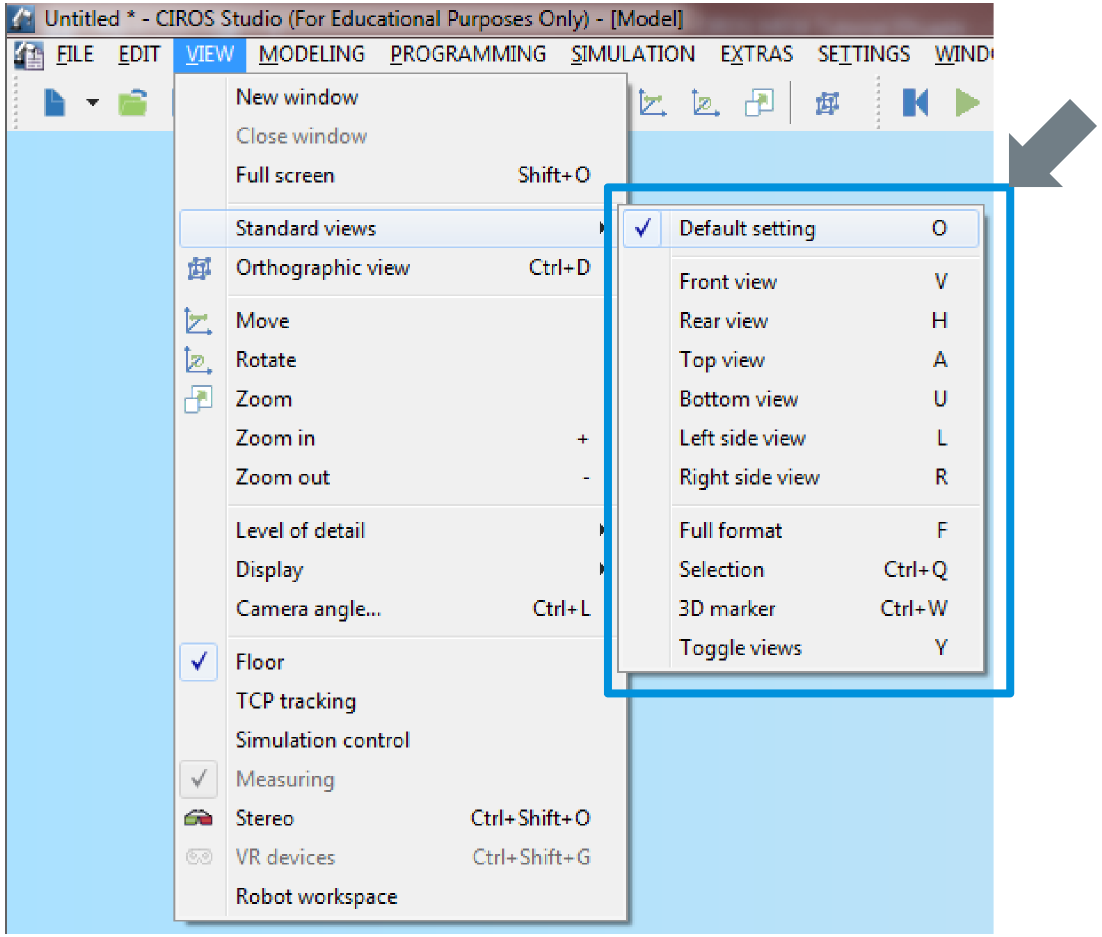
View mode

- Change user perspective onto scene
- Default cursor: 

Edit mode

- Place, move, rotate objects within scene
- Crosshair cursor: 

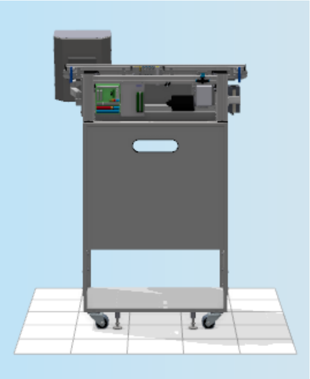
Standard Views



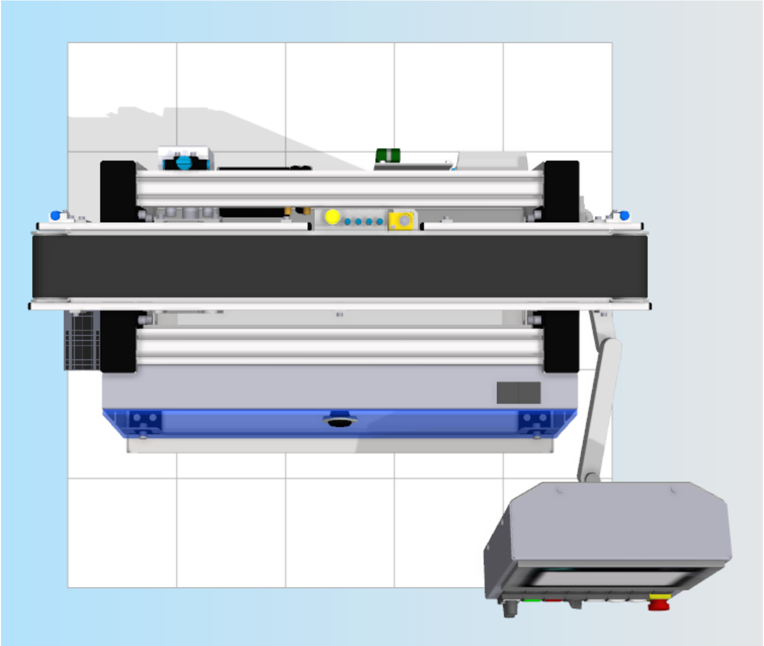
Standard Views

Default setting

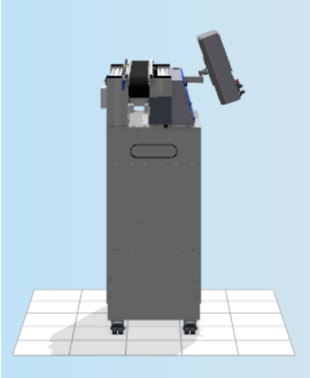
Front view



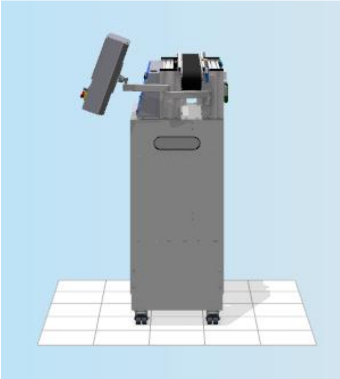
Rear view



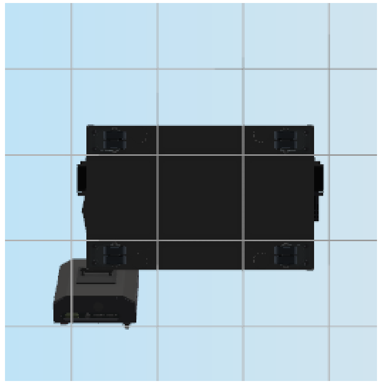
Top view



Left side view



Right side view

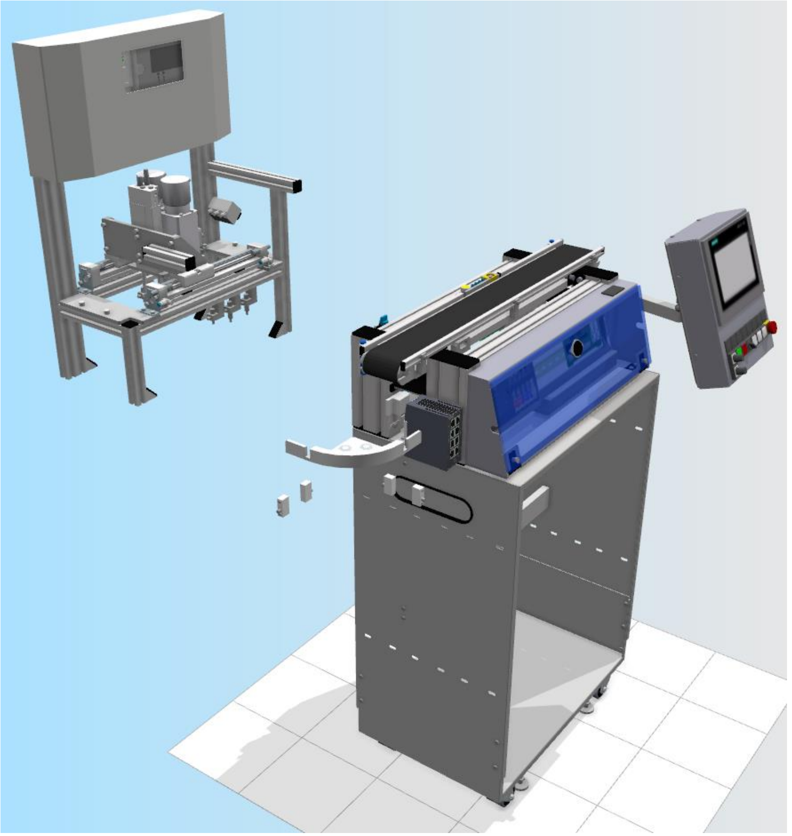


Bottom view

View Used when Working with CIROS Model

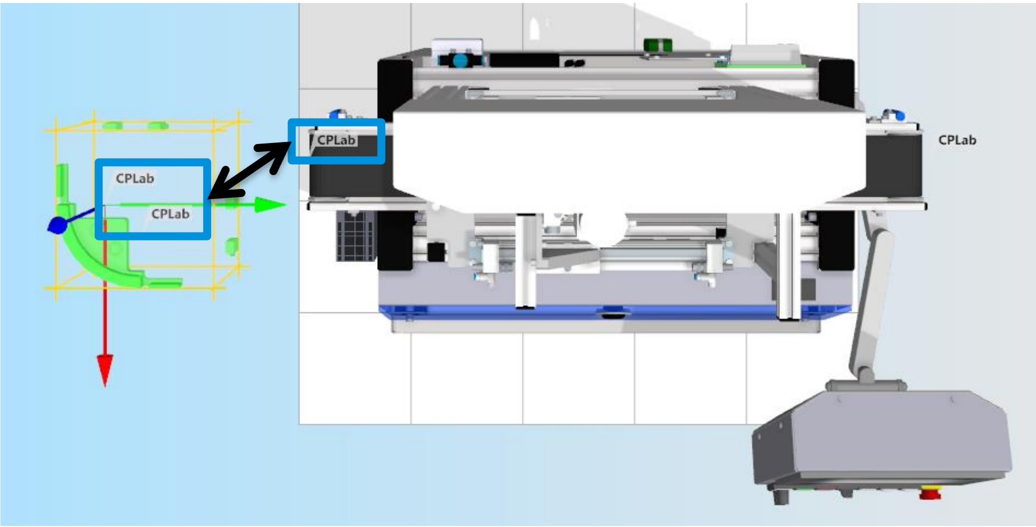
- With respect to the z-axis all CP Lab / Factory modules within the library have been prepared in such a way that
  - Conveyor belts, CNC milling stations, robot assembly stations, warehouses, and Robotinos are placed on the floor (z = 0mm)
  - Carriers, deflections, sources, sinks, and application modules flush with the conveyor belts' upper edges (z = 975mm)

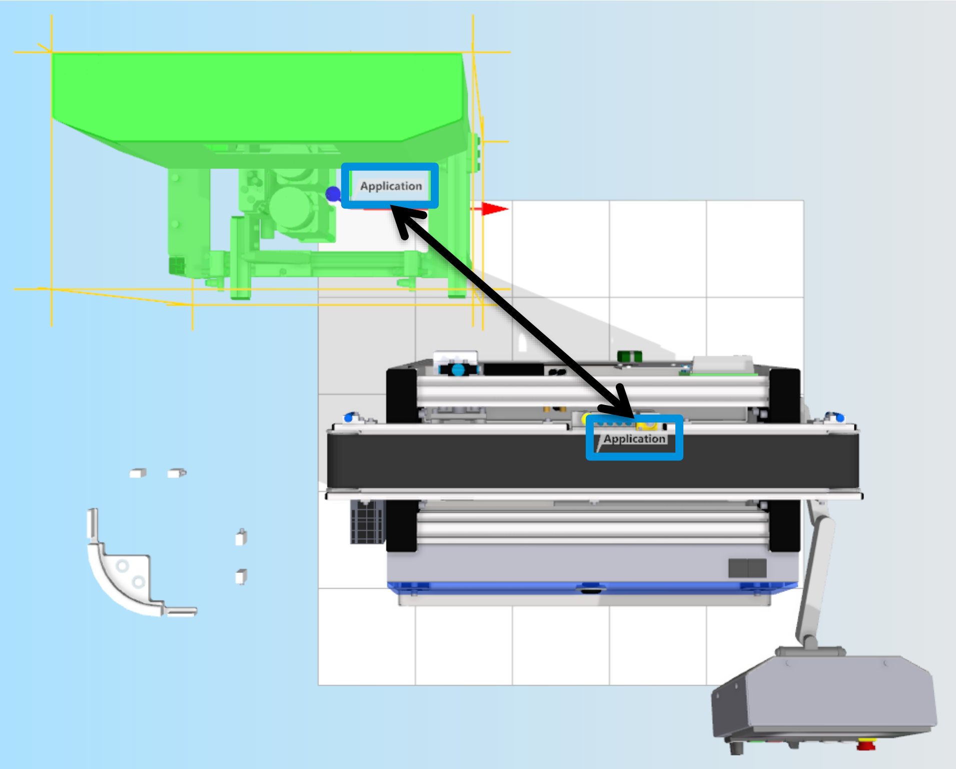
- **Important**
  - Switching to Top View ensures that the z-values remain constant when moving components within the scenery!
  - Snapping into place of modules works well in Top view only!



Snapping into Place

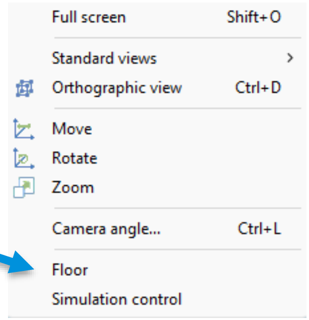
- When placing application modules onto conveyor belts, it is not necessary to adjust them as precisely as possible.
- Just putting the phrases Application, CPLab, Modul, etc. on top of each other is all one has to do!
- Same holds for mounting deflections, docking kits, and so on.

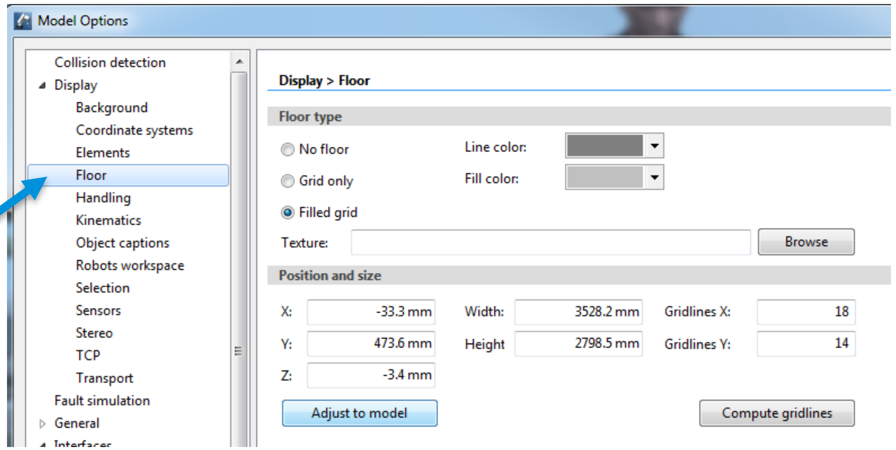




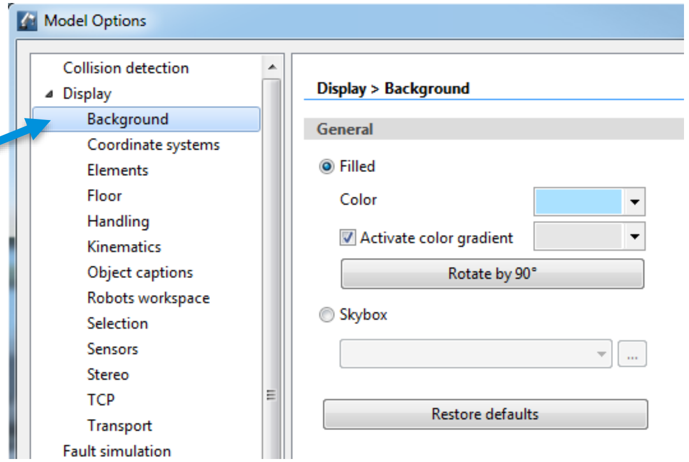
Floor and Background

- Floor can be activated or deactivated.
  1. Right click in view window.
  2. Check or uncheck Floor.





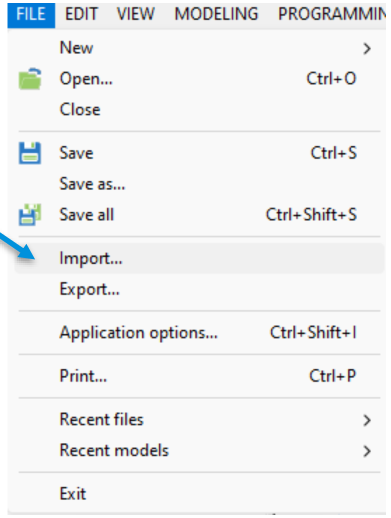
- Floor's size and muster can be adjusted.
  - Open Settings → Model options → Display → Floor.



- Background can be adjusted.
  - Open Settings → Model options → Display → Background.

Import Data

- It is possible to import a model from CAD data by selecting File → Import.
- Supported formats are as follow:
  - 3ds Max
  - AutoCAD DXF
  - Autodesk
  - Blender
  - Collada
  - IGES
  - PointCloud
  - STEP
  - STL
  - VRML
  - Wavefront Object



FESTO

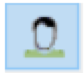
Export Data

- It is possible to export a model or selected objects in the model to CAD or picture.

1. To export a whole model, in Model Explorer, select Objects. To export objects in model, select the objects in Model Explorer.
2. Click File → Export.

- Supported export formats are:
  - AutoCAD
  - For viewing in a web browser (HTML)
  - PNG
  - IGES
  - POV-Ray scene description
  - RT toolbox file
  - STEP
  - STL
  - VRML
  - Windows bitmap

Teacher Mode

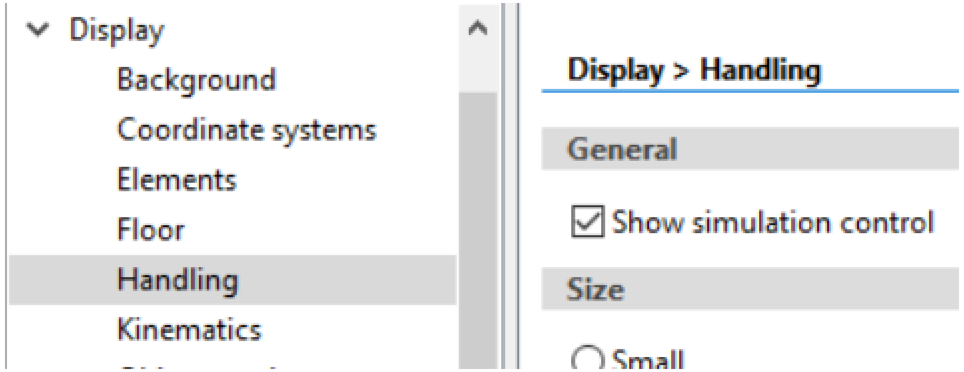
- Teacher mode can be activated to configure fault simulation in “Fault Setting “ and “Fault Log” windows.
  - Extras → Fault Simulation → Teacher Mode
  - 

- Password for Teacher Mode is “didactic”.

Simulation Control in Model Window

- Simulation control buttons can be shown in model window.
- This is to allow simulation control in full screen mode or in VR.

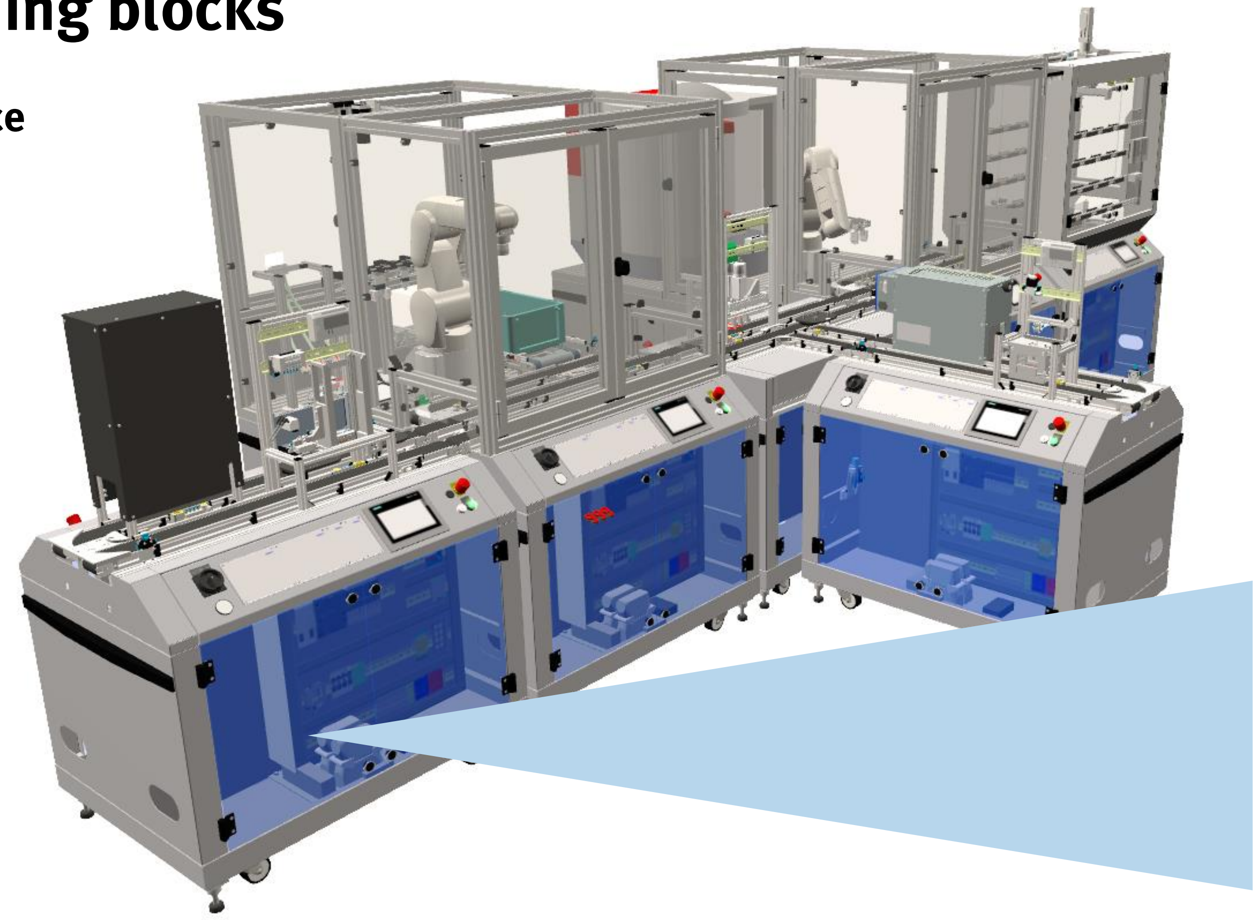
1. Go to “Settings → Model option”.
2. Select “Display → Handling”.
3. In section “General”, activate “Show simulation control”.
4. Click on “ok”.



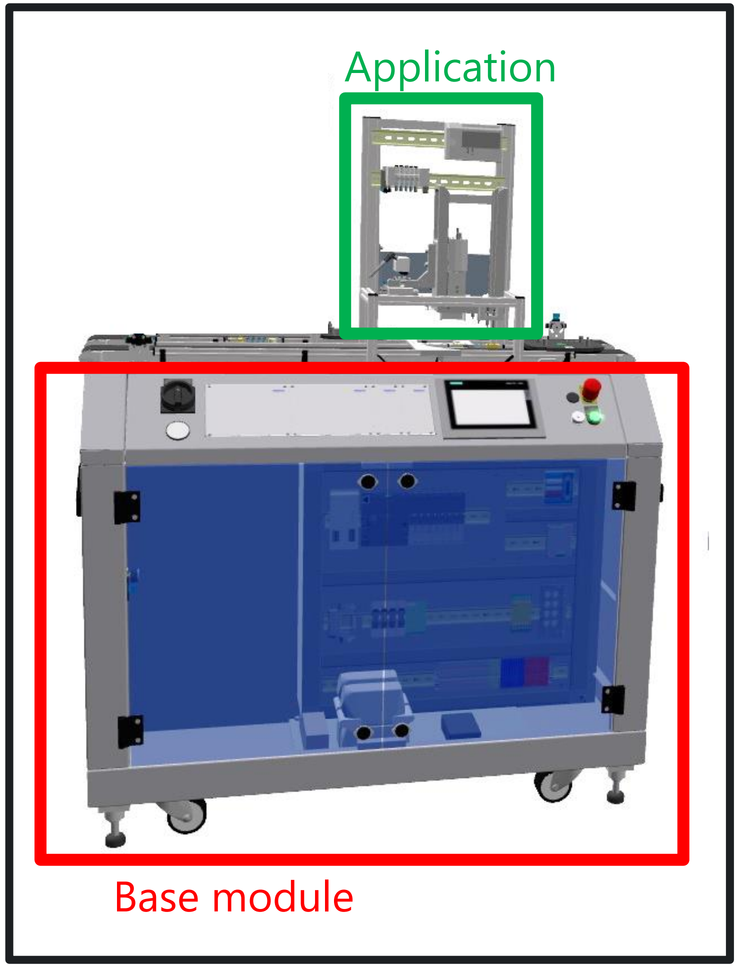
Introduction to Festo CP-System

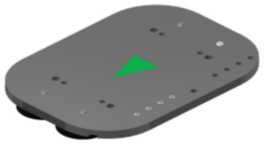
Building blocks

Resource



Module / Ressource

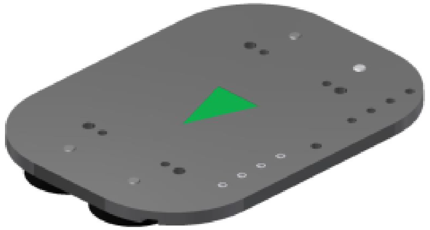




Carrier

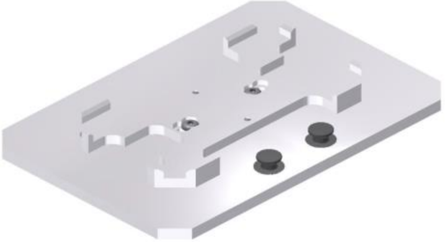
Building Blocks

Carrier and product



Carrier

+



Palette

+

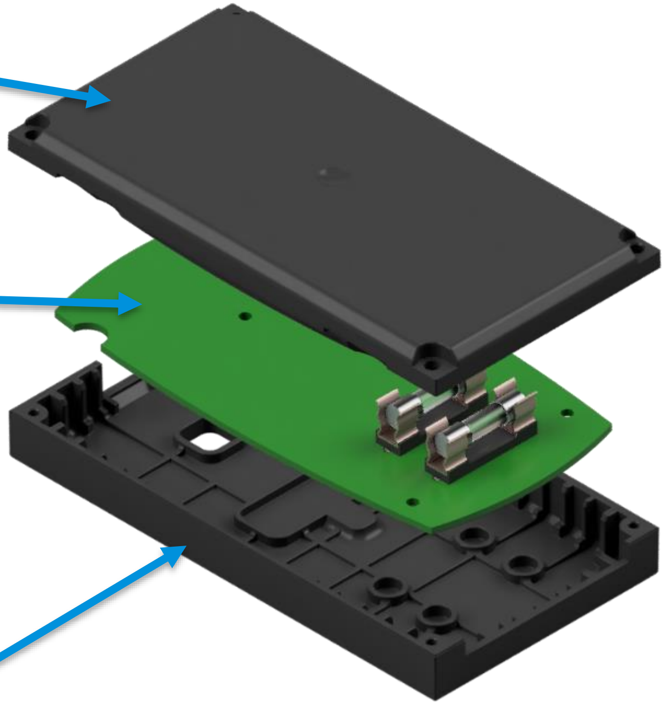
Product

Product (Smartphone)

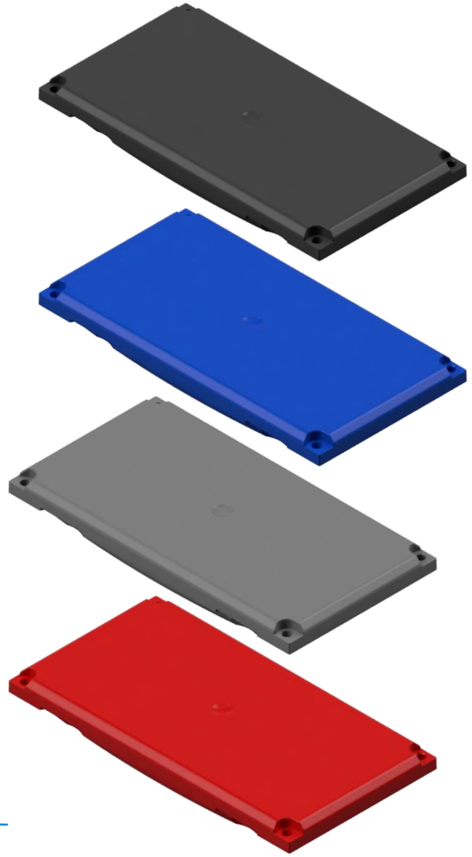
Back cover

Inlay (optional)
- PCB
- Fuses

Front cover
- Slot for screen
- An opening for Button
- Able to drill holes for screw thread



Covers are available in four colours



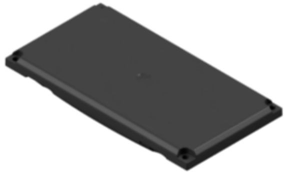
Terms & Definitions

Parts

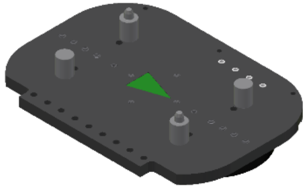
- Set of all objects, except Robotinos, that are moving around within a CP Lab / Factory.
- Parts get identified by a unique part number (PNo).
- Six different subclasses.

External production parts	Parts not produced by CP Lab / Factory
Production parts	Parts produced by CP Lab / Factory
Boxes	Transport workpieces from one production facility to another
Pallets	Mounted on top of the carriers
Carriers	Move (external) production parts on the conveyor belts.
Undefined	Containing all unknown parts or objects, typically used to represent faulty parts.

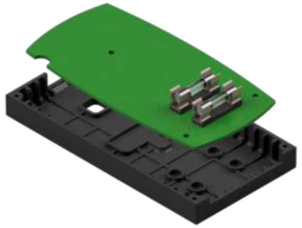
External production part



Carrier

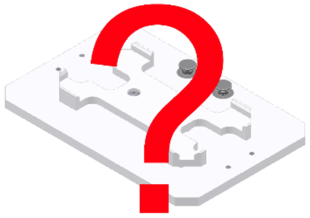


Production part

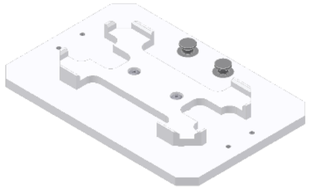


Box

Undefined



Pallet



Standard Part Numbers

Supported part numbers

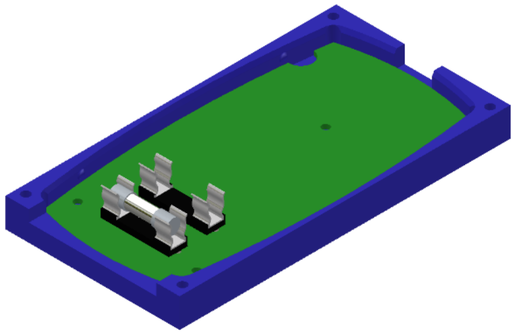
25 31	Pallet Carrier
110 / 109 / 108 / 107	Raw material front cover black / grey / blue / red
111 / 112 / 113 / 114	Back cover black / grey / blue / red
210 / 310 / 410 / 510	Front cover black / grey / blue / red
x11 / x12 / x13 / x14 (x = 2, 3, 4, 5)	Front cover in black / grey / blue / red including PCB / front fuse / rear fuse / both fuses
1y11 / 1y12 / 1y13 / 1y14 (y = 2, 3, 4, 5)	Front and back cover in black / grey / blue / red including PCB / front fuse / rear fuse / both fuses
26 / 120 / 130	Unknown workpiece / PCB / fuse

- Supported part numbers are parts that can be booked into a high-bay warehouse at simulation time t=0s.
- All other part numbers – in particular user-defined parts – can be generated and stored into a high-bay warehouse during production, but not replicated in a warehouse at simulation startup at t=0s.

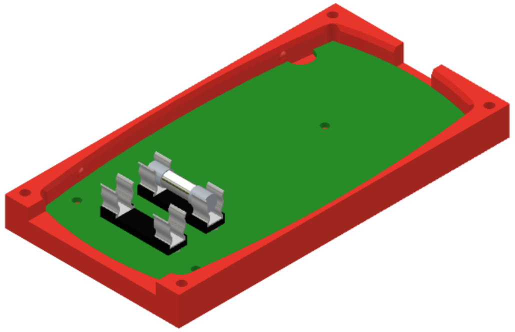
FESTO

Standard Part Numbers

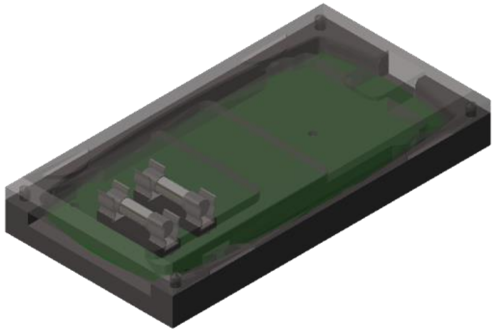
Some examples



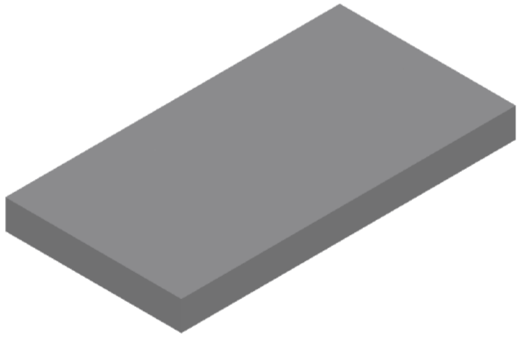
Part number 412
(blue, PCB, front fuse)



Part number 513
(red, PCB, rear fuse)



Part number 1214
(black, both fuses, back cover)



Part number 109
(grey, raw material front cover)

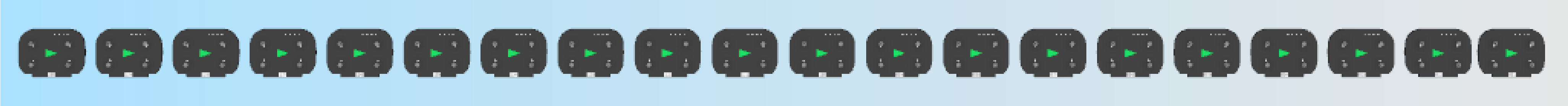
FESTO

Groups and Utilities

- Groups define sets of parts somehow belonging to each other.
- Besides the list of parts belonging to a group each group also contains a unique ID and a description.
- There is no restriction on the number of different types of parts a group consists of.
- Parts can be attached to an arbitrary number of groups or not attached to any group at all.
- Typically, groups are used to define zones and restrictions wrt. buffer positions of a high-bay warehouse.
- Utilities defines a special subclass or group of parts, containing all carriers, boxes, and pallets, which are the ones that are responsible for transport purposes.

Carriers

- In CIROS two different kinds of carriers are supported
- CP-F-CARRIER: Carriers without Pallets

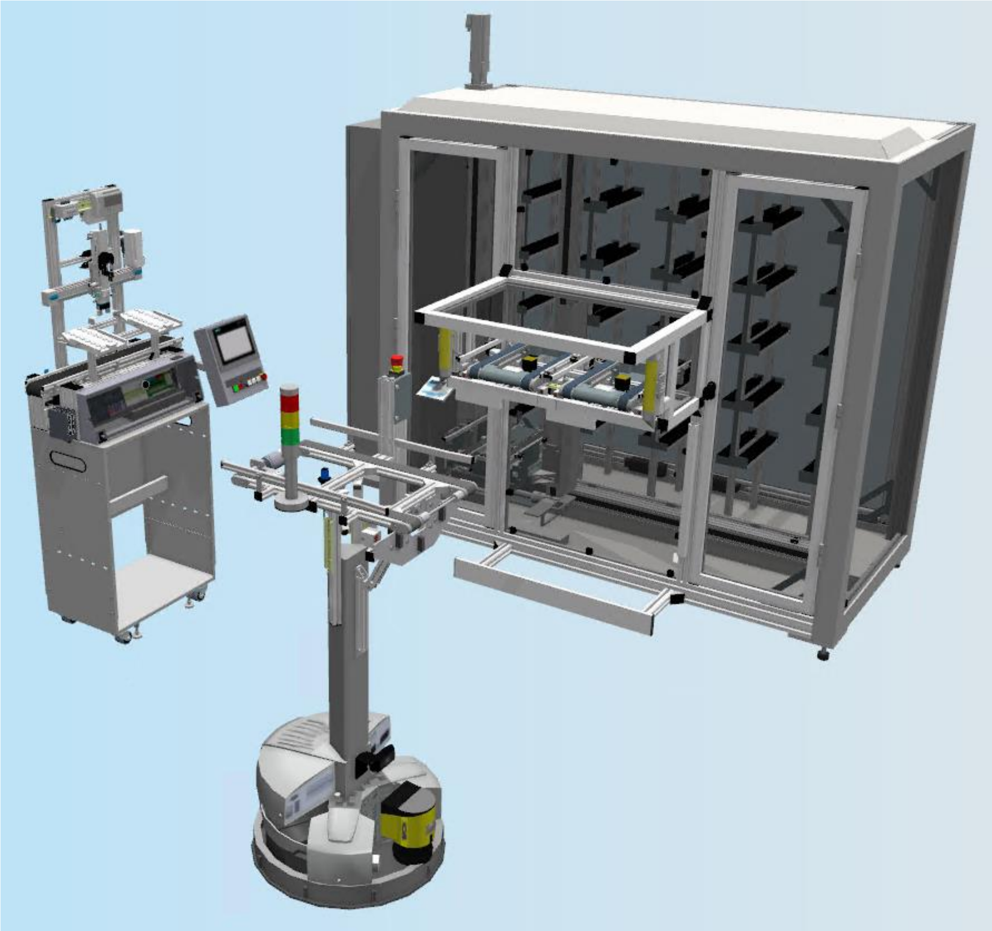


- CP-L-CARRIER: Carriers including Pallets

- Carriers without Pallets must be used in cases in which a CP Factory high-bay warehouse is part of the CIROS model, since the workpieces will be stored and released together with pallets in this situation!
- In all other cases Carriers including Pallets should be used
- Rule of thumb according to the number of carriers used within a CIROS model: One carrier per conveyor belt.

Resources and Buffers

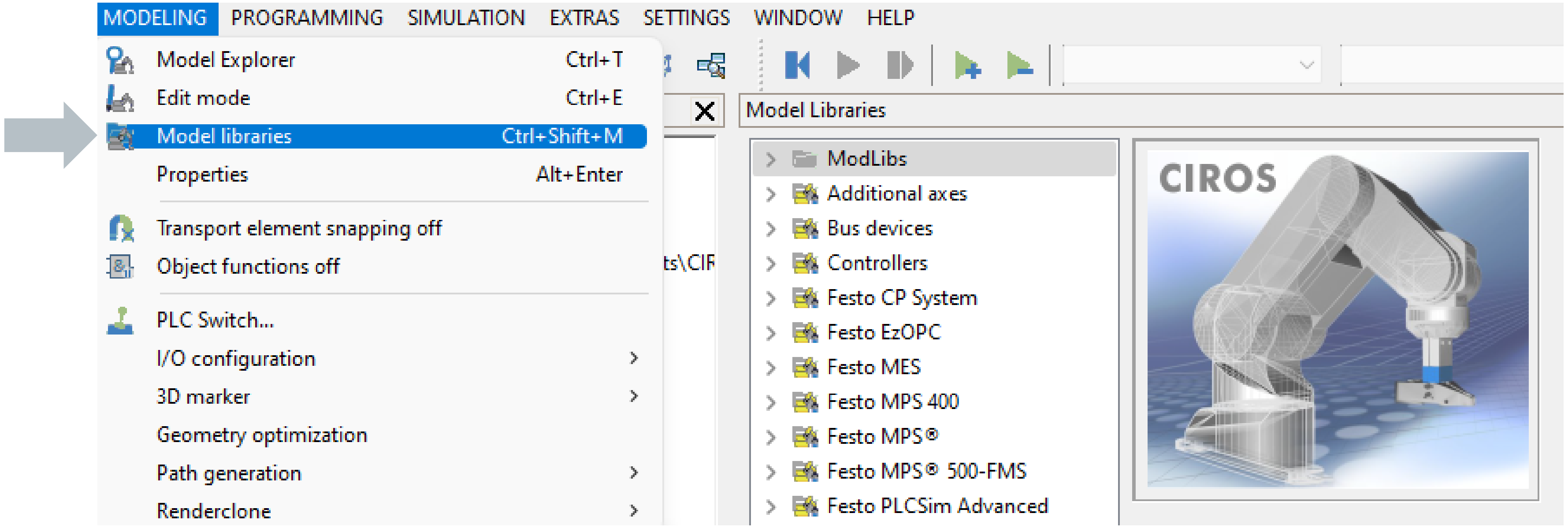
- Resources are the production facilities (modules) of the CP Lab / Factory.
- Each resource is represented by a unique ID.
- Periodically, each resource sends a status update to MES4 and gets process data back when initiating a corresponding service call.
- Robotino is a special kind of a mobile resource.
- Some resources contain buffers, like the high-bay warehouse, the robot assembly station, Robotinos, and the branches.
- Each buffer consists of at least one buffer position to store parts, one per buffer position.
- MES4 could be configured in such a way that, based on zones and restrictions, buffer positions are allowed to store specific types of parts only.



CP-System Model Libraries

Open Model Libraries Window

Modelling → Model libraries

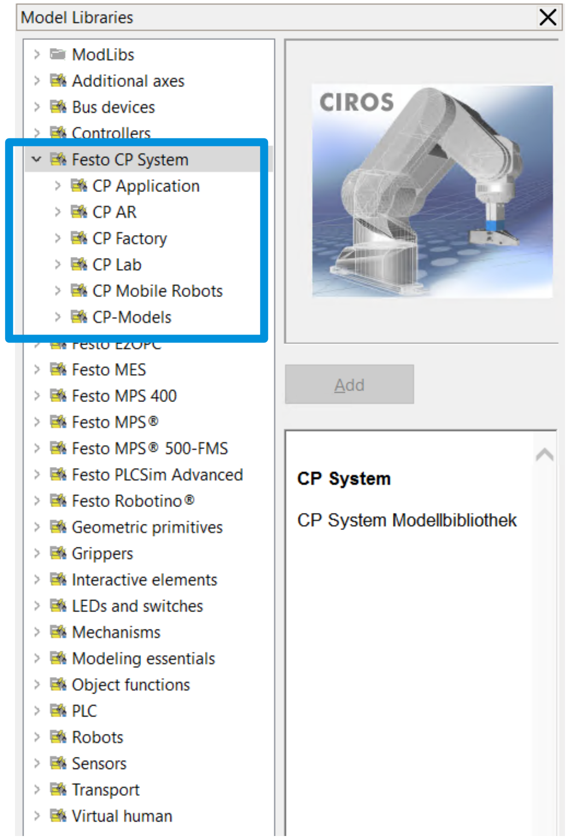


Model Library Festo CP System

- There are four groups containing CP Lab / Factory modules

Name	Description
CP-Application	Application modules
CP AR	AR Marker for Festo Didactic Augmented Reality
CP Factory	CP-Factory based modules and stations
CP Lab	CP-Lab based modules
CP Mobile Robots	Robotino related modules
CP-Models	CP-Lab standard systems with configured MES4 v1 database

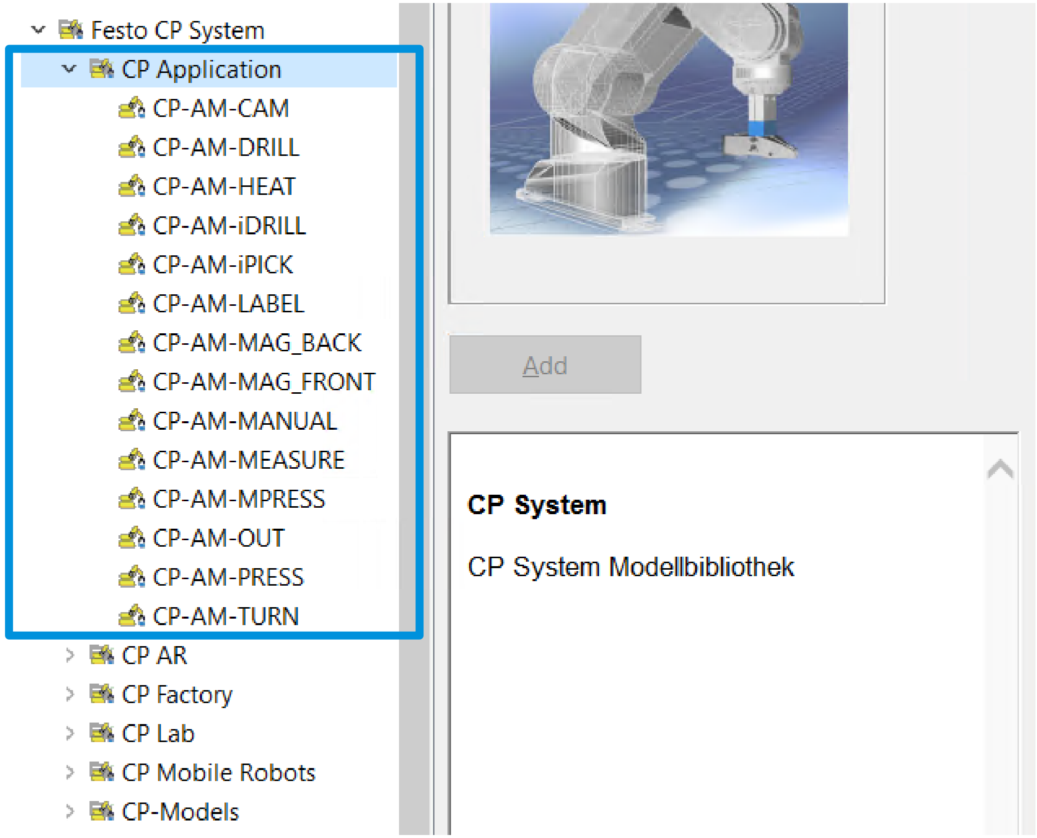
- For each module there is a brief description and a tiny image.



Model Library Festo CP System

CP Application

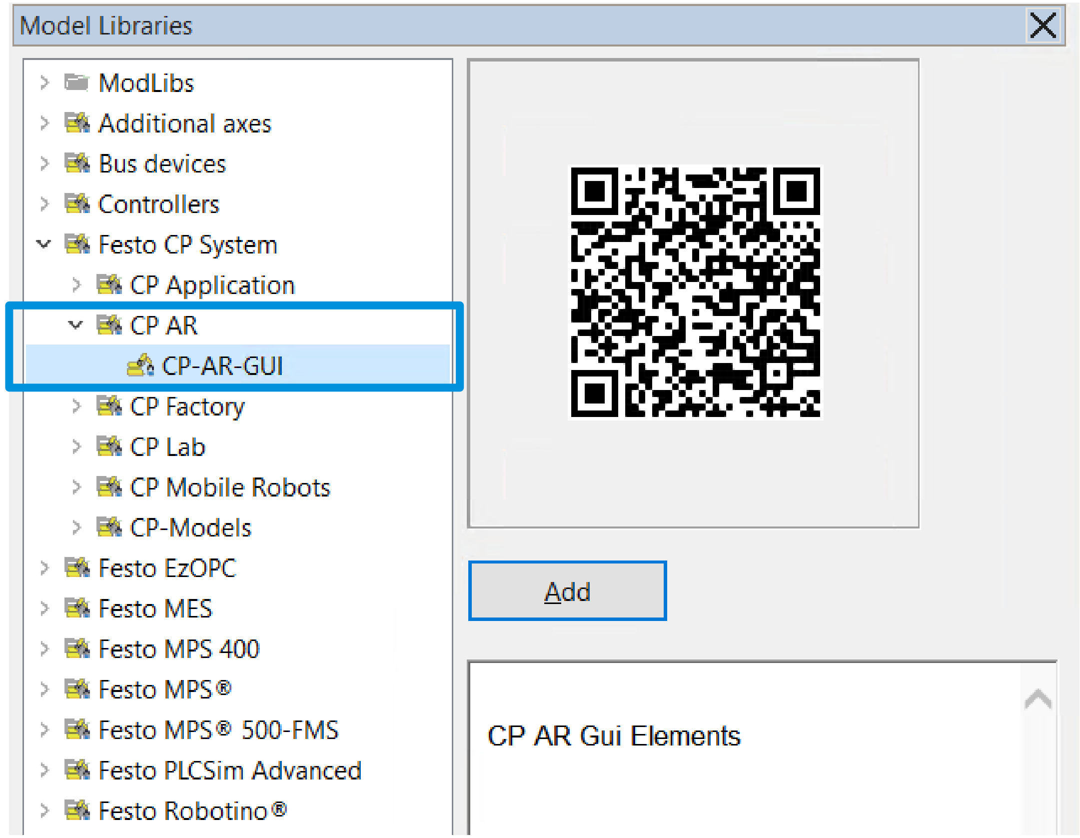
Module	Description
CP-AM-CAM	Camera inspection
CP-AM-DRILL	Drilling
CP-AM-HEAT	Heating tunnel
CP-AM-iDRILL	Drilling module with own PLC
CP-AM-iPICK	Pick by light with own PLC
CP-AM-LABEL	Labelling printer
CP-AM-MAG_BACK	Back cover magazine
CP-AM-MAG_FRONT	Front cover magazine
CP-AM-MANUAL	Manual working place
CP-AM-MEASURE	Analog measurement
CP-AM-MPRESS	Muscle press
CP-AM-OUT	Workpiece output
CP-AM-PRESS	Pressing
CP-AM-TURN	Workpiece flipping



Model Library Festo CP System

CP AR

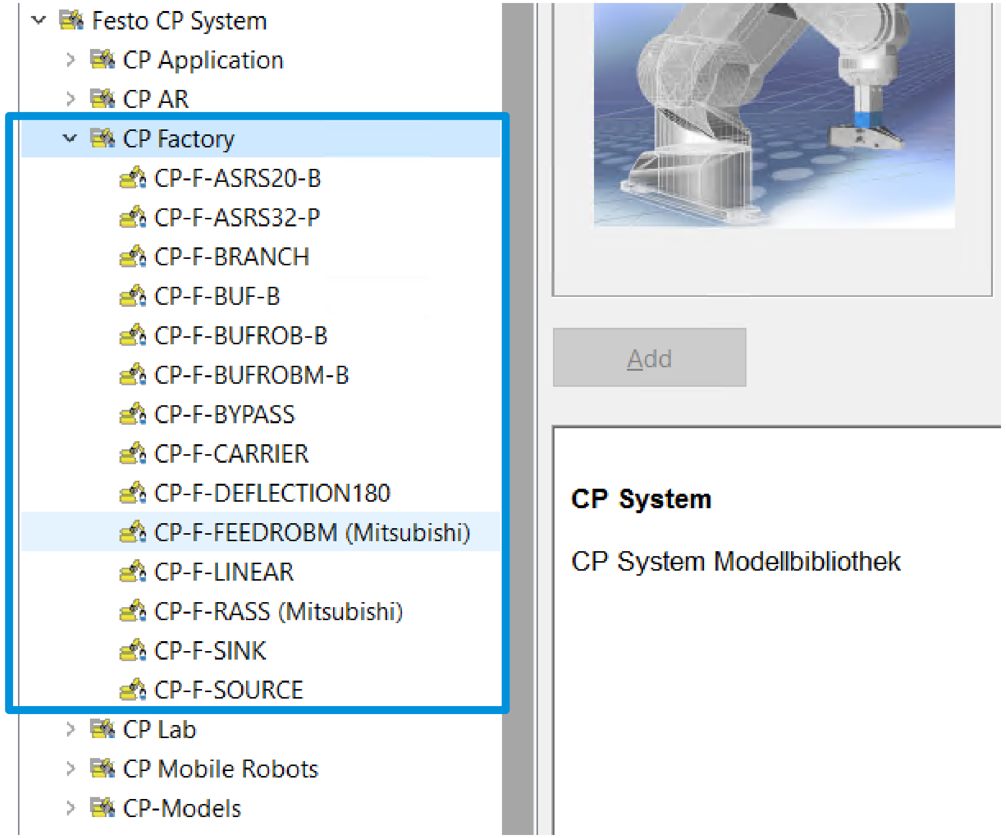
Module	Description
CP-AR-GUI	CP-AR QR-Code to the AR server.



Model Library Festo CP System

CP Factory

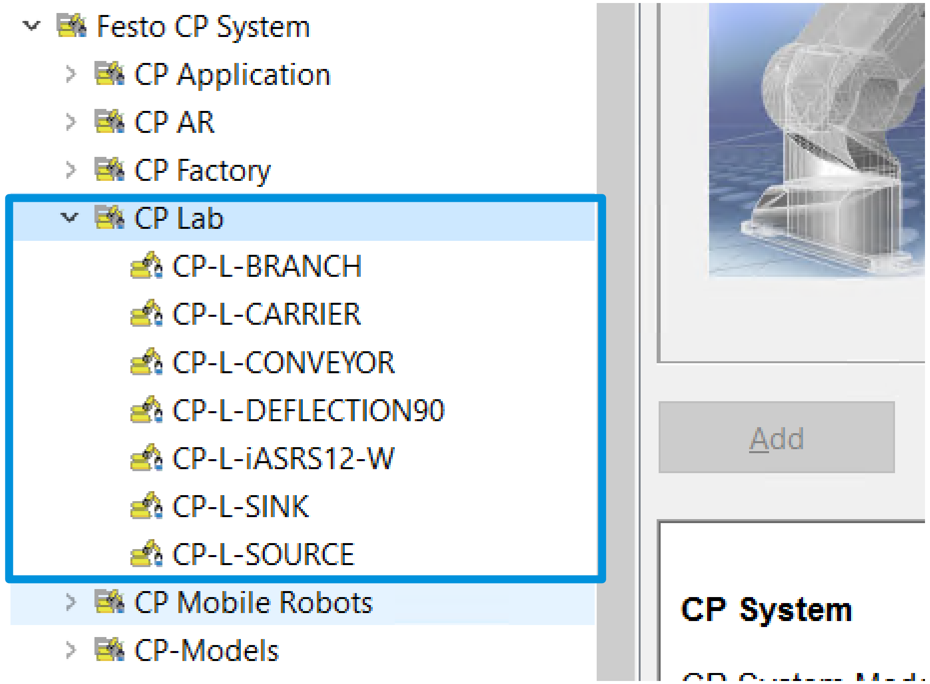
Module	Description
CP-F-ASRS20-B New	Automated Storage Retrieval System for boxes
CP-F-ASRS32-P	Automated Storage Retrieval System for parts and pallets
CP-F-BRANCH	Branch base module
CP-F-BUF-B New	Manual working place with automated box transfer
CP-F-BUFROB-B New	Part transfer between box and CP-Factory line with robot
CP-F-BUFROBM-B New	CNC milling application attached to robot and box transfer
CP-F-BYPASS	CP-Factory base module with bypass belt
CP-F-CARRIER	15 carriers for CP-Factory base modules without pallet
CP-F-DEFLECTION180	Passive 180° deflection
CP-F-FEEDROBM	CNC milling application attached to robot and CP-F linear
CP-F-LINEAR	CP-Factory base module with two parallel conveyor belts
CP-F-RASS	Robot assembly station
CP-F-SINK	Sink to remove carriers, pallets, workpieces
CP-F-SOURCE	Source to generate carriers, pallets, workpieces



Model Library Festo CP System

CP Lab

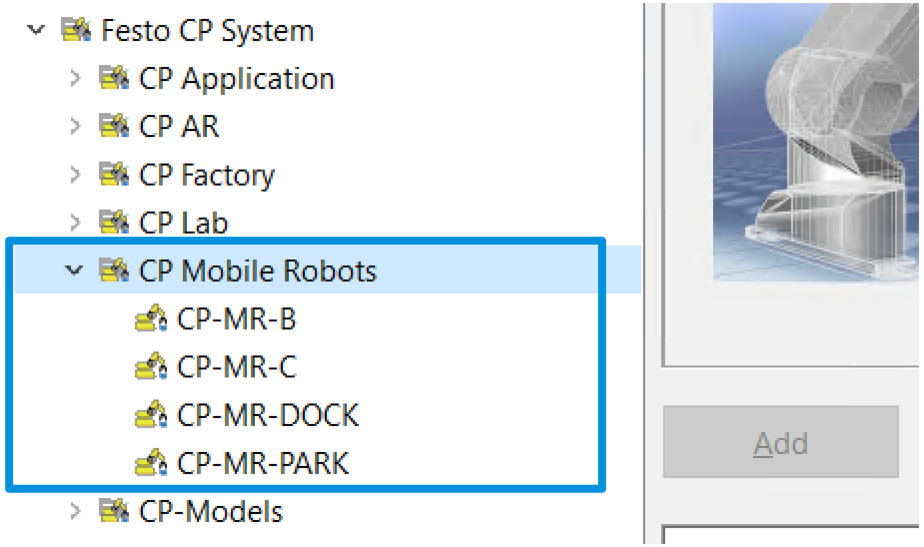
Module	Description
CP-L-BRANCH	Branch base module
CP-L-CARRIER	15 carriers for CP-Lab base modules with pallet
CP-L-CONVEYOR	CP-Lab base module with a conveyor belt
CP-L-DEFLECTION90	Passive 90° deflection
CP-L-iASRS12-W	Automated Storage Retrieval System for parts
CP-L-SINK	Sink to remove carriers, pallets, workpieces
CP-L-SOURCE	Source to generate carriers, pallets, workpieces



Model Library Festo CP System

CP Mobile Robots

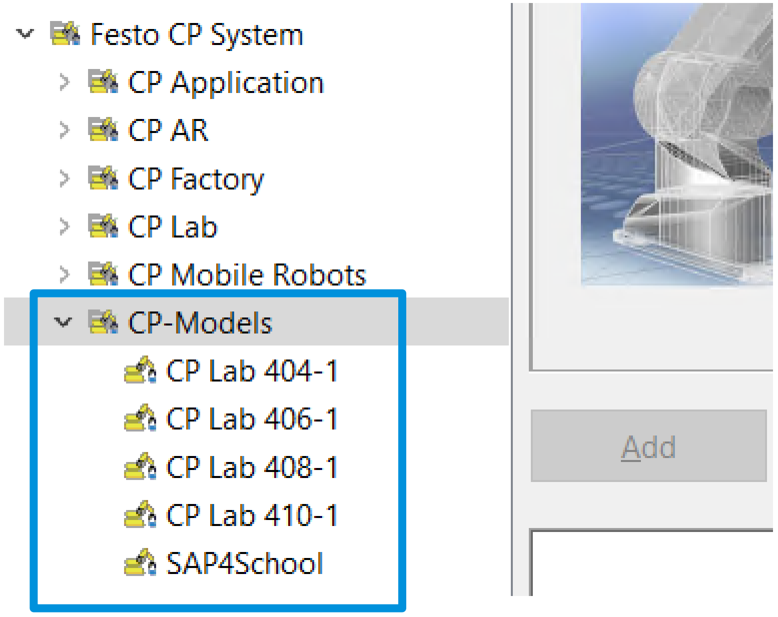
Module	Description
CP-MR-B New	Robotino for boxes
CP-MR-C	Robotino for carriers
CP-MR-DOCK	Docking kit to be mounted on branches to enable (un)docking maneuvers by a Robotino
CP-MR-PARK New	Robotino parking position



Model Library Festo CP System

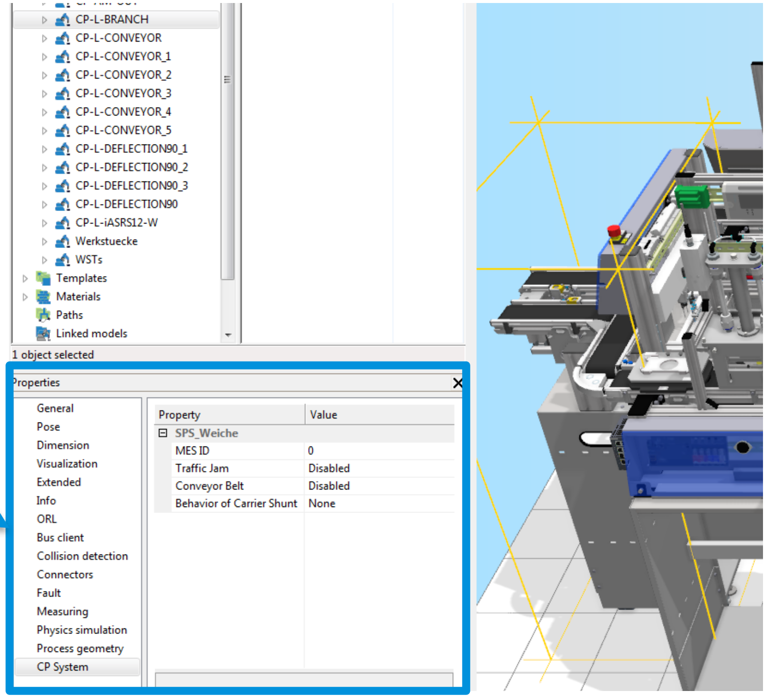
CP-Models

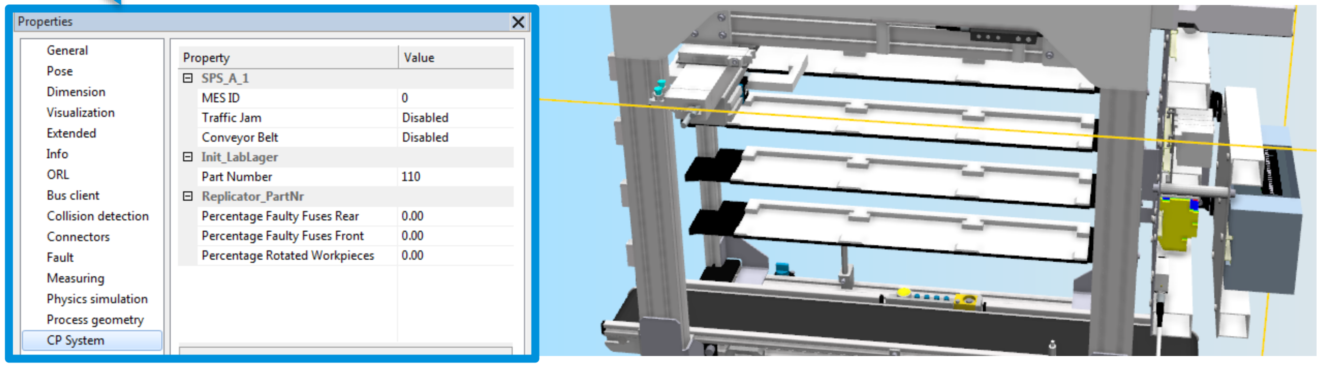
Module	Description
CP Lab 404-1	Standard CP Lab system with four stations and configured MES4 v1 1. Front cover magazine 2. Analog measurement 3. iDrill 4. Workpiece Output
CP Lab 406-1	Standard CP Lab system with six stations and configured MES4 v1 1. All modules in CP Lab 404-1 2. Back cover magazine 3. Pressing
CP Lab-408-1	Standard CP Lab system with six stations and configured MES4 v1 1. All modules in CP Lab 406-1 2. Pick by light 3. Labelling
CP Lab 410-1	Standard CP Lab system with six stations and configured MES4 v1 1. All modules in CP Lab 408-1 2. Camera inspection 3. Turning
SAP4School	Model for virtual commissioning of SAP4School



Configuration in Properties Section CP System

- For almost all components there is a section CP System as being part of the corresponding Properties menu.
- Represents the options that can be defined at the HMIs of a real CP Lab / Factory
  - MES ID
  - Traffic jam control
  - Energy saving (stopping the belts whenever possible)
  - Behavior of branches
- Additionally, one can configure some CIROS internal parameters not available on a real CP Lab / Factory.

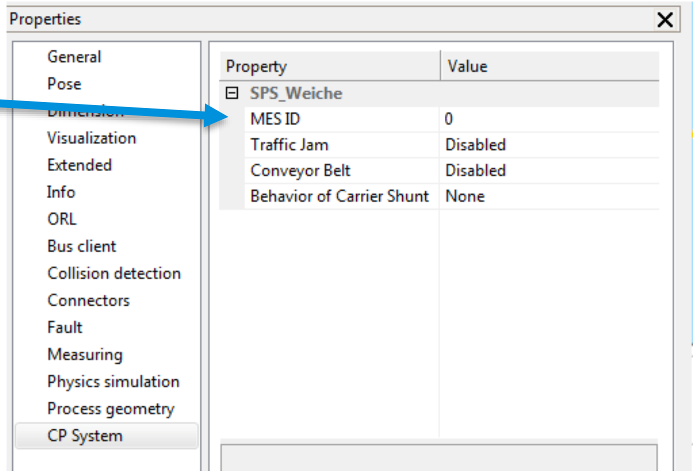




Configuration in Properties Section CP System

Define MES ID (1)

- MES ID = 0 → Default mode without using MES4
- MES ID › 0 → MES4 mode



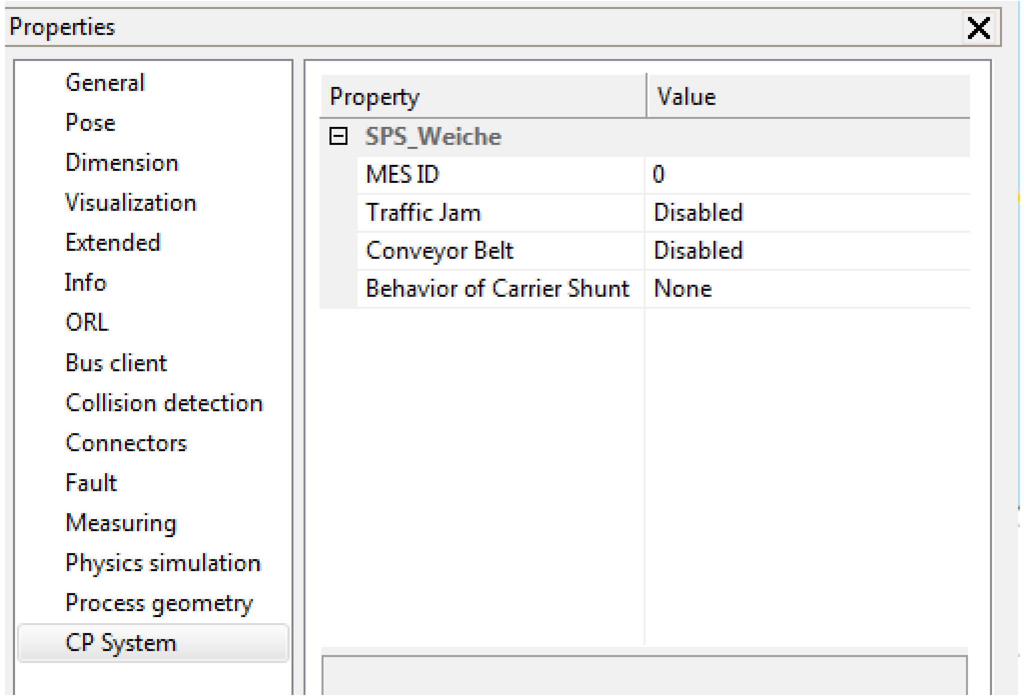
- Note, that the default mode of CIROS is not equal to the default mode of a real CP Lab / Factory!
- If the MES ID of at least one component is greater 0, running the CIROS simulation without MES4 results in an error message!

Configuration in Properties Section CP System

Define MES ID (2)

- Typically, MES ID is defined with one of the two ideas below:
  - Based on the MES IDs of a similar real CP Lab / Factory.
  - According to the default process to be performed, starting with MES ID = 1 for the component which is executing the first step of the process.

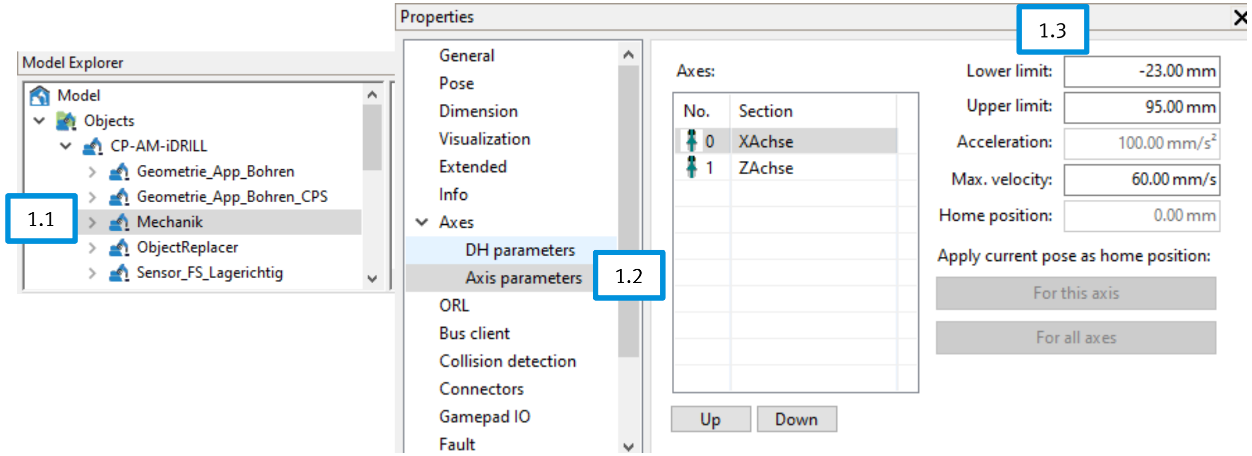
- Constraints
  - MES IDs must be unique throughout the entire model .
  - Each MES ID must be greater 0.
  - Definition of IDs within CIROS and MES4 must match each other.



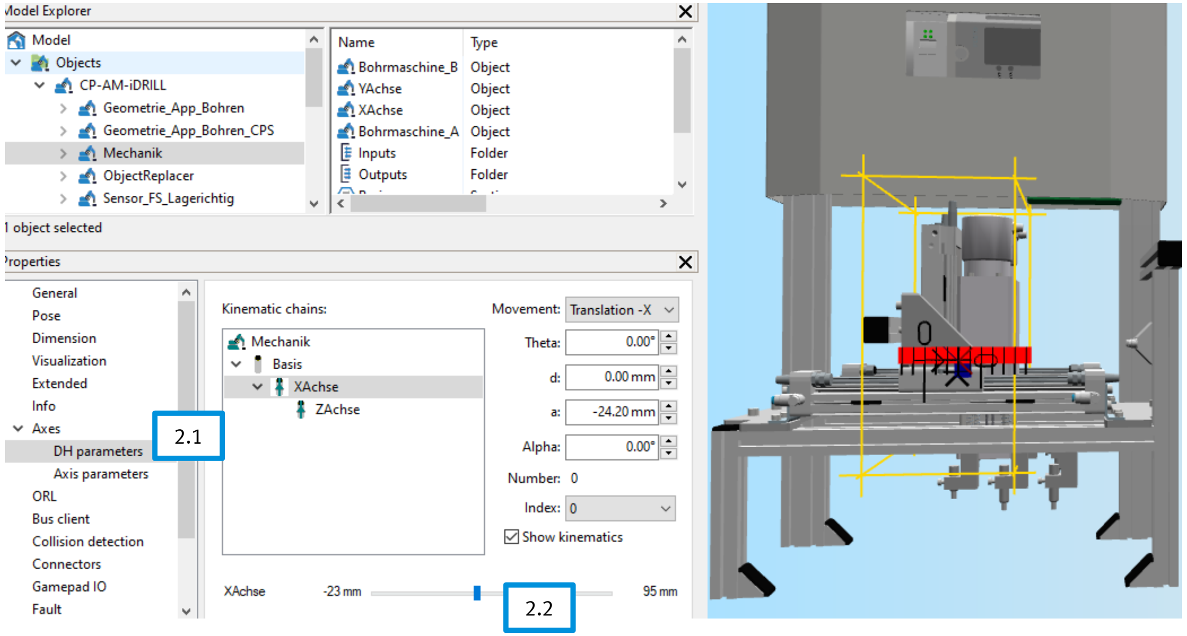
CP-AM-iDRILL

Configure x-axis linear drive

1. Configure axis lower limit, upper limit and maximum velocity.
    1. In Model Explorer, select CP-AM-iDRILL → Mechanik.
    2. In Properties, select Axes → Axis parameters.
    3. Adjust Lower limit, Upper Limit and Max. velocity.



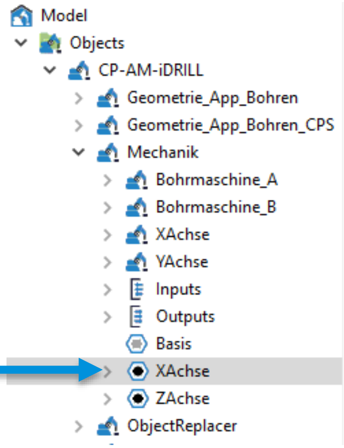
2. It is possible to see the position of the drill bit in model window.
    1. In Properties, select Axes → DH parameters.
    2. By dragging the scale XAchse at bottom, the drill bit will move accordingly



CP-AM-iDRILL

Object Mechanik (1)

- The object Mechanik has type Mechanism. It is a double acting cylinder.
- Object type double acting cylinder in CIROS has following attributes, which can be assigned to I/O. By default, following attributes are connected.
- The attributes can be viewed in CP-AM-iDRILL → Mechanik → Section XAchse → Properties → Mechanism.



Type: Cylinder, double-acting

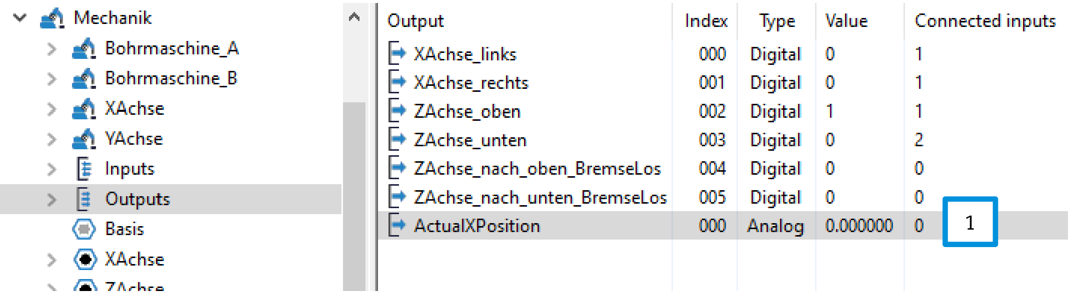
Attribute	Value
Retract [I/O]	XAchse_nach_links
Extend [I/O]	XAchse_nach_rechts
Speed [input]	
Retracted [output]	XAchse_links
Extended [output]	XAchse_rechts
Actual position [output]	
Movement characteristics (.csv)	
Min. limit defined by grip relation [gpp]	
Max. limit defined by grip relation [gpp]	
Pressure switch retracting [output]	
Pressure switch extending [output]	

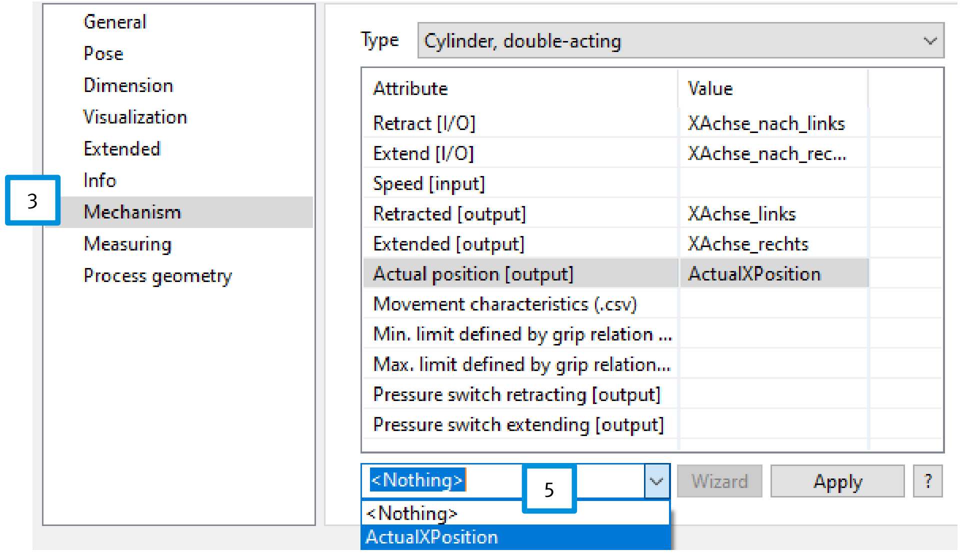
Note: Terms might varies in different language.

CP-AM-iDRILL

Object Mechanik (2)

- It is possible to assign own I/O to the attribute.
- In this example, An analog output ActualXPosition is assign to the attribute Actual position.



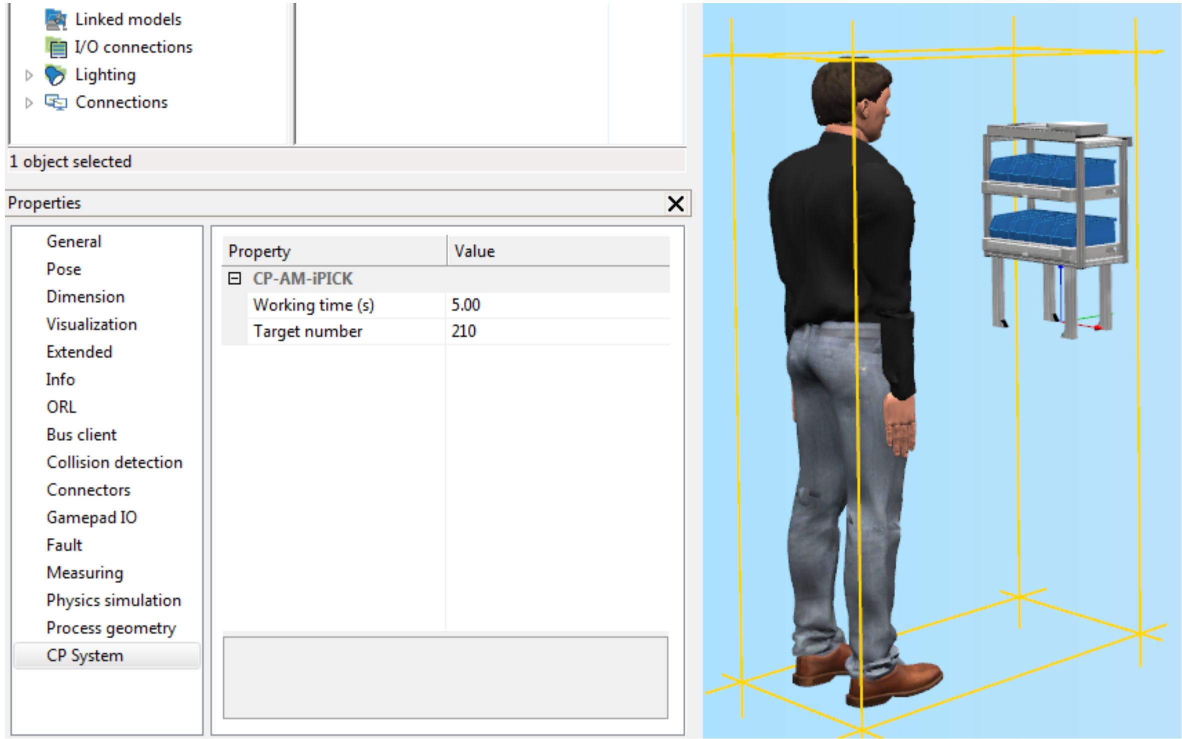


1. Add a new analog output with name ActualXPosition in object Mechanik.
2. In Model Explorer, select CP-AM-iDRILL → Mechanik → Section XAchse.
3. In Properties, select Mechanism.
4. In Attribute table, select the row Actual position [output].
5. Click on the drop down list at the bottom, select ActualXPosition.
6. Click Apply.

CP-AM-iPICK

Configuration in properties section CP System

Property	Description
Working time (s)	Working time required by the worker to finish an operation (valid for CIROS and CIROS/MES4).
Target number	Part number of the part to be replicated by the worker (valid for CIROS default mode only).

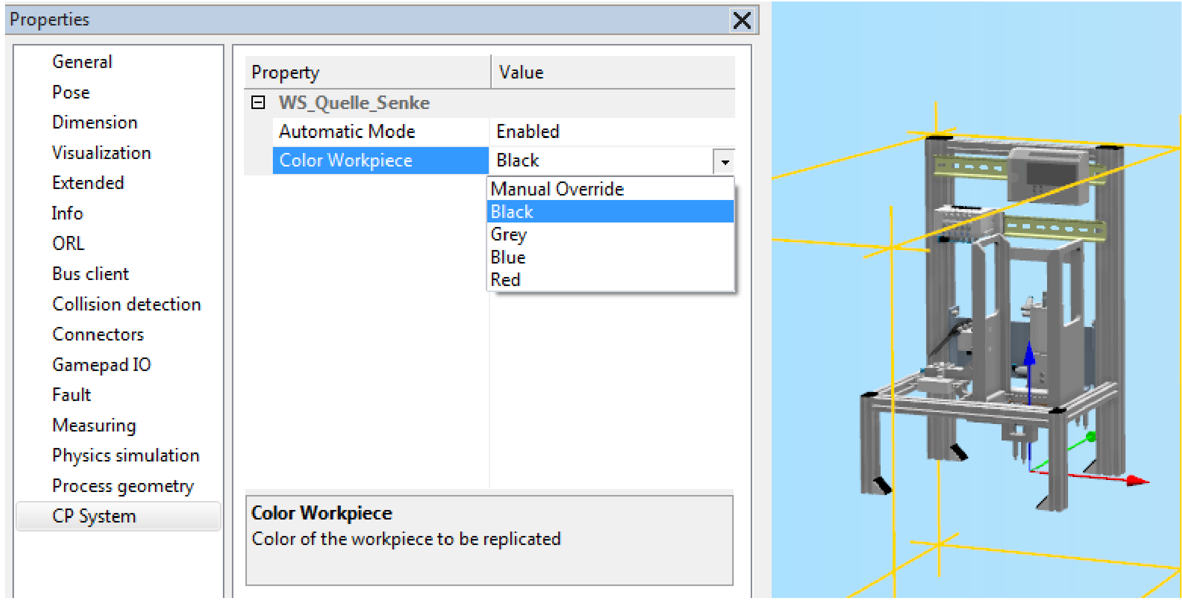


FESTO

CP-AM-MAG_BACK

Configuration in properties section CP System

Properties	Description
Automatic Mode	In “Automatic Mode” the magazine will be filled automatically during simulation.
Colour Workpiece	Defines the color of the back covers stored in the magazine. By enabling “Manual Override” the user can specify the color of each individual cover to be replicated.



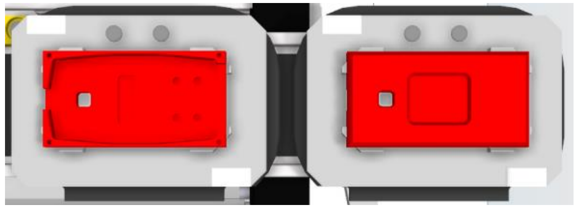
FESTO

CP-AM-MAG_FRONT

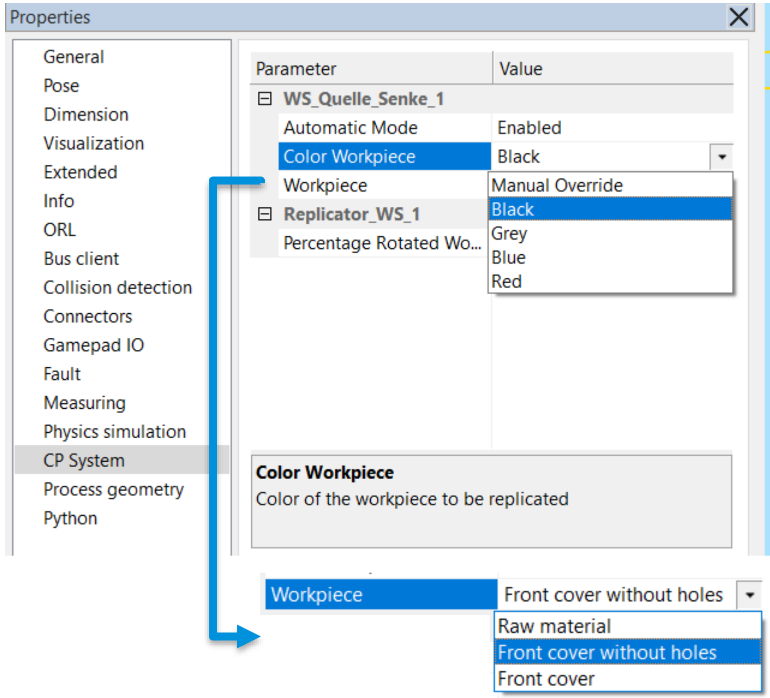
Configuration in properties section CP System

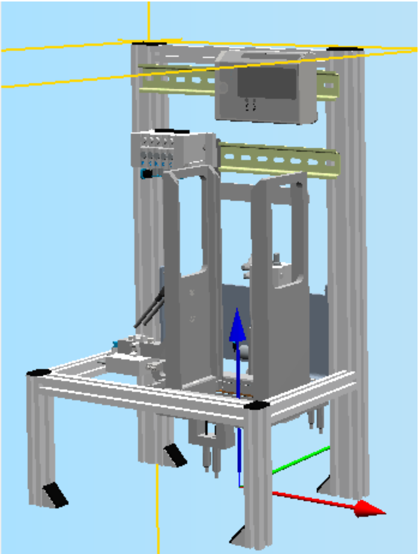
Properties	Description
Automatic Mode	In “Automatic Mode” the magazine will be filled automatically during simulation.
Colour Workpiece	Defines the color of the front covers stored in the magazine. By enabling “Manual Override” the user can specify the color of each individual cover to be replicated.
Workpiece	Defines the front cover type, is it a raw plastic block, front cover without drilled holes or final front cover.
Percentage Rotated Workpieces	Fault injection option, defines percentage of faulty part filled to the magazine.

Default orientation



Rotated orientation



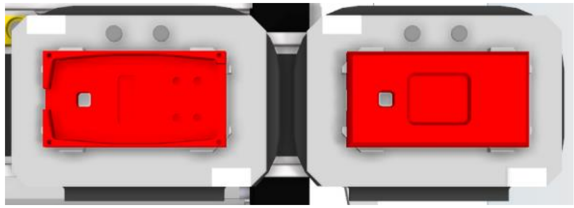


CP-AM-MANUAL

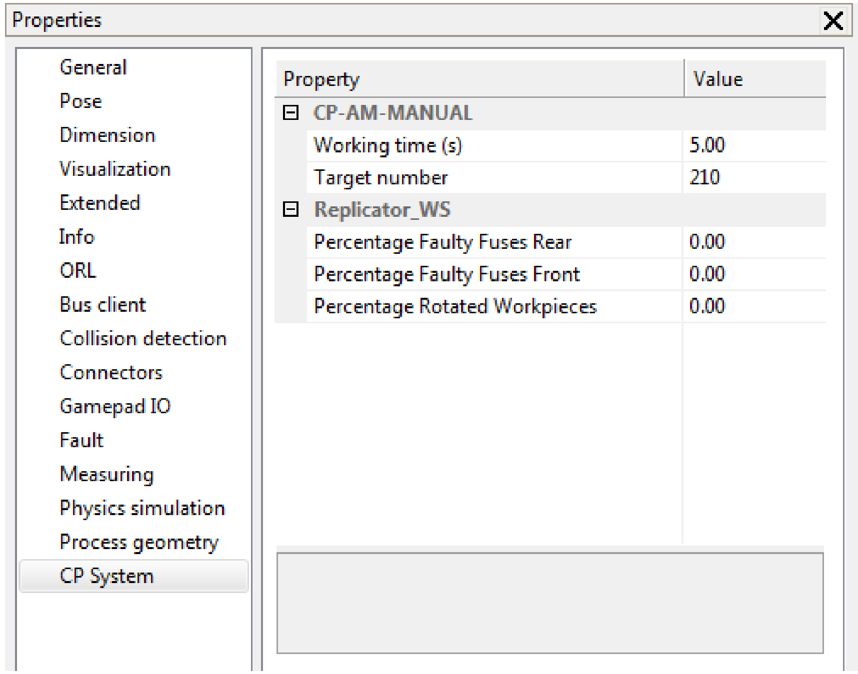
Configuration in properties section CP System

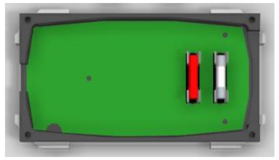
Properties	Description
Working time (s)	Working time required by the worker to finish an operation (valid for CIROS and CIROS/MES4).
Target number	Part number of the part to be replicated by the worker (valid for CIROS default mode only).
Percentage Faulty Fuses	Fault injection option. In case the worker must replicate workpieces with fuses, one can specify the percentage of “faulty” fuses assembled (faulty fuses are highlighted in red) .
Percentage Rotated Workpieces	Fault injection option. Percentage of workpieces placed in a rotated orientation (upside down).

Default orientation



Rotated orientation



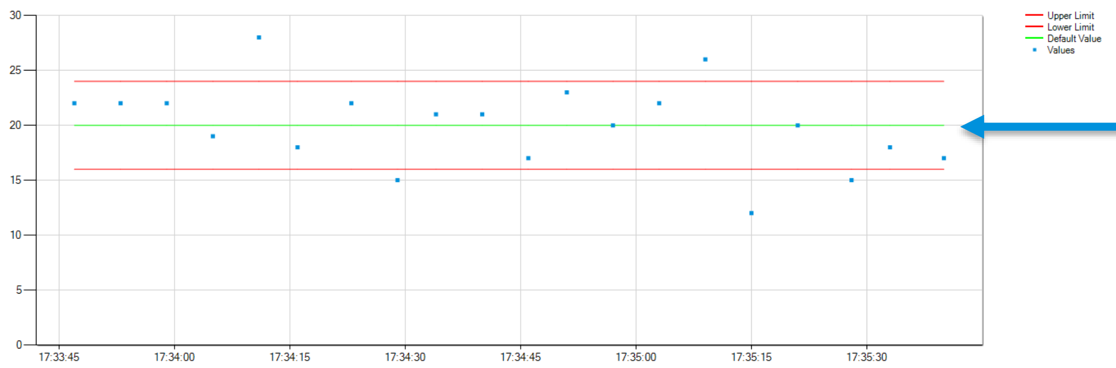


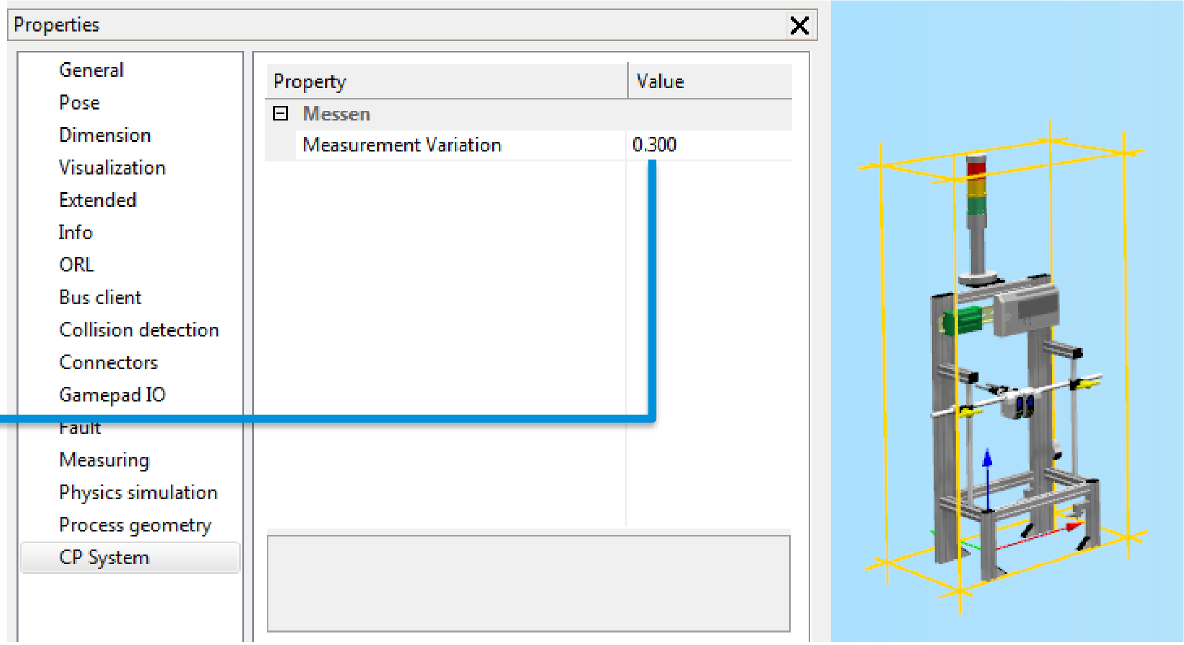
Faulty fuses are shown in red

CP-AM-MEASURE

Configuration in properties section CP System

Properties	Description
Measurement Variation	Fault injection option, adds some random “noise” with Gaussian filter to the measured value.

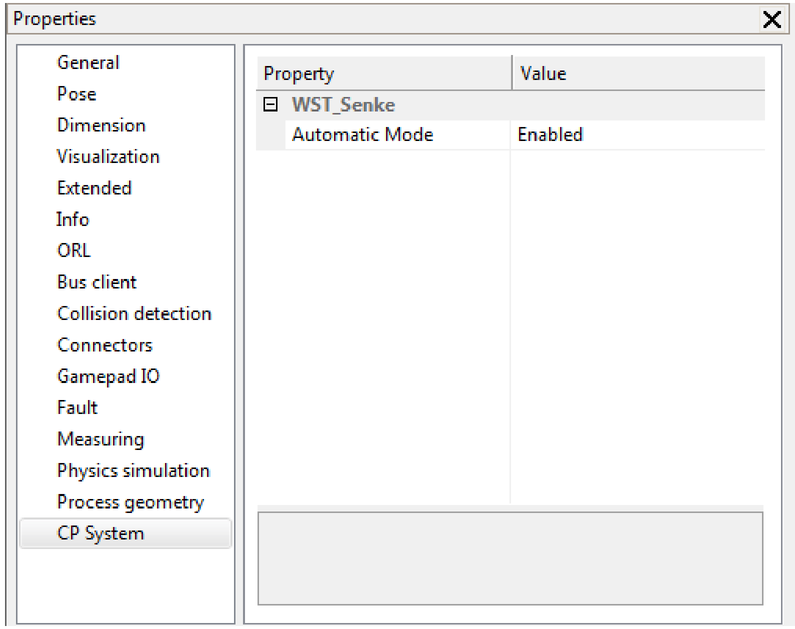


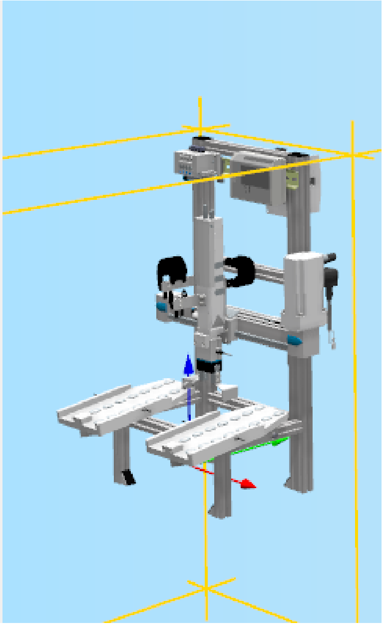


CP-AM-OUT

Configuration in properties section CP System

Properties	Description
Automatic Mode	In “Automatic Mode” workpieces will be removed from the ramps automatically during simulation. Otherwise, one might run into a traffic jam due to slides full!



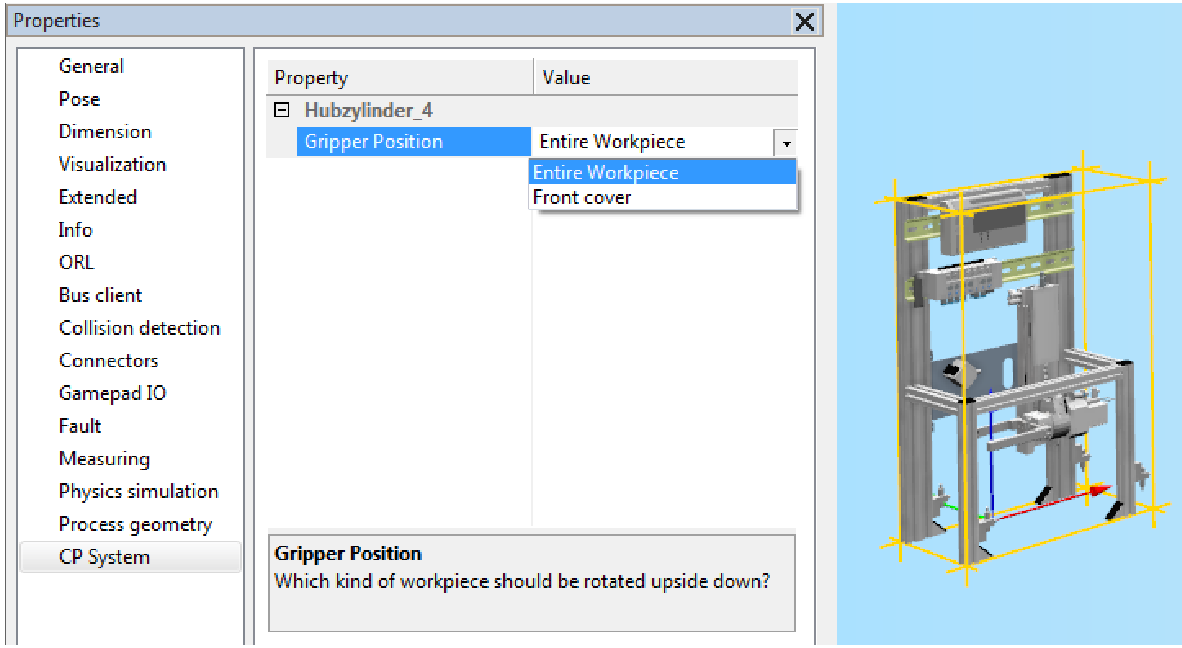


FESTO

CP-AM-TURN

Configuration in properties section CP System

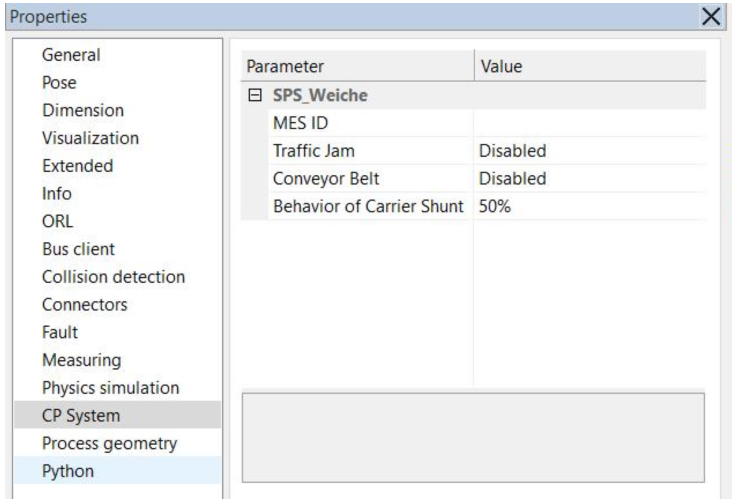
Properties	Description
Gripper Position	Before starting the simulation, one must specify whether front covers or entire workpieces (i.e., front & back covers pressed together) should be rotated upside down.

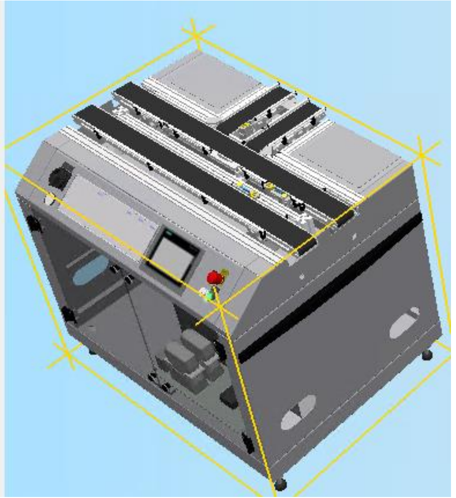


CP-F-BRANCH / CP-L-BRANCH

Configuration in properties section CP System

Properties	Description
Traffic Jam	Stop carrier if conveyor belt is occupied.
Conveyor Belt	Stop conveyor belt while application is running.
Behaviour of Carrier Shunt	Percentage of carriers without an order turning to the right.

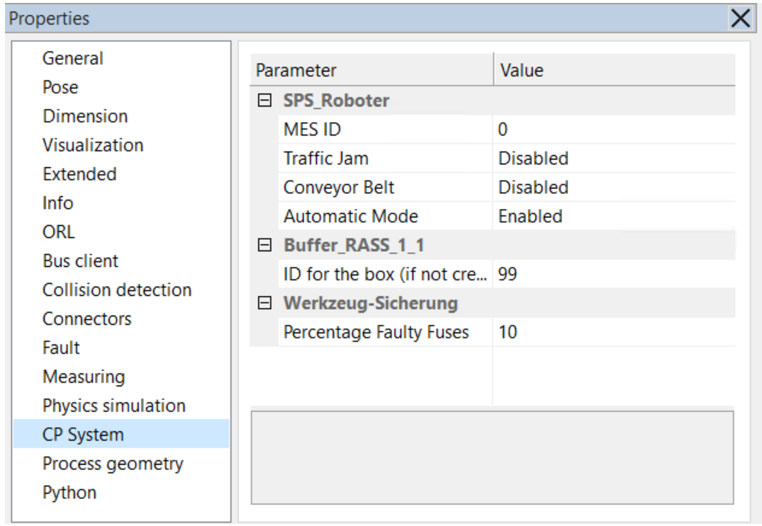


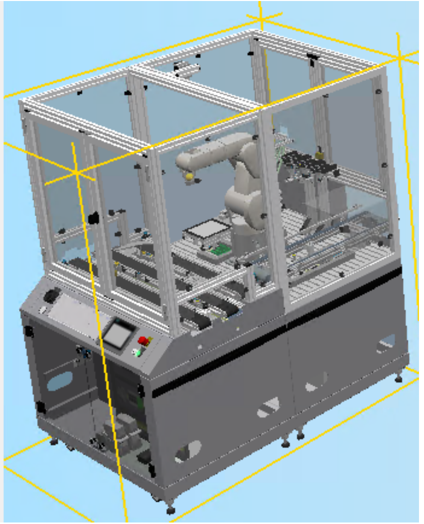


CP-F-RASS

Configuration in properties section CP System

Properties	Description
Traffic Jam	Stop carrier if conveyor belt is occupied.
Conveyor Belt	Stop conveyor belt while application is running.
Behaviour of Carrier Shunt	Percentage of carriers without an order turning to the right.
Automatic Mode	In automatic mode PCBs in the box will be created automatically regardless of buffer in MES4.
ID for the box	If the box is not created by MES, this will be the ID of the box.
Percentage Faulty Fuses	Percentage of “faulty” fuses assembled by the robot assembly station. Faulty fuses are highlighted in red.

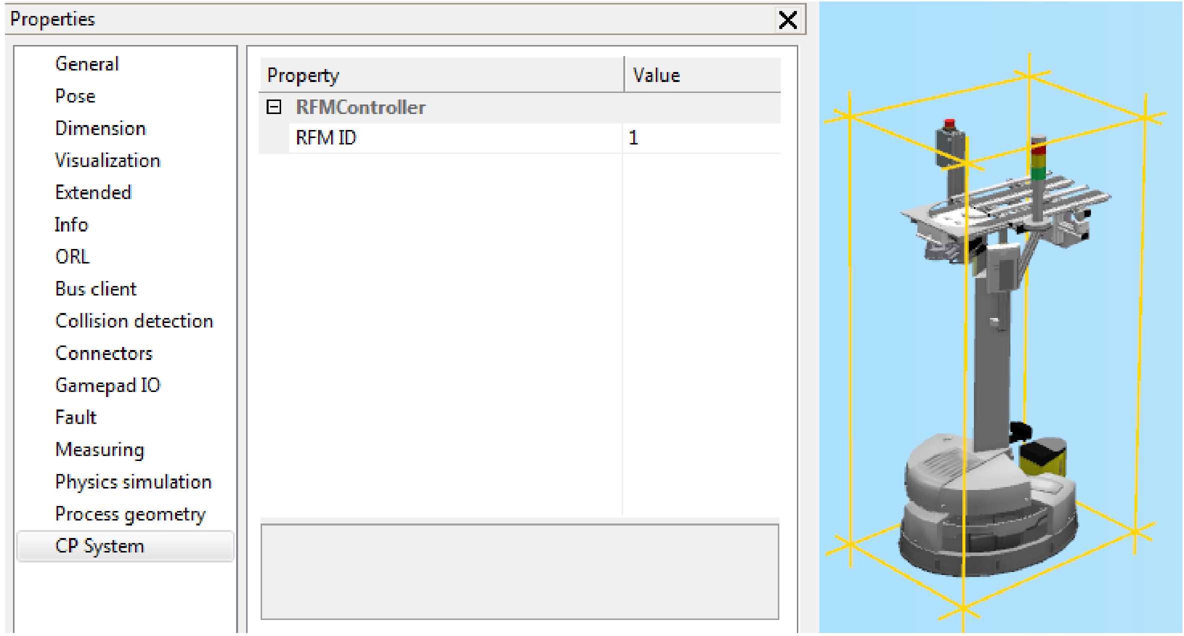




CP-MR-C / CP-MR-B

Configuration in properties section CP System

Properties	Description
RFM ID	Unique ID, mandatory for the Festo Fleet Manager to get access to a particular Robotino .

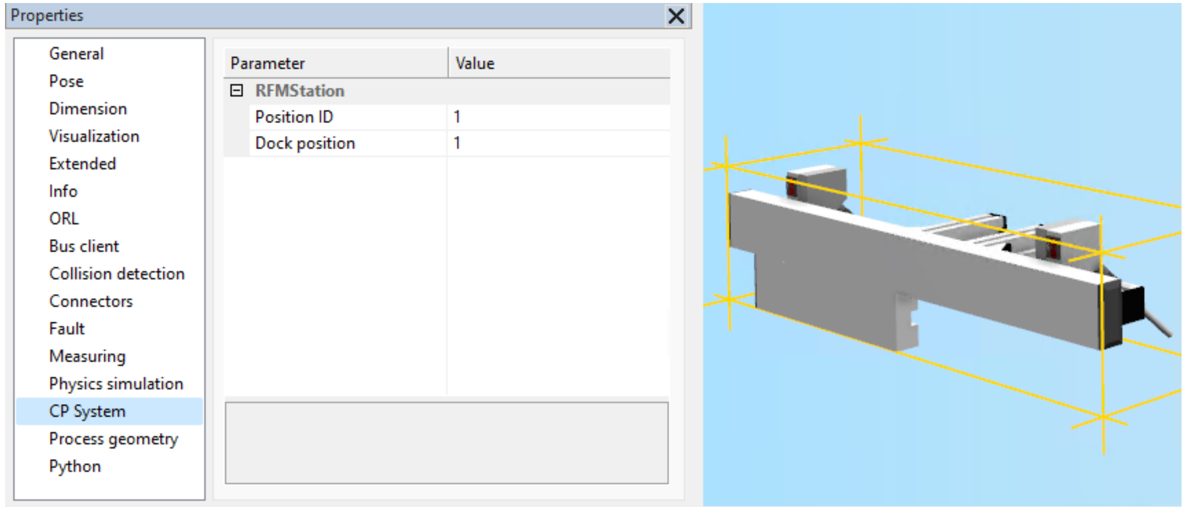


Note:
- CP-MR-C stands for Cyberphysical-Mobile Robot-Carrier.
- CP-MR-B stands for Cyberphysical-Mobile Robot-Box.
- RFM stands for Robot Fleet Manager.

CP-MR-DOCK

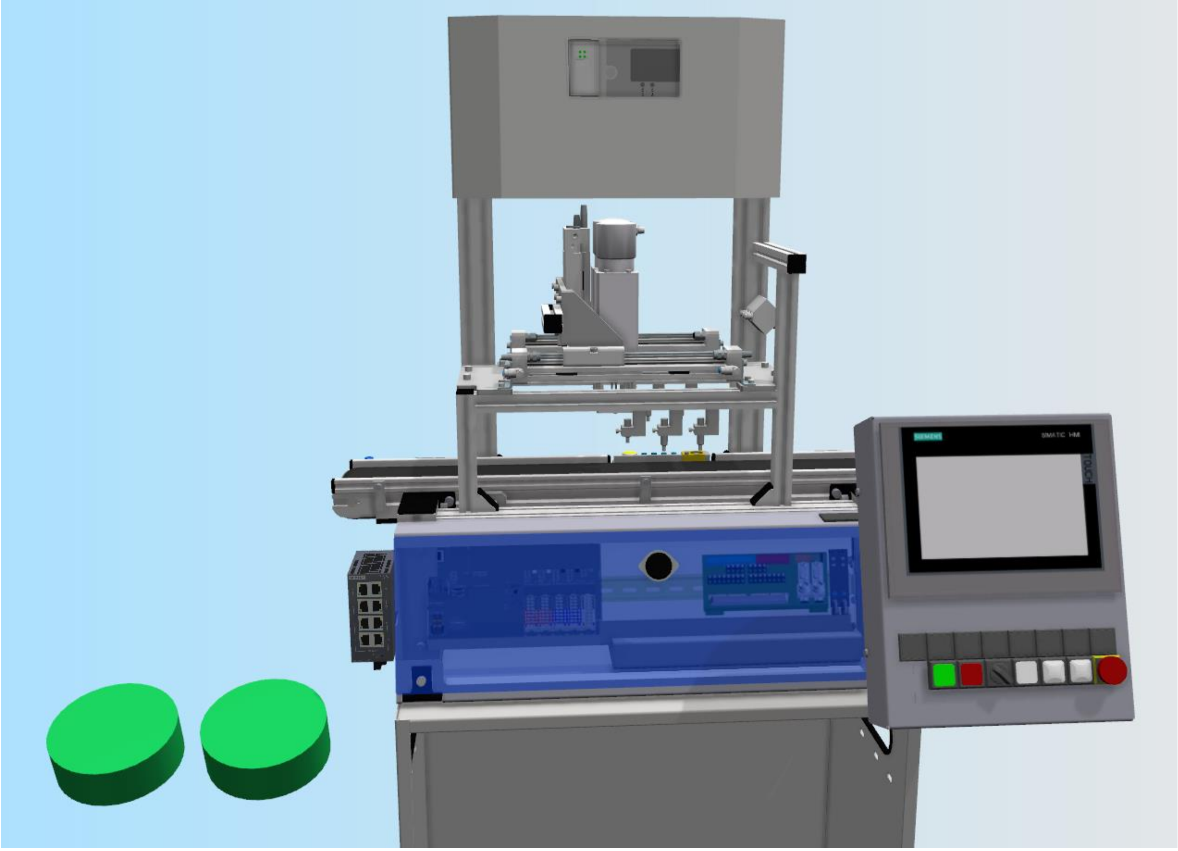
Configuration in properties section CP System

Properties	Description
Station ID	Unique ID, mandatory for the Festo Fleet Manager to differ between the various docking positions a Robotino could dock to within a model.
Dock position	Number of conveyor belts available for exchanging workpieces (always 1 for carrier Robotinos).



Sources and Sinks

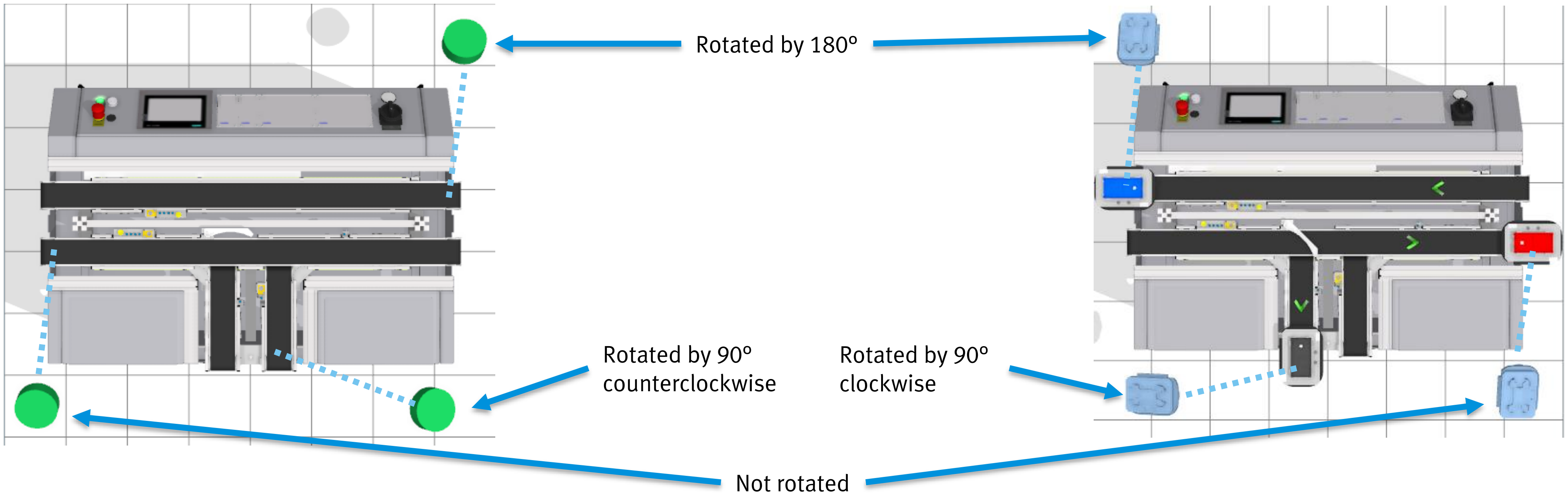
- Sources and Sinks can be used to dynamically replicate and remove workpieces during the simulation, either via pushing the corresponding button or automatically controlled by CIROS.
- There are different versions of sources and sinks, one set to be applied to CP Lab conveyor belts and one set to be used in combination with CP Factory components.
- Typically, sources and sinks come into play whenever one wants to model a single conveyor belt & application module.



Sources and Sinks

Remarks

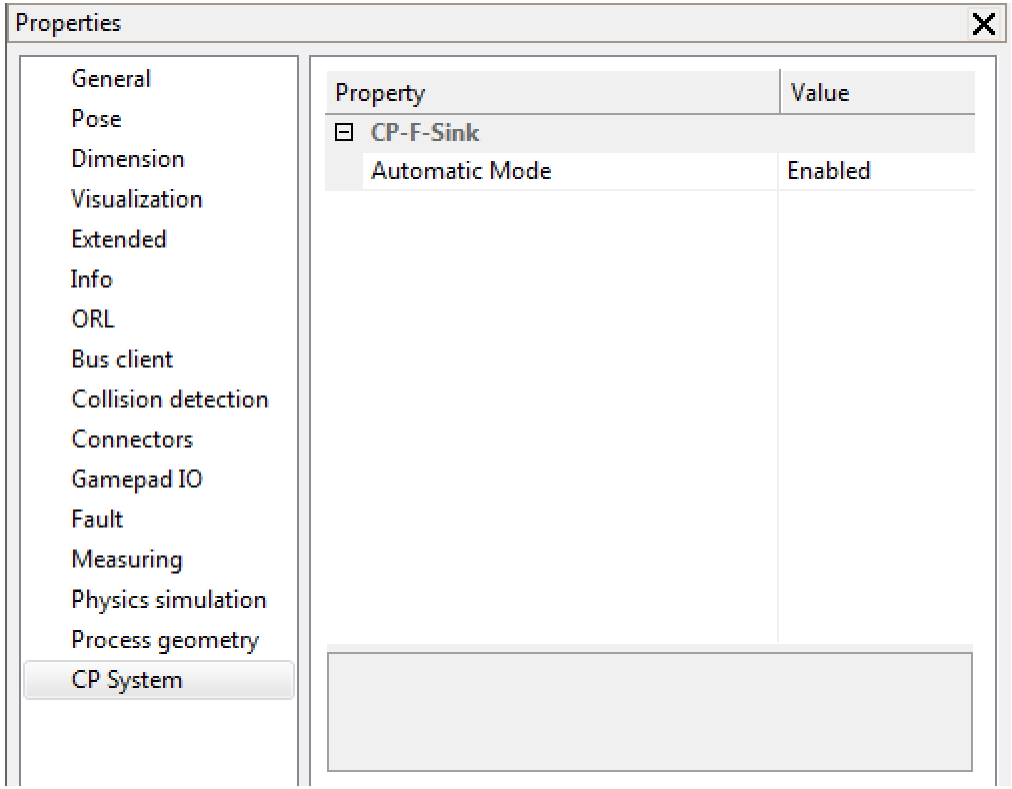
- Depending on the model, sources and sinks may have to be rotated to operate correctly!



Sources and Sinks

Configuration in properties section CP System

Properties	Description
Automatic Mode	If enabled carriers / pallets / workpieces within the range of the sink will be removed automatically during simulation.

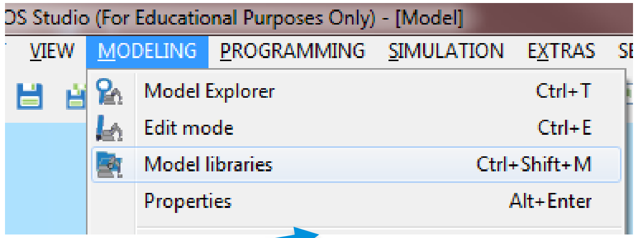


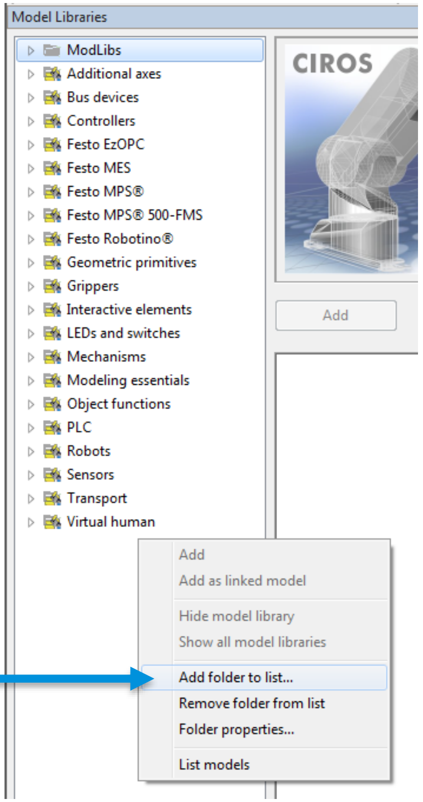
Adding New Libraries to CIROS

1. Open an already existing model or create a new CIROS model like shown before.

   Note: Model libraries window can only be opened when a CIROS model is opened or created!

2. Open the model libraries window
   - MODELING → Model libraries or
   - CTRL + SHIFT +M
3. Click on the right mouse button within the tree view part of the model libraries to open the corresponding context menu
4. Execute Add folder to list...
5. Select the folder in which the model library to be added is stored and press OK





Virtual Commissioning with MES4

Basic knowledge in Festo MES4 and Fleet Manager is required.

Virtual Commissioning with MES4 Tutorial

MES4 v2 and below

1. Create a new CIROS project.

   Video tutorial: 10_CreateNewCIROSModel.mp4
2. Build a CIROS model from model libraries.

   Video tutorial: 11_BuildAModelFromModelLibraries.mp4
3. Configure MES4.

   Video tutorial: 12_MES4ForCirosModel.mp4
4. Configure CIROS model respective to the MES4 configured.

   Video tutorial: 13_ConfigureCirosModelForMES4.mp4
5. Run the simulation

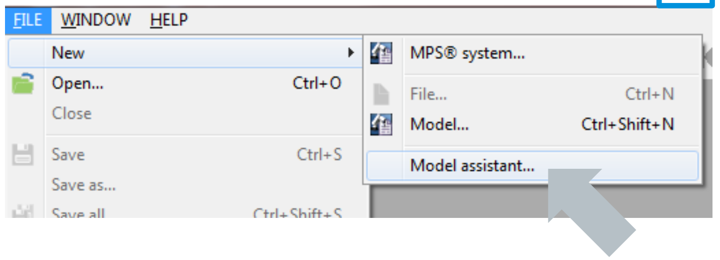
   Video tutorial: 14_SimulateAnMES4Order.mp4

Virtual Commissioning with MES4 Tutorial

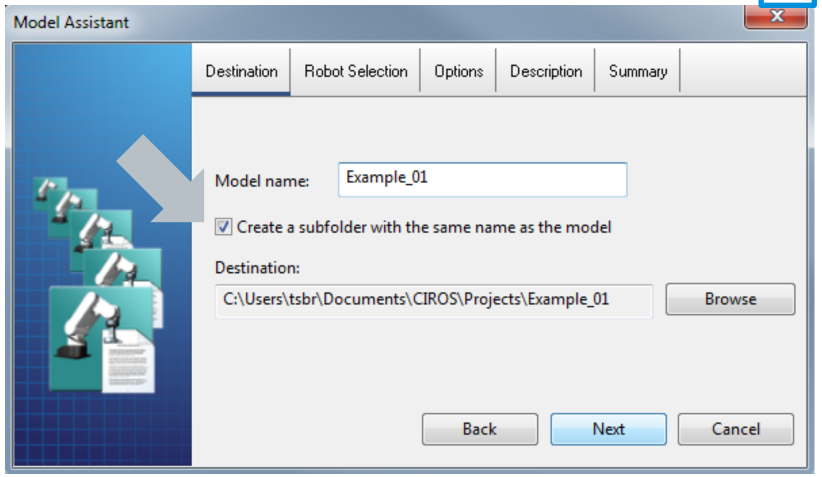
1. Create a new CIROS project.

1. Choose FILE → New → Model assistant
2. Specify the model's name and enable Create a subfolder with the same name as the model.
3. **Important:** Do not select a robot, these ones are not the ones integrated within CP Lab / Factory! Choose Empty model instead!

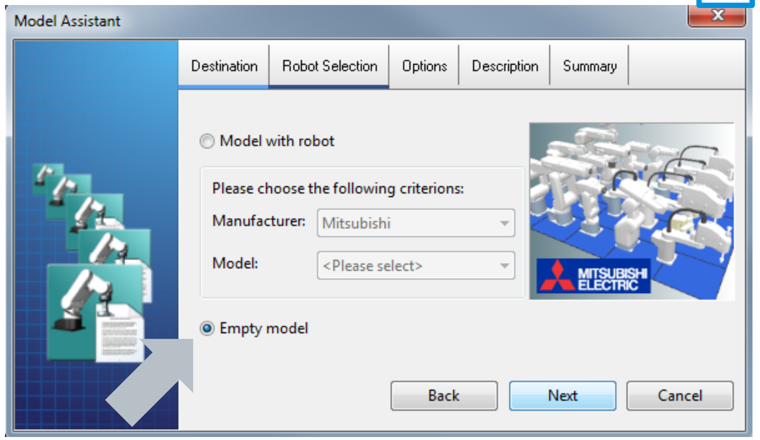
1



2



3



FESTO

Virtual Commissioning with MES4 Tutorial

2. Build a CIROS model from model libraries.

1. Switch to Top View within the list of views.
2. Open the model libraries.
3. Select all modules needed and add them to the model.

   Note: take care, that components properly snap into place.
4. After adding all modules, select each passive deflection and snap it into place again!
5. Depending on the model,
   1. Integrate sources and sinks if required.
   2. Adjust floor and background.
   3. Disable shadow simulation

Virtual Commissioning with MES4 Tutorial

Picture source: MES4 v2 or lower

3. Configure MES4.

1. According to the CIROS model add all resources in MES4.
   - Application modules and Robotino.
   - CP Lab branches (Remark: CP Factory branches are defined implicitly by the MES4 topology) .
   - Define the MES4 ID, IP address, type of PLC (i.e. Siemens) of each resource.
   - Specify the system’s topology.
2. Start the CIROS simulation and check in MES4 Production Control → Resources whether all resources are available and active.
3. Define all parts that are required by the various work plans.
4. Add work plan(s).
5. In case a high-bay warehouse is part of the model, specify the initial load of the corresponding buffers.
6. Clear all other buffers (branches, Robotino!).
7. Start a production or customer order and check that everything works fine.

FESTO

Virtual Commissioning with MES4 Tutorial

4. Configure CIROS model respective to the MES4 configured.

1. Configure the CP System options for each component.
    1. MES ID in CIROS should be the same as Resource ID in MES4.

Virtual Commissioning with FactoryViews[1]

Import Model from Python Script

- Python script to generate model with configured MES4 resource ID can be generated from Festo Factory View, which is the successor of Festo MES4 Software.
- The script can be executed in CIROS to configure the CP System setup for virtual commissioning.
- With this option, time to insert the models and to configure the resource IDs is saved.
- However, the carriers are not generated from the code. Thus, they have to be added manually from Festo CP System Model Library.

[1] FactoryViews is the new software bundle for MES and web based services. It contains MES4 v3.

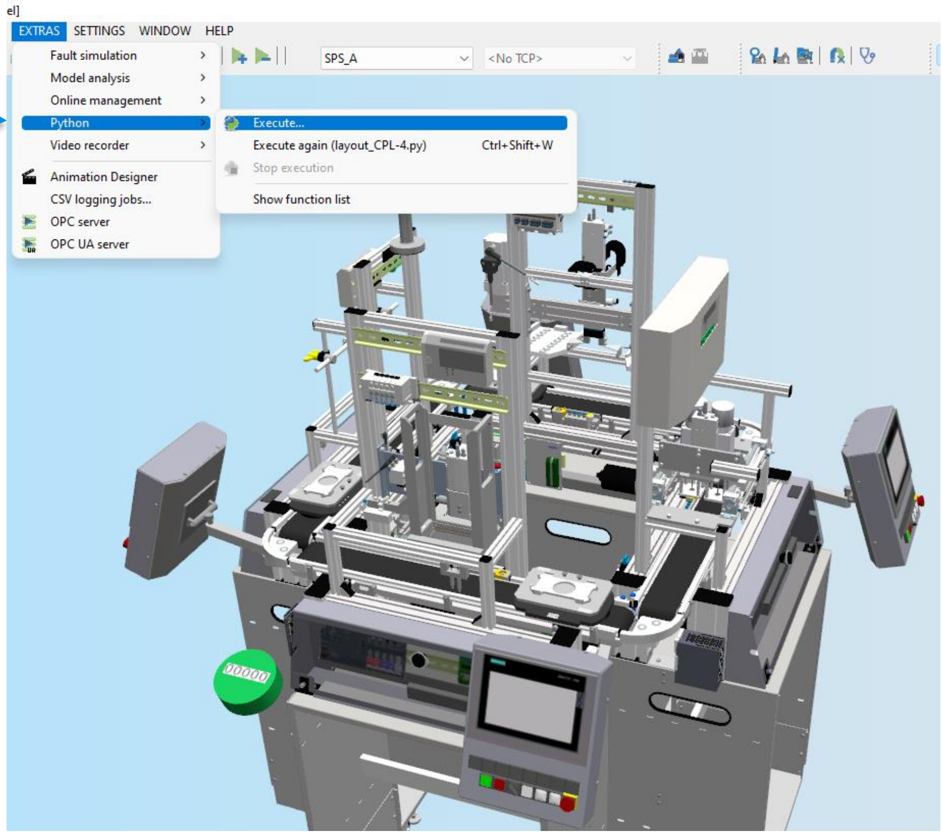
FESTO

Virtual Commissioning with FactoryViews[1]

Guide to import model from python script

1. Create a CIROS model.
2. Select Extras → Python → Execute.
3. Select the python script generated, for example, from Factory View, and click open.

   Note: Last python script can be executed again by selecting Extras → Python → Execute Again or shortcut key Ctrl+Shift+W.
4. Go to top view or shortcut key A.
5. Open Model Library or shortcut key Ctrl+Shift+M.
6. Insert carriers and delete extra carriers.
7. Snap the remaining carriers in place.
8. The model is ready.

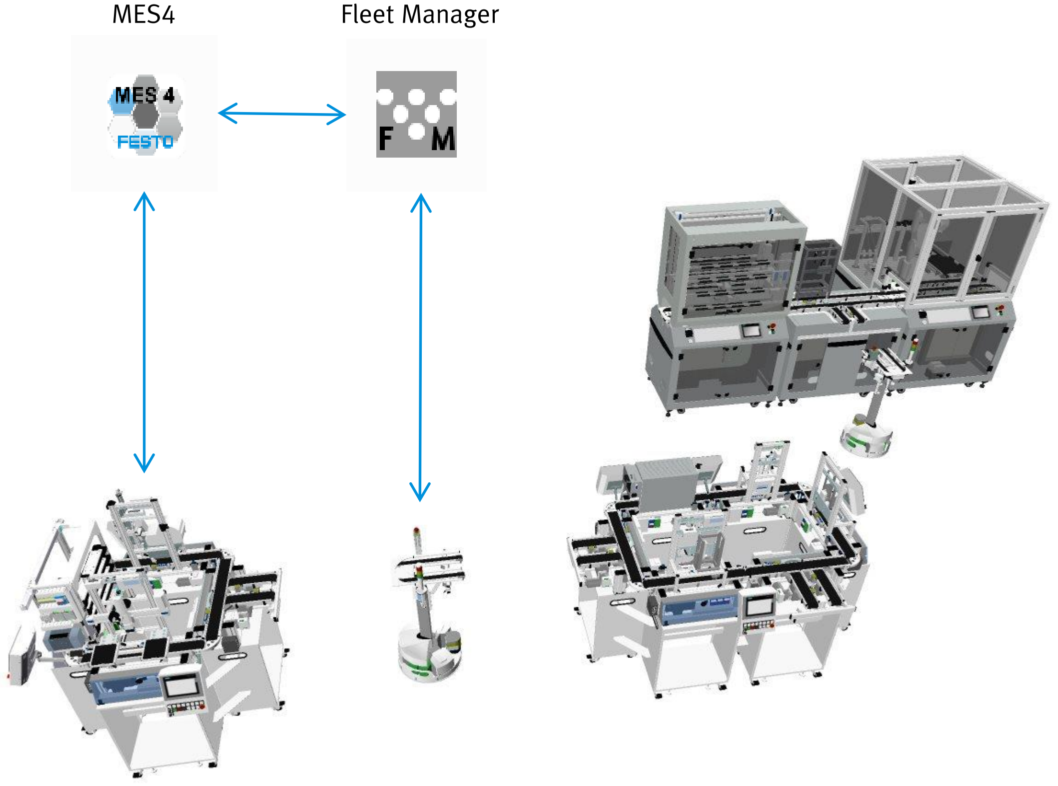


Hint:
- In the python script exported from FactoryViews, the modules are snapped in place and MES ID of the resources are set.
- To reduce graphic power consumption and avoid crash, close all windows and minimize Model Window before loading the script.

[1] FactoryViews is the new software bundle for MES and web based services. It contains MES4 v3.

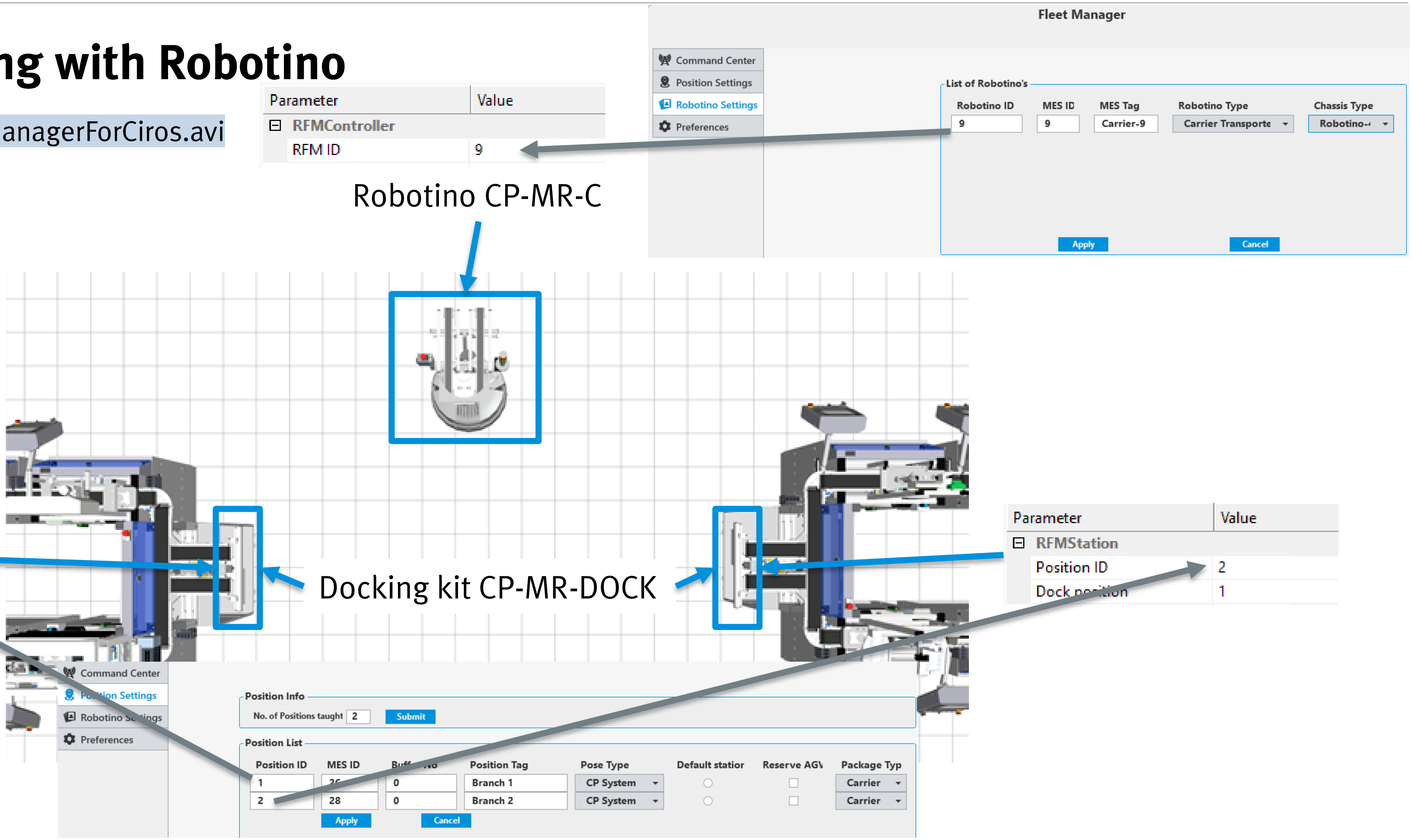
Virtual Commissioning with Robotino

- **Note:** MES4 is not communicating with Robotino(s) directly!
- Communication is carried out via the Fleet Manager.
- MES4 is just sending transportation orders like “Go to position A, grab a workpiece, move to position B, and release the workpiece over there” to the Fleet Manager
- Fleet Manager itself selects one of the available Robotino(s) and sends commands like “dock to position A” to the Robotino to fulfill the MES4 order.



Virtual Commissioning with Robotino

Videos tutorial: 21_ConfigureFleetManagerForCiros.avi

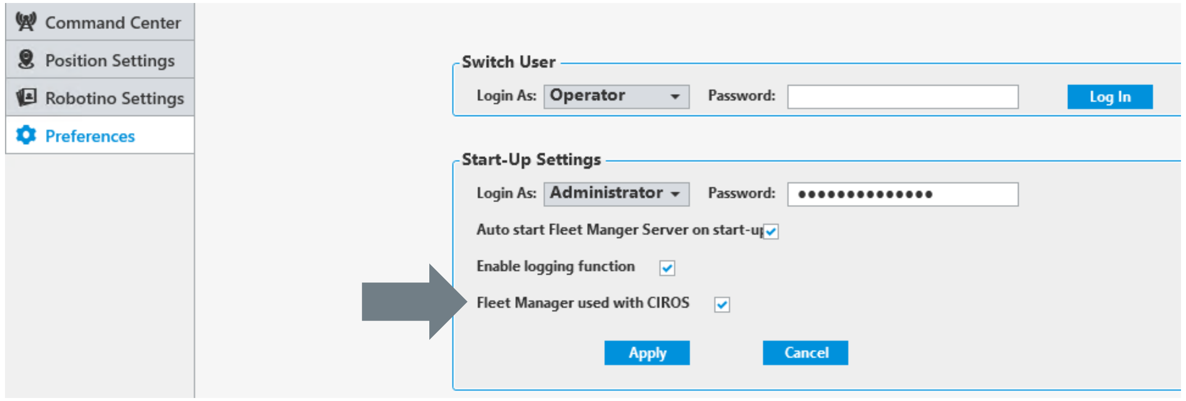


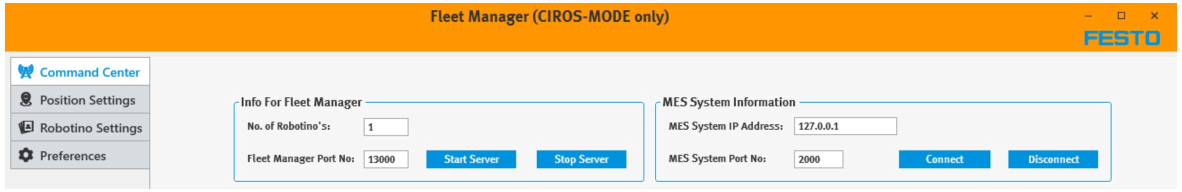
Parameter	Value
RFMStation	
Position ID	1
Dock position	1

Virtual Commissioning with Robotino

Fleet Manager v3 and above

- Option Fleet Manager used with CIROS must be activated.





Virtual Commissioning with Robotino

Simulation with CIROS, MES4 and Fleet Manager

- Start simulation
  1. Start MES4, CIROS, and Fleet Manager in any order, but **do not** start the CIROS simulation yet.
  2. In Fleet Manager, if the server is not started, start the Robotino server via the Start Server button.
  3. Start the CIROS simulation.
  4. Fleet manager: Select all available Robotinos and switch them to Automatic.
  5. Place your MES4 orders.

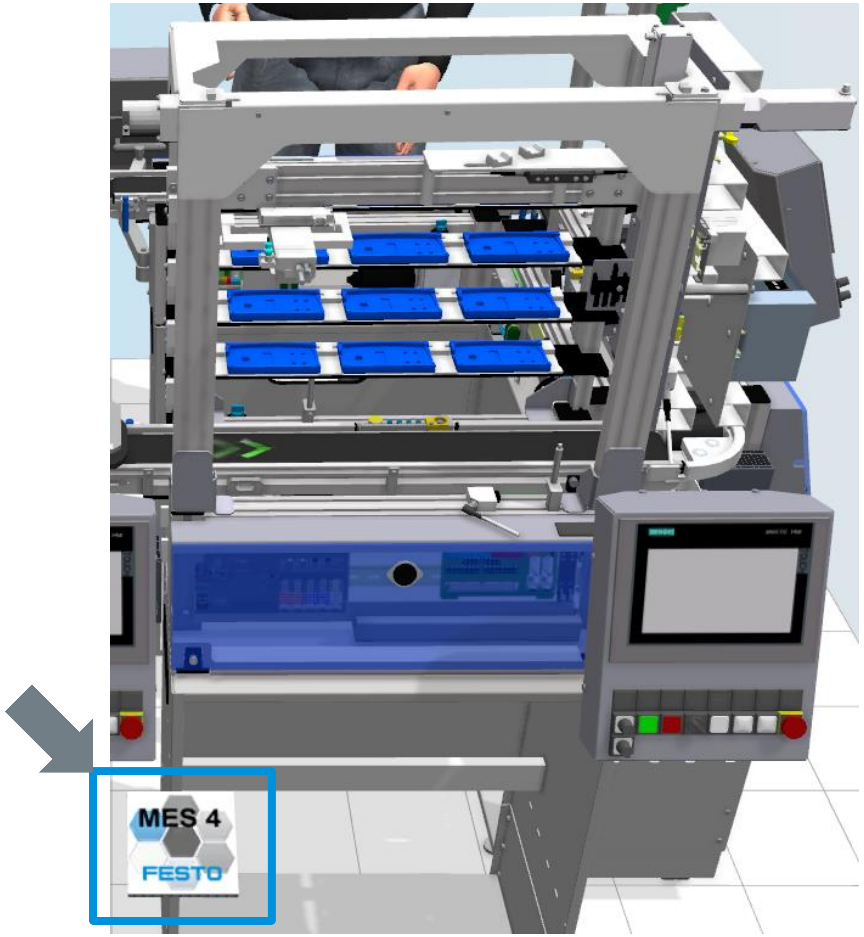
- Stop simulation
  1. Stopping simulation in CIROS.
  2. Reset the CIROS simulation to t=0s.
  3. Fleet manager: Stop the Robotino server via Stop Server.

     Note: Server must be stopped!
  4. MES4: Clear all Robotino-related buffers.

Videos tutorial:
01_StartUpRobotino.mp4
22_SimulationOfRobotino.avi

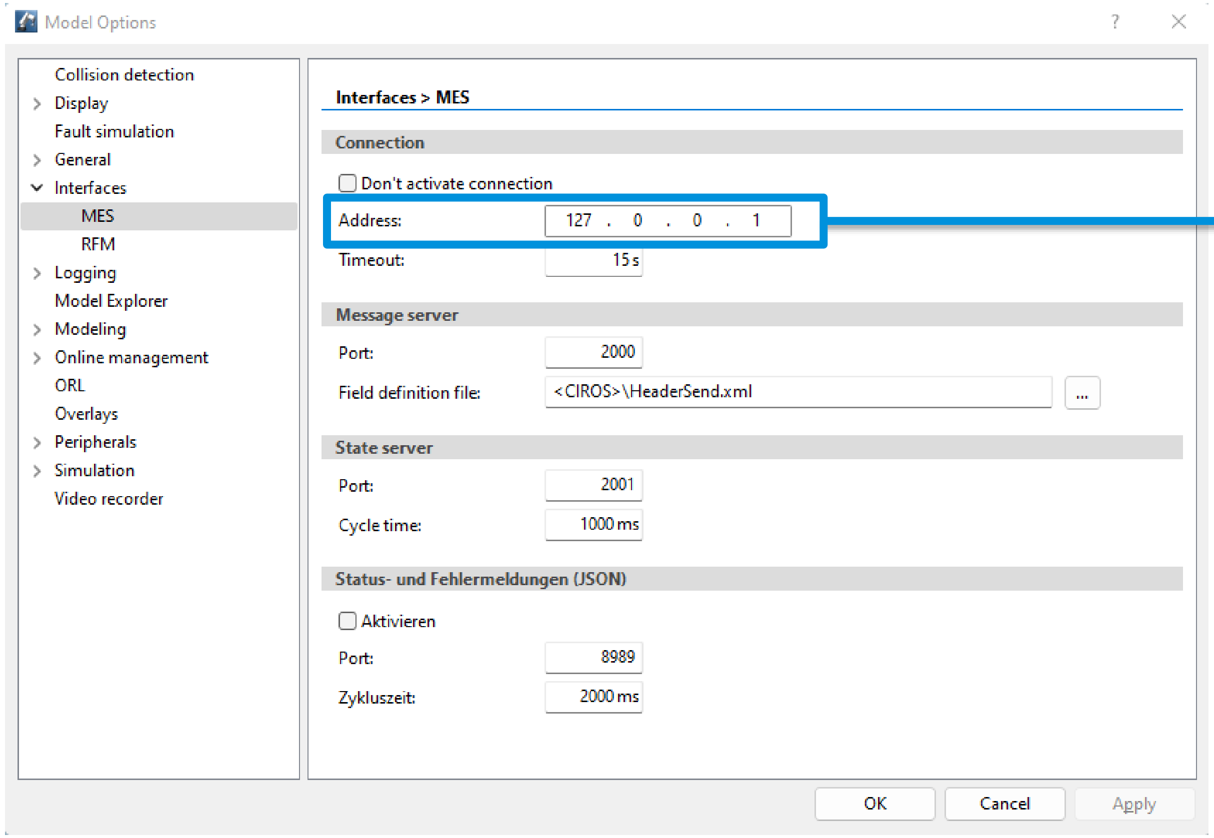
Synchronise CIROS Parts in Storage with MES4 Buffers

- During the initialization phase of a CIROS simulation run, CIROS will ask for the buffer contents of high-bay warehouses automatically .
- But, whenever the buffer of a high-bay warehouse has changed within the MES4 while the CIROS simulation is running, one must click the MES 4 button in CIROS view window to transmit these changes to CIROS.



Running CIROS and MES4 on Different PCs

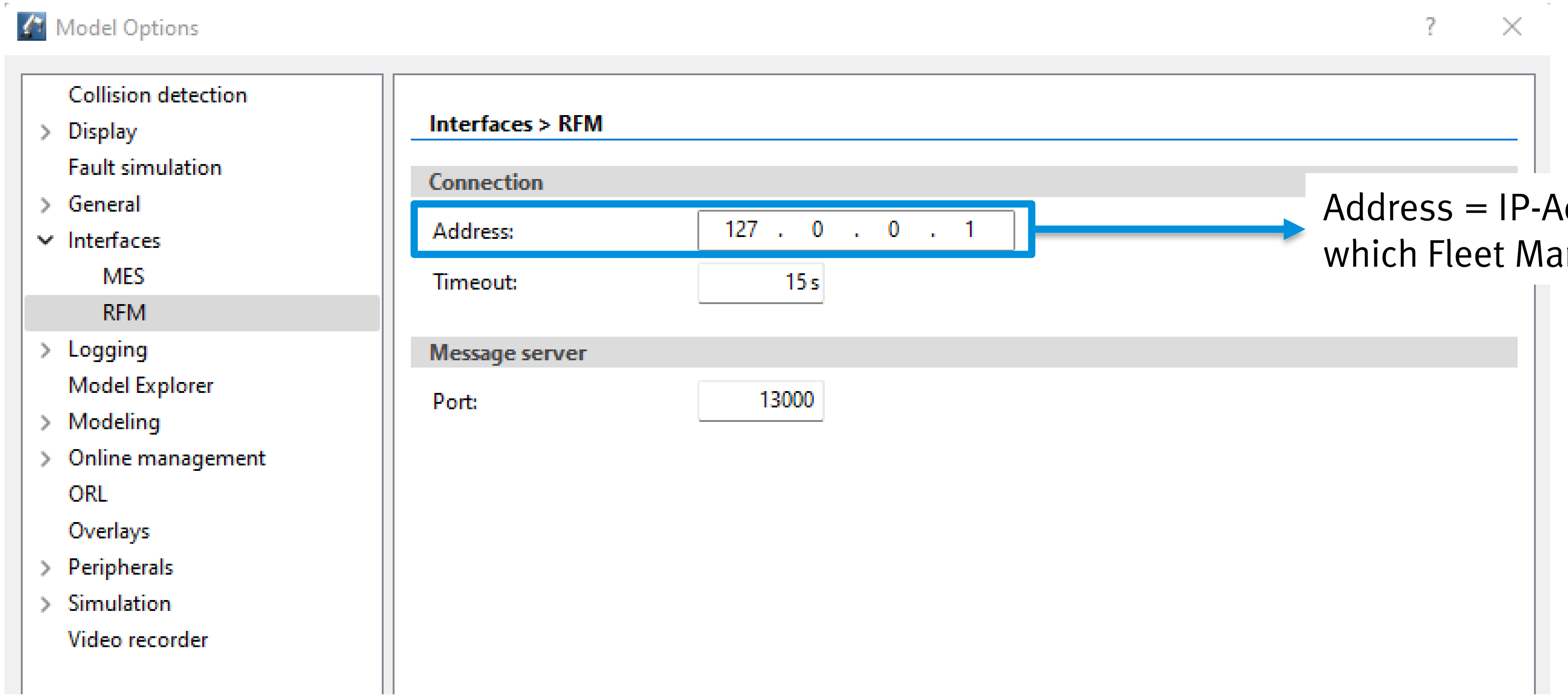
CIROS → Settings → Model Options



Address = IP-Address of PC which MES4 is running.

Running CIROS and Fleet Manager on Different PCs

CIROS → Settings → Model Options

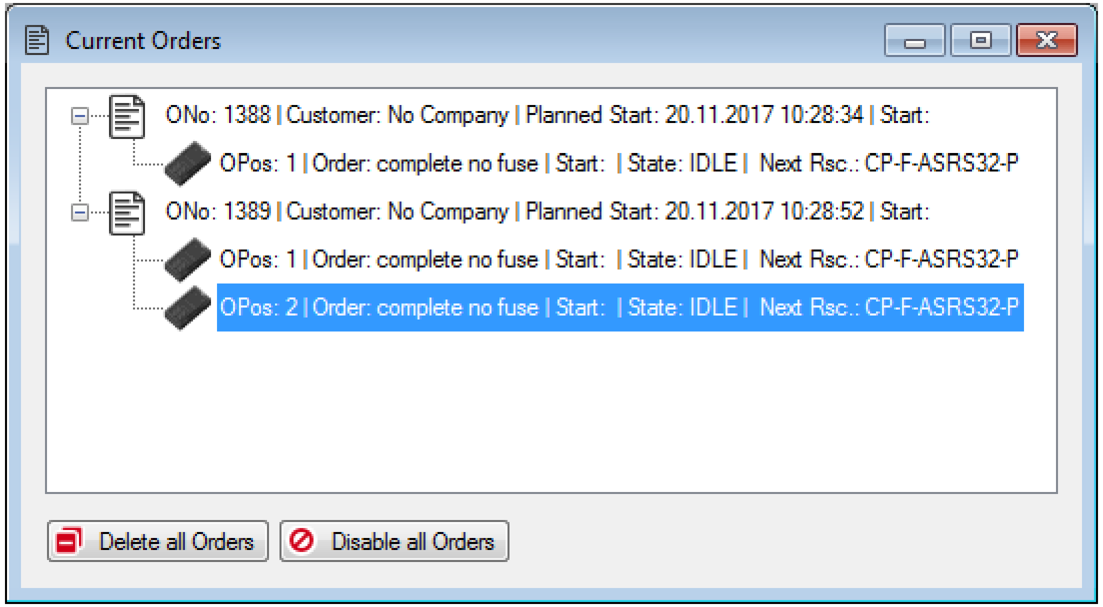


Address = IP-Address of PC which Fleet Manager is running.

Terms & Definitions in MES4

Customers, orders, order number & order position

- MES4 maintains a list of registered customers which are allowed to place customer orders.
- Each order has a unique order number (ONo) and may consist of a couple of different products and parts to be produced.
- Each production part within an order has its own order position (OPos), ranging from 1 up to the total number of parts of a particular order.
- (ONo, OPos) is a unique representation of an individual part.

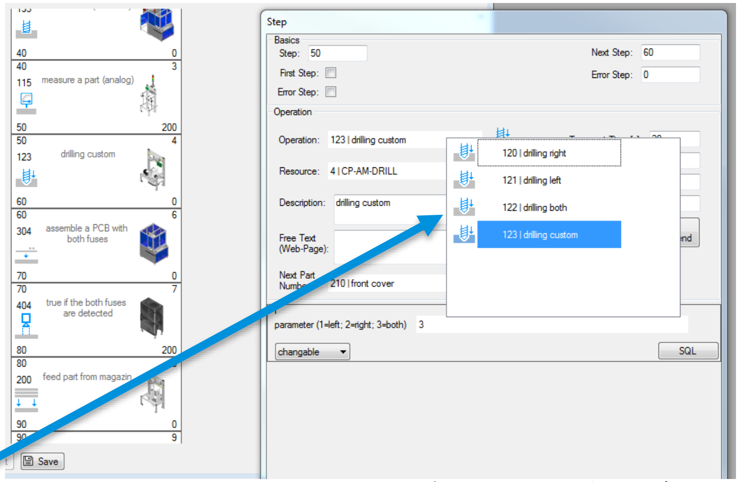


Picture source: MES4 v2 or lower

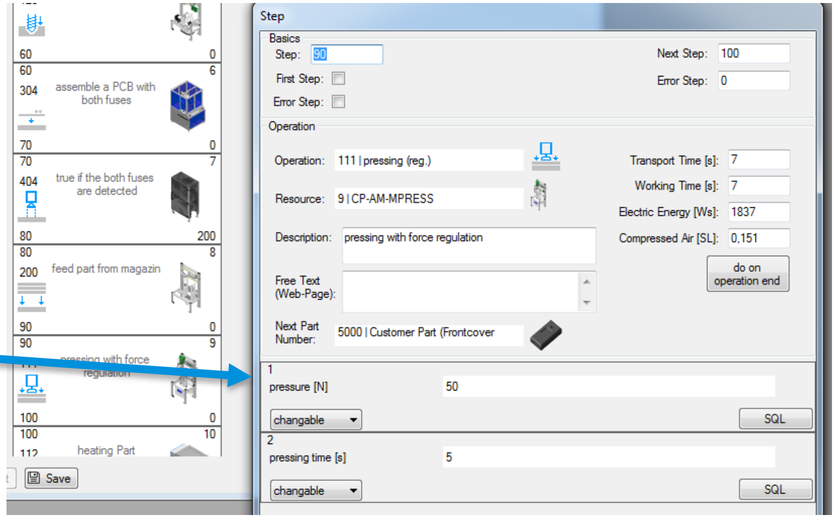
Terms & Definitions in MES4

Operations and parameters

- Operations define the functionality of a production step and are executed by resources.
- But, MES4 separates between operations and resources, since there might be several resources able to perform the same type of operation, i.e., operations are not sub-objects of resources but allocated to them.
- Vice versa, some resources can execute more than just one operation.
- Each operation has its own unique ID (OpNo) and might have no, one or even quite several parameters to adjust the production step.



Picture source: MES4 v2 or lower

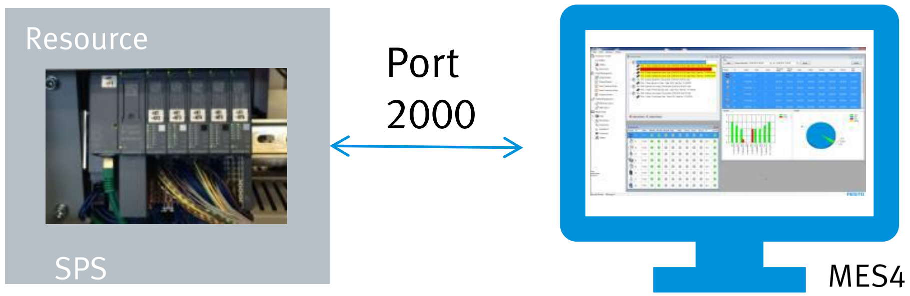


Picture source: MES4 v2 or lower

MES4 Communication Interface

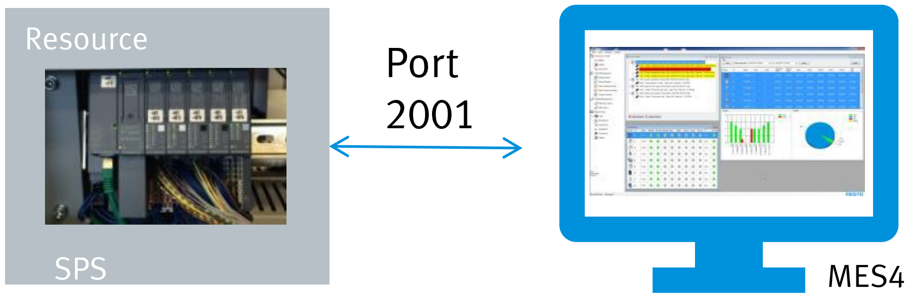
Cyclic status message

- The resources send a status update to MES4 in every second.



Service requests

- Resources or other applications query data from MES4 or write data to MES4.



FESTO

Terminology in MES4 Messages

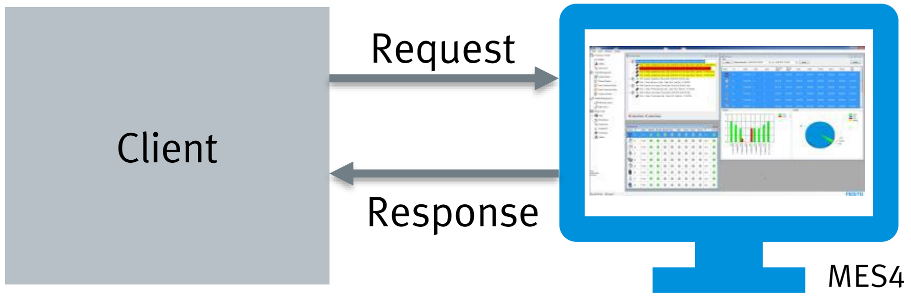
Term	Meaning
BoxID	Box ID
BoxPNo	Box part number
BoxPos	Position in box
BufNo	Buffer number
BufPos	Buffer position
ONo	Order number
OPos	Order position
Op	Operation
OpEnd	Operation end
OpNo	Operation number

FESTO

MES4 Service Requests

- MES4 offers many services that are required for the operation of a plant.
- None of the plants use all services, but all services are available at any time and can be called not only by PLCs but also by other business applications via a TCP/IP connection. In addition, a skilled operator can implement additional services, if they can be mapped to an SQL query of theMES4 database.
- Service calls always follow the request response paradigm, i.e., a client sends a call and MES4 response back.

- Internal controller of CIROS model is the virtual representation of PLC. Thus, it communicates the same way as a real PLC.



MES4 Service Requests

Message classification

- Services are uniquely identified by two characteristics:
  - MClass (service class)
  - MNo (service number)
- The MNo is only unique within an MClass. This means that the service with MClass 100, MNo 6 is different from MClass 150, MNo 6.
- They also have a name, but this is of no relevance to the client or to MES4. It is only used for recognition by humans.
- All the messages available can be seen in MES4
  - Tools → Com. Simulator
- The messages can be edited
  - MES4 → Tools → Config SQL
  - **Note:** Only Administrator can access Config SQL
  - Administrator’s password is SolutionCenter

Example:

MClass	MNo	Name
100	6	GetOpForONoOPos
100	33	GetStepDescription
101	20	OpEnd
150	5	GetBufPos

FESTO

MES4 Service Requests

Classes

- Service classes on the right are frequently used. For user defined services, any other classes can be used.

MClass	Description
100	Get information from orders and work plans
101	Write information to orders and work plans
110	Request topology related data
150	Request buffers and utilities (incl. boxes) status
151	Write buffers and utilities (incl. boxes) status
200	Request logistic and Robotino information
201	Write logistic and Robotino information

FESTO

MES4 Service Requests

Message packet overview

Request

Header
- Defined in HeaderGet.xml

Standard input parameters
- Defined in HeaderGet.xml

Service specific parameters
- Defined in MES4 DB

Response

Header
- Defined in HeaderSend.xml

Standard output parameters
- Defined in HeaderSend.xml

Service specific parameters
- Defined in MES4 DB

MES4 Service Requests

Data coding

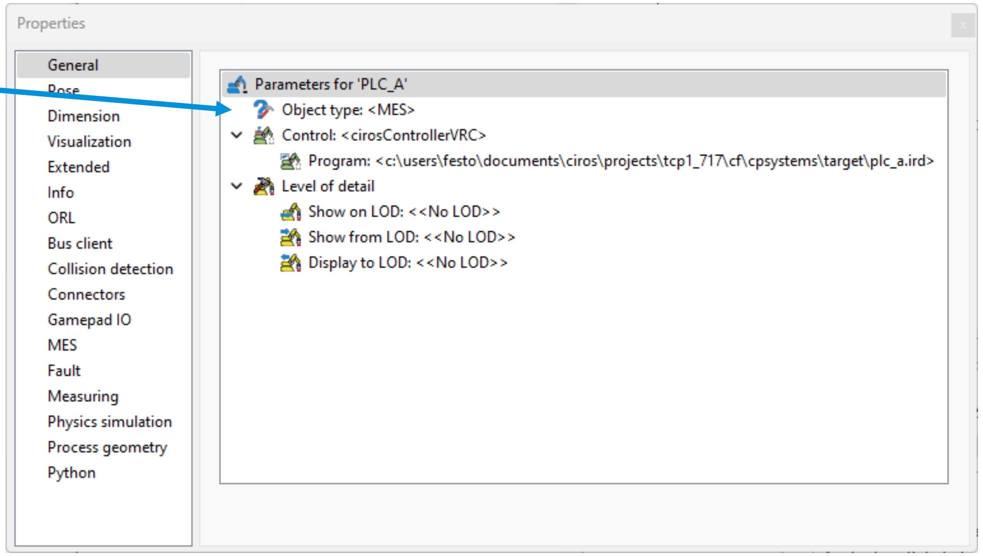
- MES4 allows two different encoding procedures for service requests and responses.
- The first three or four bytes of each packet are TCP Ident header, which indicates which method the packet uses.

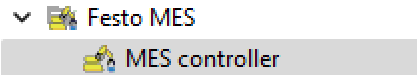
- **Binary coding**
  - Binary coding is primarily used for communication with PLCs on which it is fundamentally easier to handle fixed-address binary data than strings.
  - In the binary procedure, a distinction is still made between the Siemens format and the CODESYS format. MES4 also responds to each binary-coded request in the binary procedure of the same format.

- **String coding**
  - Well suited for implementation in high-level languages or for manual tests. Parameter names and values are transmitted in human-readable form.
  - The string procedure also has two forms.
    - The complete format can be used for both calls and responses.
    - The abbreviated format is only used in MES4 responses if this was requested in the call. was requested.

Festo MES4 Interface

- The communication between the MES4 and the CIROS model requires an object of type MES in the CIROS model.
- A corresponding object named MES Controller can be found in the CIROS model library Festo MES.
- The MES object from the model library already has all required communication inputs and outputs. Those inputs and outputs shall are only be used for external communication and should not be connected to other inputs or outputs in CIROS.





Festo MES4 Interface

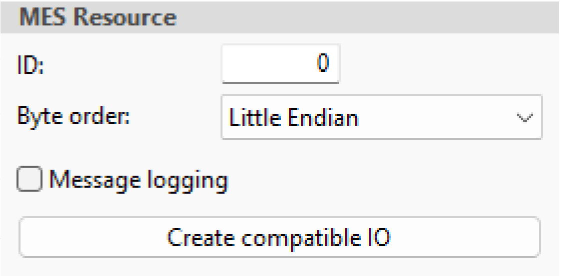
- The communication with the MES object inside CIROS is exclusively done via IRL functions. These functions are provided by a program named MES_commands.irl which is automatically added to a Programs subfolder of your model folder when you add the MES object. The required functions can be made accessible in your own IRL program by stating FROM MES_commands IMPORT ALL;
- By default, the MES4 interface uses field definitions from the HeaderSend.xml file in the CIROS root directory. If you need own field definitions, these can be configured in the model options. After configuring field definitions, go to the MES property page of the MES object and press the Create compatible IO button to create the required object inputs and outputs.

individual ID of the MES object

big / little endian

log the communication messages to the messages window

creates the required inputs and outputs based on the configured field definition file



Use Case: Update Resource Status with MES Controller

- At the end of this tutorial, user is able to update to MES4 that the resource is in Automatic, MES mode when simulation starts.

1. Create a CIROS project.
2. In Model Libraries, insert following:
   1. Festo MES \ MES controller.
   2. Controllers \ Simulation controller
3. Open Project Management window.
   1. Create a new project in Industrial Robot Language (IRL) in location ‹project folder›\Programs. Name it ResourceState.prjx.
   2. Assign the project to controller MES_Controller.
   3. Add MES_commands.irl in the project.
   4. Create a new program. Name it ResourceState.irl. Make it main program.
   5. Program ResourceState.irl as in next page.
   6. Save and compile the program.
   7. Save the project.
   8. Run simulation and observe in MES4 Production Control \ Resources.

FESTO

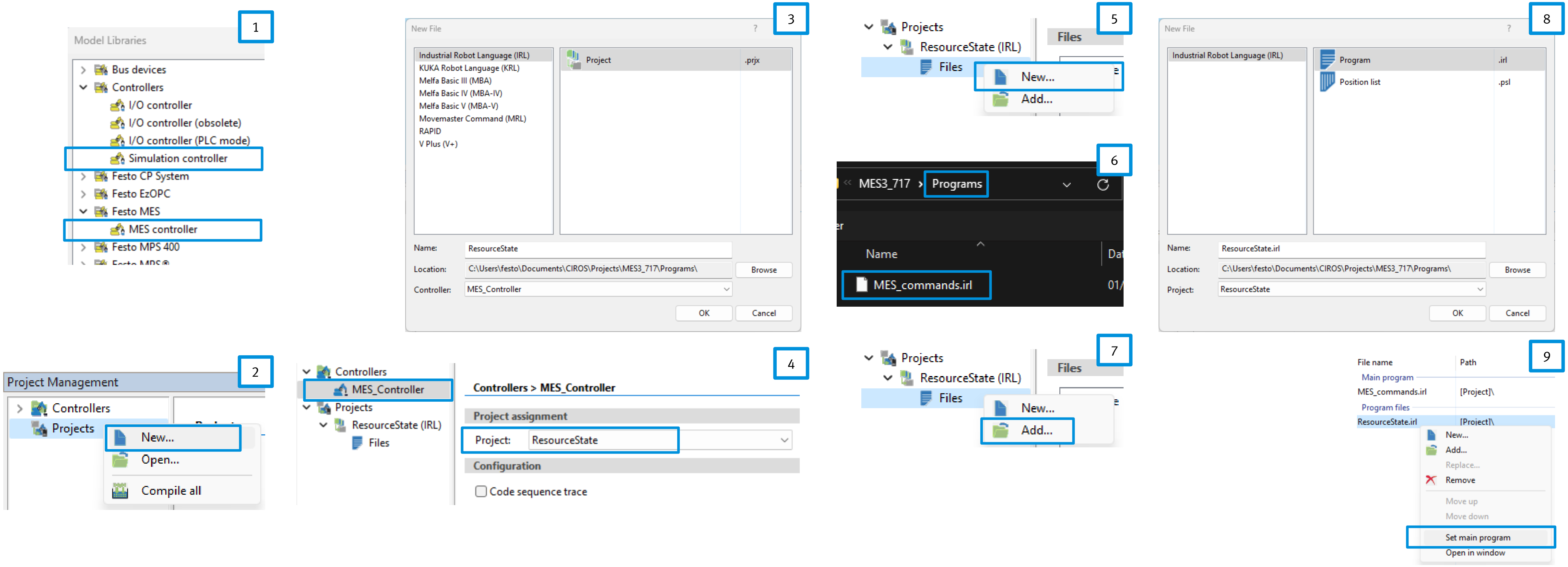
Use Case: Update Resource Status with MES Controller

ResourceState.irl

```
PROGRAM ResourceState;

FROM MES_commands IMPORT ALL;

BEGIN
        {Initiate MES status}
        setStateMESMode(true);
        setStateAuto(true);
        setStateReset(false);
        setStateError(-1);

ENDPROGRAM
```

Use Case: Update Resource Status with MES Controller

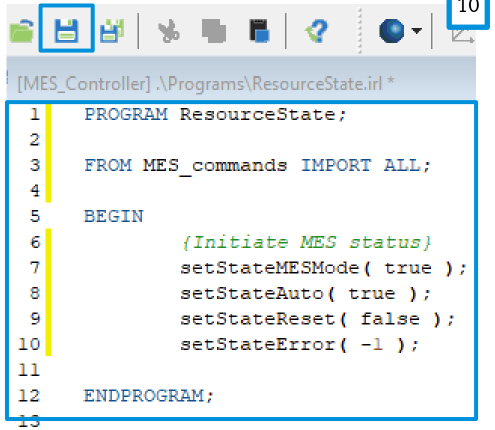
Steps in screenshots



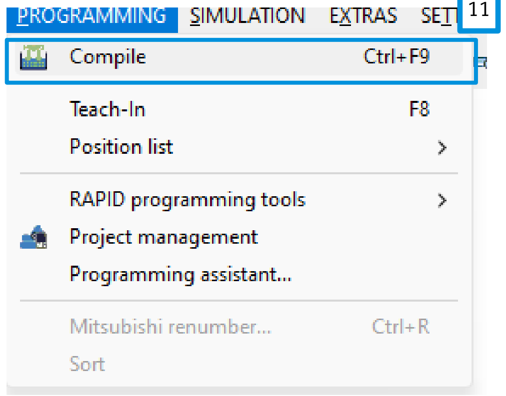
Use Case: Update Resource Status with MES Controller

Steps in screenshots

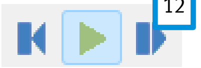
10



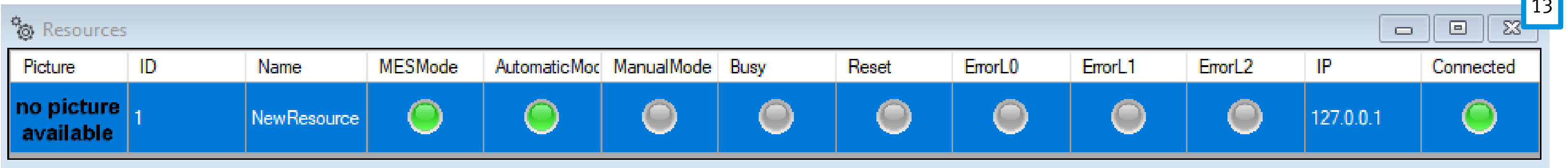
11



12



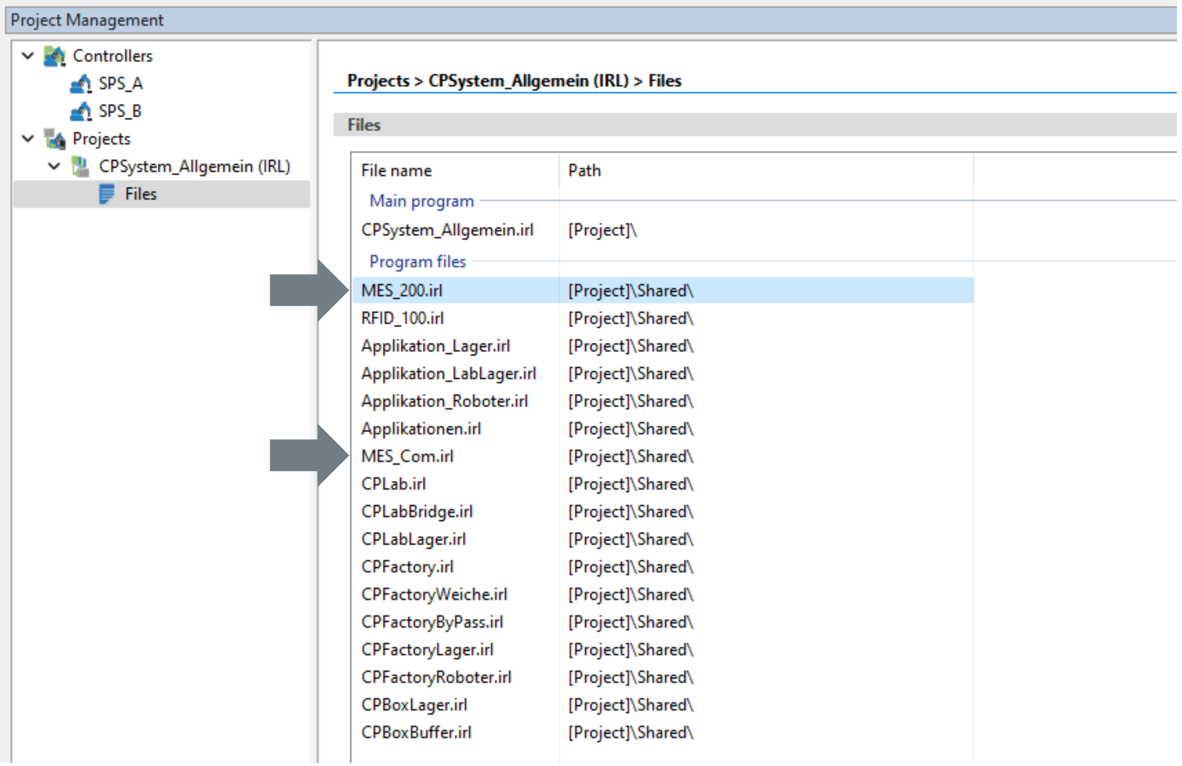
13



Message Request from CP System to MES4

- CIROS behaves the same as PLC in terms of MES4 communication. Thus a bit coding is used.
- The messages are written in IRL format and can be found in Project Management.

1. In toolbar, open Programming \ Project Management.
2. In Project Management window, if not yet exist, add following project.
   - ‹project folder\CF\CPSystems\CPSystem_Allgemein.prjx›
3. Open Projects \ CPSystem_Allgemein(IRL) \ Files.



MES Communication Flow Chart

Example: CP-L-CONVEYOR

SPS_A main program
- CPSystem_Allgemein.irl

Lab conveyor ID
- CIROS_Serie_ID at AI97 = 1
- CIROS_GM_ID at AI98 = 0
- CIROS_App_ID at AI99 = 0

CPLab.irl
- Function CPLab
  - Wait for E-stop

Function CPLab_Linearmodul
- Start the belt
- Wait RFIDInventory = 1

MES_Com.irl
- Function checkForWorkInThisStation
  - If Resource ID = 0, standard mode
  - If Resource ID › =, MES mode

Start communications
- GetOpForONoOPos

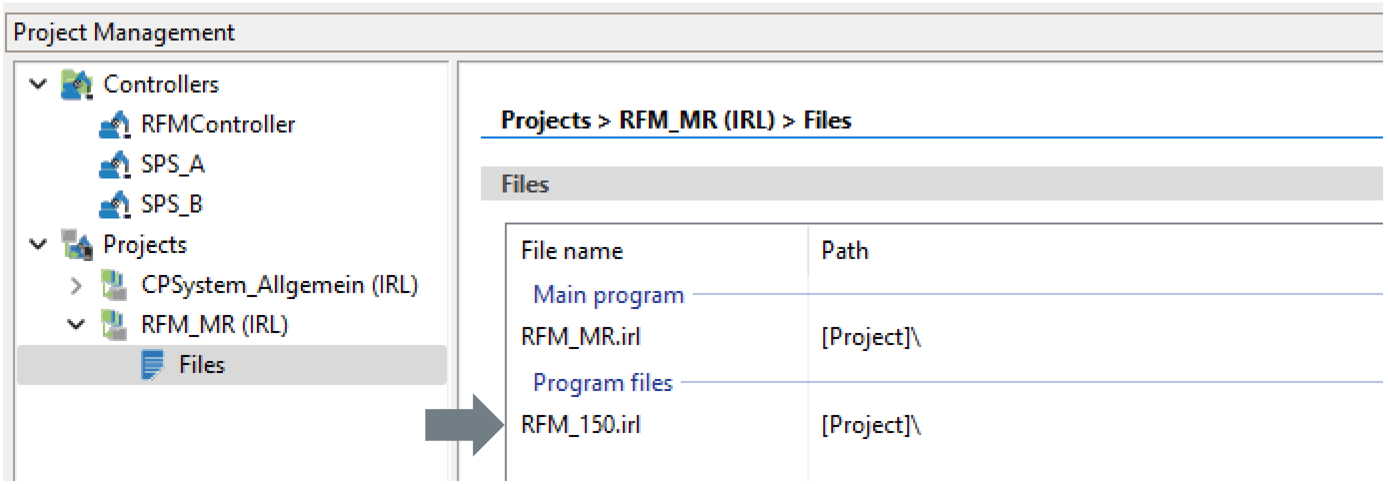
MES_Com.irl
- Function callApp
  - OpStart

Applications.irl
- Function App
  - start the corresponding app

OpEnd
- Write info of next operation

Message Request from CIROS to Fleet Manager

- Like PLC, CIROS Robotino model is the virtual representation of real Robotino. Thus it communicates with Fleet Manager the same way as a real Robotino.
- The communication messages are written in IRL format and can be found in CIROS → Programming → Project Management → Projects →RFM_MR.



- **Note:** RFM = Robot Fleet Manager, MR = Mobile Robot

Virtual Commissioning with Soft PLC

Basic knowledge in PLC programming and TIA Portal is required.

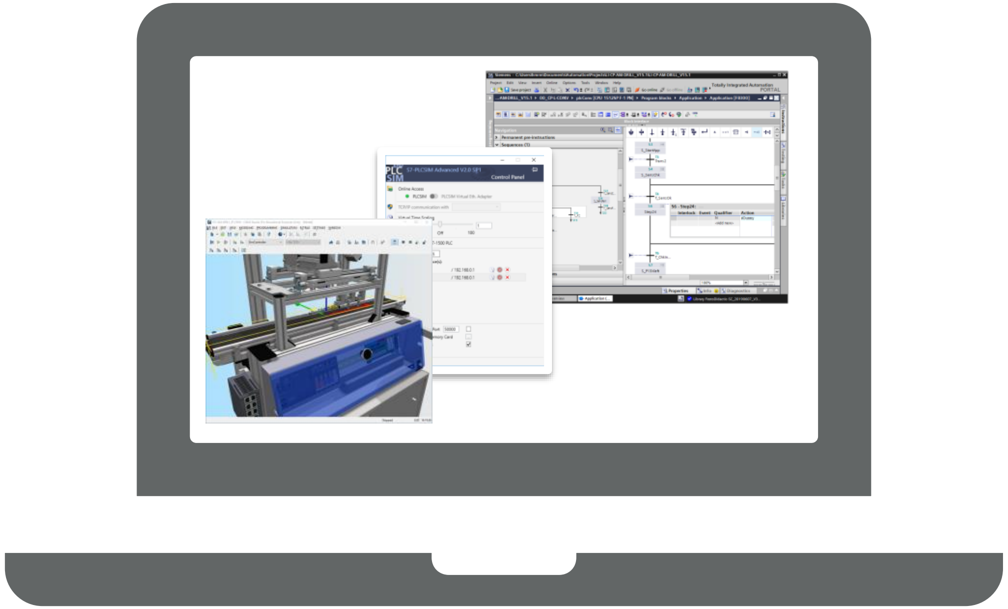
FESTO

Scenario Overview

- Program your PLC against a virtual mechatronic model
- No risk to your hardware if students make mistakes in program code
- Program modules that you don‘t physically own or let dozens of students program the same module even if you only own it once

Scenario Overview

All on single PC



Software on different PCs

Process Summary

1. Prepare a CIROS model with the hardware you want to program
2. Create your hardware configuration and I/O tags in TIA Portal
3. Create a PLCSIM Advanced instance and download the hardware configuration
4. Configure the interface between CIROS and your instance
5. Start programming!

Important:

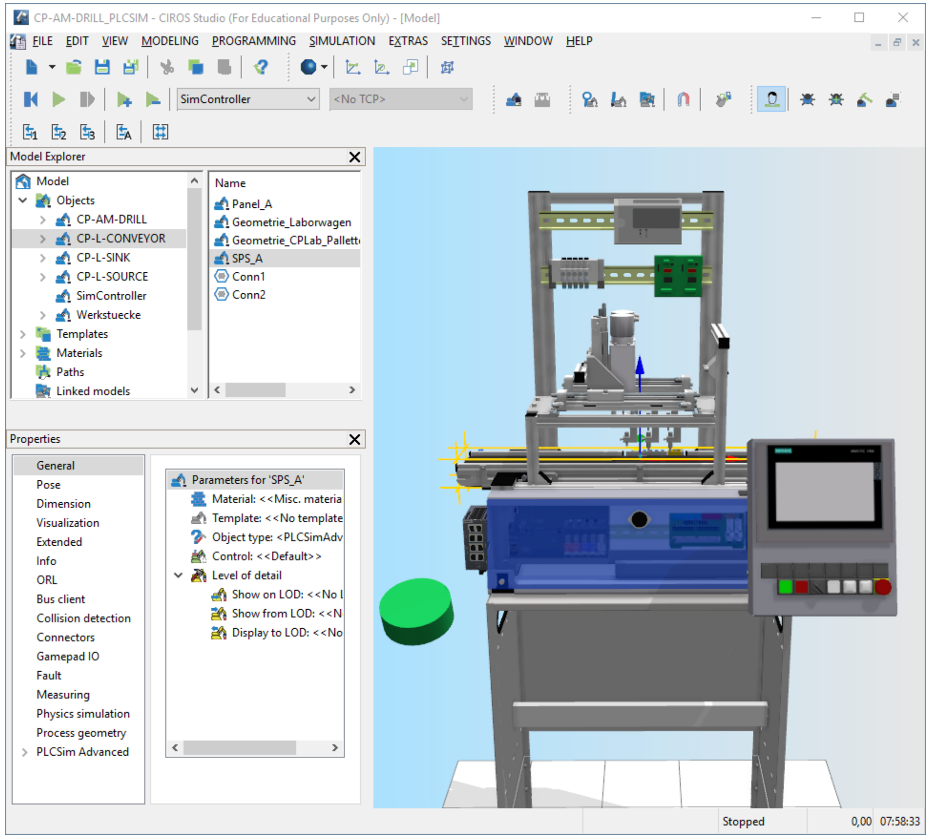
- CIROS v7.1 is only compatible with Siemens **PLCSIM Advanced v3.0** or above!
- CIROS v7.0 or below is only compatible with Siemens PLCSIM Advanced v2.0 or below.

Preparing a CIROS Model

- Two approaches are possible:

1. Create a model from scratch
   - Maximum flexibility
   - Program any CP station you like

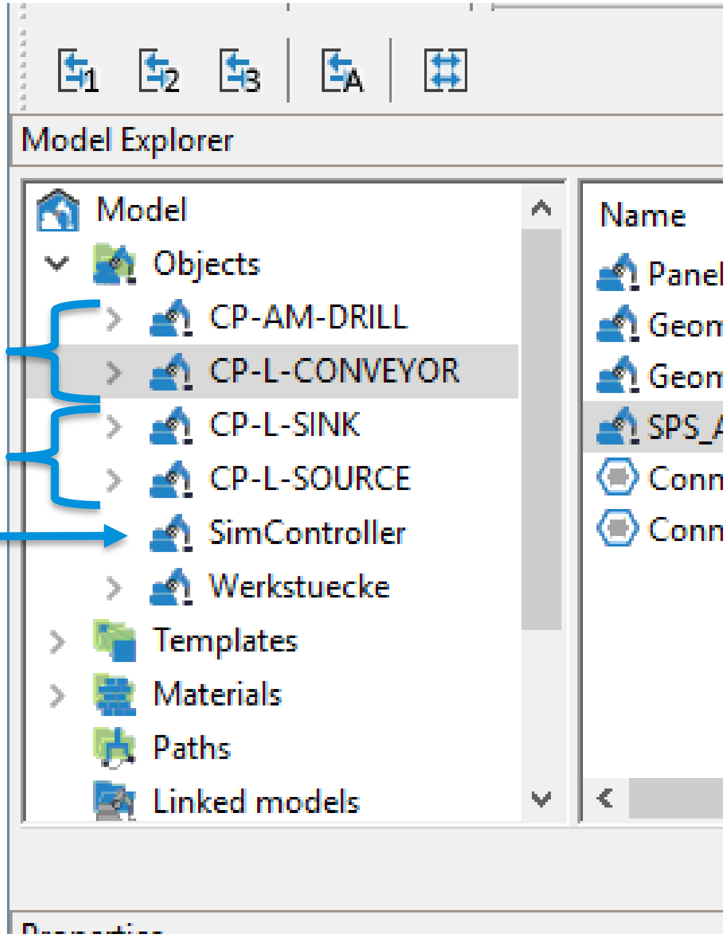
2. Load a premade model from the model library
   - Get started quickly with minimum effort
   - Limited selection of CP systems available



Preparing a CIROS Model

Your model usually needs three basic elements to serve for virtual commissioning with PLCSIM Advanced:

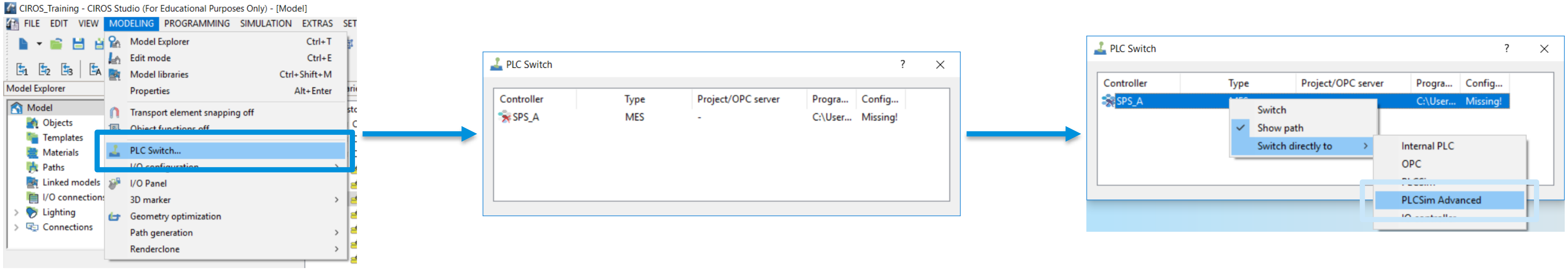
- The mechatronic system you want to program
- A source and sink to generate and remove carriers with parts
- A SimController so CIROS is able to simulate your model



Exercise

Preparing a CIROS model from scratch

1. Create a new empty model.
2. Add your mechatronic system from the model library.
   For this exercise, add a CP-L-CONV.
3. Add a source and sink from the model library that matches your system.
   For this exercise, add a CP Lab source and sink.
4. Connect the source and sink to your CP Lab module.
5. Switch the PLC in your CP Lab module (it's named SPS_A) to PLCSIM Advanced mode.



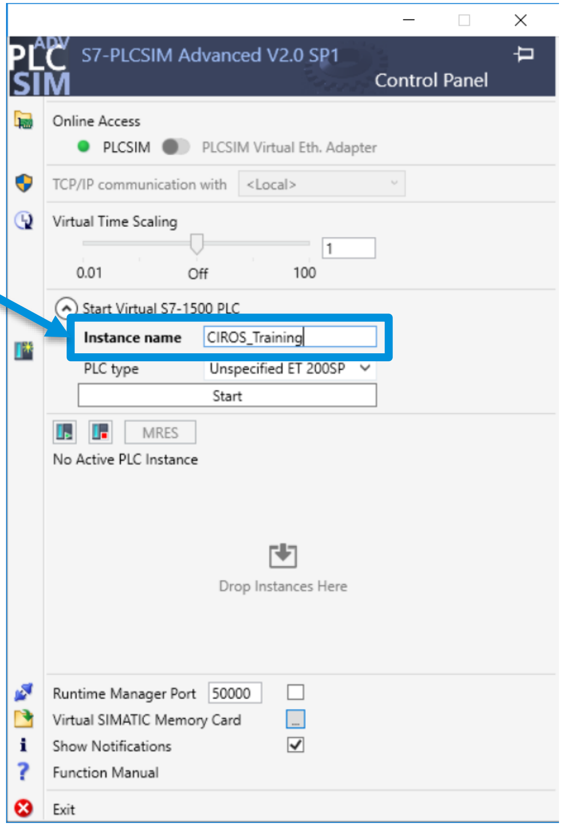
Starting a PLCSIM Instance

Feel free to choose any PLCSIM Advanced settings that work for you.

The only setting relevant to CIROS is the instance name. Choose one you like and remember it. You'll need it later.

Some recommendations:

- For Online Access, choose PLCSIM unless your simulated PLC needs to communicate over the network. This mode makes the connection to TIA Portal effortless
- Leave Time Scaling off. CIROS has its own time scale and will make sure the PLC keeps track if you speed up the simulation beyond real-time
- Choose ET 200SP for PLC type as that matches the physical PLC in most CP hardware systems

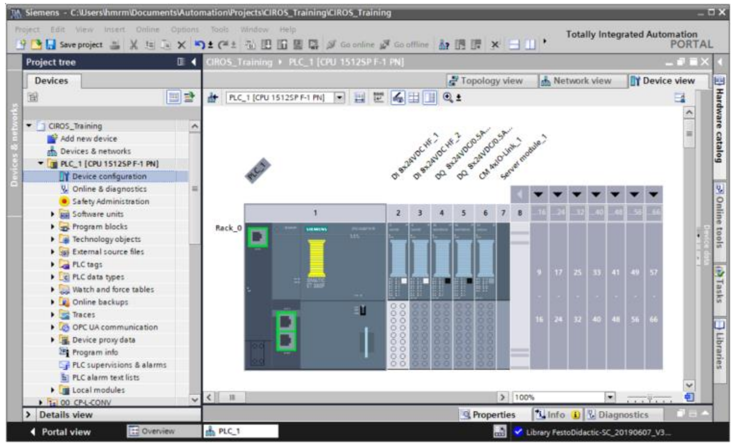


Creating the Hardware Configuration and IO Tags in TIA Portal

Hardware configuration

You can configure your PLC in any way you like.

Ideally, it should have at least the number of digital and analog I/Os that the physical PLC inside your chosen CP system has.

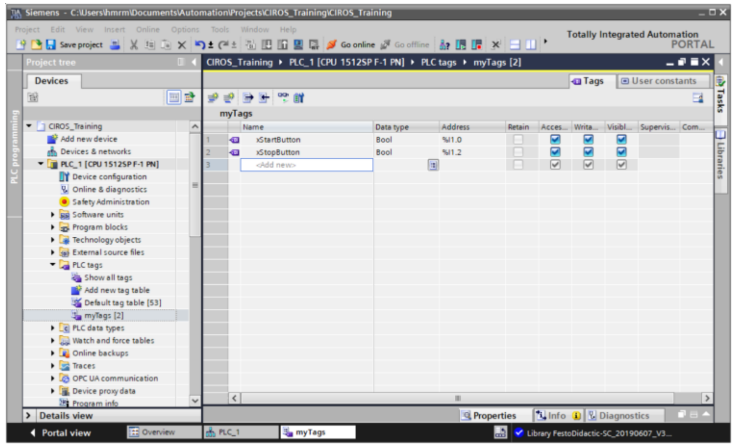


IO tags

You can freely name your inputs and outputs, as long as the address and the type of an input/output is correct.

If you like you can skip the inputs and outputs that are not connected to anything in your CP system.

Refer to the Festo Didactic Infoportal (https://ip.festo-didactic.com) for an I/O listing of your CP system. Alternatively, find the relevant information in your manual or circuit diagram.



FESTO

Exercise

Creating the hardware configuration

Detailed instructions how to do the hardware configuration in TIA Portal are beyond the scope of this document. Refer to the courseware ‘Device configuration’ if you’re having trouble.

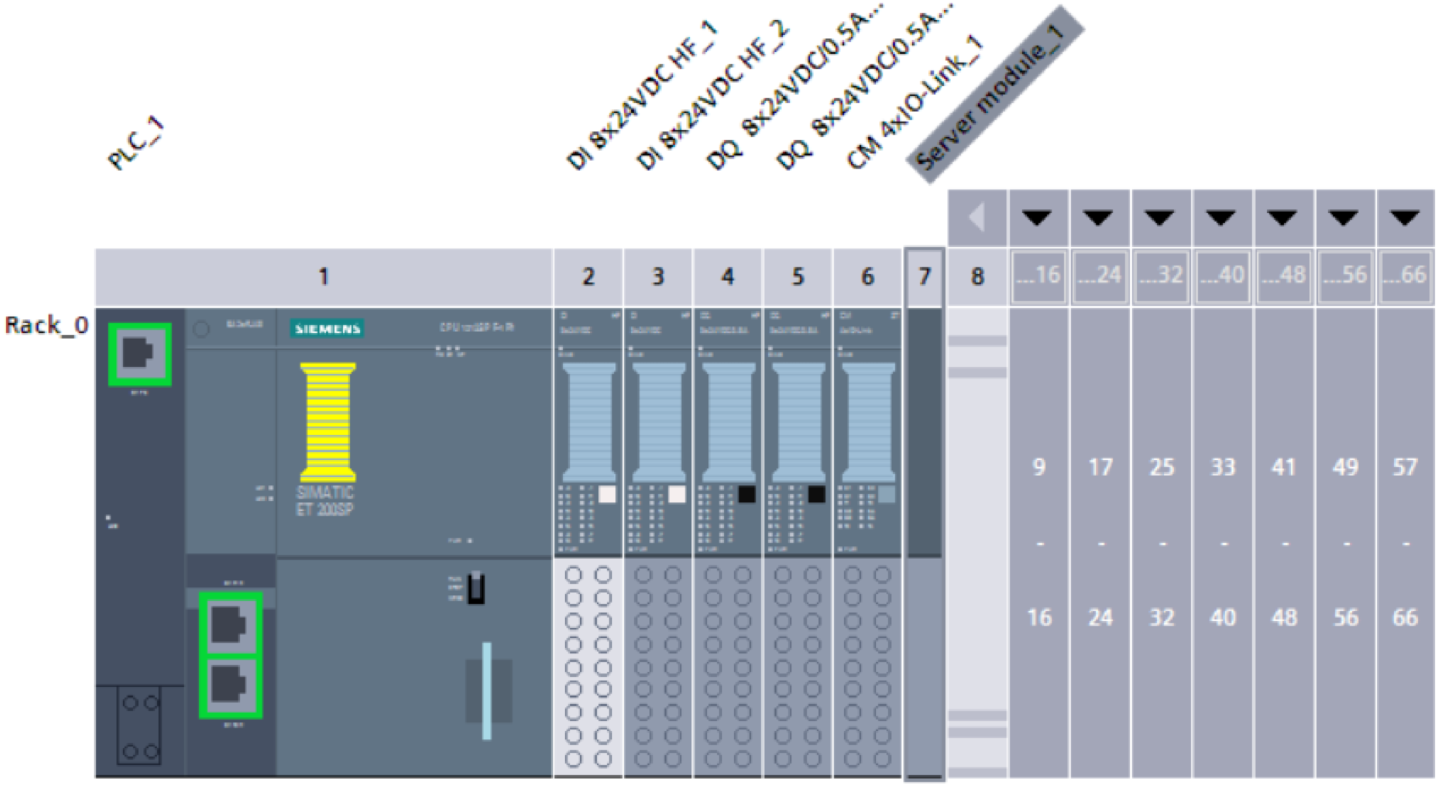
For this exercise, we’re configuring the PLC as we would a real CP-L-CONV module with a Siemens IO-Link 1.1 conformant RFID device.

1. Create a new TIA project
2. Add a S7-1512SP F-1 PN PLC to your project (6ES7 512-1SK01-0AB0)
3. Add two DI 8x24VDC HF (6ES7 131-6BF00-0CA0)
4. Add two DQ 8x24VDC/0.5A HF (6ES7 132-6BF00-0CA0)
5. Add a CM 4xIO-Link (6ES7 137-6BD00-0BA0)
6. Add a server module (6ES7 193-6PA00-0AA0)
7. Set the IO-Link master’s input/output type to 64/64 and shift the starting I/O addresses to address 10

FESTO

Exercise

Creating the hardware configuration



Exercise

Setting up the I/O tags

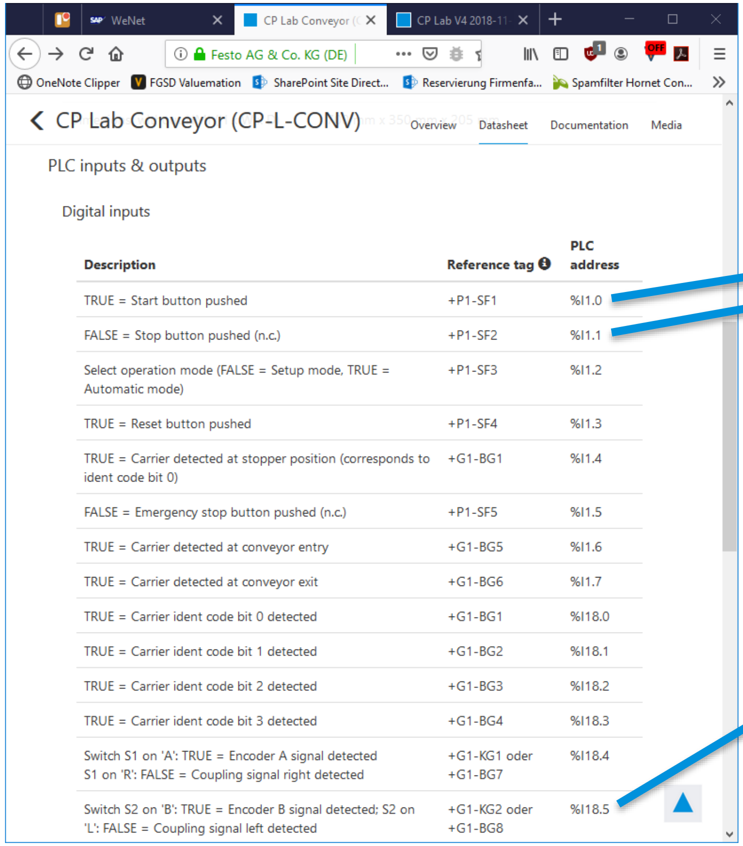
1. Find the list of I/O addresses for the CP-L-CONV at https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/hardware/base/datasheet.php?model=CP-L-CONV&lang=en
2. Create a new tag table
3. Enter all tags listed on the Infoportal into your tag table

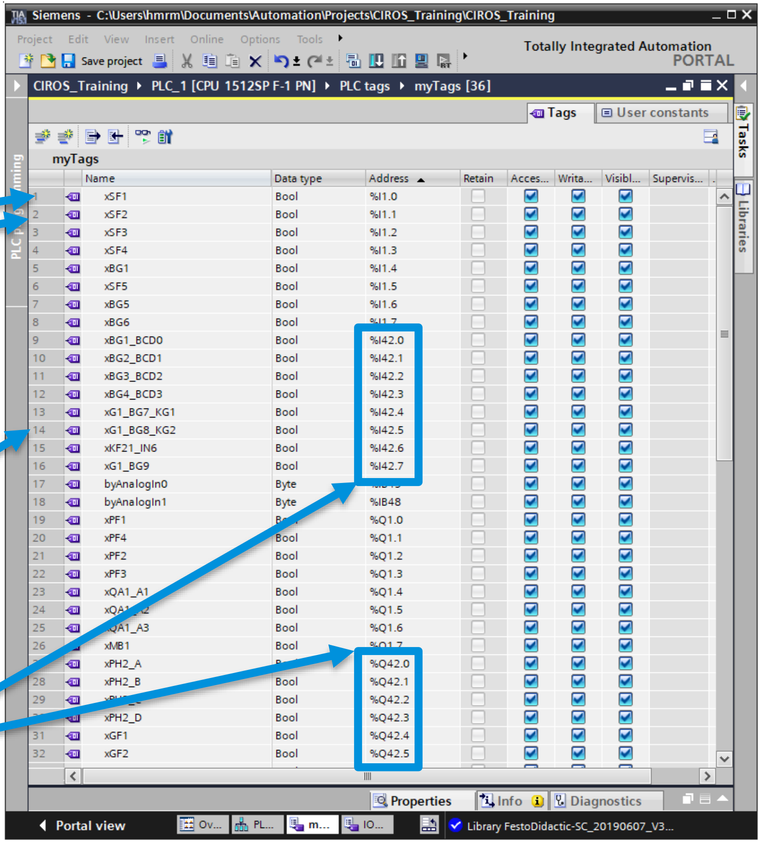
Note:

Depending on the revision of a physical CP-L-CONV, any address listed on the Infoportal in byte 18 might require to be shifted to byte 42. This is only relevant if you plan to download this TIA project to a real CP-L-CONV. In CIROS the absolute I/O addresses don't matter.

Exercise

Setting up the I/O tags





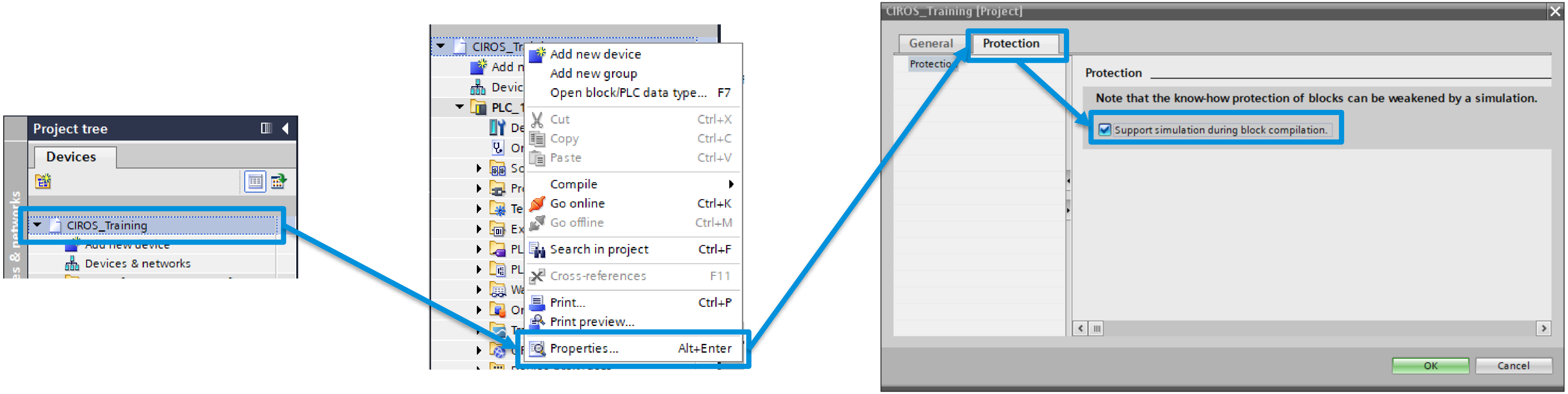
Note the addresses are shifted from 18.x to 42.x.

FESTO

Exercise

Enabling simulation support

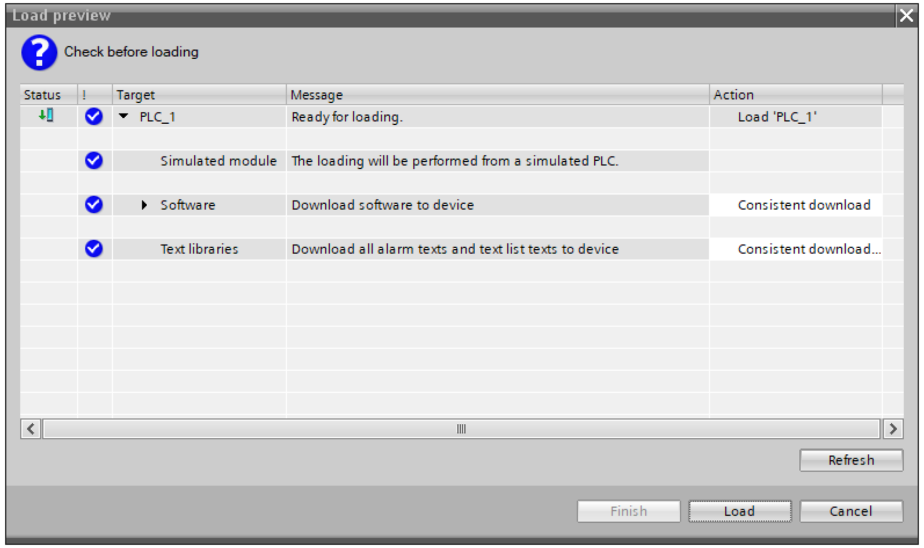
1. Open the project properties.
2. On the Protection tab, check Support simulation during block compilation.
3. Compile and download it again. It should work without a problem now



Exercise

Downloading project to PLC instance

1. Compile the project.
2. Download it to your simulated PLC.
   If Online Access in PLCSIM Advanced is set to PLCSIM mode, this is almost fully automatic.



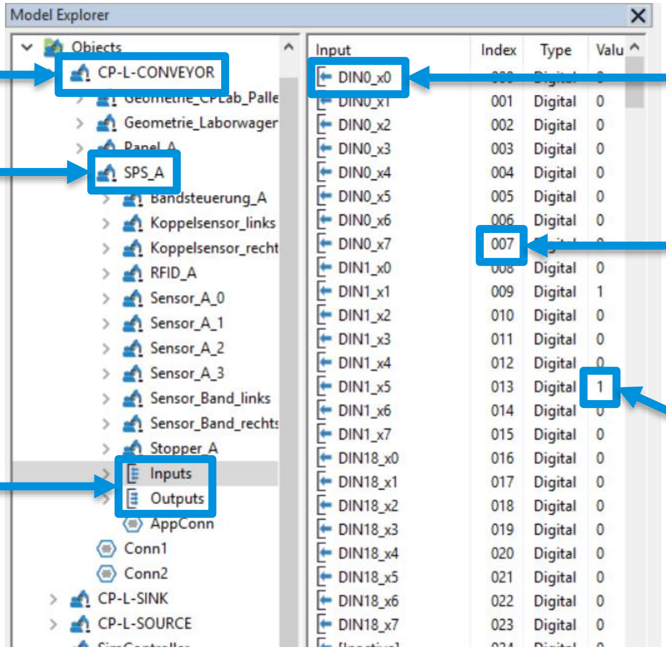
Configuring the Interface

The virtual PLC in CIROS has a large number of inputs and outputs, most of which are unused or used for internal processes inside CIROS. A few virtual I/Os correspond to the I/Os of the PLC inside a real CP system, though. These are named DIN0_x0 to DIN18_x7 and DOUT0_x0 to DOUT18_x7, after the absolute addresses of the real PLC’s I/Os.

CP system you want to program

Virtual PLC inside module

Virtual inputs and outputs



Name of a virtual I/O, corresponding to %I0.0 in the real CP module.

I/O address inside CIROS. Completely irrelevant to your TIA project.

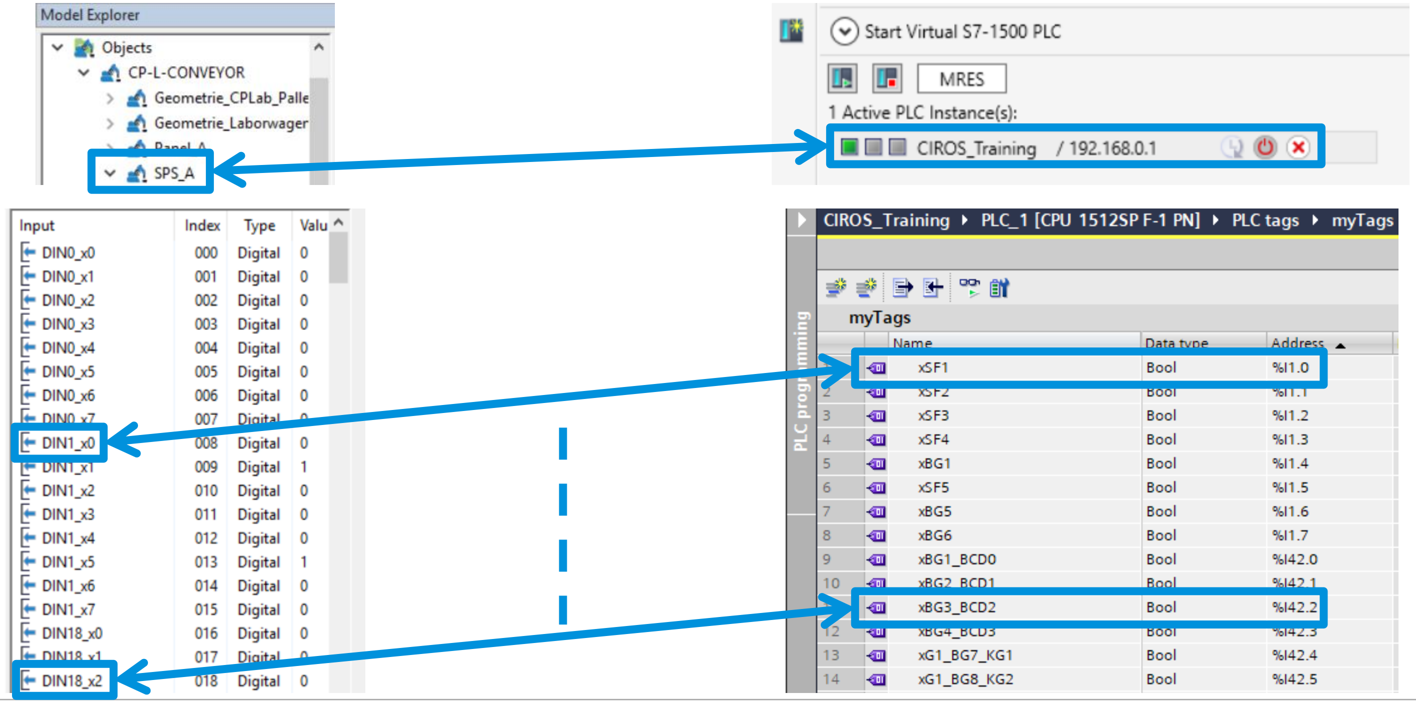
Current value of a virtual I/O. For inputs these come from simulated sensors. For outputs, they control a simulated actuator.

FESTO

Configuring the Interface

You can connect each virtual PLC in your CIROS model to exactly one PLCSIM Advanced instance.

You have to configure our CIROS PLC to connect to the right instance and to hook up the virtual CIROS I/Os to the correct TIA I/Os.



FESTO

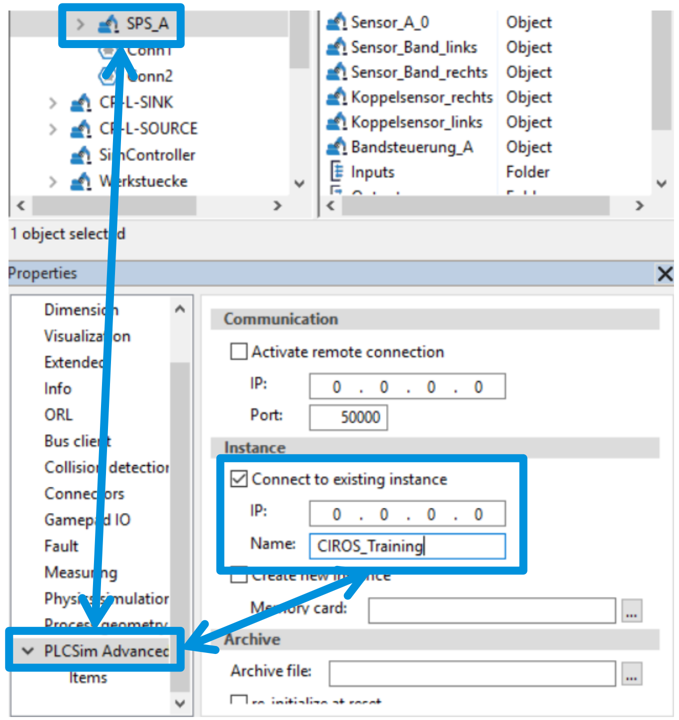
Exercise

Configuring the interface

Configure your CP-L-CONV model to connect to your instance.

All of this is done in CIROS. TIA doesn't know anything about the CIROS interface.

1. Open the properties of your virtual PLC (SPS_A)
2. On the PLCSIM Advanced page, select Find instance by name.
3. Enter the name of your PLCSIM Advanced instance

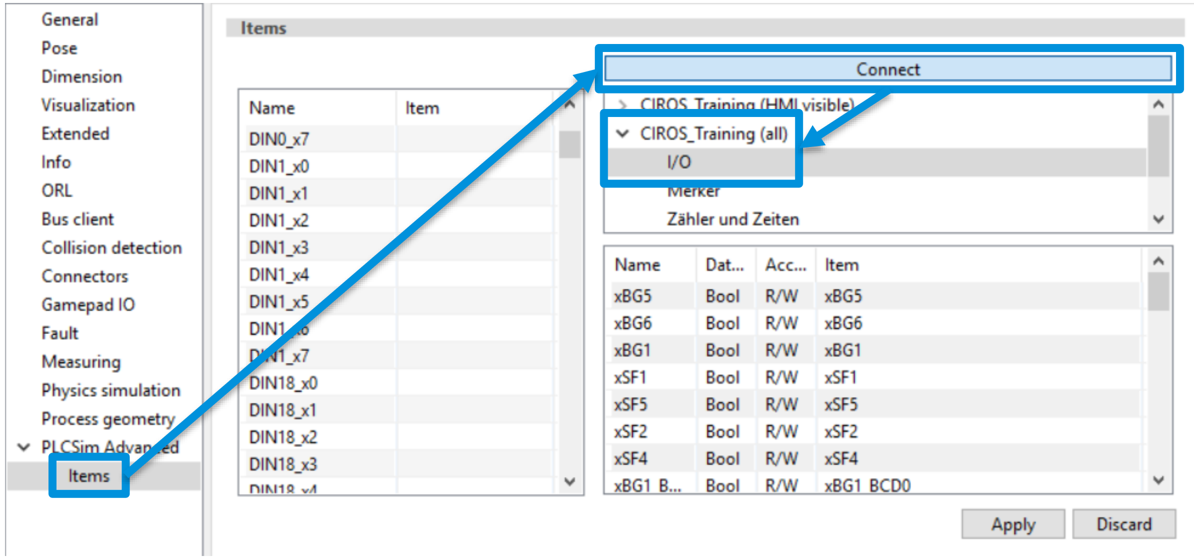


Exercise

Configuring the interface

1. Go to the subpage Items
2. Click on Connect
3. Open either the entry that says all or HMI visible, depending on your preference. The latter only offers you I/O tags that have been declared as *Visible in HMI engineering* inside TIA
4. Under this entry, open I/O

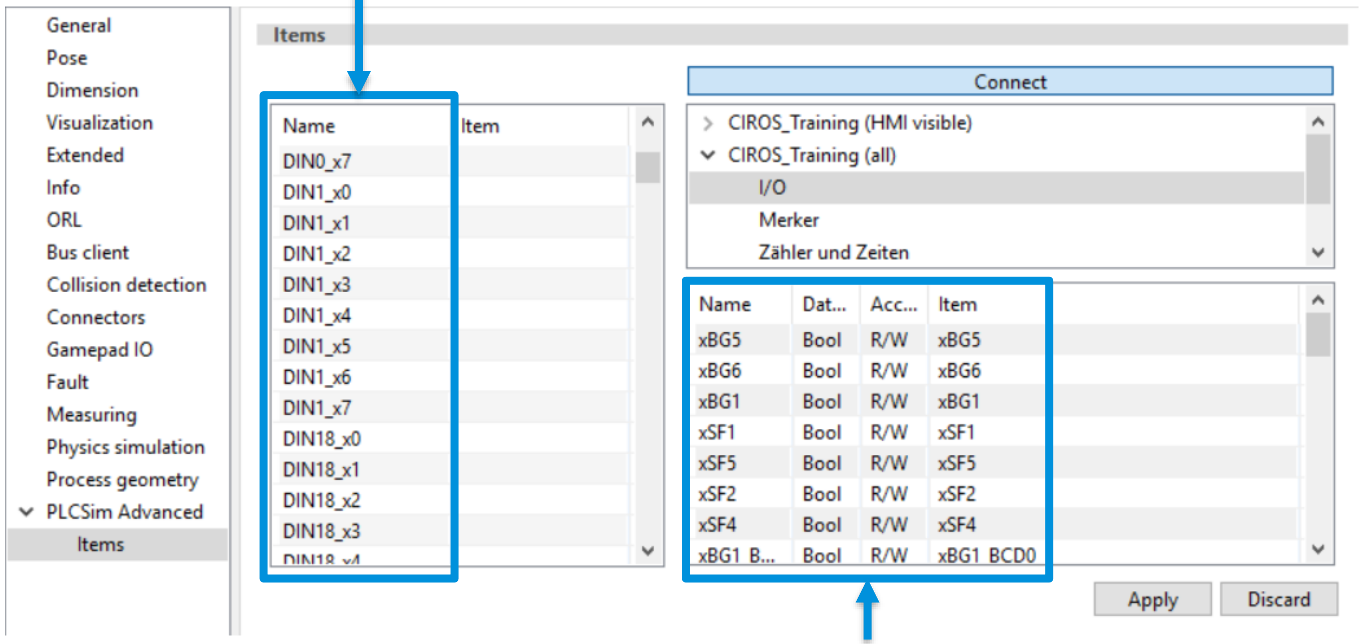
Note that you're also offered Memory, Counters and Timers and Data blocks. You can hook up CIROS I/Os to any of these but in this exercise, we'll only use I/Os.



Exercise

Configuring the interface

Virtual I/Os of the CIROS PLC



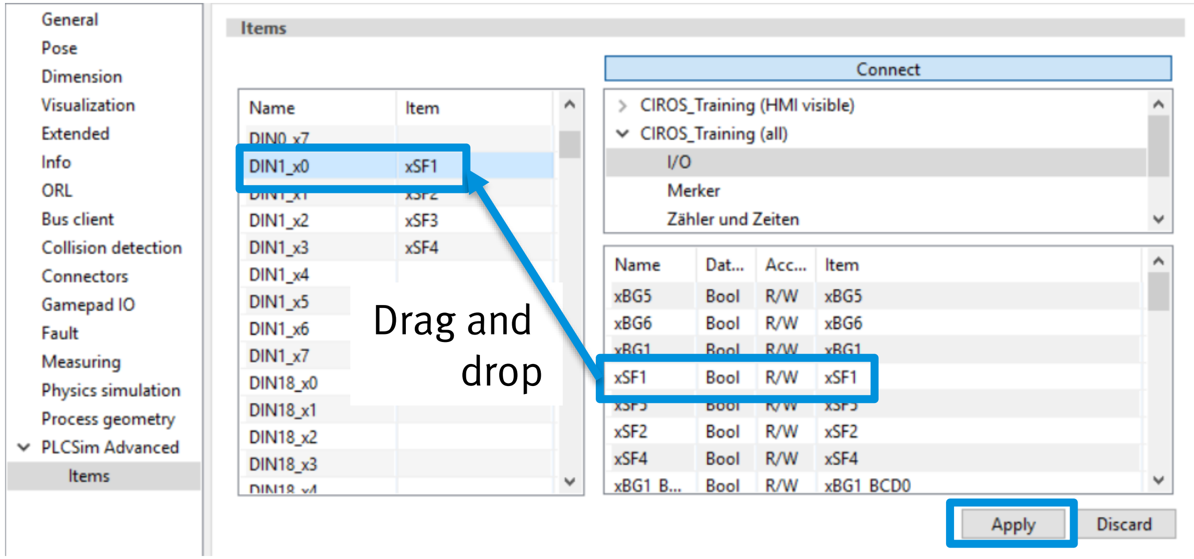
I/O tags configured in TIA

Exercise

Configuring the interface

1. From the list of I/O tags on the right, drag and drop each I/O to the matching entry on the left.
   Note: Connect CIROS inputs with PLC inputs, and CIROS outputs with PLC outputs.
2. When done, click "Apply".
3. Optionally, click "Connect" again to disconnect.

Hint: You can select multiple I/Os from the right if they are in the right order and drag them to the left at once.

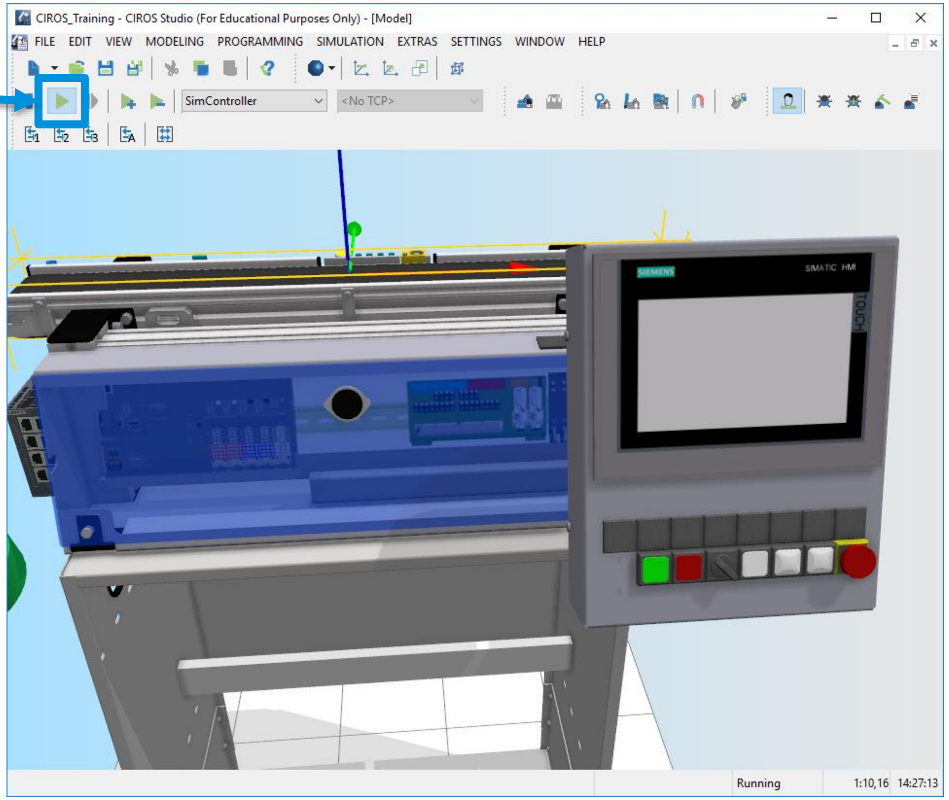


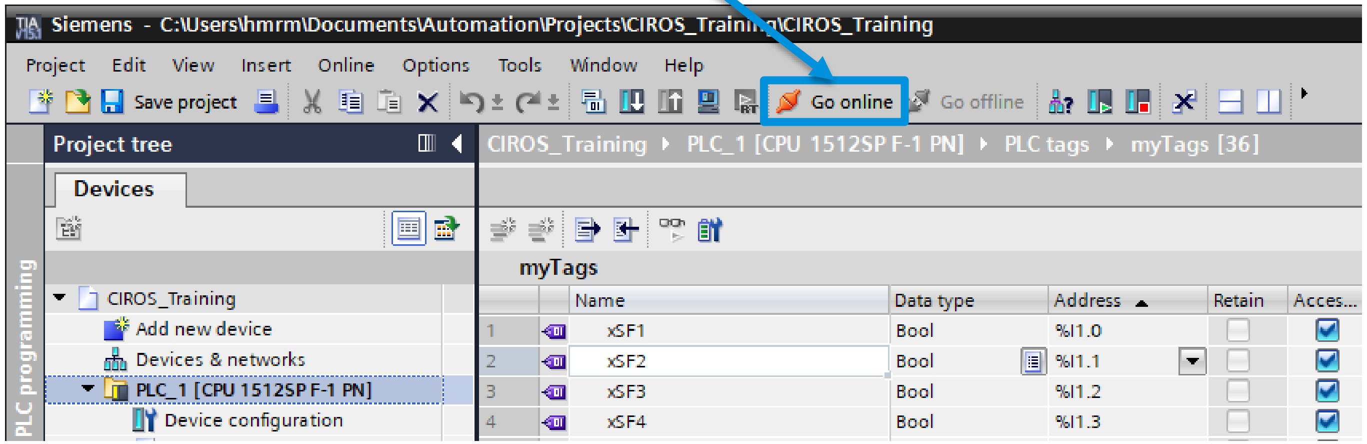
FESTO

Exercise

Run the simulation

1. Click the play button to start the simulation
2. Use TIA to go online / connect to your PLC instance

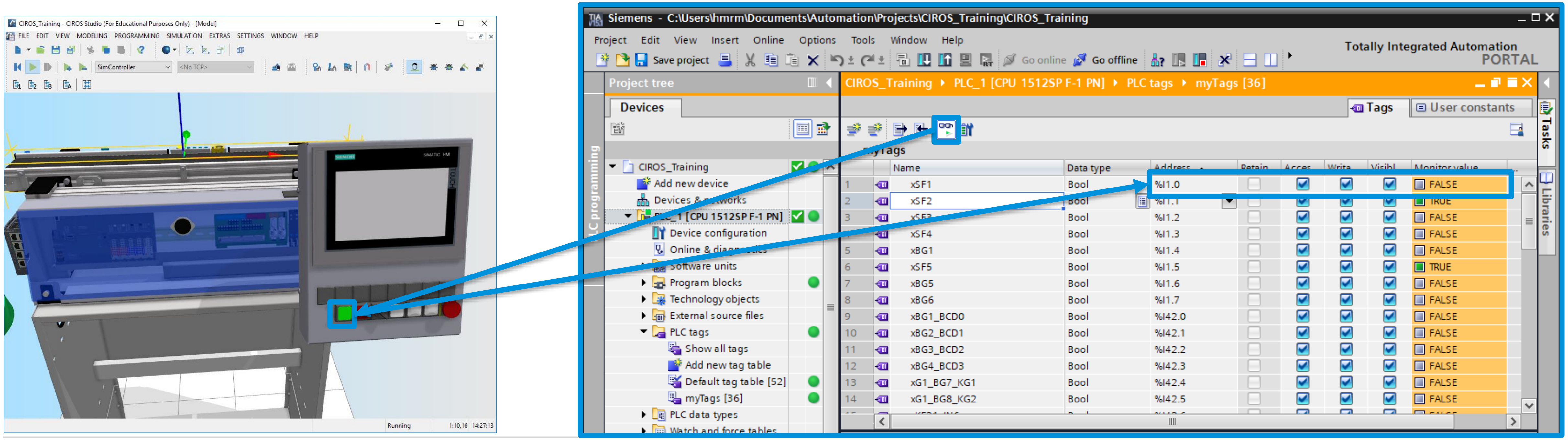




Exercise

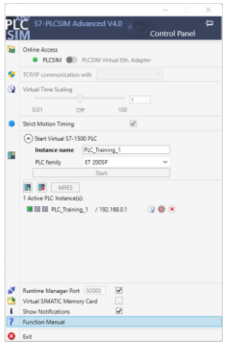
Run the simulation

1. In TIA, open your tag list and monitor it
2. Test the connection by clicking on the virtual green start button inside CIROS. You should see the value of %I1.0 change in TIA



More Information

Mapping CIROS I/Os to PLCSIM Advanced instance I/Os



Inputs ⟷ Inputs

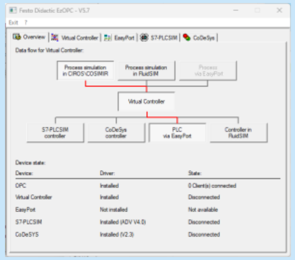
Outputs ⟷ Outputs

FESTO

More Information

Mapping CIROS I/Os to real PLC via EasyPort and EzOPC

Software



Hardware

OPC DA

USB

Syslink

Inputs

Inputs

Inputs

Inputs

Outputs

Outputs

Outputs

Outputs

Common Issues

Can’t download project to PLC instance anymore

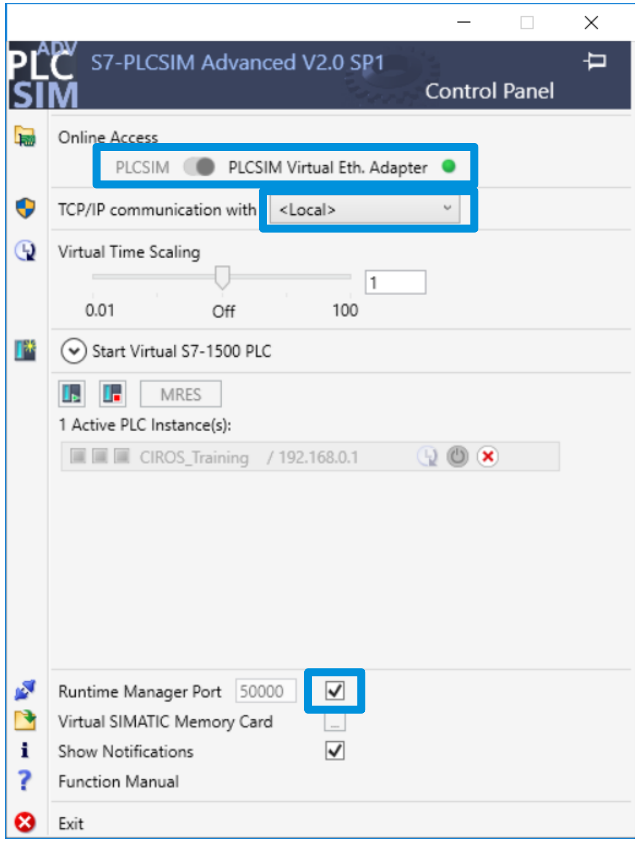
Once CIROS has established a link to a PLCSIM instance, that instance is bound to the simulation. Only when the CIROS simulation is running, will the instance run as well.

Should TIA appear to be stuck when downloading to the instance that is likely because your CIROS simulation is paused. As soon as you start the simulation, the download will continue.

Remote Connection between CIROS and PLCSIM Advanced

If you’re running CIROS and PLCSIM Advanced on different machines, you need a little bit of extra configuration.

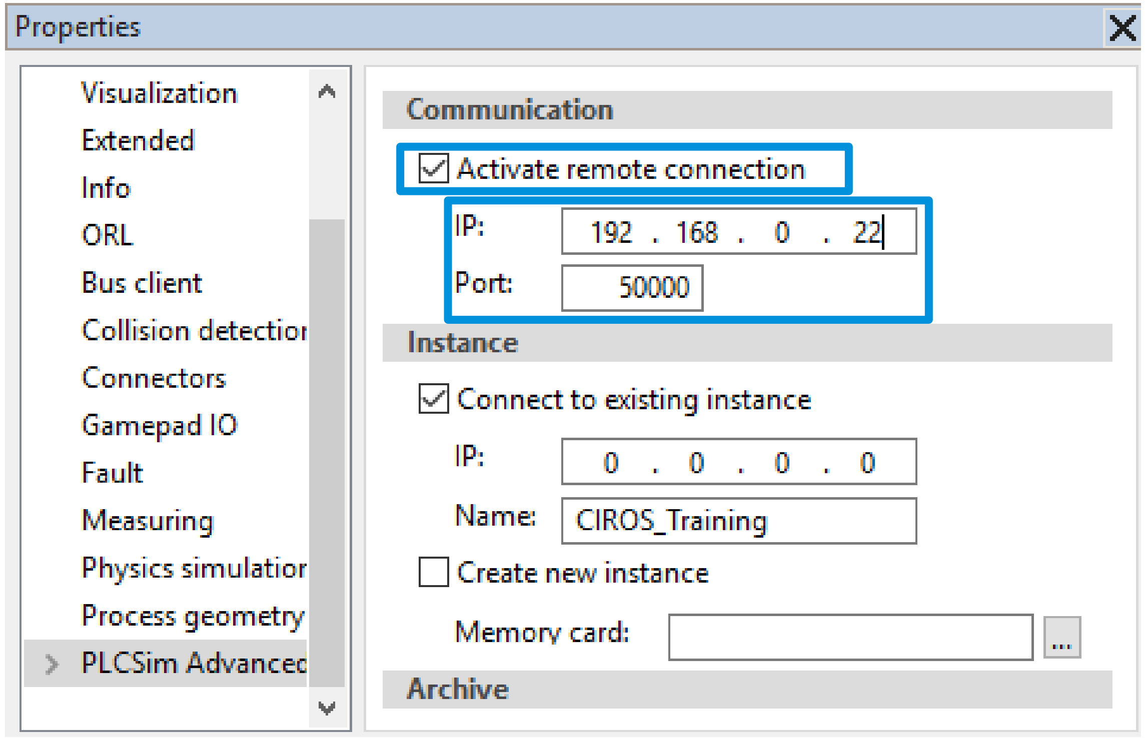
1. PLCSIM Advanced must use the Virtual Ethernet Adapter
2. The communication interface must be set to the network interface through which you’re connecting to CIROS
3. The runtime manager port must be enabled. Note the port number written here



Remote Connection between CIROS and PLCSIM Advanced

You also have to let CIROS know, where to find PLCSIM Advanced.

1. Go to the virtual PLC’s properties
2. Open the PLCSIM Advanced page
3. Check Activate remote connection
4. Enter the IP address of the PC where PLCSIM Advanced is running
5. Enter the runtime manager port number that is configured in PLCSIM Advanced

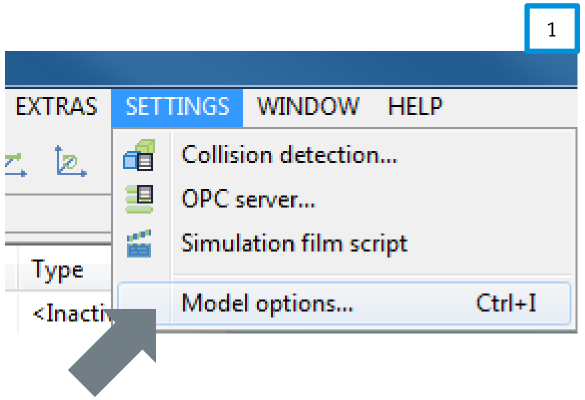


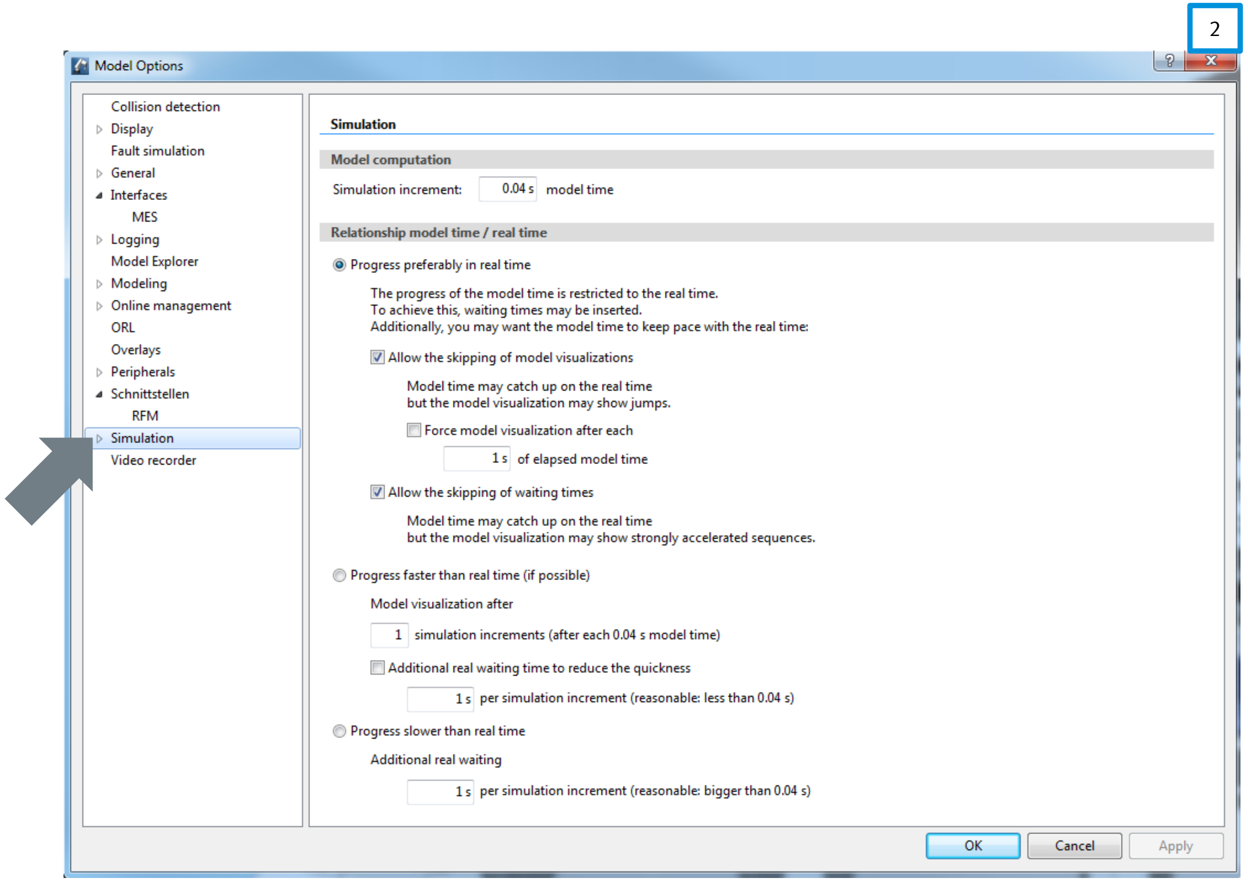
Simulation

Simulation Kernel

- Settings → Model options allows for configuring the way in which the simulation status will be updated during simulation.

- Mainly characterized by two parameters,
    - model computation
    - relationship model time / real time



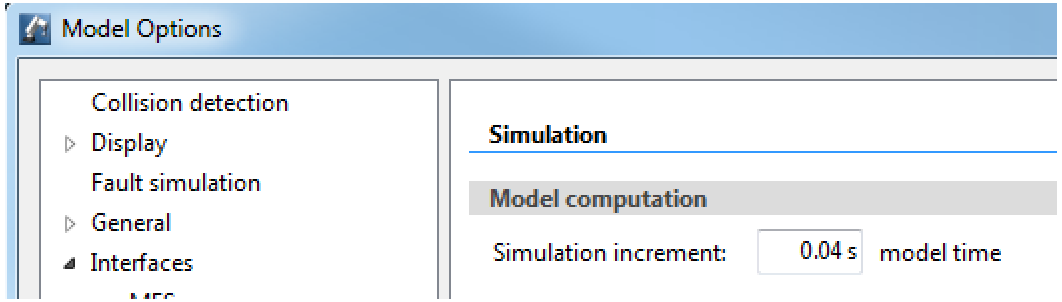


FESTO

Simulation Kernel

Model computation

- Simulation increment specifies the intervals in which the simulation status and its visualization will be updated.
- Default of 0.04s means 25 updates per second of model time.
- Provided that the computer CIROS is running on is powerful enough, this results in real-time behavior!
- Increasing the Simulation increment leads to fewer calculations of simulation states which might lead to some strange behavior.

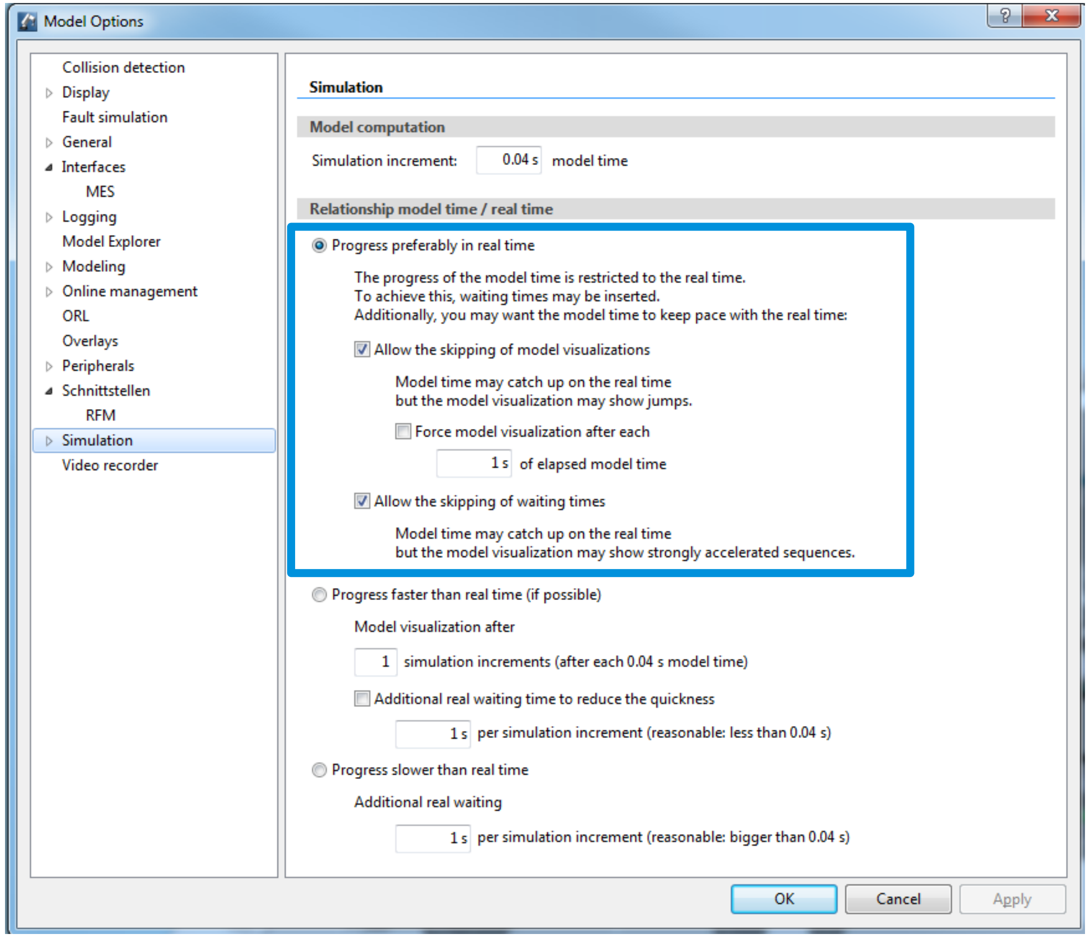


Rule of thumb: Do not touch the default value for the simulation increment!

Simulation Kernel

Relationship model time / real time (1)

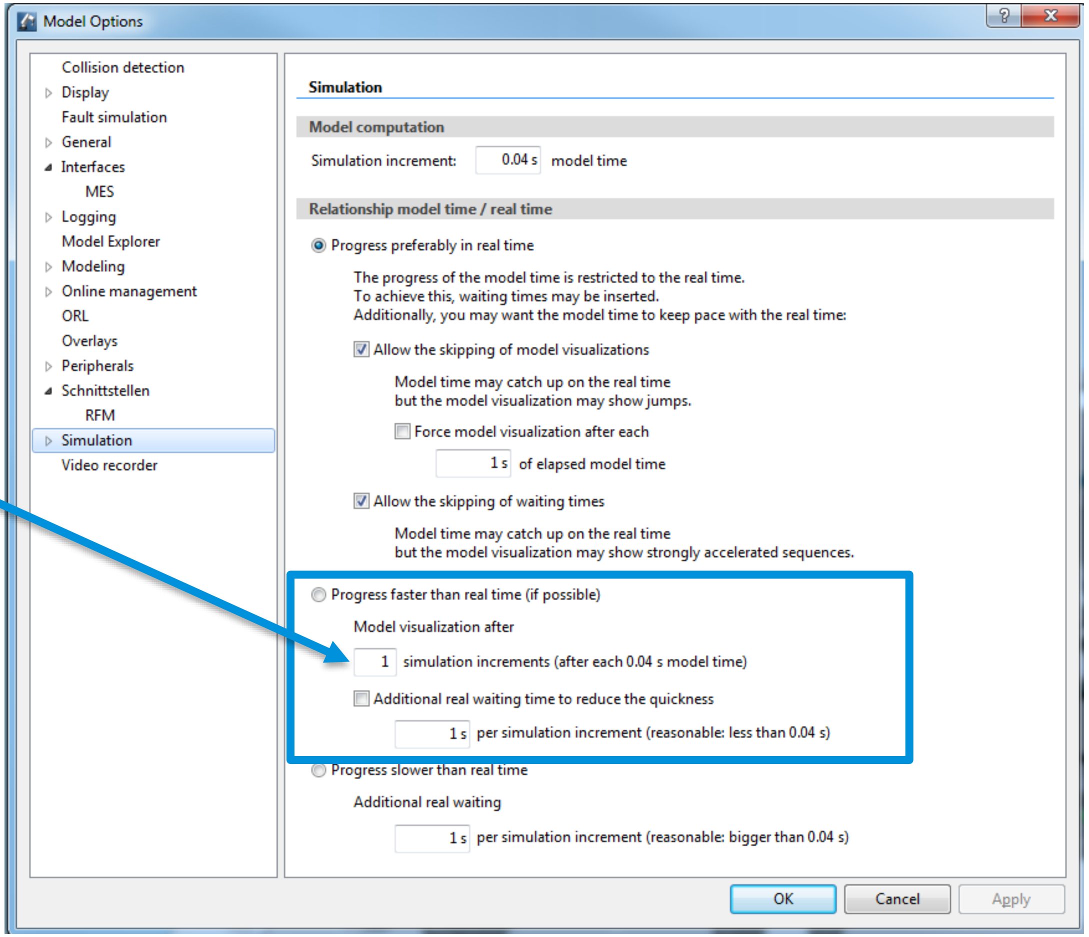
- Progress preferably in real time
  - Progress of model time is restricted to real time.
  - Default option, ensuring that a CIROS model behaves like a similar real CP Lab / Factory system (wrt. process times).
- Allow the skipping of model visualizations
  - By default, the visualization gets updated with every simulation increment.
  - Skipping some of these calculations might give CIROS the chance to keep track with real time.
- Allow the skipping of waiting times
  - To keep track with real time, one can also skip simulation updates when nothing has changed.



Simulation Kernel

Relationship model time / real time (2)

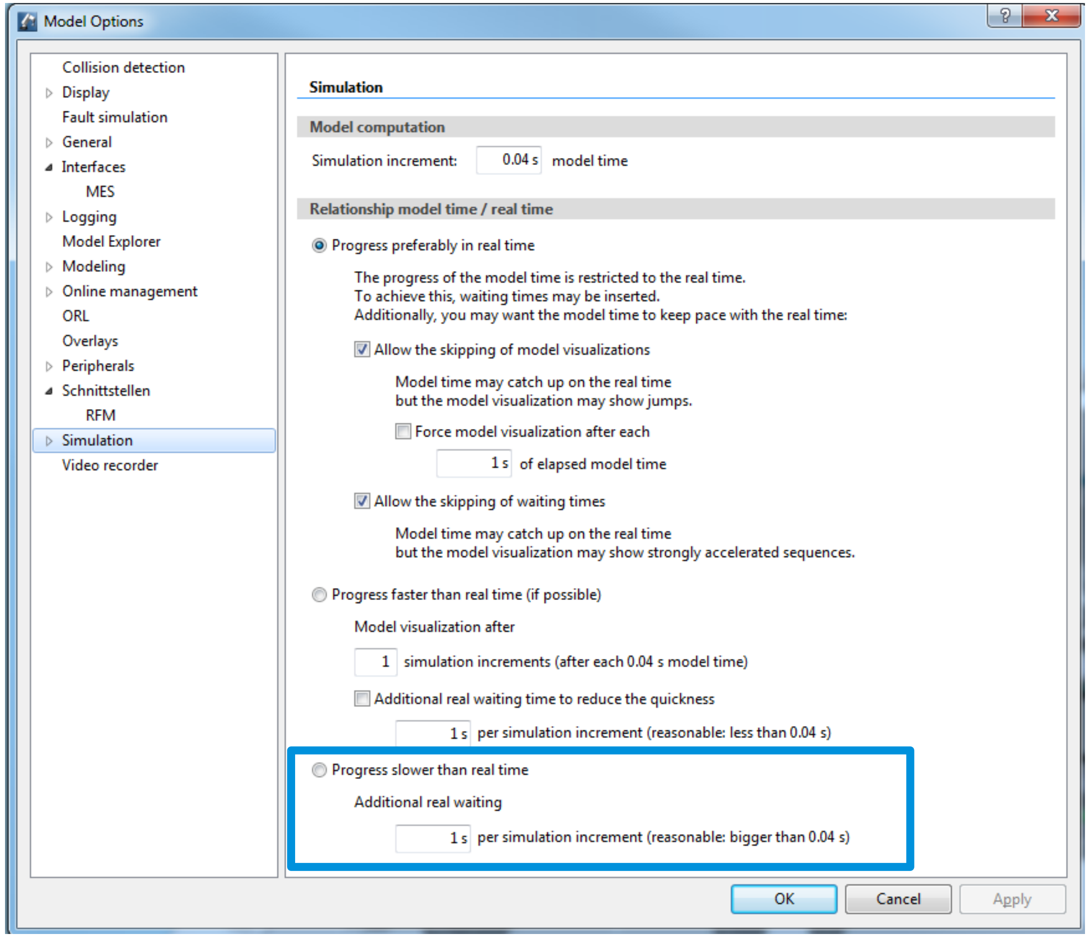
- Progress faster than real time (if possible)
  - Allows CIROS to simulate as fast as the underlying hardware environment allows.
  - Skipping some visualizations results in even faster computations.
  - The speed can be set to adjust how fast the simulation should be in Model visualization after __ simulation increments ...
    - For example, to simulate 5x faster than real time, configure it as follow: Model visualization after 5 simulation increments ...



Simulation Kernel

Relationship model time / real time (3)

- Progress slower than real time
  - Mainly for debugging purposes.



FESTO

Simulation Kernel

Remarks

- Timestamps in CIROS are based on model time, while in MES4 timestamps are based on real time!
- Example of what could happen if model time differs from real time.
  - Assume that a single step of an MES4 workplan takes 10s of real time on a corresponding real application module.
  - If CIROS is keeping track with real time (model time = real time), the step will be simulated in 10s of real time, too.
  - If CIROS is running faster, this step still requires 10s of model time, but CIROS can simulate that 10s of model time in (to give an example) 2s of real time. MES4 will record that the operation required only 2s of real time!
- Therefore, if CIROS is simulating faster/slower than real time the process times the MES4 is measuring are no longer reliable!
- Due to the different ways of measuring time, it is not possible to simulate the annual production of a CP Lab / Factory within a few hours!

Reduce Simulation Computing Requirement

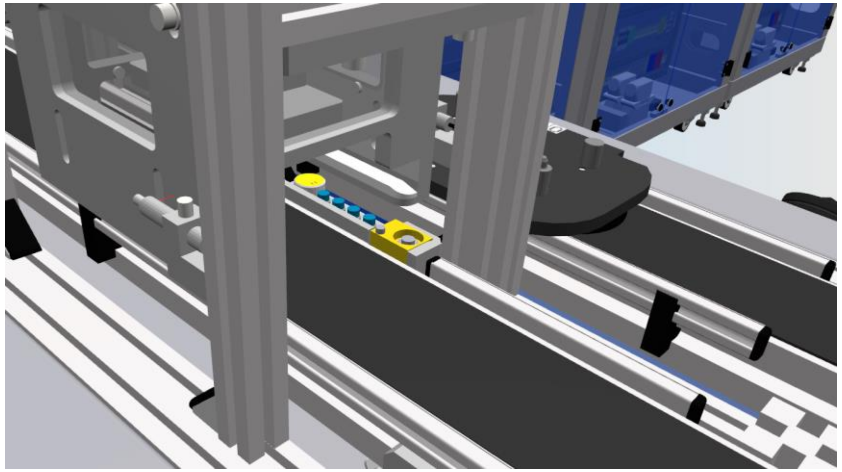
Disabling shadow simulation

- Calculating shadows during simulation allows for a more natural appearance of the model but requires a lot of CPU/GPU performance. **Simulation might slow down significantly!**
- For less powerful hardware environments disabling shadows improves overall simulation performance.
  - Screen Space Ambient Occlusion (SSAO)
  - Shadow light sources

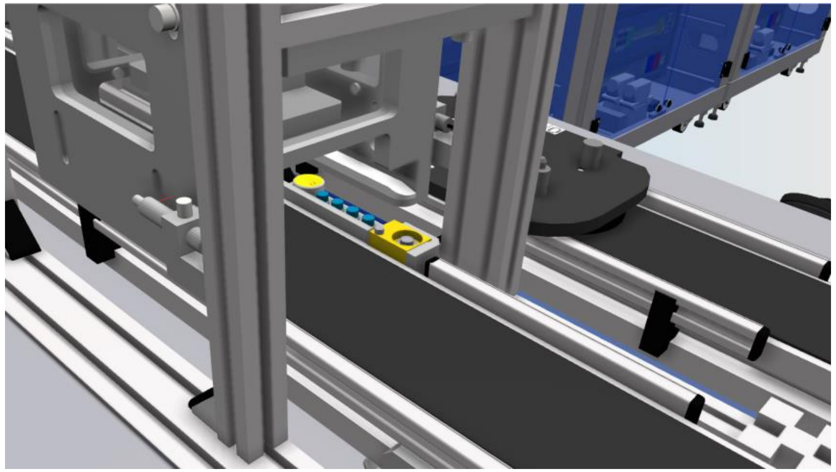
Reduce Simulation Computing Requirement

Screen Space Ambient Occlusion (SSAO) (1)

- SSAO is a computer graphics technique for efficiently approximating the ambient occlusion effect, caused by ambient lightning, in real time.
- While the implementation in principle is quite fast, it nevertheless requires substantial computation power.



without SSAO

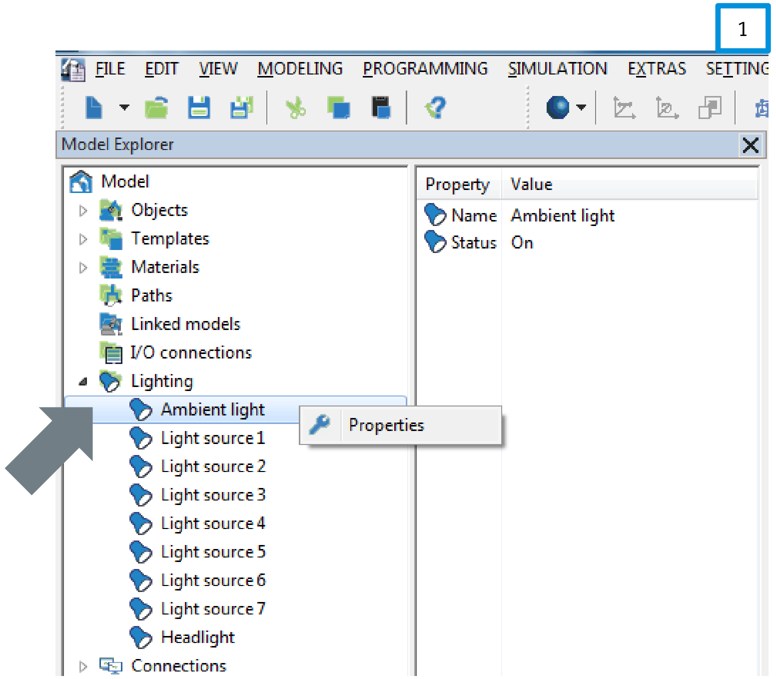


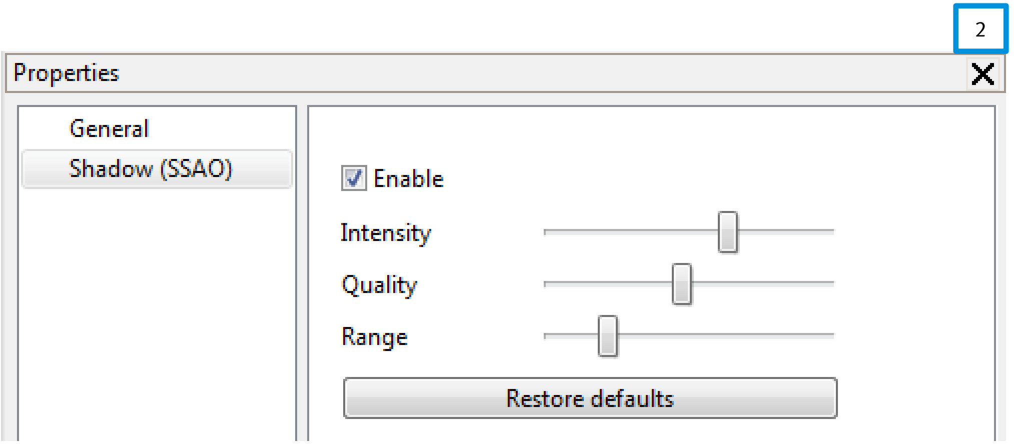
with SSAO

Reduce Simulation Computing Requirement

Screen Space Ambient Occlusion (SSAO) of a model (2)

- Use Model Explorer → Model → Lightning → Ambient light → Properties to disable or enable Screen Space Ambient Occlusion.

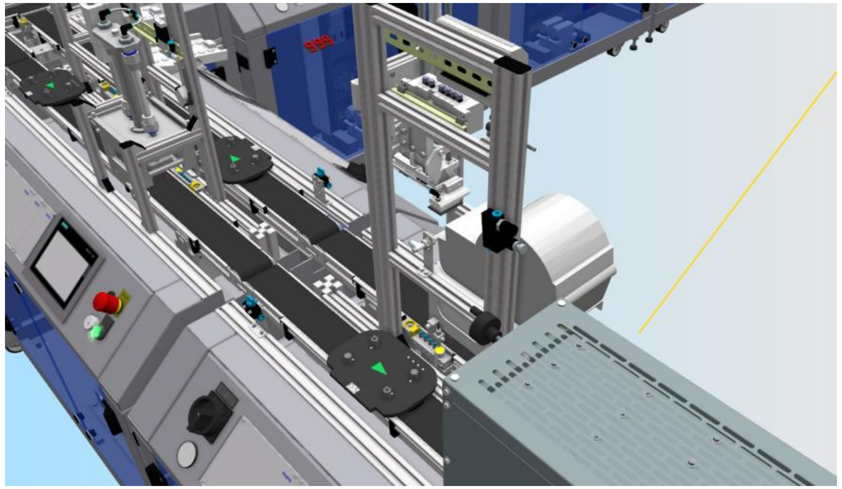




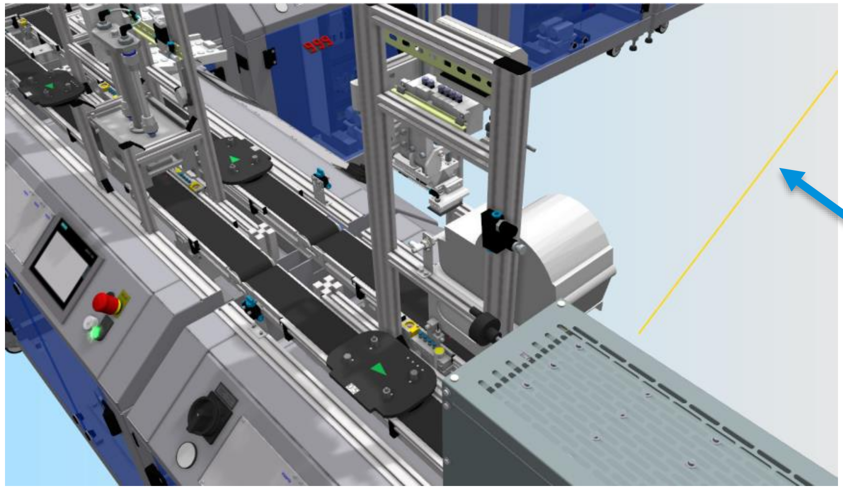
Reduce Simulation Computing Requirement

Shadow light source 1

- Usually, light source 1 is causing shadows, while light sources 2 to 7 and headlight do not have that option
- If enabled the simulation of that shadow depends on the position of light source 1 and its properties



without shadows caused by light source 1



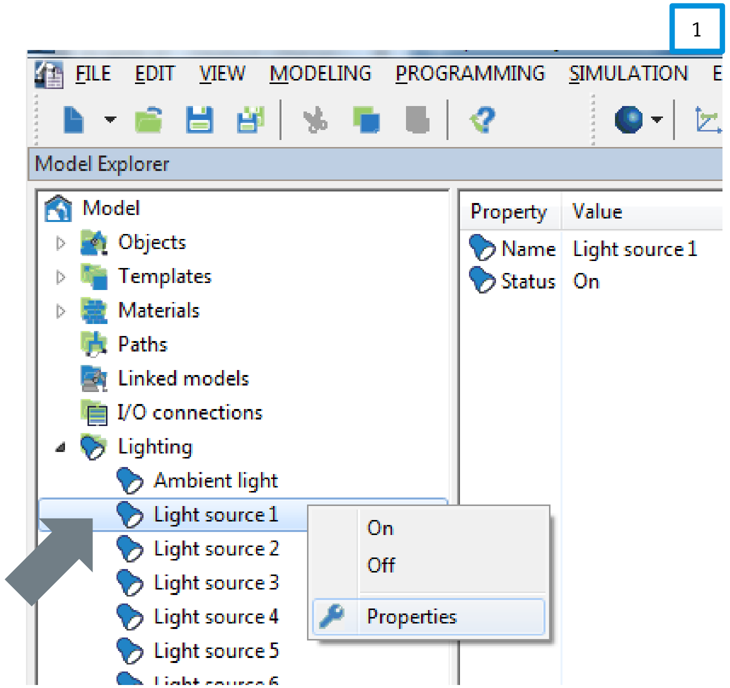
including shadows

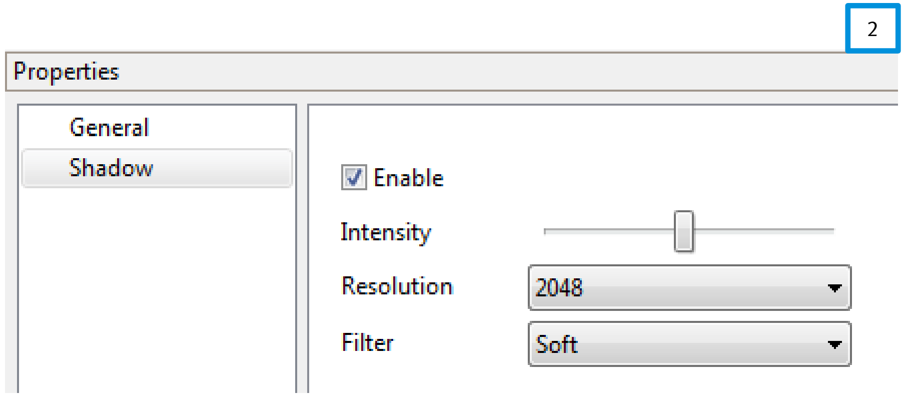
Position & direction of light source 1

Reduce Simulation Computing Requirement

Shadow light source 1 of a model

- Use Model Explorer → Model → Lightning → Light source 1 → Properties to disable or enable shadow simulation.

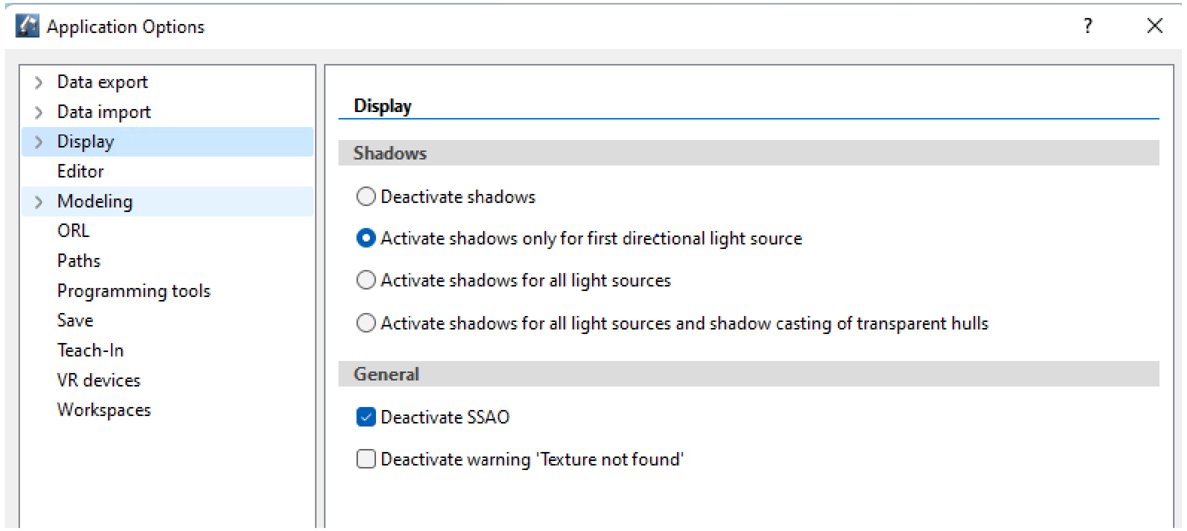




Reduce Simulation Computing Requirement

SSAO and shadow source of CIROS application.

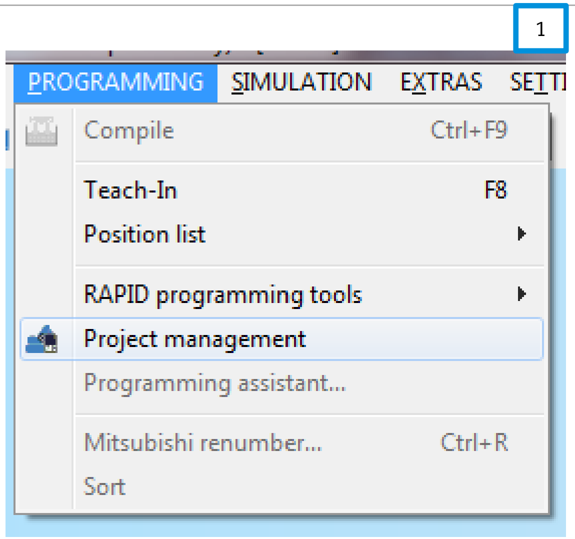
- Disabling shadows like shown on the slides before is valid for a particular model only!
- To disable or enable shadows regardless of the model use File → Application options → Display.

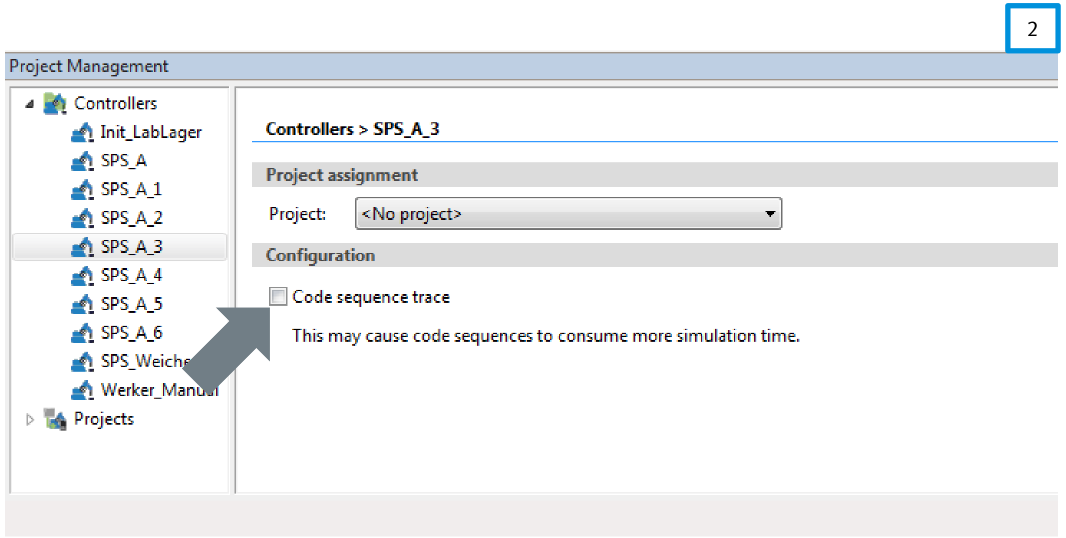


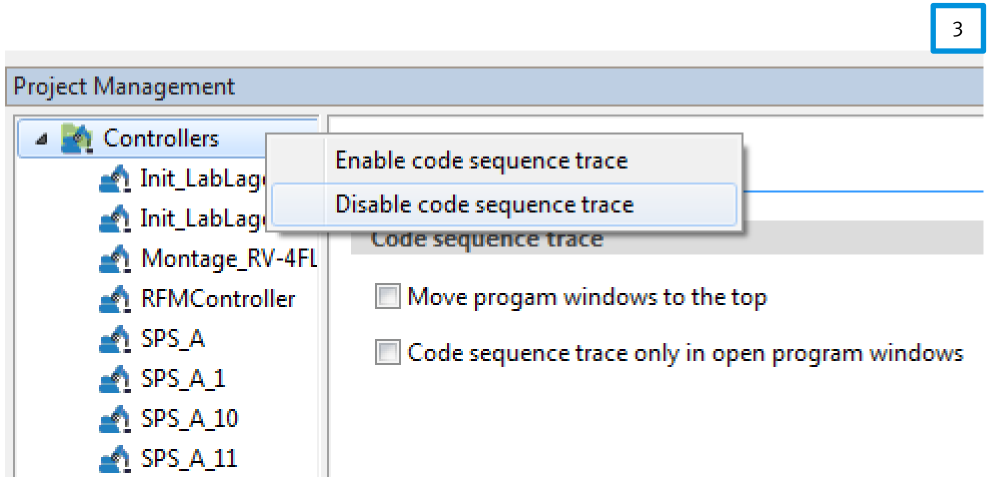
Code Sequence Trace

- Simulation is controlled by the chosen controller in the project and its code, whether PLC controller or robot program, can be traced during the simulation.

1. The function can be activated in Programming → Project Management.
2. To enable or disable tracing of a particular controller, check the option Code sequence trace for the controller.
3. To enable or disable tracing of all controllers in the project, right click on Controllers and choose the option.







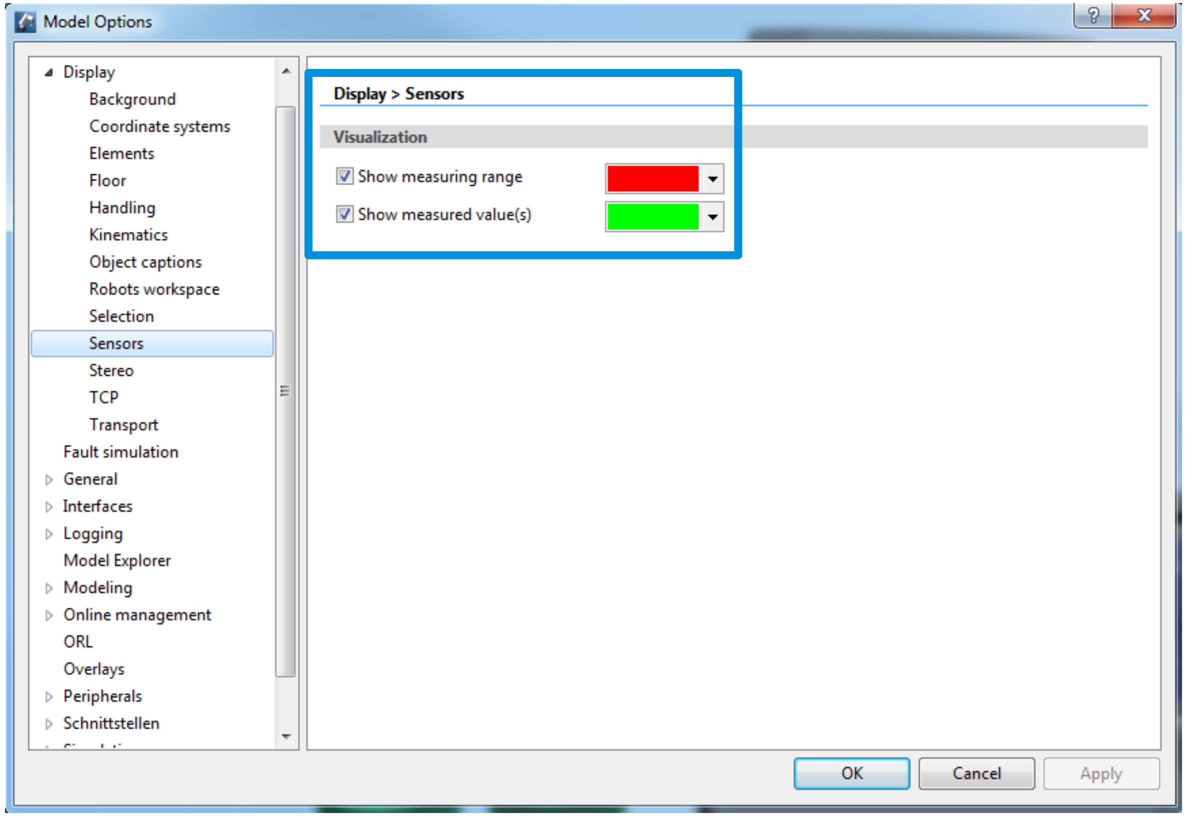
Visualising Sensor Data

- Depending on the model options, measured range and values are not visualized by default in view window.

- Displaying sensors can be achieved by configuring Settings → Model options → Display → Sensors

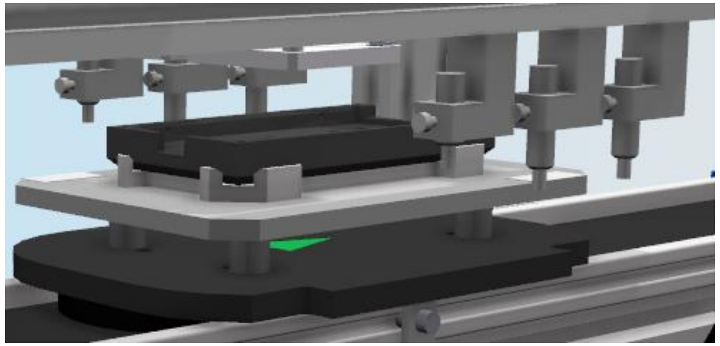
- Measured value depends on the sensor type

Light barrier	Obstacle detected
Distance sensor	Distance to obstacle
Colour sensor	Colour

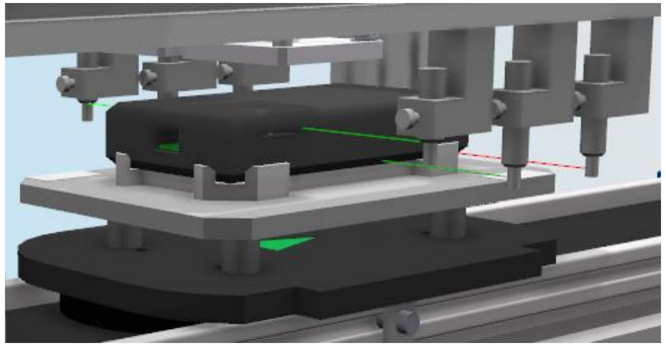


Visualising Sensor Data

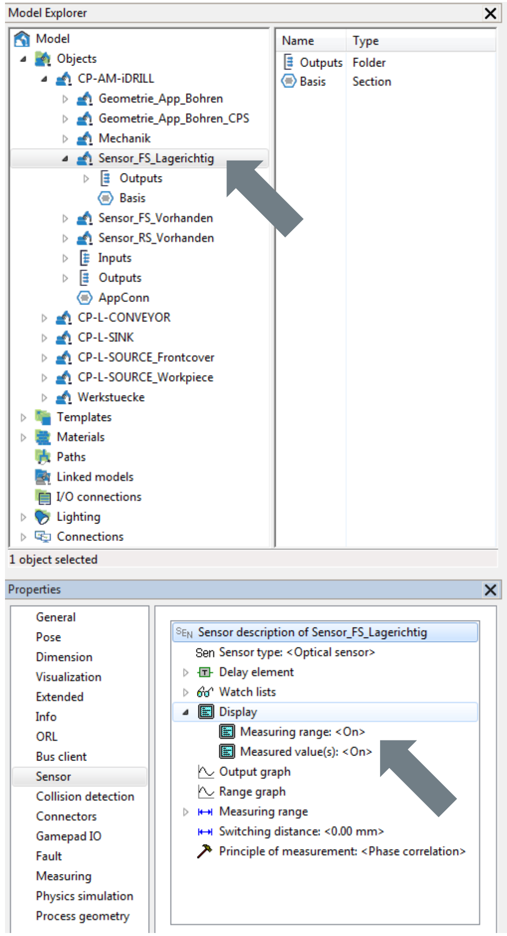
- In rare cases one must enable the visualization of sensor data by modifying the properties of each individual sensor.



Sensor visualisation off



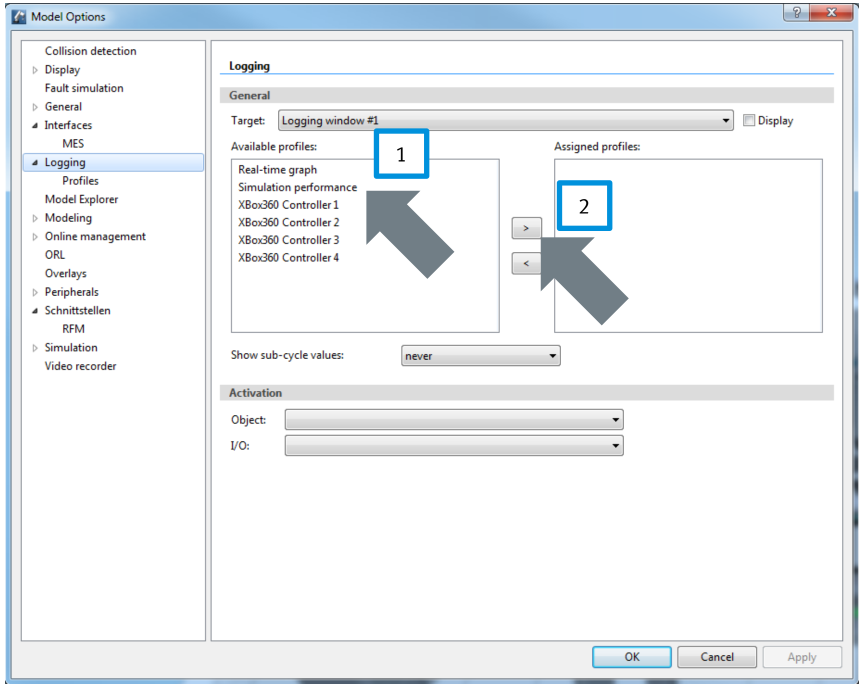
Sensor visualisation on

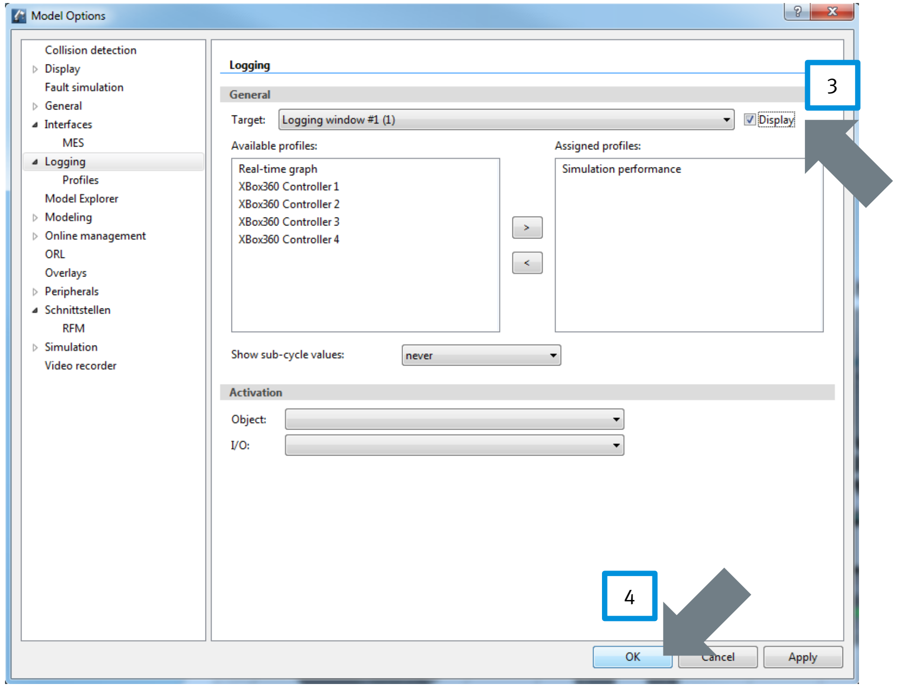


FESTO

Data Logging

- Data logging in CIROS allows for storing and visualization of values of inputs, outputs, variables, etc. during simulation.
- Example shown here: Visualization of the time needed for performing a status update.
- Configuration of data logging is available at Settings → Model options → Logging.



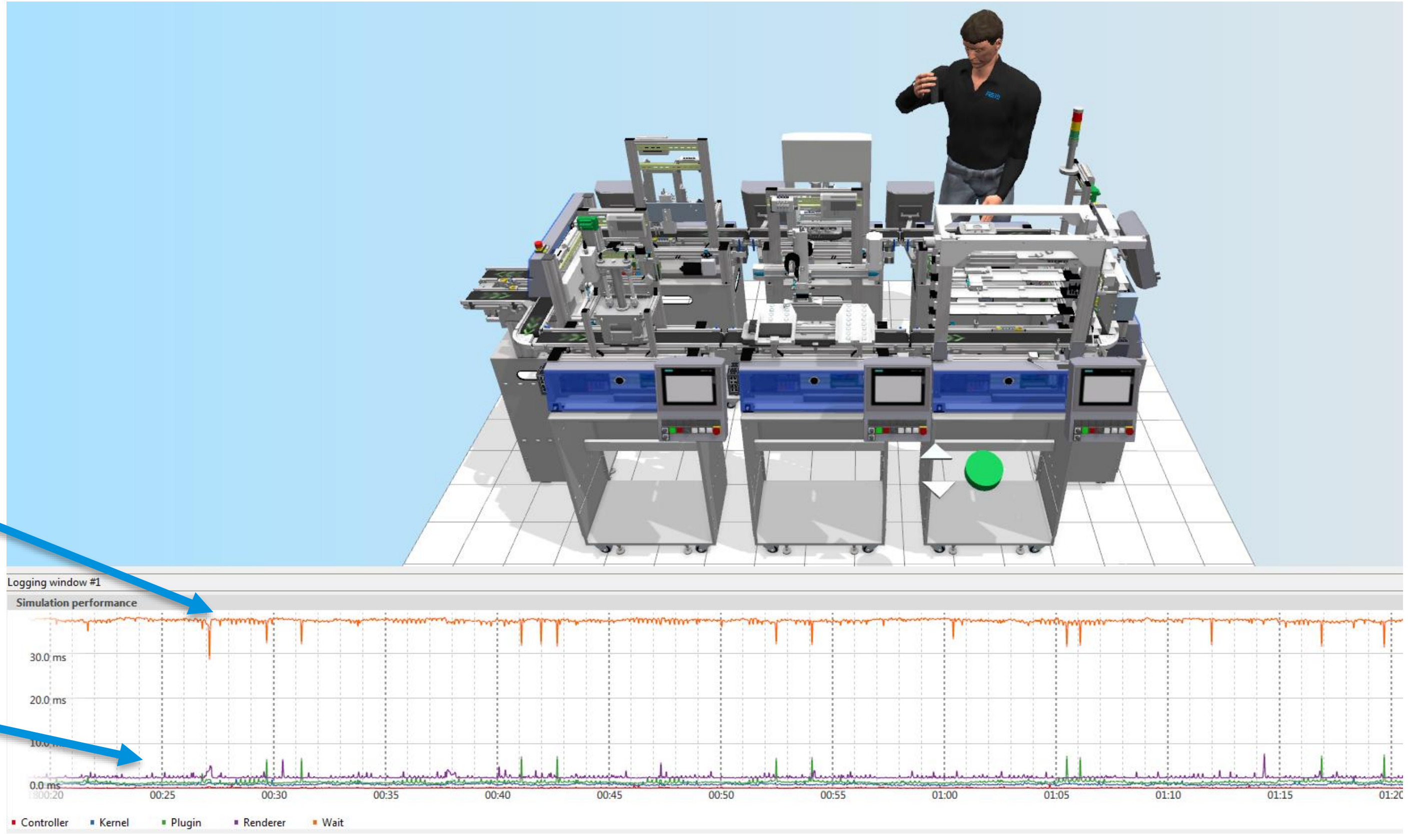


Data Logging

Simulation speed restricted to real time

Almost all the 40ms the simulation is waiting to avoid being faster than real time.

Time required for updating the simulation status is almost zero in this setup.

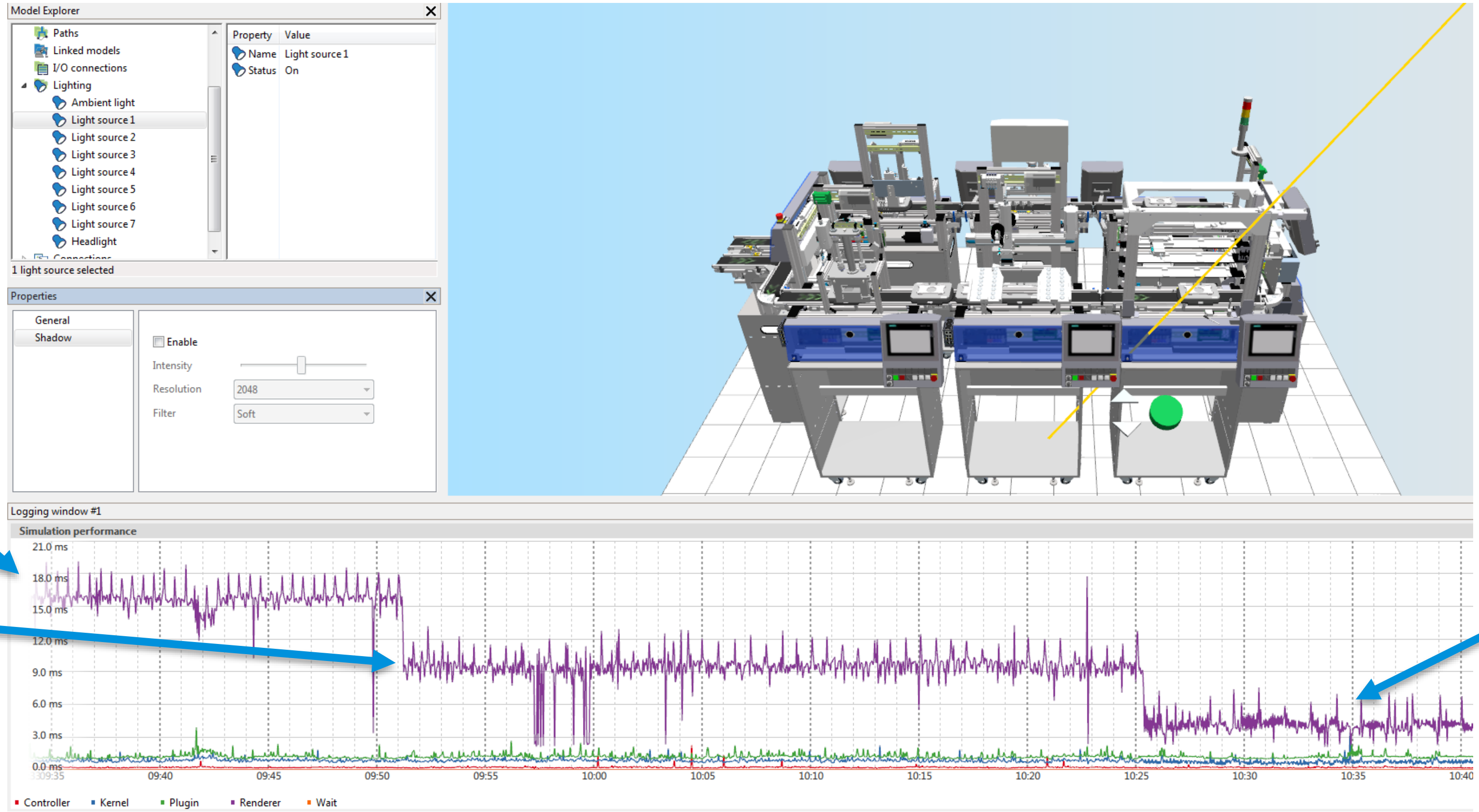


Data Logging

Simulation speed as fast as possible without manual working place

Time required for updating the simulation status.

without SSAO shadows



without both, SSAH and light source 1 shadows

FESTO

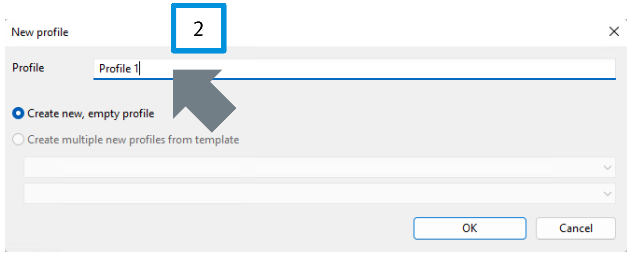
Data Logging

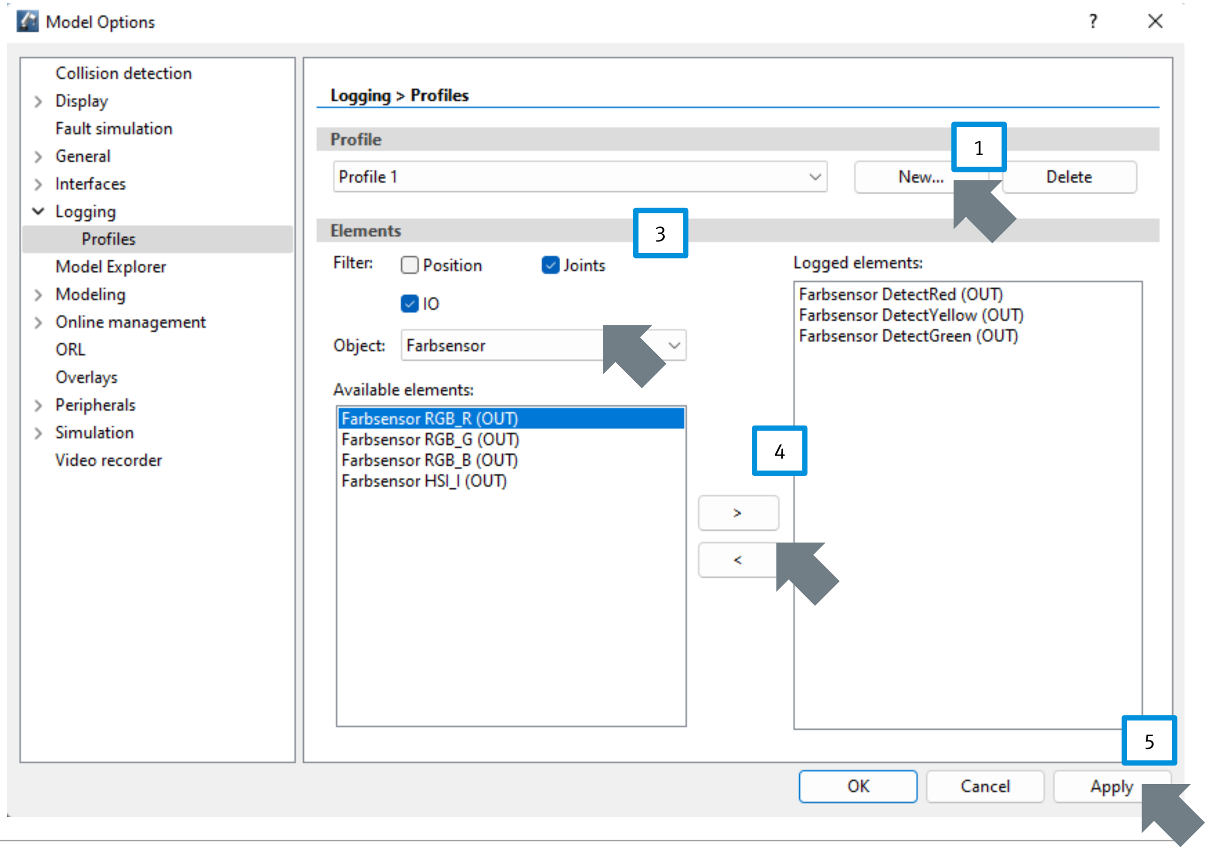
Create new profile

- Apart from default data logging profiles, it is possible to log own data to simulate the model built.
- To log own data, a logging profile containing desired data must be created, it can be created in Settings → Model Options → Logging → Profiles.

1. In section Profile, select New...
2. Give a Name.
3. Use the filter option to filter the selection.
   - Possible data are robot positions, joint coordinates and IOs of the objects.
4. Use the arrow buttons to select and deselect data.
5. Finally, click Apply to create the profile.

- Profile created can then be added to logging window in Settings → Model Options → Logging, section General.





Simulation Control in CP System

Industrial Robotic Language (IRL)

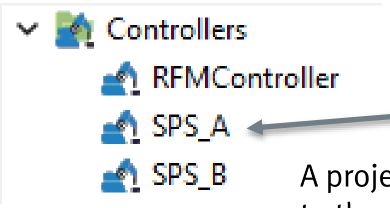
- Located in ‹project file›\CF\CPSystems.
- Scripts can be opened in CIROS in project management window.
  - Programming → Project management
- Main simulation controller.
- Communicates with MES4 and Fleet Manager.
- Control the model movements.
- Read input values and write the output values.

- **Note:** RFID data structure in CIROS is different from in real system.

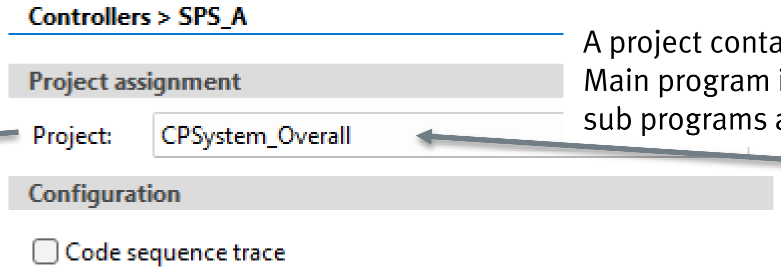
Python

- Located in ‹project file›\CF\py.
- Translate MES4 part number to CIROS part number.
- Replicates workpieces and boxes.
- Send string requests to MES4

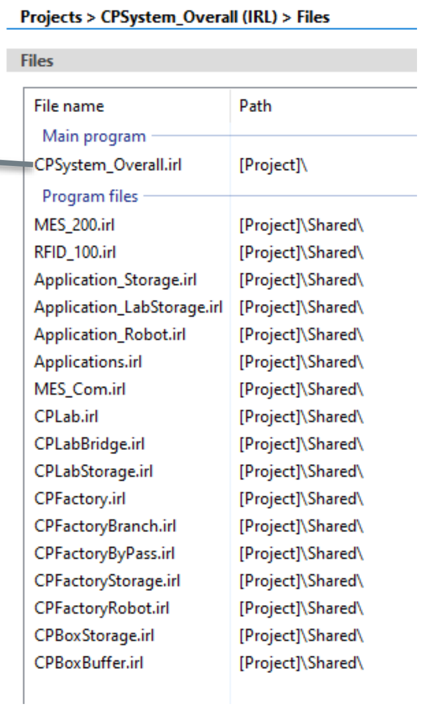
CP System Simulation Controller

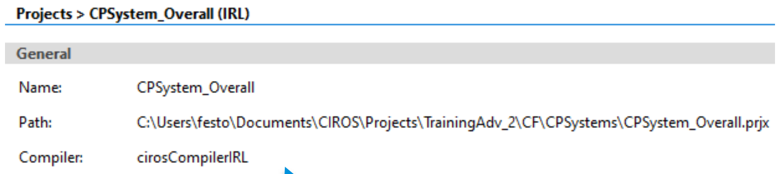


A project is assigned to the controller.



A project contains different controller scripts.
Main program is called cyclically and
sub programs are called in main program.





Project information.

Python

Python in Model Libraries

- Python 3.7 or higher is required for the support of CP Lab / Factory model library.
- Replication of CP Lab / Factory workpieces within CIROS is based on various Python scripts, compared to previous versions this change within the CIROS kernel simplifies the integration of user defined workpieces.

- If not already installed, CIROS installation wizard will install Python and add it to Windows path during the setup in silence mode.

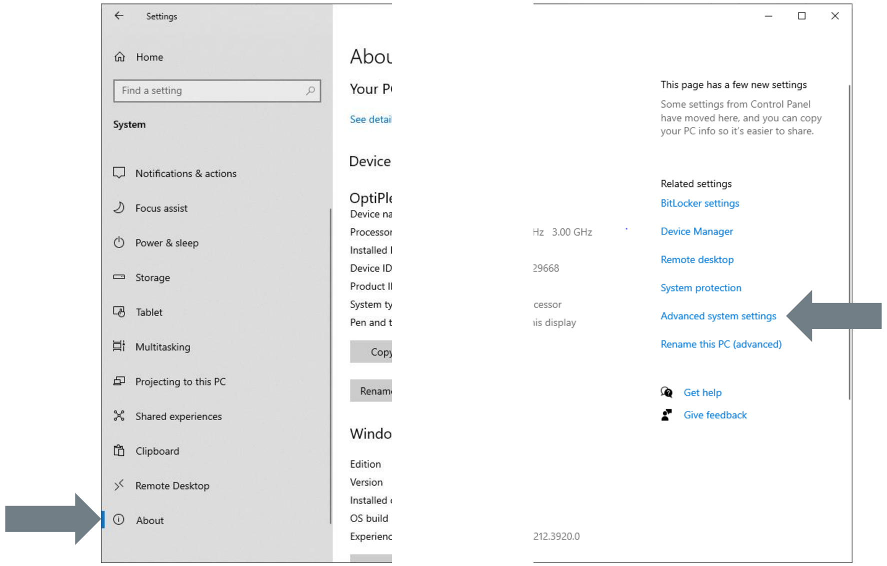
- In case Python in removed...
  - Due to various reasons, like uninstallation of other applications in the same PC, Python can be uninstalled or removed from Windows path.
  - In this case, CIROS will throw an error, most commonly cirospluginpython.dll not loaded.
  - When this happens, user should check whether the correct Python version is installed and is it in the Windows path.

Python in Model Libraries

Adding python to Windows path (1)

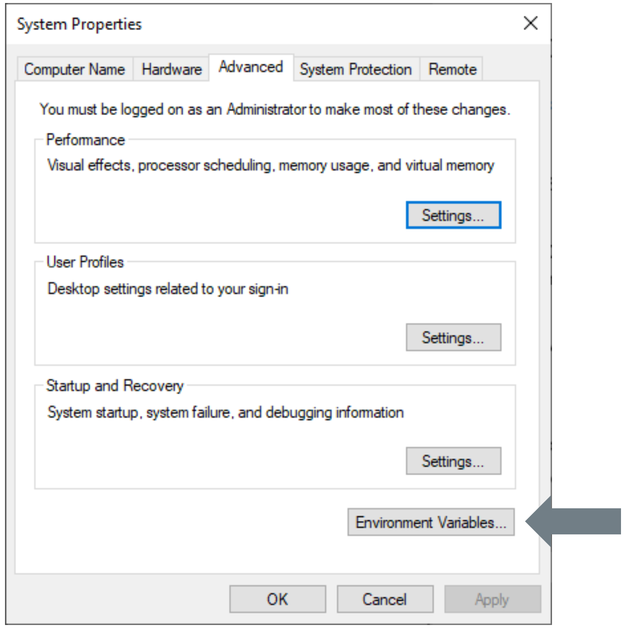
- To define Python in System Environment Variable Path in Windows 10:
  1. Open System Properties.

     Menu → Settings → System → About → Advanced System Settings



2. Open Environment Variables.

   Advanced → Environment Variables

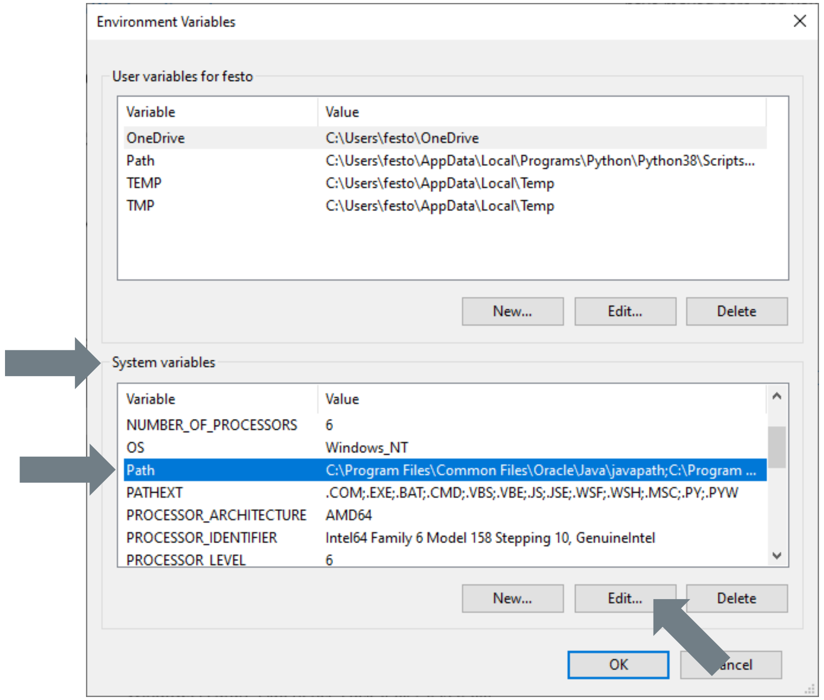


Python in Model Libraries

Adding python to Windows path (2)

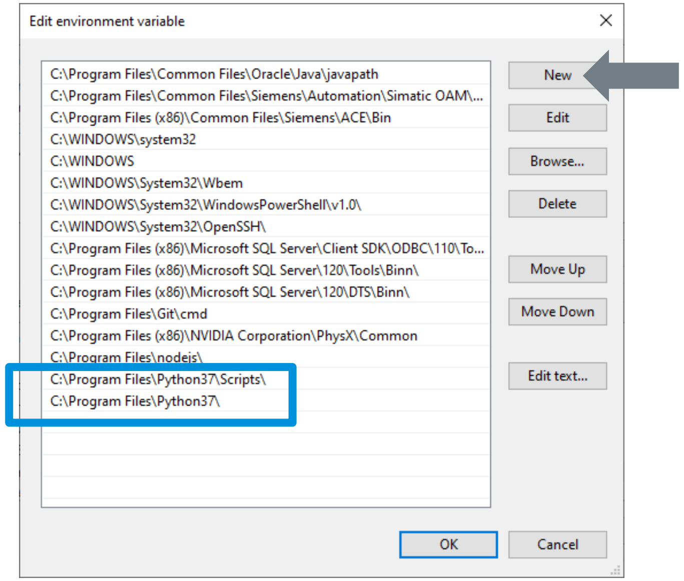
3. Edit Variable Path in System Variables.

   System Variables → Path → Edit



4. Insert the path to Python and Python Scripts folder here.

   For example: C:\Program Files \Python 38\ and C:\Program Files \Python 38\Scripts\

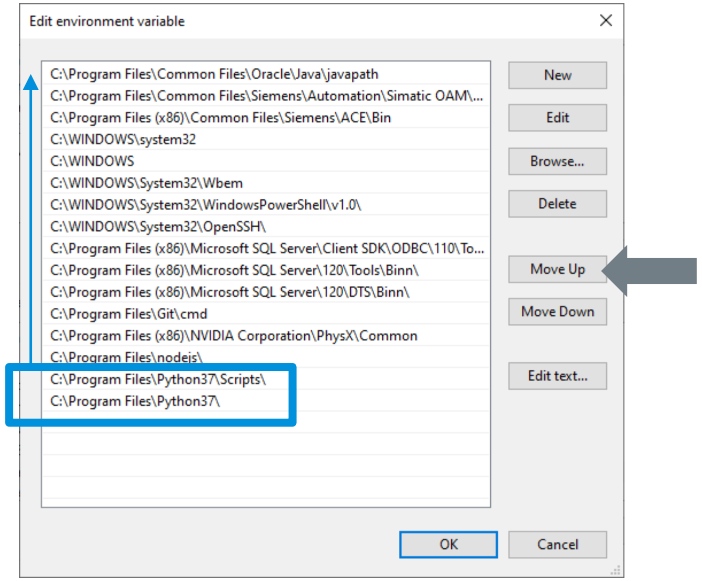


5. Restart the computer.

Python Installed but Not Working

- When different python versions are installed in the same computer, CIROS always selects the version on top of the Windows environment path list.
- When the version on top of the list is not supported by CIROS, there will be an error.
- Typical error message is “cirosPluginPython.dll not loaded”.

- Steps to solve this problem
  1. Open Windows Environment Variable.
  2. Edit System variables.
  3. Move the python with correct version up to the top.



Python Scripts

- CIROS v7 and above works with Python scripts. The scripts can be called for following purposes
  - Creation and modification of model.
  - Controller for simulation or components.
  - Define user defined commands in context menu.

- There is an integrated python module for CIROS, “Ciros”
  - Overview of the functions in the module can be called from Menu → Extras → Python → Show function list
  - In CIROS 7.1, there are currently 19 classes and 275 commands
  - Example models can be found in C:\Program Files\Festo Didactic\CIROS 7.0\CIROS Studio\Example Models\Help\Python

- Requirements
  - Python 3 is installed
    - Supported versions are 3.6, 3.7, 3.8 and 3.9
    - The used version will be chosen automatically
    - First line in python function list states the version used.

Note:

- Python must be defined in windows PATH-variable.
- The script cannot contain endless loop.

Python Scripts

Python scripts can be executed by following methods

1. Manually

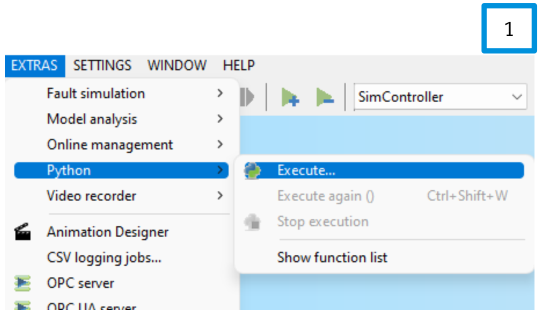
- Application script (Concurrent or blocking)
- With or without simulation
- Menu Extras → Python → Execute

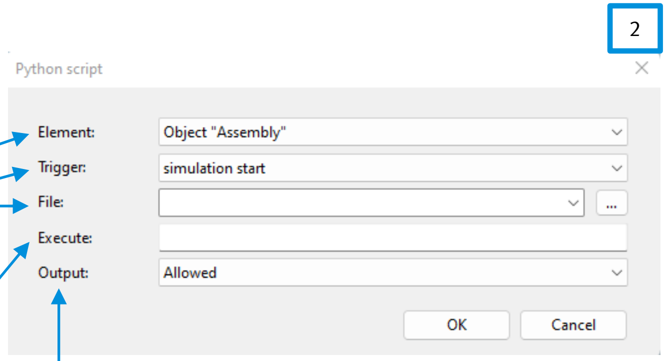
2. By a trigger during simulation

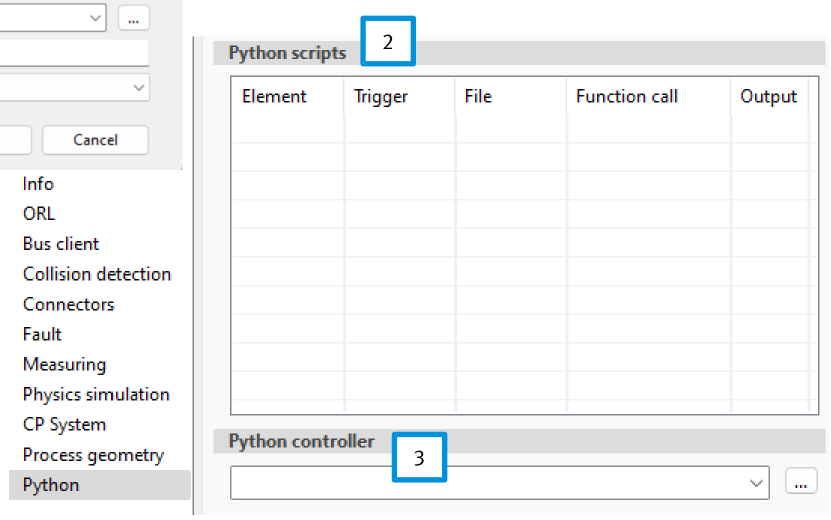
- Simulation script (Blocking)
- Parameters
  - Element: Specify the element that calls the script.
  - Trigger: Specify event that trigger the call.
  - File: The script to be called.
  - Function call: Function in the script to be called. Optional.
  - Output: Defines whether outputs should be written in message window or not.

3. Python controller

- Controller script (Concurrent)
- A controller object, e.g. simulation controller is required.







Python Scripts

Execution type

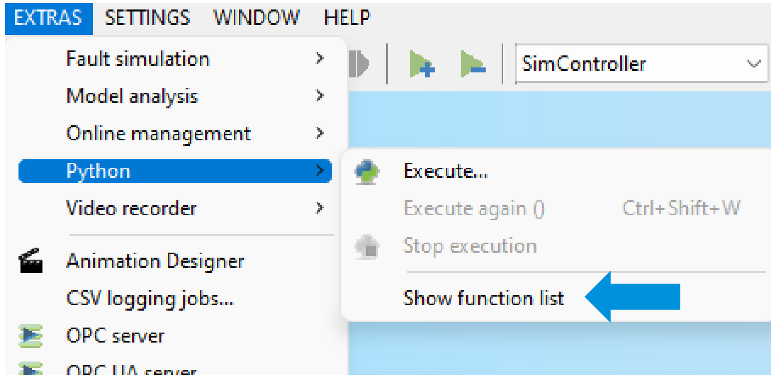
1. Concurrent
   - The python script is executed while CIROS is running.
   - This execution type is required for scripts containing GUI elements.
2. Blocking
   - First, CIROS is interrupted. Then, the python script is executed up to its end. Finally, CIROS is resumed.
   - This execution type is required for scripts containing special commands e.g. 'muteGUI'.

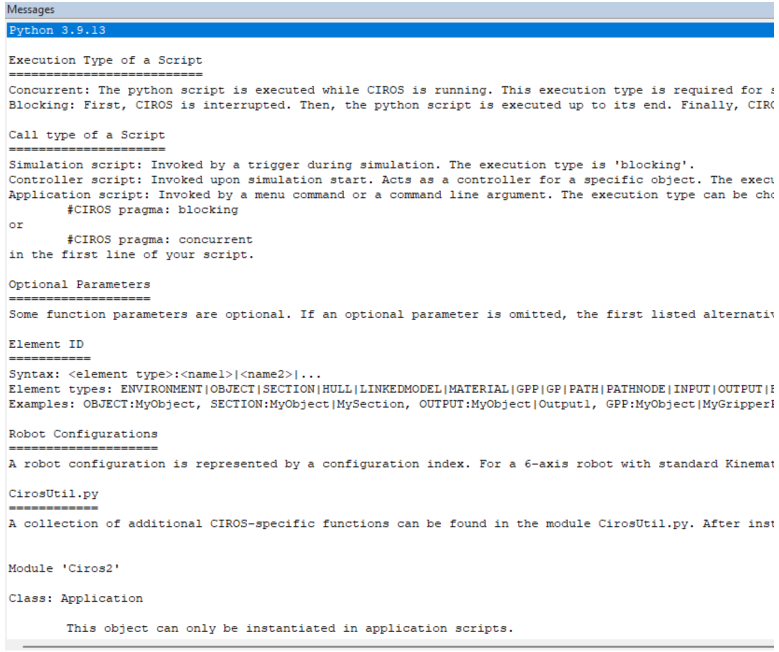
Call type

1. Simulation script
   - Invoked by a trigger during simulation.
   - The execution type is 'blocking'.
2. Controller script
   - Invoked upon simulation start. Acts as a controller for a specific object.
   - The execution type is 'concurrent'.
3. Application script
   - Invoked by a menu command or a command line argument. The execution type can be chosen by the user.
   - Please use *#CIROS pragma: blocking* or *#CIROS pragma: concurrent* in the first line of your script.

Built-In Function List

1. Show the function list by clicking Menu → Extras → Python → Show function list.
2. Copy the function list and paste it in an editor, for example ‘Notepad++’.





CP System Construction Helper

- A python library contains of four classes.

Class	Explanation
CPFactoryConstructionHelper	Constructs a single line of CP-Factory base modules from left to right.

CPLabConstructionHelper	Constructs a single island of CP-Lab modules from left to right, base modules and corners must be added in the correct order, build order is from left to right.
	`* 6 5 4 *`
	`7       3`
	`* 0 1 2 *`

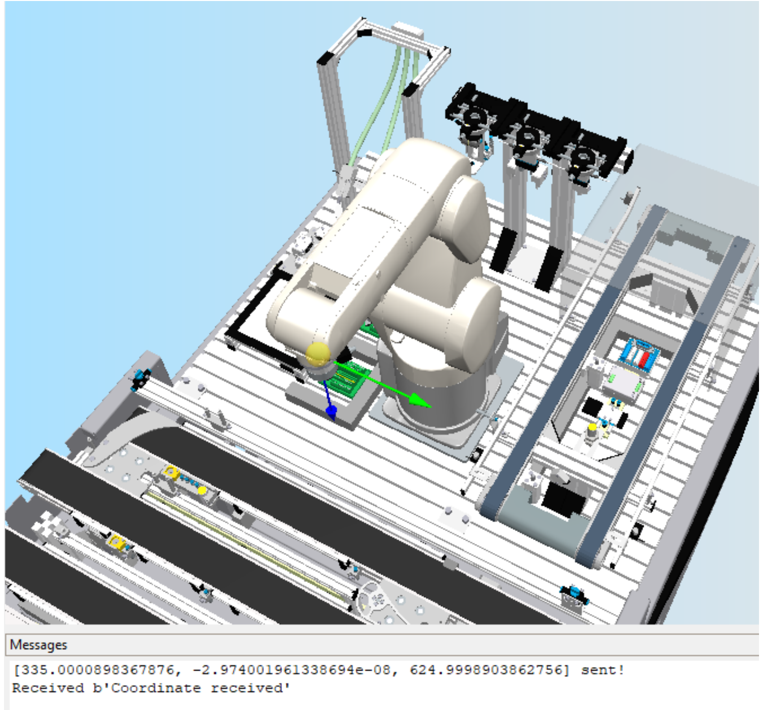
CPRobotinoConstructionHelper	Construct Robotino and related modules
CPSystemConfigurationHelper	This can be used to change the configuration of certain modules. Please note that u need to give the correct object name as it appears in the model, for example CP-L-BRANCH_1 for the second added branch module.

- **Note:** This library does not come in default with the installation. It might be found in the CIROS project configured by Festo Didactic in folder python. If it is not found, please contact us to request for it.

Use Case: Common TCP/IP Communication

Tutorial: Send robot TCP to a TCP/IP server.

- At the end of this tutorial, user is able to get a robot TCP in joint coordinate and send it to an external TCP/IP server using common TCP/IP communication protocol with a python script.
- The example will be demonstrated with a CP-F-RASS_Mitsubishi modell.



Use Case: Common TCP/IP Communication

1. Insert CP-F-RASS_Mitsubishi model in CIROS.
2. Create a python script named TcpIpClient.py.
3. Program the script as shown in next page.
4. Start a TCP/IP server.

   Example: With open source software ‘Hercules’.
5. In CIROS, select Extra \ Python \ Execute.
6. Select the python script.
7. Cartesian coordinate is received in the server.
8. Send a reply to close the connection.

Use Case: Common TCP/IP Communication

Python script

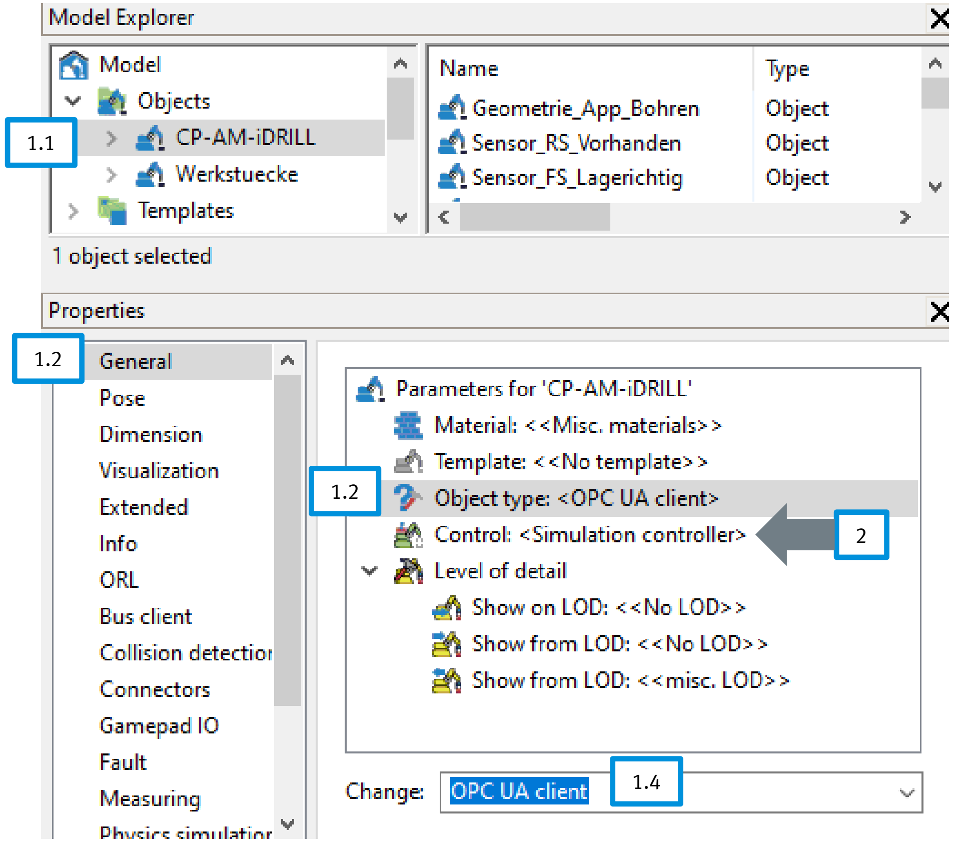
```
#CIROS pragma: concurrent

import Ciros2 as ciros
import CirosUtil
import socket

def TcpIpClient(msg):
    HOST = "127.0.0.1"
    PORT = 9000
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        s.sendall(msg.encode('utf-8'))
        data = s.recv(1024)
    print(f"Received {data!r}")
```

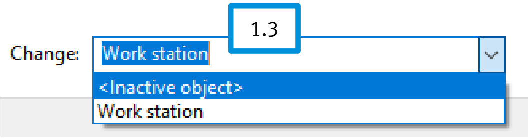
```
def robotTCP():
    rass = ciros.Object('Montage_RV-4FL')   #Find the object
    flange = rass.getGripperpoint('Flange') #Find the gripper
    coord = flange.getFrame(origin='OBJECT').getTranslation()  #Get translation
coordinate of gripper / TCP of robot
    return str(coord)

TcpIpClient(robotTCP())
```

OPC UA Interface

CIROS as OPC UA Client

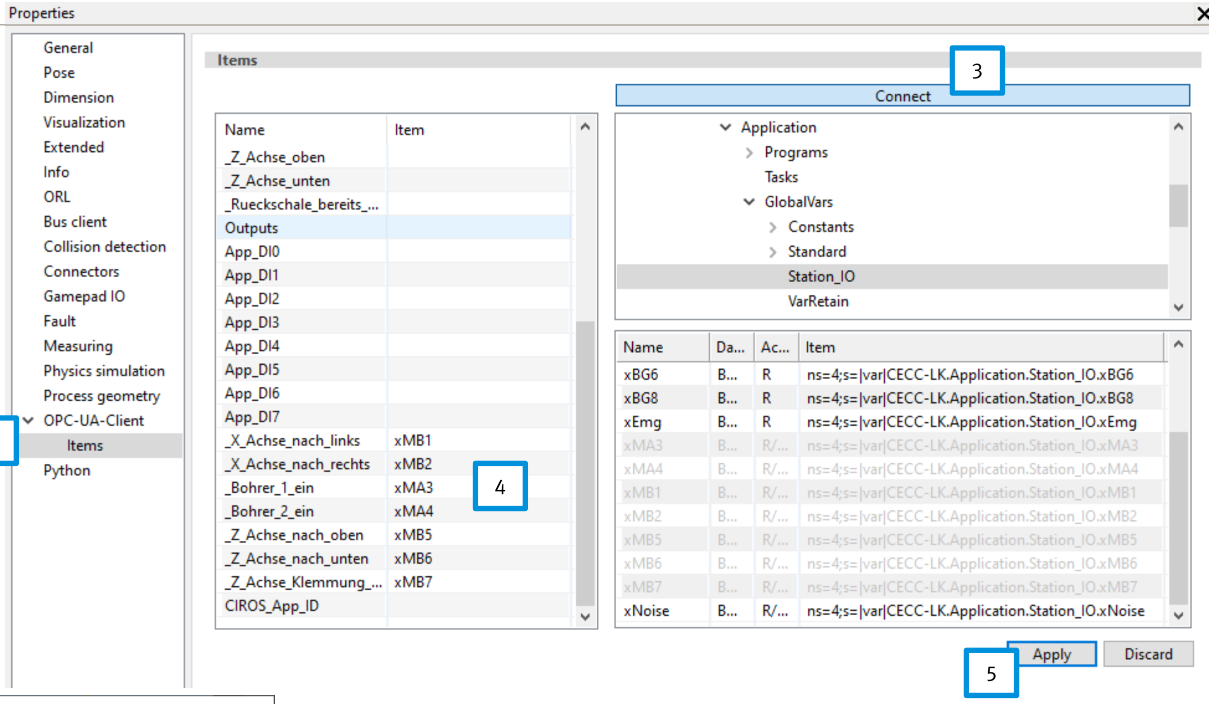
1. Change controller object type to OPC UA client.
    1. Select the object in Model Explorer.
    2. In Properties window, select General → Object type.
    3. First, change object type to inactive object to clear filter.
    4. Then, all the available object types will be shown, select OPC UA client.
2. Change controller control to Simulation controller.

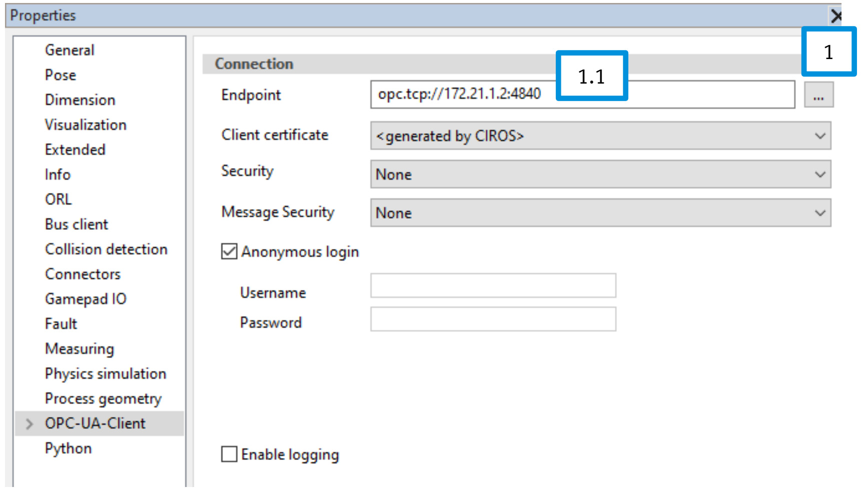


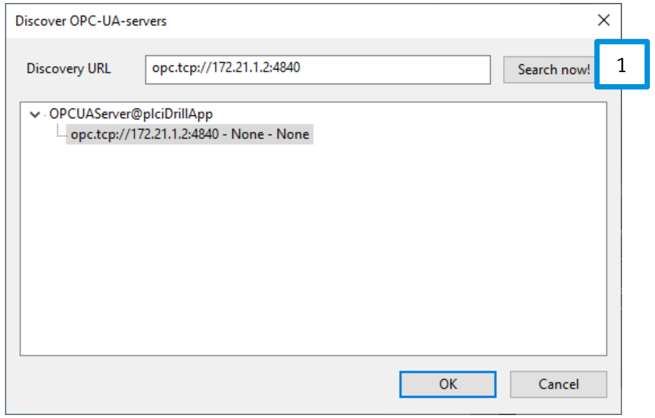


CIROS as OPC UA Client

1. In Properties → OPC-UA-Client, search for the endpoint of OPC UA server.
   1. If endpoint did not change, you can type the URL directly in the field.
2. Expand OPC-UA-Client and select Items.
3. Click on Connect.
4. Assign the OPC UA nodes in server to the internal IOs.
5. Select Apply.







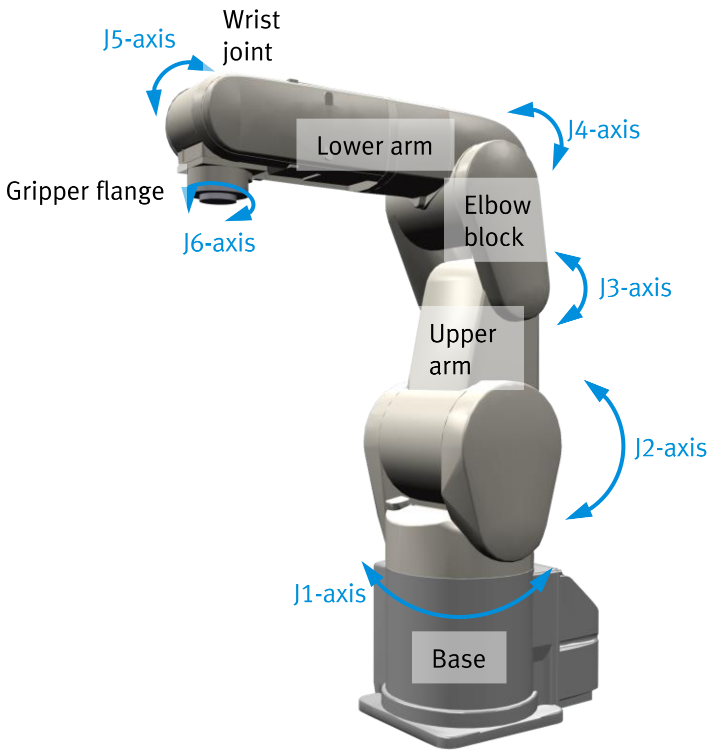
Robot Programming

Mitsubishi

Mitsubishi Industrial Robot

Electric RV-4FL robotic arm

Type	Articulated robot
Number of axes	6
Ultimate load	4 kg
Maximum reach radius	649 mm
Movement range	480° / 240° / 164° / 400° / 240° / 720°
Maximum composite speed	9048 mm/s
Cycle time	0.36 s
Position repeatability	± 0.02 mm
Weight	41 kg
Tool wiring	8 I/O
Protection rating	IP67



Mitsubishi Industrial Robot

Robot controller CR750-D

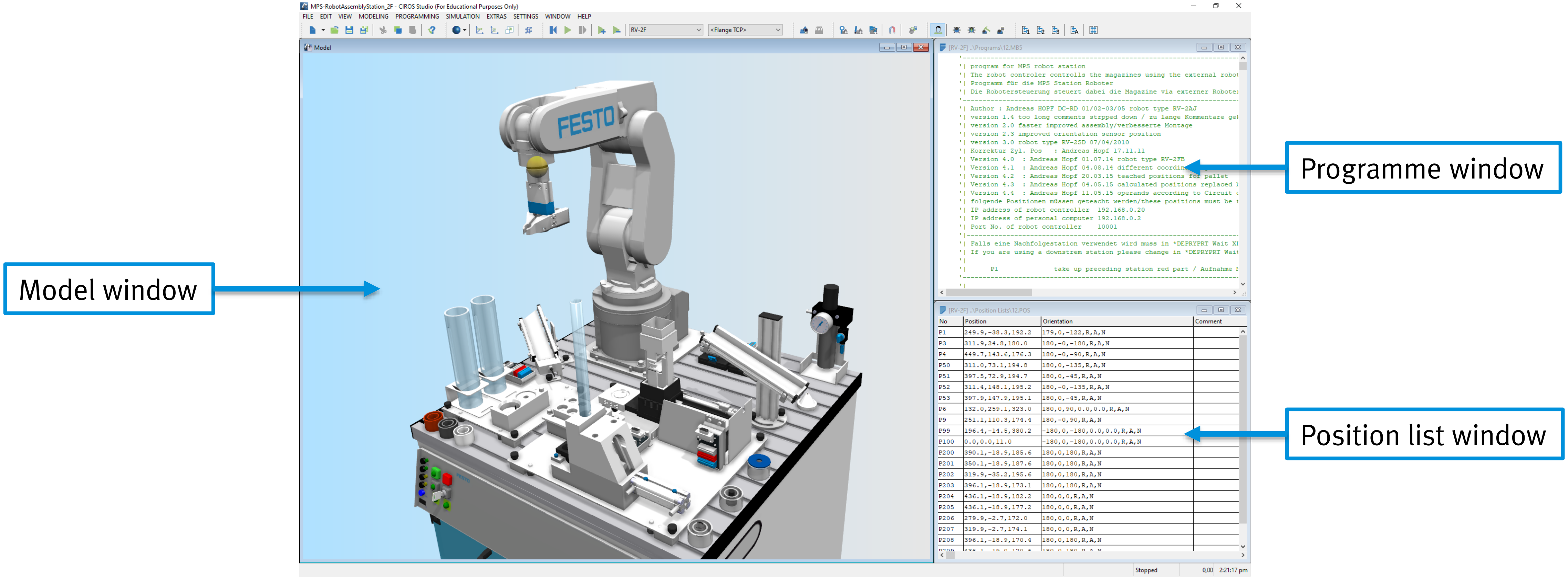
Programming language	MELFA-BASIC-V
Number of programs	512
Positions / program	3900
Programming	Teach box / PC
Power supply	Single-phase 180 – 253 V AC
Interface	RS422 / ethernet / USB / digital I/O
Dimensions (H x W x D)	430 mm x 425 mm x 174 mm
Weight	16 kg
Protection rating	Ground position / IP20

Mitsubishi Industrial Robot

Teach box R56TB

Menu navigation (language)	German, English, French, Italian
Features	Operating, programming and monitoring all robot features
Programming and Monitoring	Reading out information even during the running system; Programming using a virtual keyboard; Display of up to 14 lines of programming code; I/O Monitoring of up to 256 inputs and 256 outputs; Maintenance display of service intervals; Trouble indication of the last 128 alarms
Display	Touchscreen with background lighting 6,5" TFT display (640 x 480 pixel), 65536 colours
Interface	USB, combined RS422 and ethernet interface
Connection	Direct connection to the robot controller, cable length 7m
Protection rating	IP65
Weight	1,25 kg

Layout and Windows

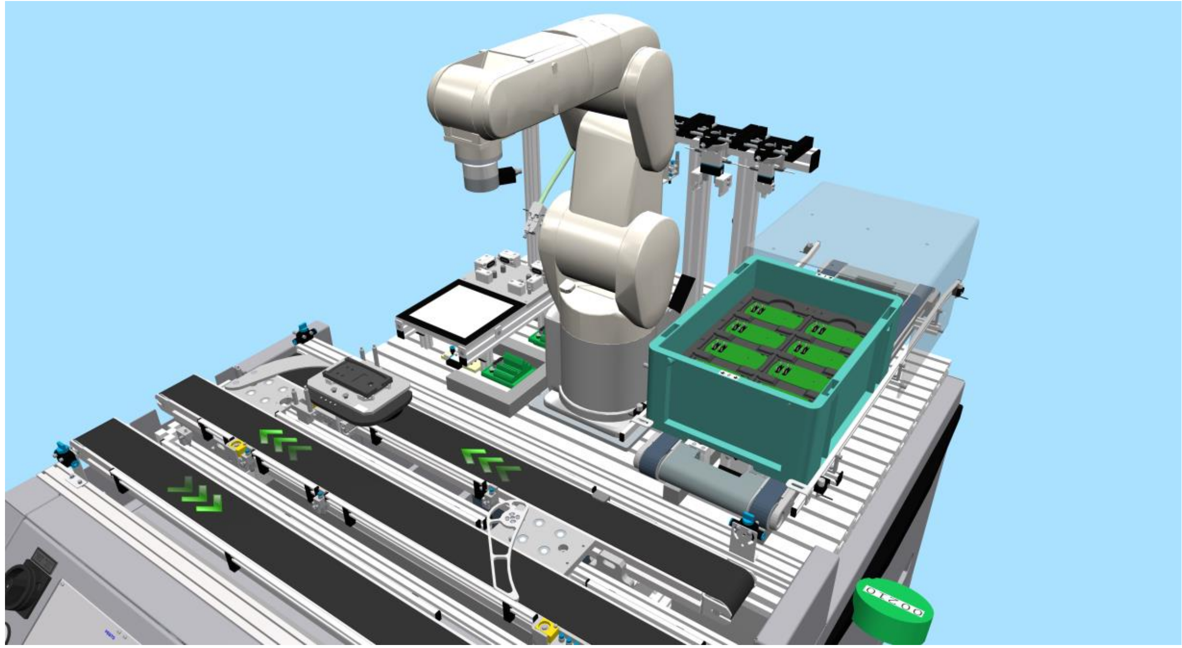


CP-F-RASS

Robot simulation

- RASS stands for Robot ASsembly Station.
- It is possible to simulate the Mitsubishi robot program in CIROS environment.
- The robot program requires several input parameters, which usually comes from MES4 or PLC. For standalone robot simulation in CIROS, user has to provide the input parameters manually.
- Following are the input parameters are required for standalone CIROS robot simulation:

1. Program number.
   - Program 1 : Assemble PCB
   - Program 2 : Assemble PCB and front fuse
   - Program 3 : Assemble PCB and rear fuse
   - Program 4 : Assemble PCB and both fuses
2. Position in box.



CP-F-RASS

General information

- **Attention:** General knowledge of Mitsubishi robot and robot programming are required to proceed in this chapter. Please be informed about Mitsubishi robot program structure, programming language, and I/O connections before proceeding.
- The robot programs are written in Melfa Basic V (MBA5) with extension .mb5, which is the programming language of Mitsubishi robot.
- Positions are stored in the position list with extension .pos.
- There is a robot project in CP-F-RASS model, which have the same programs as in actual robot. However, the I/O channels and tool changing mechanism are modified to suit the modelling environment.
- In CIROS simulation, a main program RobotSystemDriver.mb5 is used to simulate the slot allocation in actual robot controller.
- Generally, a program name can be any string characters. However, PLC only calls the program by integers. To allow the new program to be able to function in real robot, the program name should be numbers.

CP-F-RASS

Programs list

RobotSystemDriver.mb5	Main program. Assign subprograms to slot.
999.mb5	Reset robot
1.mb5	Assemble PCB to front cover
2.mb5	Assemble PCB and front fuse to front cover
3.mb5	Assemble PCB and rear fuse to front cover
4.mb5	Assemble PCB and both fuses to front cover
5.mb5	Demo program. Assemble and disassemble PCB in front cover.
123.mb5	Check all positions in PCB box. Assemble and disassemble all PCBs in box.
234.mb5	Camera test program. Pick front cover from stopper, place to vision field and run camera. Repeat four times.
255.mb5	Calculate positions in box based on four positions taught
ENERGSAVEVACU.mb5	Switch on vacuum gripper when workpiece is loose
GETCAMRESULT.mb5	Get camera result
GETCURTOOLNO.mb5	Get current tool number
GETFUSEMAGNO.mb5	Get fuse magazine number
GRPCLOSE.mb5	Close gripper
GRPLOCK.mb5	Lock the gripper to robot arm flange
GRPOPEN.mb5	Open gripper
GRPRELEASE.mb5	Release gripper from robot arm flange

GRPVACOFF.mb5	Switch off vacuum gripper
GRPVACON.mb5	Switch on vacuum gripper
INITIALIZE.mb5	Initialize input and output variables
MONITORHOME.mb5	Check if robot arm is at home position
MONITORPALWS.mb5	Check if robot arm is at PCB box position
MOUNTBOTFUSE.mb5	Mount rear fuse
MOUNTPCB.mb5	Mount PCB to front cover
MOUNTTOPFUSE.mb5	Mount front fuse
MOVHOME.mb5	Move robot arm to home position
PCBTRAYCNTRL.mb5	Check PCB box lock signal and PCB position count
PICKFRMSTOPR.mb5	Pick workpiece from stopper
PICKFRMVISION.mb5	Pick workpiece from vision field
PICKFUSFRMAG.mb5	Pick a fuse from fuse magazine
PICKNEWTOOL.mb5	Pick a new gripper
PICKPCBFRPAL.mb5	Pick PCB from PCB box
PICKWPFROMASS.mb5	Pick workpiece from assembly position
PLACETOSTOPR.mb5	Place workpiece to stopper.
PLACETOVISION.mb5	Pick workpiece from stopper to vision field
SENSORCHECK.mb5	Check sensors at stopper, assembly position, fuse magazines and input parameter for PCB position in box.
SENSORCHECK1.mb5	Check sensors at stopper, assembly position and input parameter for PCB position in box.
SENSORCHECK6.mb5	Check sensors at stopper position
UBP.mb5	User base program. Global program contains all global variable, flags and positions.

CP-F-RASS

Controllers and I/Os

- Station CP-F-RASS has two controllers,
  - PLC controller
  - Robot controller
- In CIROS, the station has two controllers as well.
  - SPS_Roboter is the virtual representation of PLC controller
  - Montage_RV-4FL is the virtual representation of robot controller
- The I/Os are linked internally in CIROS model.

Montage_RV-4FL	Index	Type	SPS_Roboter	Index	Type	Description
I_Stop	100	DI	DOUT_100_x0	40	DO	Stop robot program
I_Start	101	DI	DOUT_100_x1	41	DO	Start a robot program
IDATA_(0-15)	116 – 131	16 Bit DI	AOUT_W102	002	AO	Program number in binary
DI_PCBPalletNo_(0-7)	172 – 179	8 Bit DI	AOUT_B109	004	AO	Position in PCB box
HOpen_1	900	DO				Open gripper
HClose_1	901	DO				Close gripper
HOpen_3	904	DO				Release gripper
HClose_3	905	DO				Lock gripper

CP-F-RASS

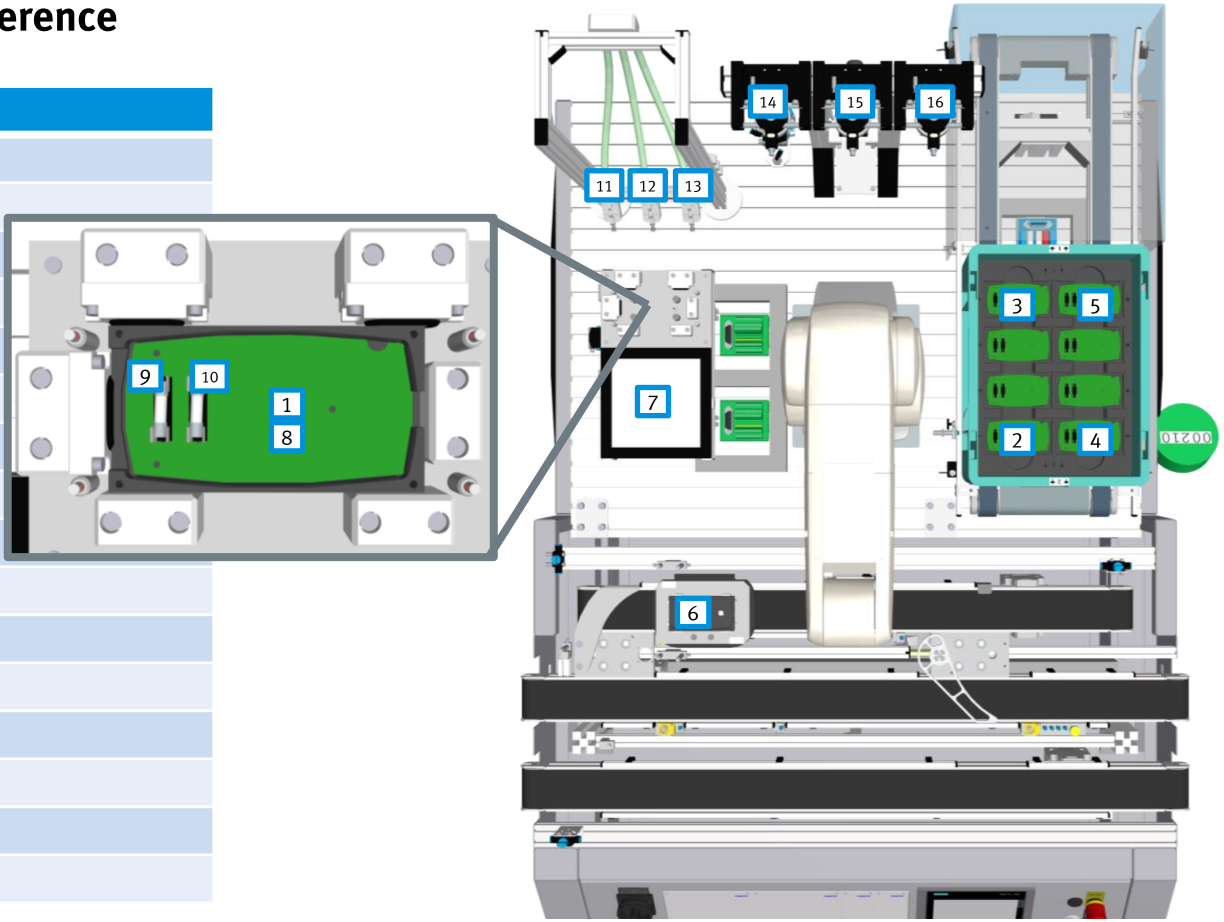
Grippers' tool number

- Each gripper has a tool number. It is important to select the right tool as different TCP is used for different tools.
  - Tool 1 : Vacuum gripper.
    - Z-offset relative to flange TCP 205 mm.
    - C-rotation relative to flange TCP 33.5°.
  - Tool 2 : Parallel gripper for front cover.
    - Z-offset relative to flange TCP 170 mm.
    - C-rotation relative to flange TCP 33.5°.
  - Tool 3 : Parallel gripper for fuse.
    - Z-offset relative to flange TCP 151.5 mm.
    - C-rotation relative to flange TCP 33.5°.
  - Tool 4 : No gripper.
    - Z-offset relative to flange TCP 0 mm.
    - C-rotation relative to flange TCP 0°.

CP-F-RASS

List of default positions in CIROS environment for reference

No	Definition	Position
1	P_AssemblePCB	(-140.03,-367.01,109.01,-179.73,0.09,89.95)(7,0)
2	P_PCBPalletOrigin	(93.00,276.00,213.00,-180.00,0.00,90.00)(7,0)
3	P_PCBPaletXDir	(-140.10,276.00,213.00,-180.00,0.00,90.00)(7,0)
4	P_PCBPaletYDir	(93.00,398.00,213.00,-180.00,0.00,90.00)(7,0)
5	P_PCBPaletXYDir	(-140.10,398.00,213.00,-180.00,0.00,90.00)(7,0)
6	P_CarrierStop1	(402.50,-267.00,171.50,179.73,-0.09,90.05)(7,0)
7	P_Vision	(30.00,-350.00,101.50,180.00,-0.00,90.00)(7,0)
8	P_AssembleWp	(-139.00,-362.00,114.00,-180.00,-0.00,90.00)(7,0)
9	P_AssembleFuse1	(-138.97,-401.18,127.13,-180.00,0.00,180.00)(7,0)
10	P_AssembleFuse2	(-138.97,-388.18,127.13,180.00,-0.00,180.00)(7,0)
11	P_FuseMagazine1	(-300.65,-430.30,160.30,177.86,-42.56,-177.96)(7,0)
12	P_FuseMagazine2	(-299.24,-360.21,159.46,177.87,-42.57,-177.96)(7,0)
13	P_FuseMagazine3	(-299.44,-290.40,159.99,177.87,-42.57,-177.96)(7,0)
14	P_GrpStorageVac	(-395.65,-125.00,484.90,-180.00,-0.00,-0.00)(7,1)
15	P_GrpStorageWp	(-395.65,0.00,484.90,-180.00,0.00,0.00)(7,1)
16	P_GrpStorageFuse	(-395.65,125.00,484.90,-180.00,-0.00,-0.00)(7,0)



Steps to configure CP-F-RASS for simulation

1. Create a new CIROS project.
2. Insert and snap following from CP System model libraries.
   1. CP-F-RASS (Mitsubishi)
   2. CP-F-SOURCE
   3. CP-F-SINK
3. Optional: Hide safety glass.
   1. In Model Explorer, choose Objects → CP-F-RASS_Mitsubishi → Geometrie → Geometrie_Umhausung
   2. In Properties, select Visualization.
   3. Click Invisible.
4. Open robot program in project management.
   1. In Project Management, right click on Projects and select Open.
   2. Select ‹project folder›\CF\Rob_Montage\RV-4FL\Montage_RV_4FL.prjx.
   3. In Project Management, select Controllers → Montage_RV_4FL
   4. Assign project Montage_RV_4FL to the controller.
5. Assign required I/Os to IO monitor.
   1. Open an I/O monitor window, it can be any I/O monitor.
   2. Drag the Outputs from Model Explorer to I/O monitor window.
   3. Required Outputs are in Objects → CP-F-RASS_Mitsubishi → SPS_Roboter → Outputs
      1. AOUT_W102 (analogue 002) : robot program
      2. AOUT_B109 (analogue 004) : position in PCB box
6. Change source part number to 210.
   1. In Model Explorer, select CP-F-SOURCE.
   2. In Properties → CP System, change Part Number to 210.

Note: The configured model can be saved as a template which can be opened with CIROS Studio and Education in robot programming lessons.

Video tutorial: 51_ConfigureRASSForRobotProgramSimulation.mp4

Steps to simulate CP-F-RASS

1. Optional: To reduce computing power, close all windows except model window and I/O monitor, e.g. Model Explorer, Properties, Project Management, etc.
2. Start simulation.
3. Give a robot program and position in box in I/O monitor and activate override.
    1. For example: robot program 1 and box position 3.
        - AOUT_W102 = robot program → Value = 1
        - AOUT_B109 = position in box → Value = 3
5. Press on green source button in model window.
6. Observe the program.

- To run another robot program, repeat step 3 to 6.
- To restart simulation, repeat step 2 to 6.

Video tutorial: 52_SimulateRASSRobotProgram.mp4

Simulate Real Robot Program in CP-F-RASS Model

- It is possible to simulate real CP-F-RASS robot program in CIROS.
- However, the program has to be modified to suit the simulation environment, for example, the I/Os address and tool changing mechanism.
- Besides, the simulated program does not connect to a camera. Thus, the subprogram which connects with camera has to be commented out.
- The modified robot program can be saved as a template project and be used repeatably in robot programming classes as it is portable with both CIROS Studio and CIROS Education.

Ready to use project: CP-F-RASS_RobotProgramming_v717

Simulate Real Robot Program in CP-F-RASS Model

Steps to modify robot program (1)

1. Create a new CIROS project.
2. Load and snap following models in place from CP System model libraries.
   1. CP-F-RASS (Mitsubishi)
   2. CP-F-SOURCE
   3. CP-F-SINK
3. Change Part Number of CP-F-SOURCE to 210.
4. Place all the real robot programs in a single folder and place the folder in CIROS project folder.
5. In the folder, delete following files.
   1. Files with type .bak.
   2. Files with type .prjx.
6. Copy following files from ‹project folder›\CF\Rob_Montage\RV-4FL to the robot program folder.
   1. RobotSystemDriver.mb5
   2. Montage_RV_4FL.prjx
   3. ENRGSAVEVACU.mb5, if not exist
   4. MonitorHome.mb5, if not exist
   5. MonitorPalWS.mb5, if not exist
   6. PCBTrayCntrl.mb5, if not exist
7. In CIROS, open Project Manager, open the copied project Montage_RV_4FL.prjx in robot program folder and assign it to controller Montage_RV-4FL.
8. In Project Manager, open Projects → Montage_RV_4FL (MBA5) → Files and delete the files which do not exist.

FESTO

Simulate Real Robot Program in CP-F-RASS Model

Steps to modify robot program (2)

9. In CIROS, change the following in all files in the project.
   1. Bits and bytes in I/O definitions

From	To
2032	132
2033	133
2040	140
2064	164

From	To
2072	172
2144	244
2147	247

From	To
2148	248
2150	250
2151	251

   2. Tool changing mechanism

From	To
M_Tool = m_GripperFuse	Tool P_tGripperFuse
M_Tool = m_GripperNone	Tool P_tGripperNone
M_Tool = m_GripperVac	Tool P_tGripperVac
M_Tool = m_GripperWP	Tool P_tGripperWP
HOpen 6	HOpen 3
HClose 6	HClose 3

   3. Make following changes.
      1. Add following lines in 999.mb5.

```
P_tGripperVac=(0,0,205,0,0,33.50)
P_tGripperWP=(0,0,170,0,0,33.50)
P_tGripperFuse=(0,0,151.5,0,0,33.50)
P_tGripperNone=(0,0,0,0,0,33.50)
```

      2. Comment out or delete all position declarations in UBP.mb5 and add following lines.

```
Def Pos P_tGripperVac
Def Pos P_tGripperWP
Def Pos P_tGripperFuse
Def Pos P_tGripperNone
```

      3. Change following positions to reference positions in UBP.pos. The location of tool magazine in CIROS model is different from actual robot. Thus, the offsets are too large to be ignored.

```
P_GrpStorageVac = (-395.65,-125.00,484.90,-180.00,-0.00,-0.00)(7,1)
P_GrpStorageWp = (-395.65,0.00,484.90,-180.00,0.00,0.00)(7,1)
P_GrpStorageFuse = (-395.65,125.00,484.90,-180.00,-0.00,-0.00)(7,0)
```

   4. Comment out or remove all lines calling camera related subprograms, for example the line calling GetCamResult in program 1 to 5.mb5.

```
REM CallP "GetCamResult", CamPrgNumber%
```

10. Save all and compile the project.

CP-F-RASS Robot Programming

Robot programming example ‘gripper test’ (1)

- The program is independent of all the other robot programs, but uses positions in UBP.pos.
- It is written in MBA V and does the following.
  1. Move robot arm to home position.
  2. Move robot to parallel workpiece gripper magazine.
  3. Mount the gripper.
  4. Remove gripper from magazine.
  5. Open gripper.
  6. Close gripper.
  7. Store gripper back to magazine.
  8. Move robot arm back to home position.

Note: For more programming example, see tutorial video: 50_RASS-Programming.mp4.

CP-F-RASS Robot Programming

Robot programming example ‘gripper test’ (2)

1. Create a CIROS project and load CP-F-RASS from CP System model libraries.
2. Make sure a project is assigned to controller Montage_RV-4FL and UBP.pos is in the project.
3. In Project Management, right click on Projects → ‹project name› → Files and choose new.
4. Create a Melfa Basic V program and name it RASS-GripperTest.mb5.
5. With the programming window being the active window, select Programming → Programming assistant.
6. Uncheck Declare inputs and outputs and click OK.
7. In the programming window, add the lines shown in right.
8. Save the program.
9. In Project Management, right click on the program and select Set main program.
10. Compile the project.
11. Run simulation.

```
' TODO add your code here

' Move to home position
MOV P_Home
DLY 1

' Mount parallel workpiece gripper
MOV P_PCBPalletHelp
MOV P_GrpStorageWp, -30
JOVRD 50
MVS P_GrpStorageWp
HClose 3
MVS P_GrpStorageWp + P_ToolX80
DLY 1

' Open and close gripper
HClose 1
HOpen 1
DLY 1
HOpen 1
HClose 1
DLY 1

' Store gripper back to magazine
MVS P_GrpStorageWP
HOpen 3

' Move back to home position
MVS P_GrpStorageWp, -30
JOVRD 100
MOV P_PCBPalletHelp
MOV P_Home

END
```

Move Robot Manually

- There are several ways to move robot manually in CIROS.

1. Move the robot directly to a position on position list.
   - Double click on the position.

[RV-2F] ..\Position Lists\12.POS

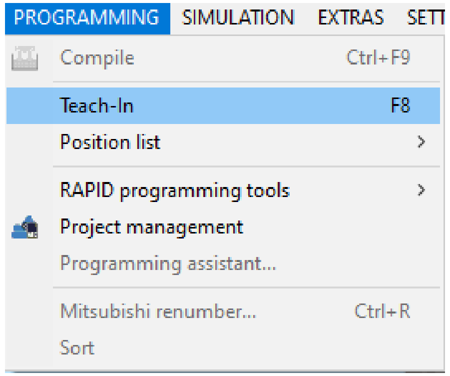
No	Position	Orientation	Comment
P1	249.9,-38.3,192.2	179,0,-122,R,A,N	
P3	311.9,24.8,180.0	180,-0,-180,R,A,N	
P4	449.7,143.6,176.3	180,-0,-90,R,A,N	
P50	311.0,73.1,194.8	180,0,-135,R,A,N	
P51	397.5,72.9,194.7	180,0,-45,R,A,N	
P52	311.4,148.1,195.2	180,-0,-135,R,A,N	
P53	397.9,147.9,195.1	180,0,-45,R,A,N	
P6	132.0,259.1,323.0	180,0,90,0.0,0.0,R,A,N	

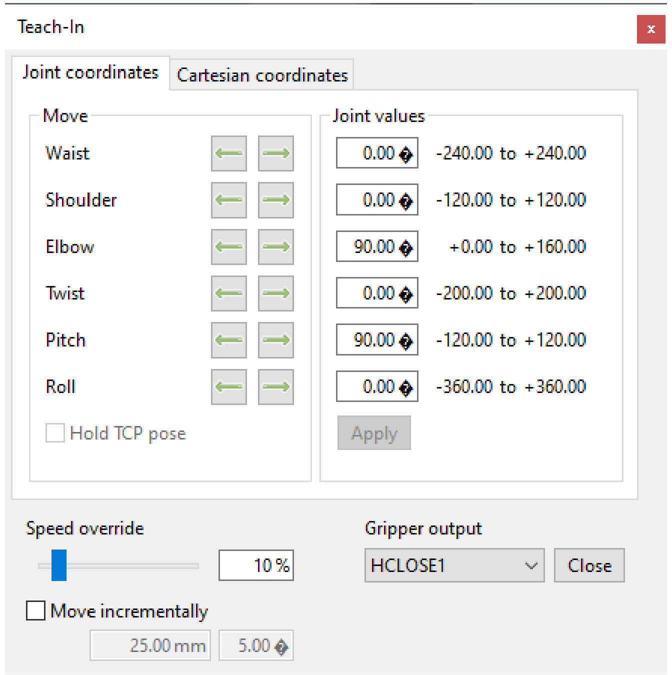
2. Double click on any location in model window.

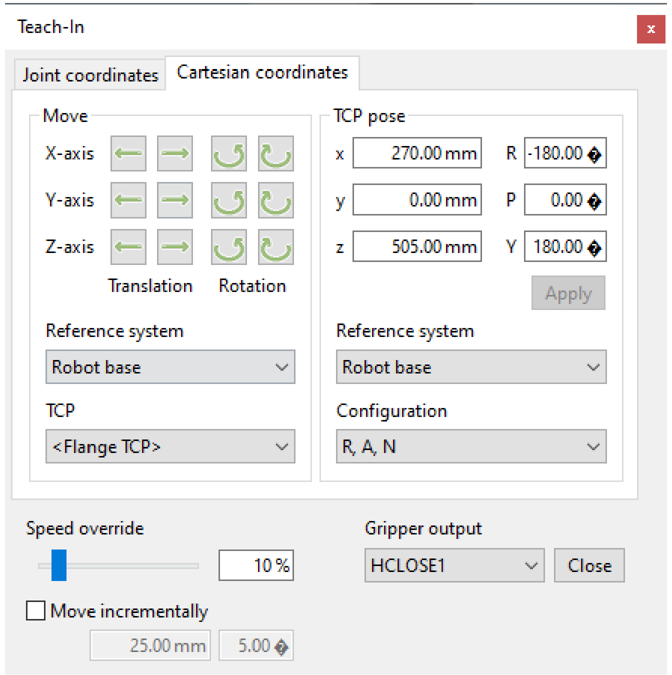
FESTO

Move Robot Manually

3. Move the robot with Teach-In panel.
   - Gripper can be controlled in section Gripper output.







Mount and Release a Gripper Manually

- In some system, for example, CP-F-RASS, TCP changes relative to the gripper.

- Steps to mount a gripper manually with example CP-F-RASS model.

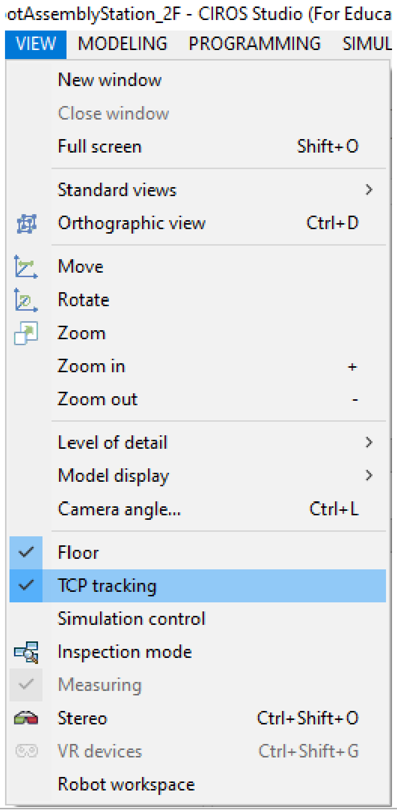
1. Select ‹Automatic TCP›.
2. Move the robot arm to the tool position.
3. In Teach-In panel → Gripper output, close HClose_3.
4. Start simulation (F5).
5. Stop simulation (F5).
6. The gripper is mounted.

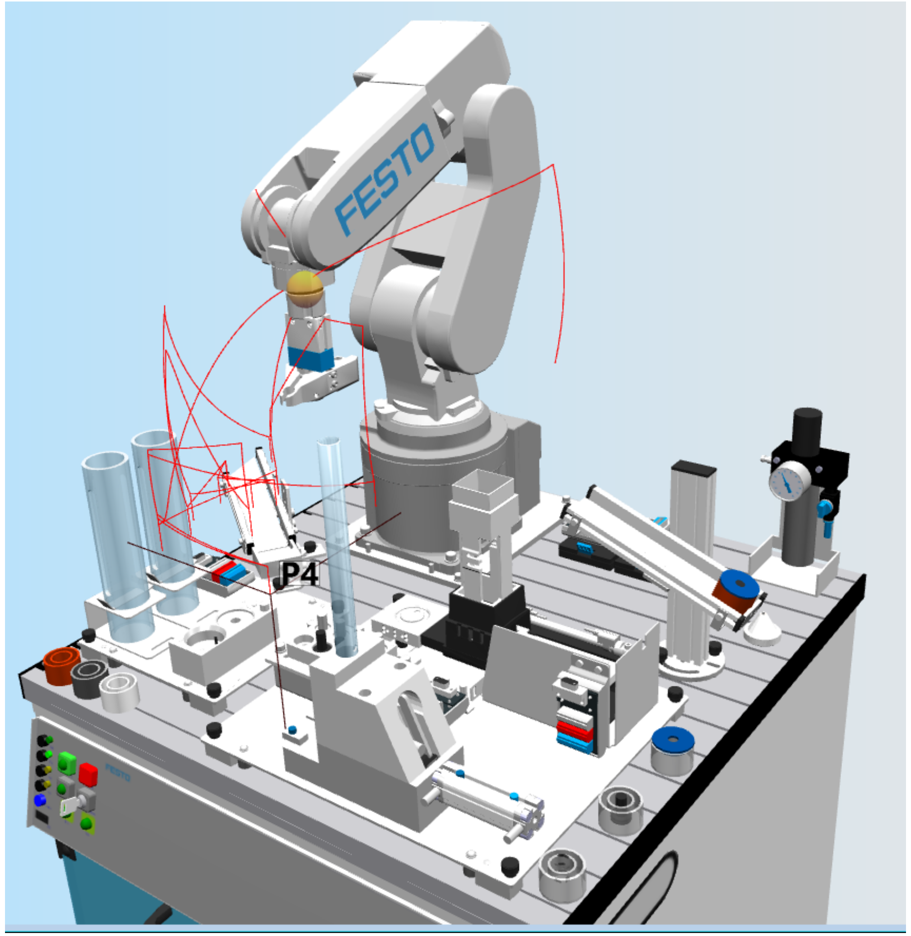
- Steps to release a gripper manually with example CP-F-RASS model.

1. Move the robot arm to the desired position.
2. In Teach-In panel → Gripper output, open HClose_3.
3. Start simulation (F5).
4. Stop simulation (F5).
5. The gripper is released.

TCP Tracking

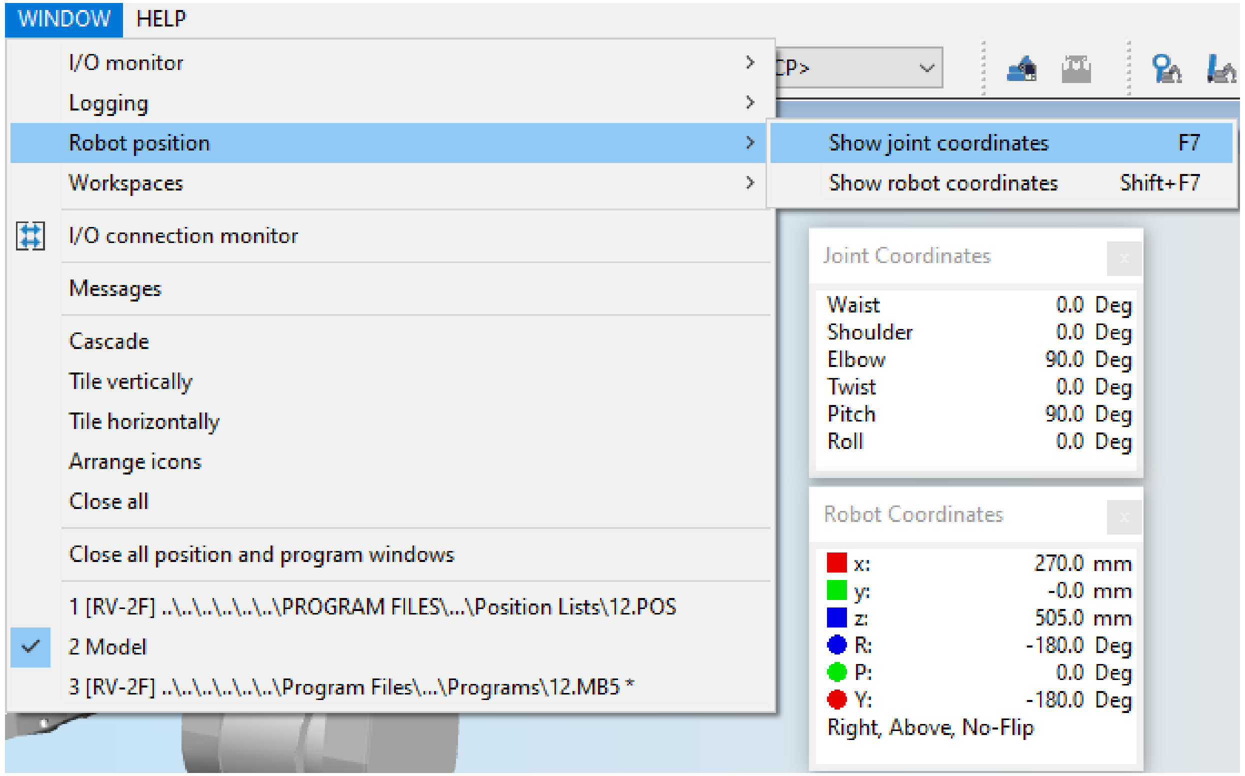
- TCP path of robot movement can be monitored.
- Active TCP tracking from View → TCP tracking.





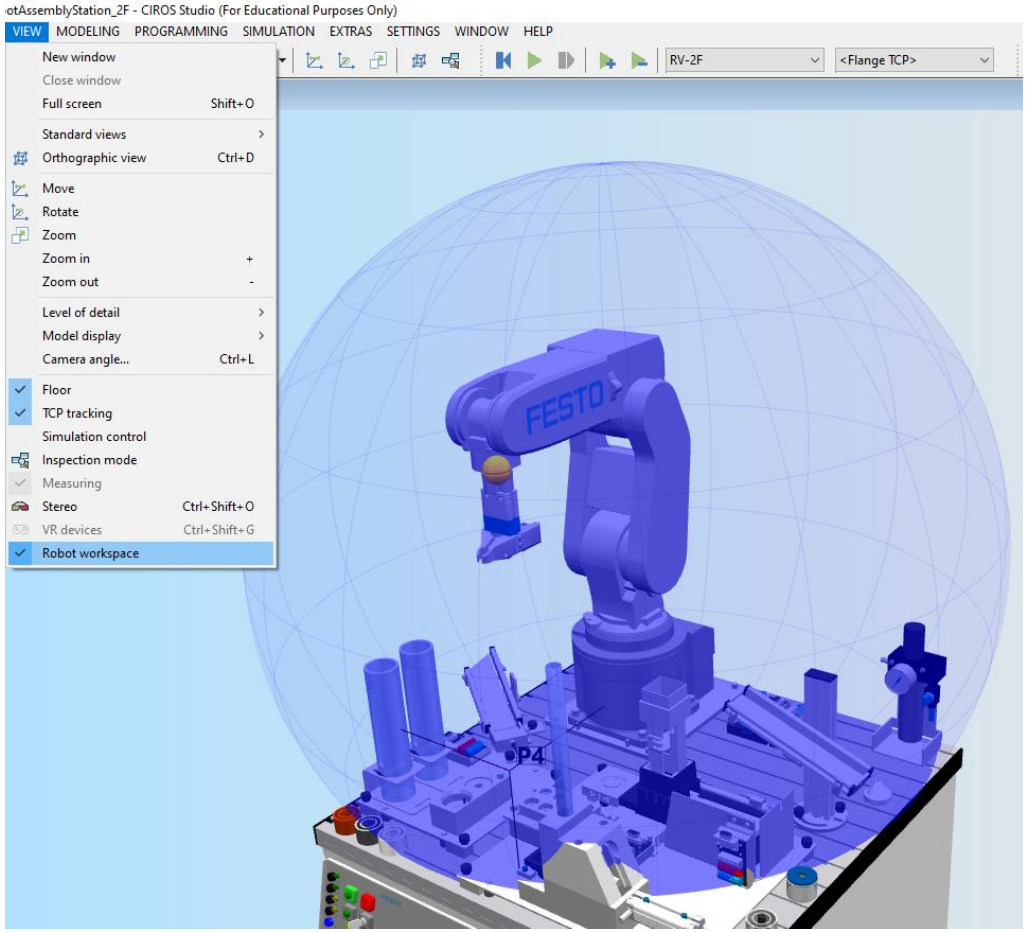
View TCP Coordinate

- Joint coordinate and cartesian coordinate of the active TCP can be monitored.



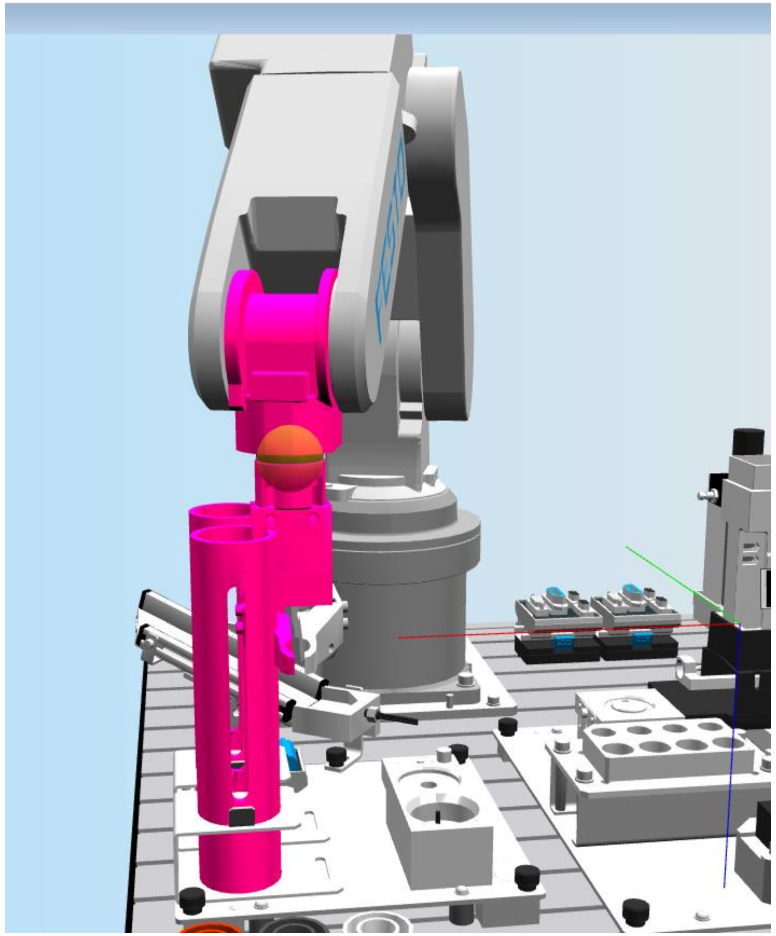
Robot Workspace

- The room that TCP can reach.
- Activate at View → Robot workspace.



Collision Detection

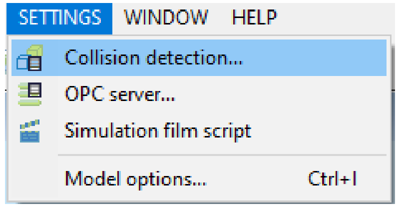
- Collision detection can be activated.
- In collision detection mode the movements are always incremental.
- It is useful in testing a new robot program to avoid collision.



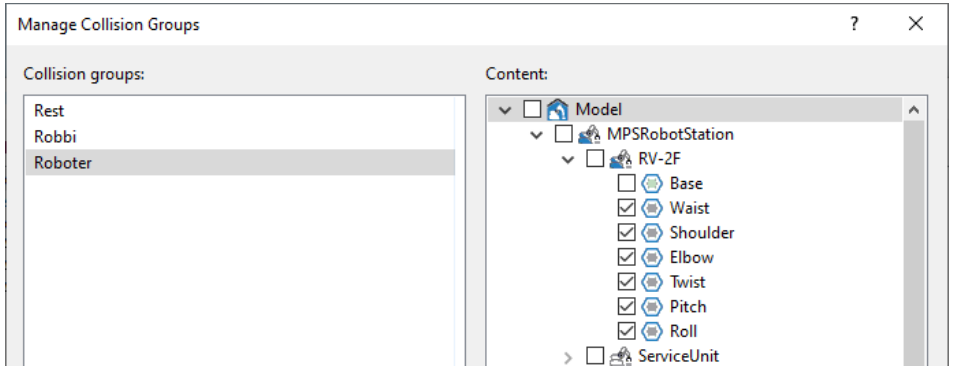
Collision Detection

Configuration (1)

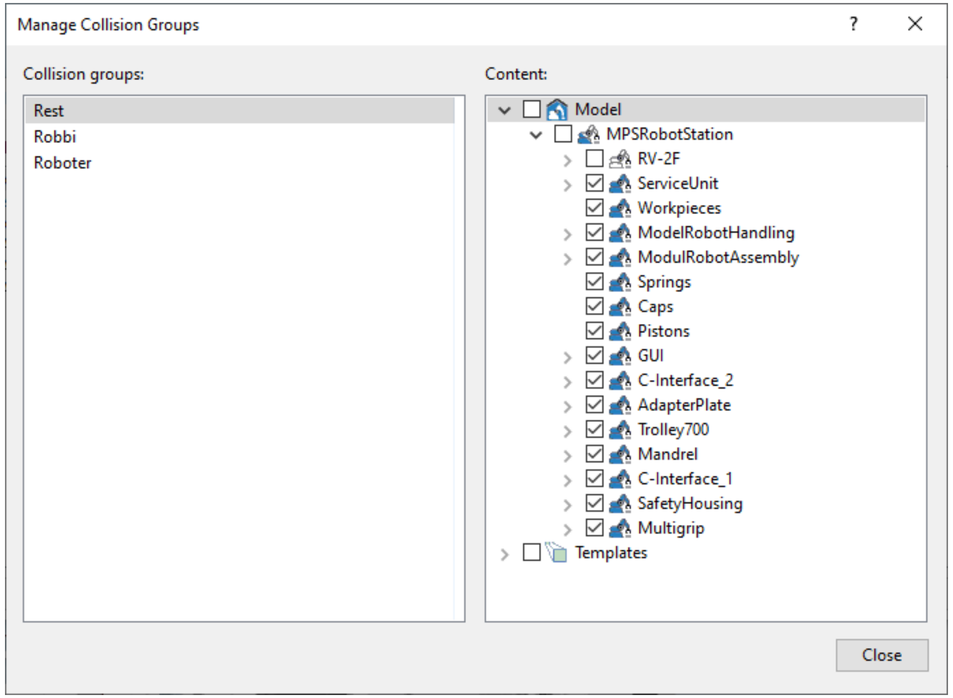
1. Assign group. Collision is only detected when elements in deferent groups cross each other.
    1. Select Settings → Collision detection.



    2. In Collision Detection window, select Manage collision groups.
    3. Assign content to Roboter group.



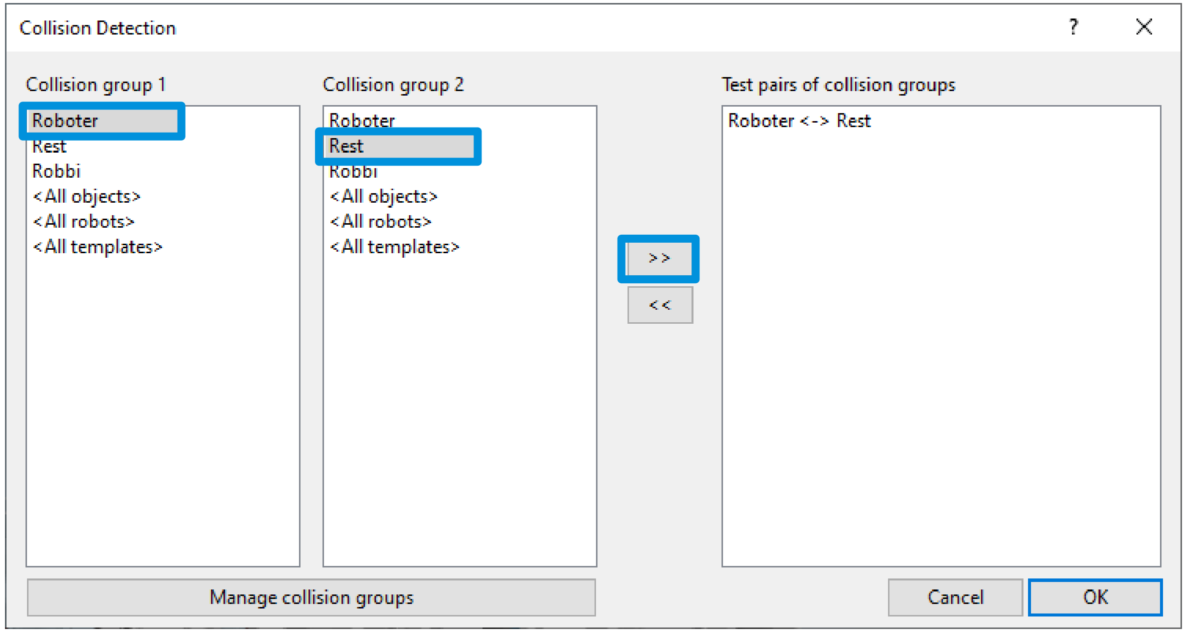
4. Assign content to Rest group.



Collision Detection

Configuration (2)

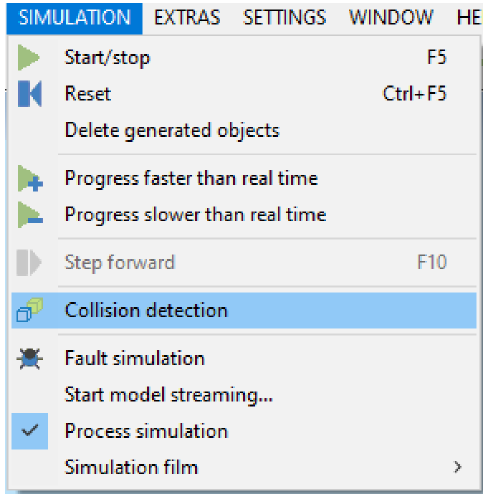
2. Assign the collision group pair.
    1. Close Manage collision groups window.
    2. Select Roboter as group 1 and Rest as group 2.
    3. Move the pair to the right to activate it.
    4. Select OK.



Collision Detection

Activate simulation

1. Select Simulation → Collision detection.

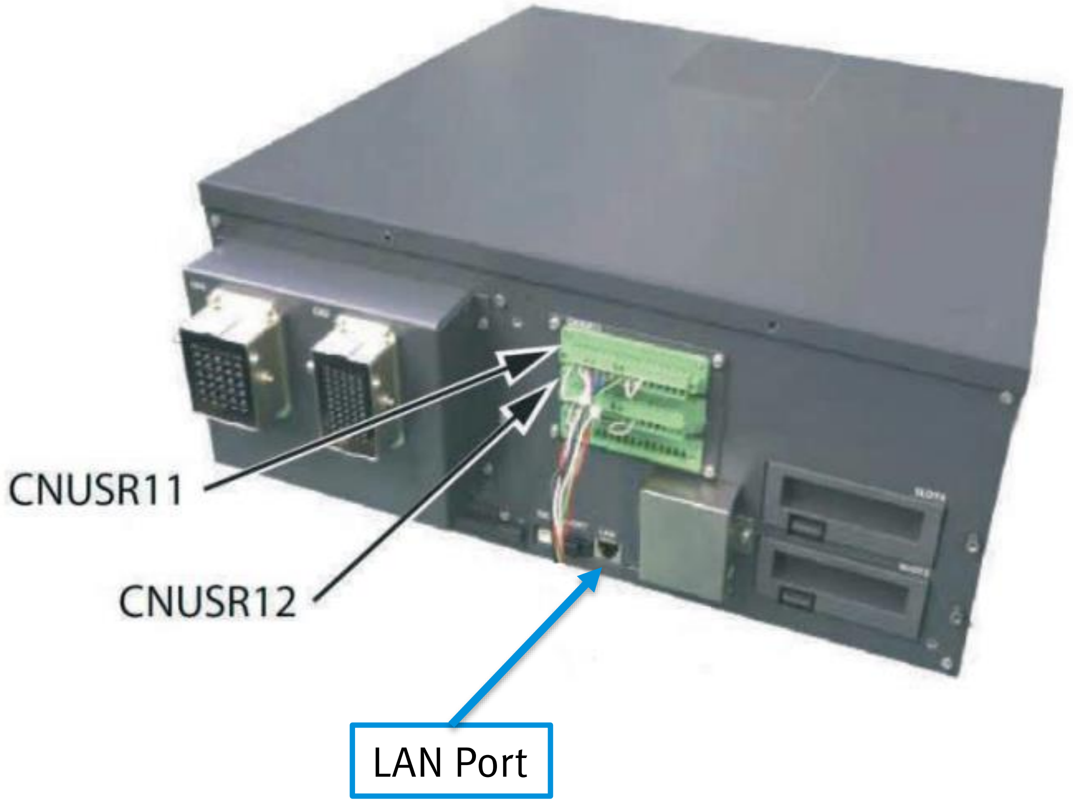


2. Move the robot against an object and observe the simulation.

Connect to Robot Controller

Communication interface

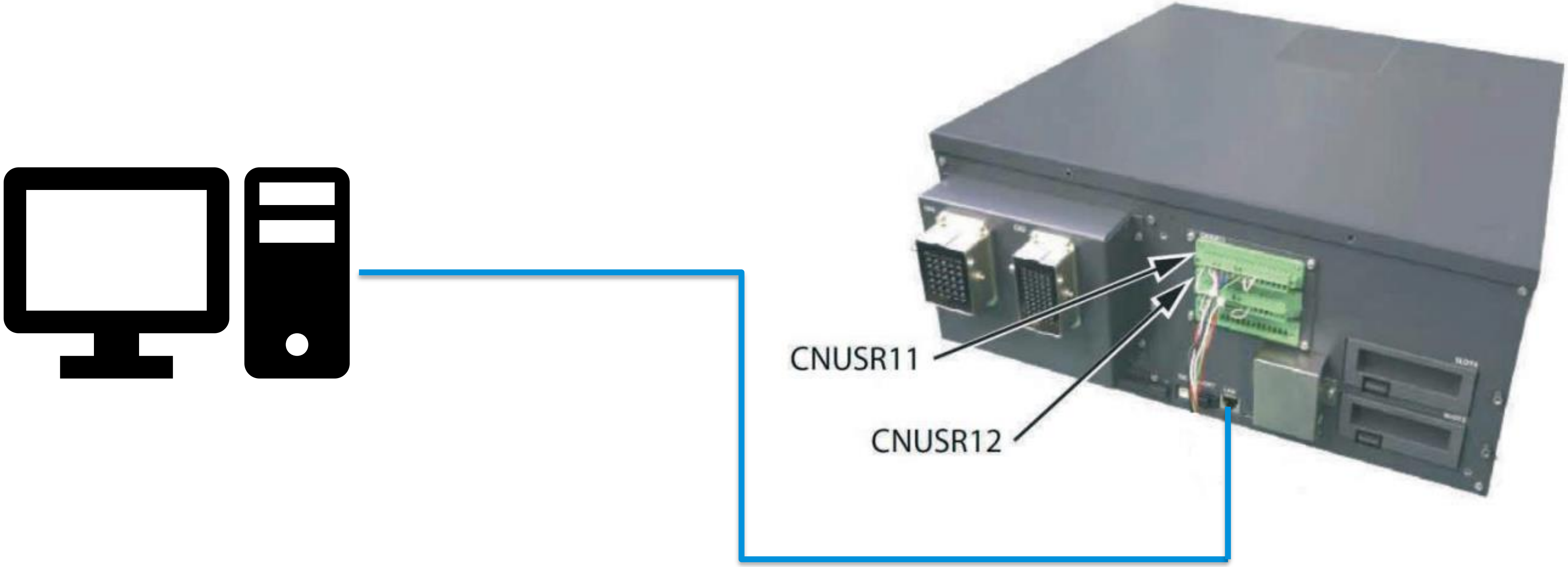
- Ethernet cable
- TCP/IP protocol
- IP-Address and Port



Only CIROS Studio

Check the Ethernet Cable

Please check the ethernet connection between robot controller and the computer.

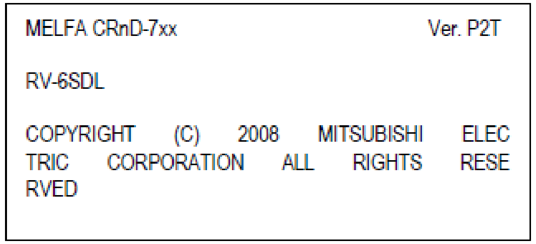


Connect to Robot Controller

Find the IP-address from R32TB

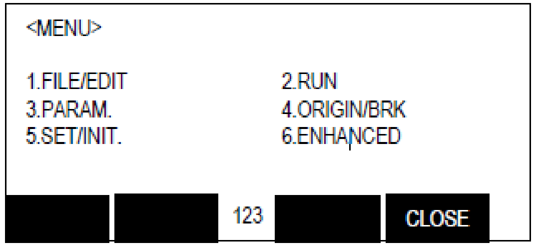
1. The IP-address can be read from robot teach box (TB).

2. Exit everything until home page.

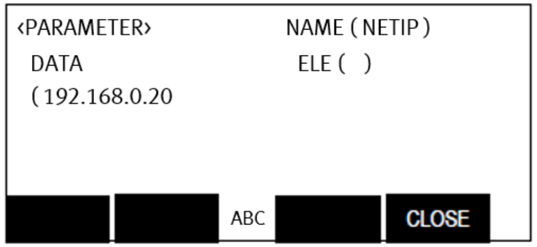


3. Press F1 to enter menu.

4. Select **3.PARAM.**



5. Search for parameter **NETIP**. Read the IP-address.



FESTO

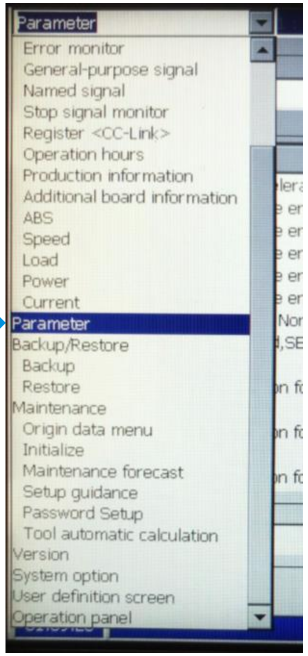
Only CIROS Studio

Connect to Robot Controller

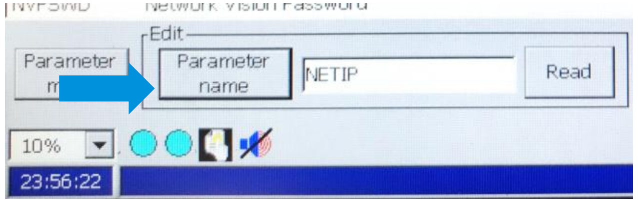
Find the IP-address from R56TB

1. The IP-address can be read from robot teach box (TB).

2. Select “Parameter” from the menu.

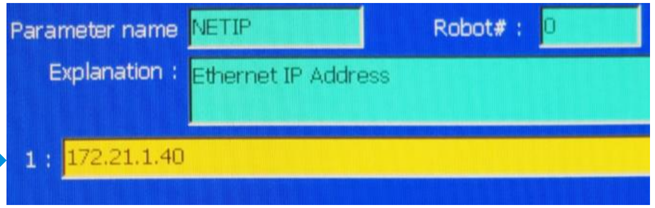


3. Click on “Parameter name” to search for parameter “NETIP”.



4. Key in “NETIP” and click “Enter”.

5. IP Address is shown.

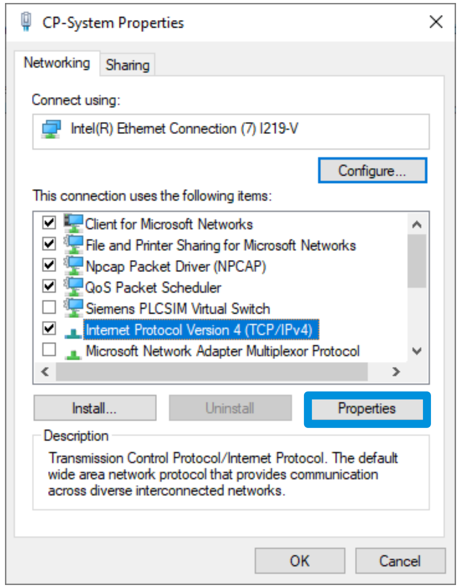


Video Tutorial: 53_FindMitsuIpR56TB.mp4

Connect to Robot Controller

Change the IP-address of the computer to the same network

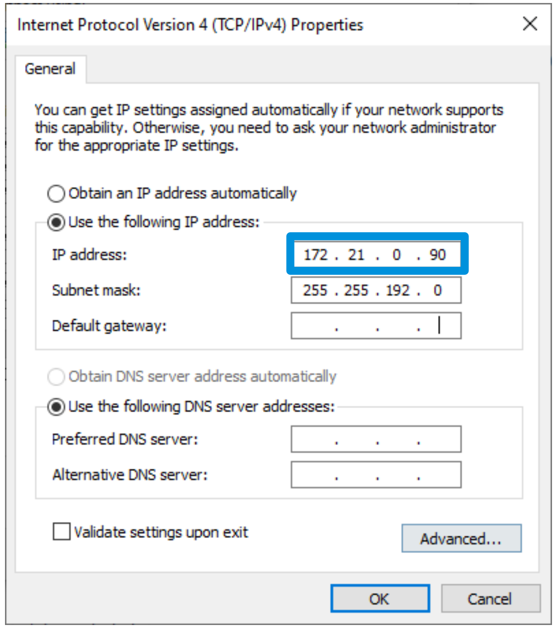
1. Open network adapter properties.



2. Select Internet Protocol Version 4 (TCP/IPv4).

3. Select Properties.

4. Change the IP address to the same network as robot controller.



FESTO

Only CIROS Studio

Connect to Robot Controller

Set robot controller to Automatic and switch of teach mode.

Only CIROS Studio

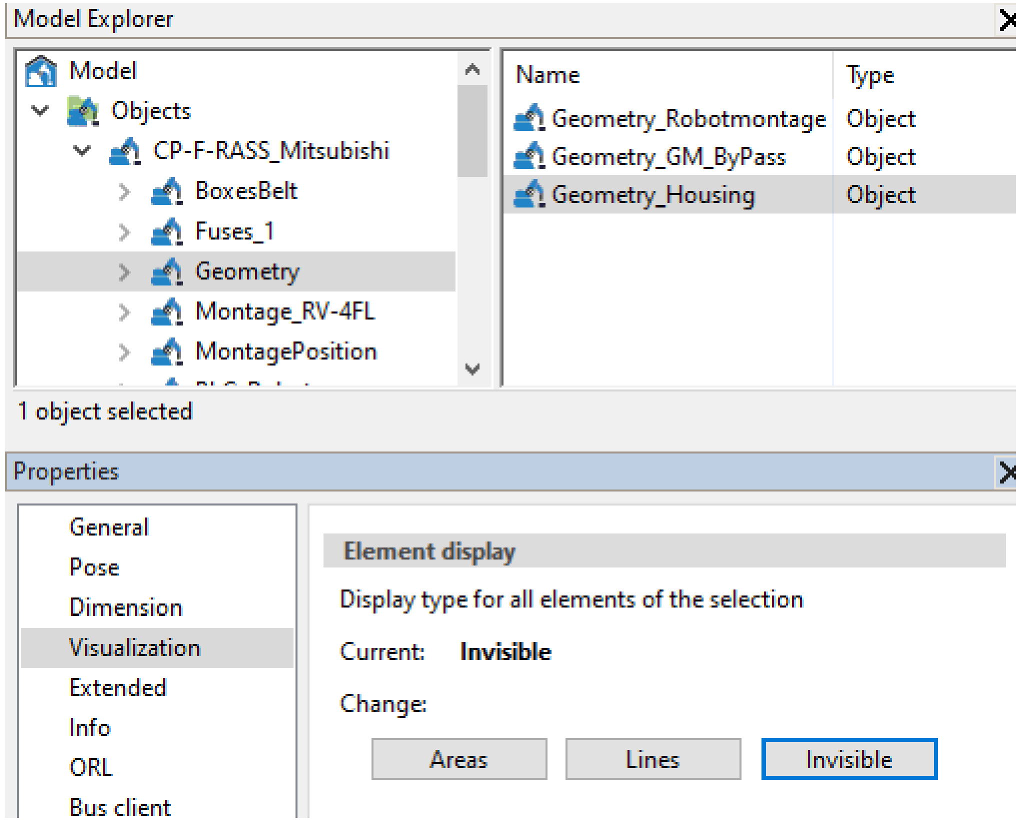
Connect to Robot Controller

Insert CP-F-RASS model

1. Create a new CIROS project.
2. Insert “CP-F-RASS_Mitsubishi” from Festo CP System model library.

   Video tutorial: 54_InsertCP-F-RASS.mp4
3. Optional: To hide robot housing, in Model Explorer, select “Geometrie\Geometrie_Umhausung”. In Properties\Visualization, select “Invisible”.

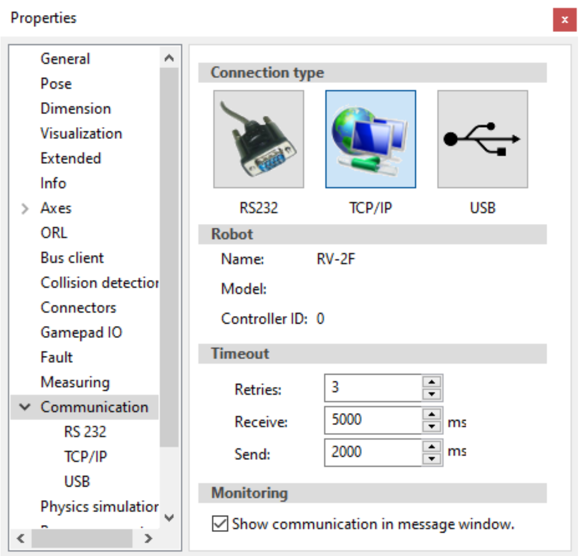
   Video tutorial: 56_HideHousing.mp4



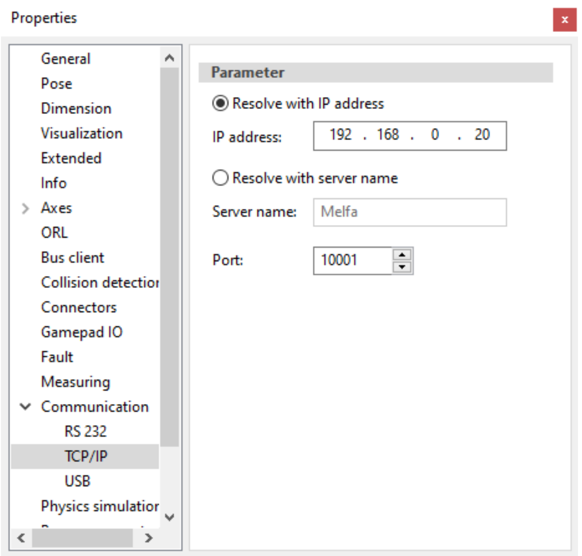
Connect to Robot Controller

Configure communication setting

1. In Model Explorer, select “CP-F-RASS_Mitsubishi\Montage_RV-4FL”.
2. In properties window, select **Communication**.
3. Select **TCP/IP** as connection type.



4. Expand Communication in properties window and select **TCP/IP**.
5. Enter the IP-address of the robot controller.
6. Port = 10001

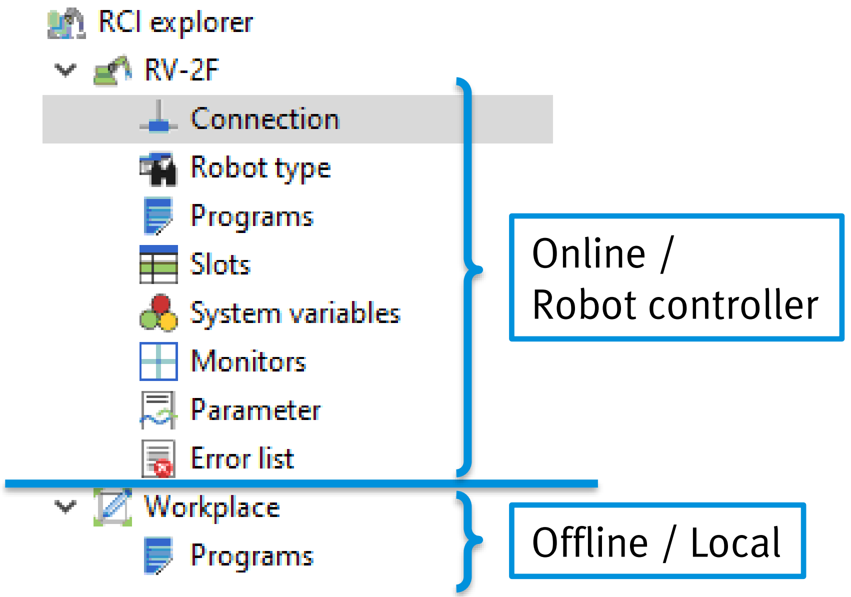


Video tutorial: 57_ConnectMitsuRobot.mp4

Connect to Robot Controller

RCI-Explorer

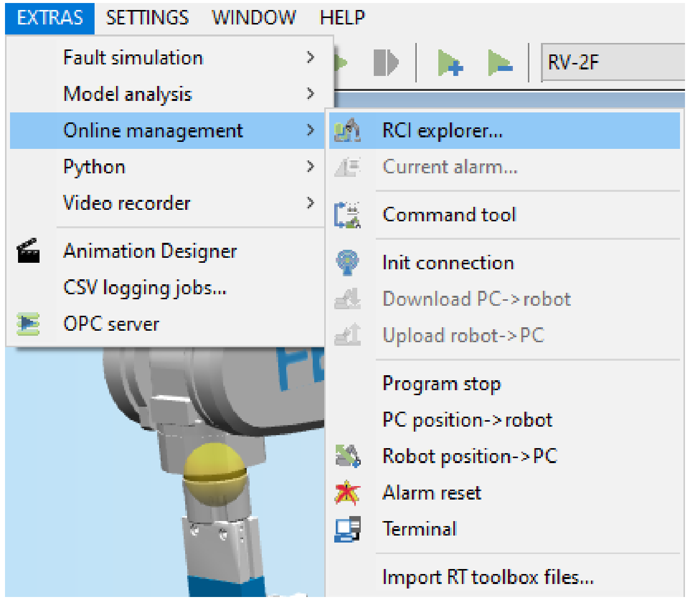
- RCI = Robot Control Interface
- Allow user to read information, program and control the robot controller in CIROS.
- Can create / load robot controller backup.
- Edit the program by uploading the robot program into local workspace.



Connect to Robot Controller

Open RCI-Explorer

- Select **Extras › Online management › RCI explorer...**

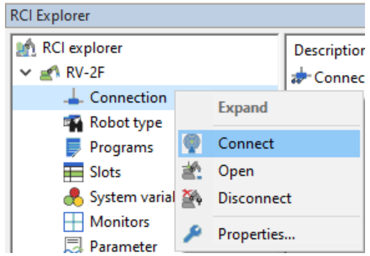


Only CIROS Studio

Connect to Robot Controller

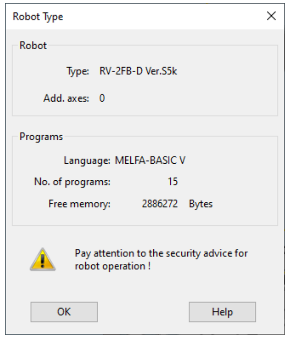
Build the connection

1. In RCI Explorer, right click on Connection and select **Connect**.

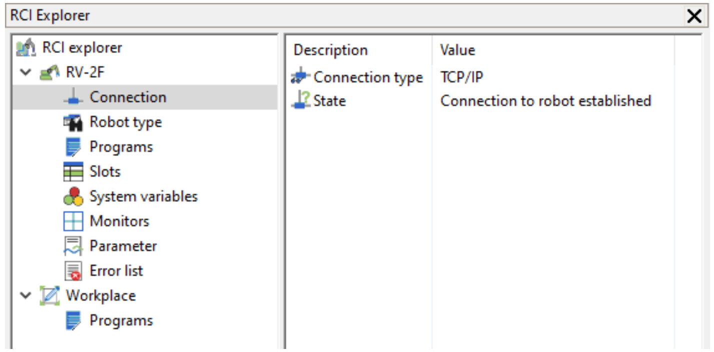


2. **Please pay attention to the security advice and environment of real robot!**

3. Select **OK** when the window pops up.



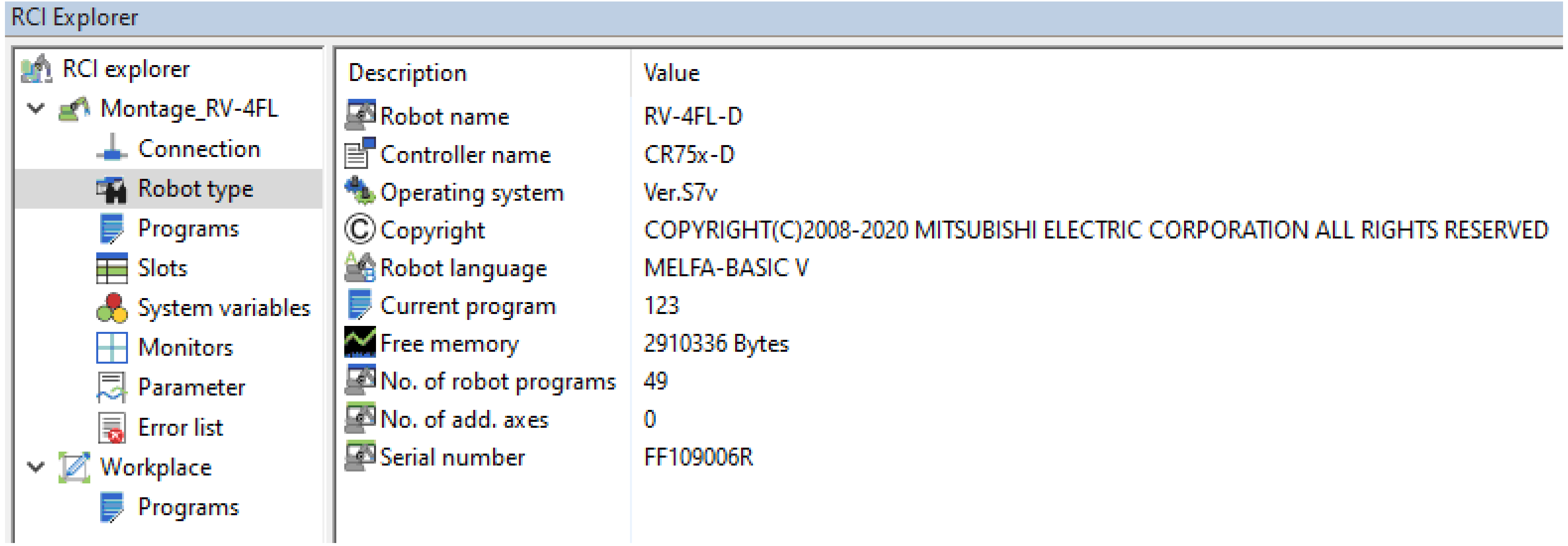
4. Connection is established.



Video tutorial: 57_ConnectMitsuRobot.mp4

Online Information from Robot Controller

Robot type



Only CIROS Studio

Online Information from Robot Controller

Robot programs

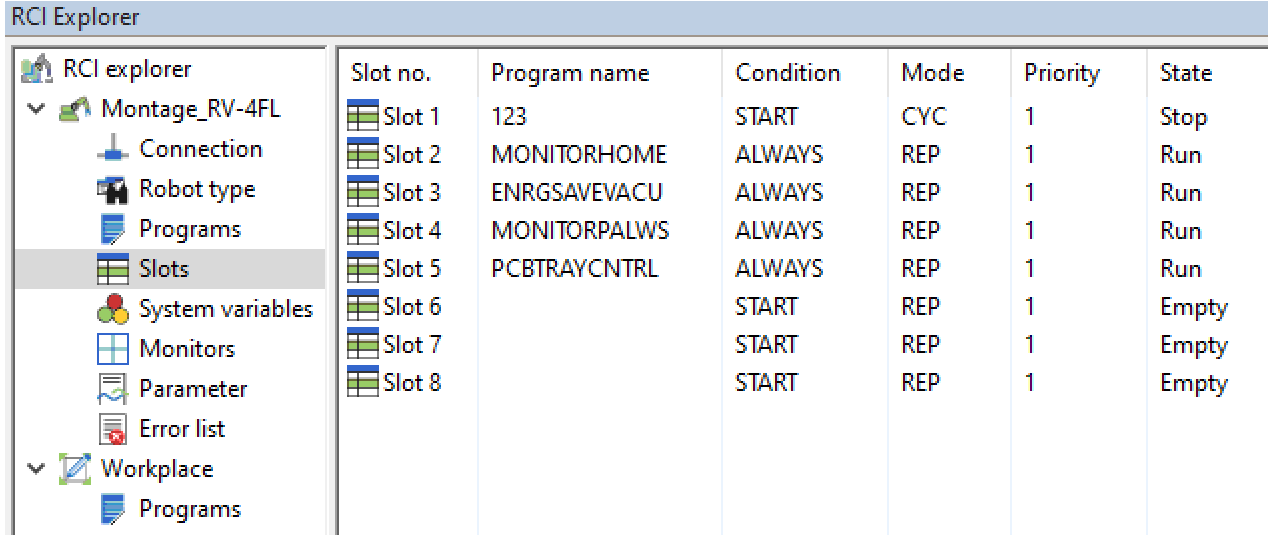
RCI Explorer

- RCI explorer
  - Montage_RV-4FL
    - Connection
    - Robot type
    - Programs
    - Slots
    - System variables
    - Monitors
    - Parameter
    - Error list
  - Workplace
    - Programs

File name	Size	Saved at	Lines
1	3285 Bytes	22-11-30	91
2	3495 Bytes	22-11-30	102
3	3507 Bytes	22-11-30	102
4	3770 Bytes	22-11-30	114
5	3028 Bytes	22-11-30	84
6	1974 Bytes	22-11-30	50
11	962 Bytes	22-11-30	28
12	1312 Bytes	22-11-30	26
13	1115 Bytes	22-11-30	25
14	1117 Bytes	22-11-30	25
123	5086 Bytes	22-11-30	141
234	3140 Bytes	22-11-30	83
255	1240 Bytes	22-11-30	18
900	1871 Bytes	22-11-30	37
999	3305 Bytes	22-11-30	63
UBP	7393 Bytes	22-12-01	112
GRPLOCK	765 Bytes	22-11-30	20
GRPOPEN	1090 Bytes	22-11-30	26
MOVHOME	977 Bytes	22-11-30	23
GRPCLOSE	1092 Bytes	22-11-30	26
GRPVACON	806 Bytes	22-11-30	20
MOUNTPCB	1399 Bytes	22-11-30	36
GRPVACOFF	798 Bytes	22-11-30	19
GRPRELEASE	792 Bytes	22-11-30	20
INITIALIZE	1481 Bytes	22-11-30	30
CAMERACALIB	9420 Bytes	22-11-30	240
MONITORHOME	2243 Bytes	22-11-30	53
PICKNEWTOOL	5361 Bytes	22-11-30	178
PLROTWP2STP	3122 Bytes	22-11-30	84
SENSORCHECK	3747 Bytes	22-11-30	85
DEMOMOUNTPCB	4746 Bytes	22-11-30	135
ENRGSAVEVACU	1459 Bytes	22-11-30	35
GETCAMRESULT	7994 Bytes	22-11-30	157
GETCURTOOLNO	2953 Bytes	22-11-30	61
GETFUSEMAGNO	2716 Bytes	22-11-30	68
MONITORPALWS	2268 Bytes	22-11-30	53
MOUNTBOTFUSE	1217 Bytes	22-11-30	34
MOUNTTOPFUSE	1211 Bytes	22-11-30	34
PCBTRAYCNTRL	1341 Bytes	22-11-30	31
PICKFRMSTOPR	3008 Bytes	22-11-30	83
PICKFRMVISON	1493 Bytes	22-11-30	39
PICKFUSFRMAG	2892 Bytes	22-11-30	79
PICKPCBFRPAL	3897 Bytes	22-11-30	97
PICKWPFRMASS	1767 Bytes	22-11-30	42
PLACETOSTOPR	3102 Bytes	22-11-30	85

Online Information from Robot Controller

Robot programs in slots

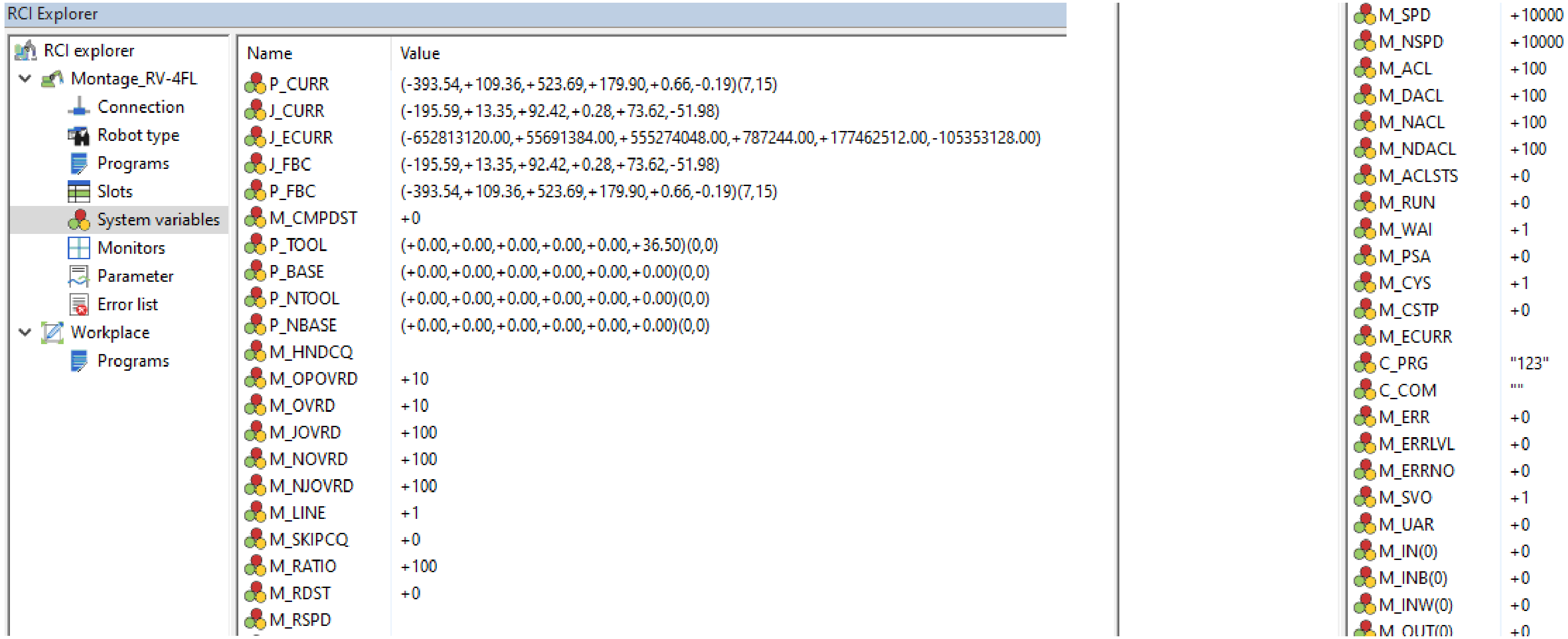


Slot no.	Program name	Condition	Mode	Priority	State
Slot 1	123	START	CYC	1	Stop
Slot 2	MONITORHOME	ALWAYS	REP	1	Run
Slot 3	ENRGSAVEVACU	ALWAYS	REP	1	Run
Slot 4	MONITORPALWS	ALWAYS	REP	1	Run
Slot 5	PCBTRAYCNTRL	ALWAYS	REP	1	Run
Slot 6		START	REP	1	Empty
Slot 7		START	REP	1	Empty
Slot 8		START	REP	1	Empty

Only CIROS Studio

Online Information from Robot Controller

System variables



Only CIROS Studio

Online Information from Robot Controller

Parameters

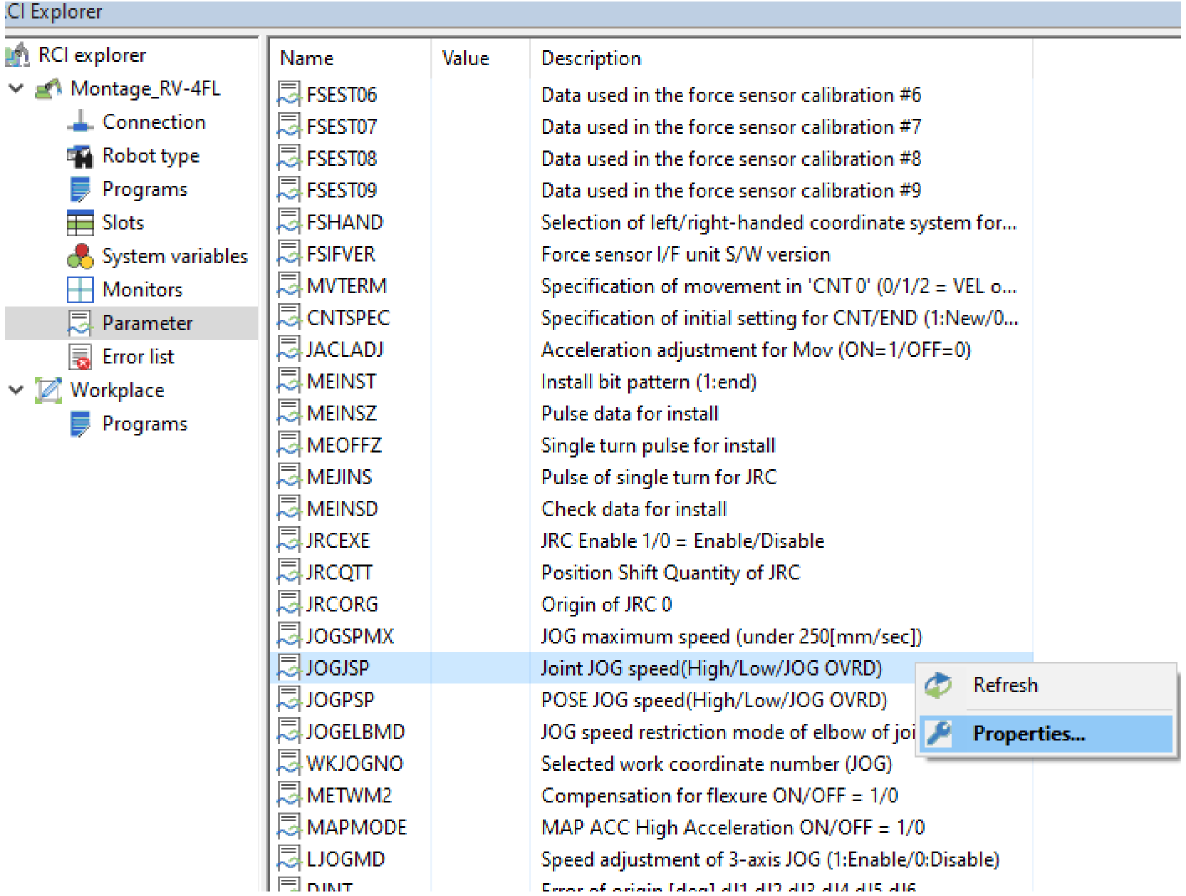
RCI Explorer

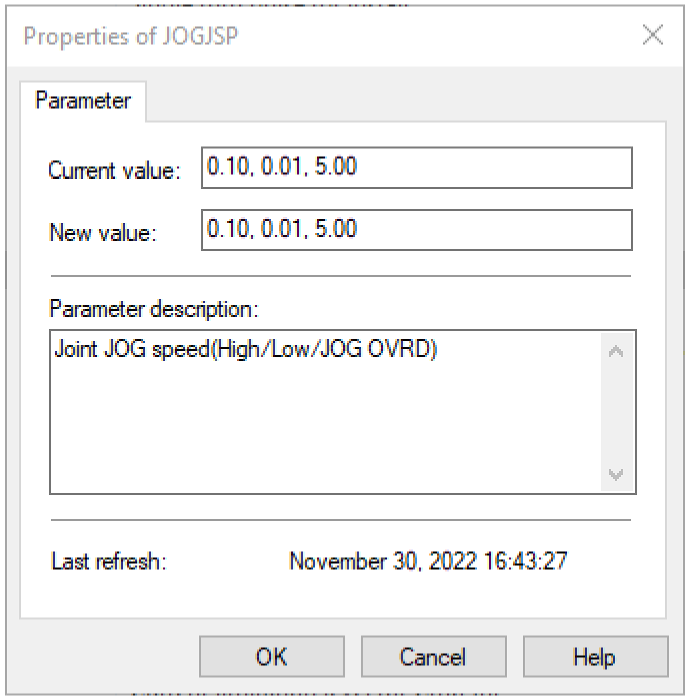
- RCI explorer
  - Montage_RV-4FL
    - Connection
    - Robot type
    - Programs
    - Slots
    - System variables
    - Monitors
    - Parameter
    - Error list
  - Workplace
    - Programs

Name	Value	Description
TASKMAX		Number of Multitask
RETPATH		Use resumed buffer 2:ON(MVS)/1:ON(MOV)/0:OFF
INB		Stop input signal normally open(0)/close(1)
CTN		Continue function Disable/Enable (0/1)
TBCAN		
TBBZR		T/B Buzzer (1:ON, 0:OFF)
COMDEV		Definition of device for COM:1-8
CDTR422		DTR control[0:OFF,1:ON]
CBAU232		Baud rate
CLEN232		Length[8,7]
CPRTY232		Parity[0:None,1:odd,2:even]
CSTOP232		Stop bit[1,2]
CTERM232		Termination[0:CR 1:CR+LF]
CPRC232		Protocol[0:Non,1:Process ,2:Data link]
CDTR232		DTR control[0:OFF,1:ON]
CBUF232		Receive buffer mode (0:Ring,1:Fix)
CBAUE11		Baud rate
CLENE11		Length[8,7]
CPRTYE11		Parity[0:None,1:odd,2:even]
CSTOPE11		Stop bit[1,2]
CTERME11		Termination[0:CR 1:CR+LF]
CPRCE11		Protocol[0:Non,1:Process,2:Data link]
CDTRE11		DTR control[0:OFF,1:ON]
CBUFE11		Receive buffer mode (0:Ring,1:Fix)
CBAUE12		Baud rate
CLENE12		Length[8,7]
CPRTYE12		Parity[0:None,1:odd,2:even]
CSTOPE12		Stop bit[1,2]
CTERME12		Termination[0:CR 1:CR+LF]
CPRCE12		Protocol[0:Non,1:Process,2:Data link]
CDTRE12		DTR control[0:OFF,1:ON]
CBUFE12		Receive buffer mode (0:Ring,1:Fix)
CBAUE13		Baud rate
CLENE13		Length[8,7]
CPRTYE13		Parity[0:None,1:odd,2:even]
CSTOPE13		Stop bit[1,2]
CTERME13		Termination[0:CR 1:CR+LF]
CPRCE13		Protocol[0:Non,1:Process,2:Data link]
CDTRE13		DTR control[0:OFF,1:ON]
CBUFE13		Receive buffer mode (0:Ring,1:Fix)
CTERME14		Termination[0:CR 1:CR+LF]
CPRCE14		Protocol[0:Non,1:Process,2:Data link]
CBUFE14		Receive buffer mode (0:Ring,1:Fix)
CTERME15		Termination[0:CR 1:CR+LF]
CPRCE15		Protocol[0:Non,1:Process,2:Data link]

Only CIROS Studio

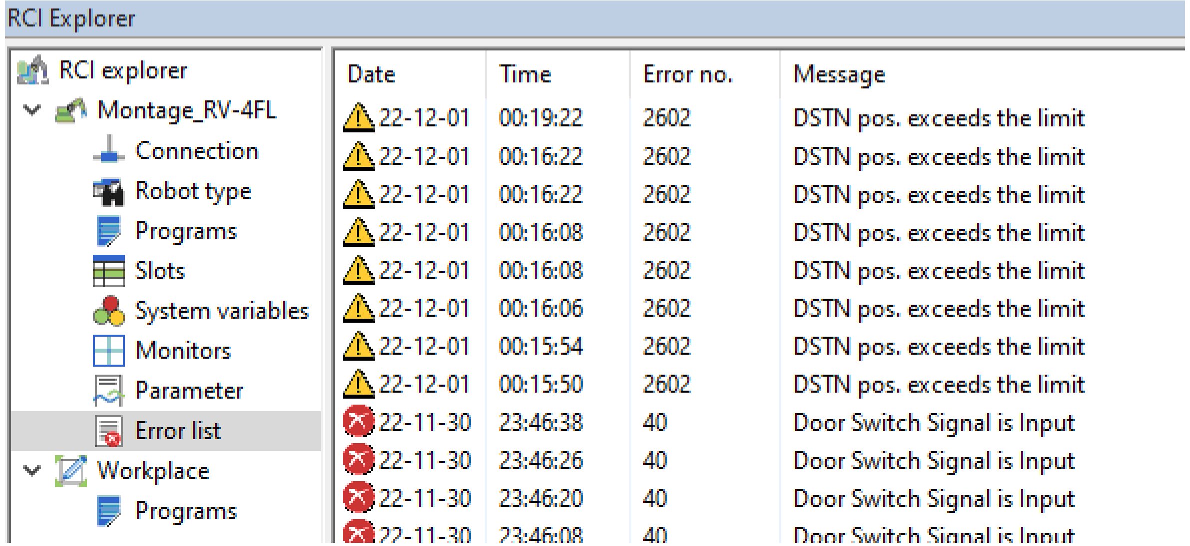
Read Live Parameter Value





Online Information from Robot Controller

Error list



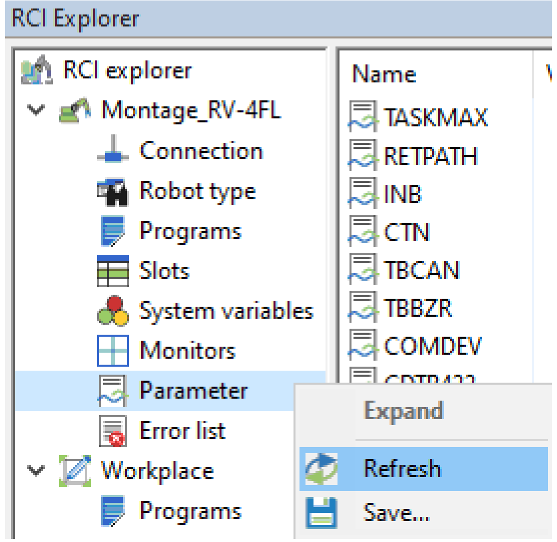
FESTO

Only CIROS Studio

Online Information from Robot Controller

Refresh window

- Sometimes, the window is not updated. In this case, the windows can be refreshed.

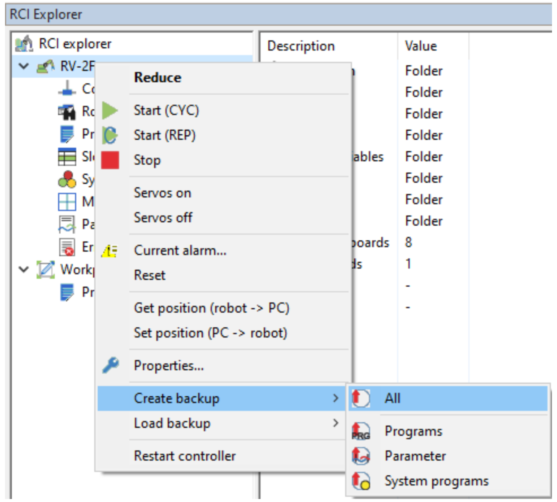


Video tutorial: 58_RefreshRCIWindow.mp4

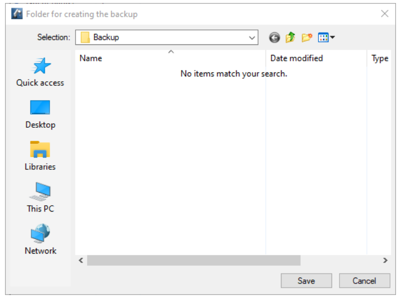
Only CIROS Studio

Create Robot Controller Backup

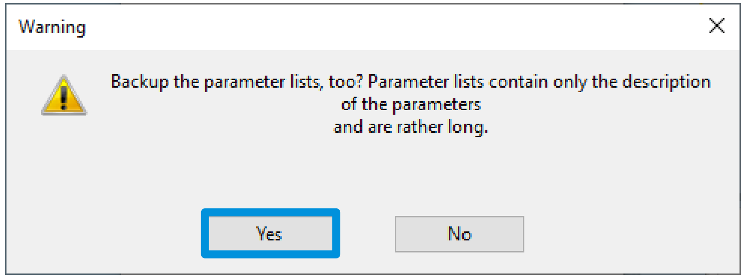
1. Right click on the robot and select Create backup\All.



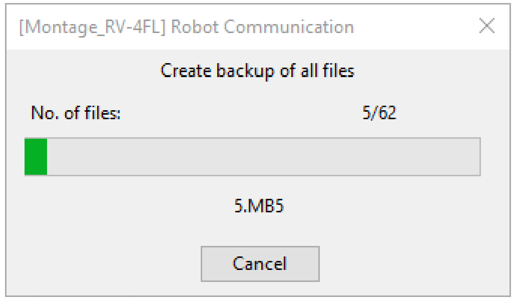
2. Create the backup in an empty folder.



2. Confirm the warning.



3. Creating backup...



Video tutorial: 59_CreateRobotBackup.mp4

Only CIROS Studio

Robot Controller Backup Folder

- By default, backup is located in ‹project folder›\Backup.

Name	Date modified	Type	Size
Backup	30/11/2022 16:36	File folder	
CF	30/11/2022 16:23	File folder	
Montage_RV-4FL	30/11/2022 16:40	File folder	
Textures	30/11/2022 16:24	File folder	
RASS2_717.err	30/11/2022 16:33	ERR File	6 KB
RASS2_717.ini	30/11/2022 16:24	Configuration sett...	24 KB
RASS2_717.modx	30/11/2022 16:24	CIROS Model	35.471 KB
RASS2_717.par	30/11/2022 16:32	PAR File	1 KB
translate.py	30/11/2022 15:10	Python File	9 KB

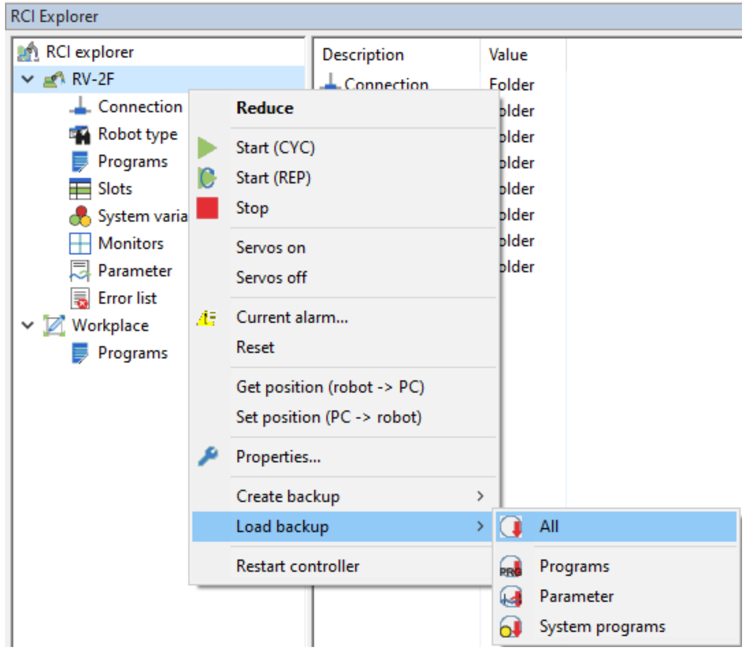
This PC › Documents › CIROS › Projects › RASS2_717 › Backup

Name	Date modified	Type	Size
1.MB5	30/11/2022 16:35	MB5 File	4 KB
2.MB5	30/11/2022 16:35	MB5 File	4 KB
3.MB5	30/11/2022 16:35	MB5 File	4 KB
4.MB5	30/11/2022 16:35	MB5 File	4 KB
5.MB5	30/11/2022 16:35	MB5 File	3 KB
6.MB5	30/11/2022 16:35	MB5 File	2 KB
11.MB5	30/11/2022 16:35	MB5 File	1 KB
12.MB5	30/11/2022 16:35	MB5 File	2 KB
13.MB5	30/11/2022 16:35	MB5 File	2 KB
14.MB5	30/11/2022 16:35	MB5 File	2 KB
123.MB5	30/11/2022 16:35	MB5 File	5 KB
234.MB5	30/11/2022 16:35	MB5 File	4 KB
255.MB5	30/11/2022 16:35	MB5 File	2 KB
900.MB5	30/11/2022 16:35	MB5 File	2 KB
999.MB5	30/11/2022 16:35	MB5 File	4 KB
AError.log	30/11/2022 16:35	Text Document	57 KB
BACKUP.COS	30/11/2022 16:37	COS File	1 KB
CAMERACALIB.MB5	30/11/2022 16:35	MB5 File	10 KB
CError.log	30/11/2022 16:35	Text Document	4 KB
COMMON.LST	30/11/2022 16:36	LST File	160 KB
COMMON.PRM	30/11/2022 16:35	PRM File	2 KB
DEMOMOUNTPCB.MB5	30/11/2022 16:36	MB5 File	5 KB
ENRGSAVEVACU.MB5	30/11/2022 16:36	MB5 File	2 KB
GETCAMRESULT.MB5	30/11/2022 16:36	MB5 File	8 KB
GETCURTOOLNO.MB5	30/11/2022 16:36	MB5 File	3 KB
GETFUSEMAGNO.MB5	30/11/2022 16:36	MB5 File	3 KB

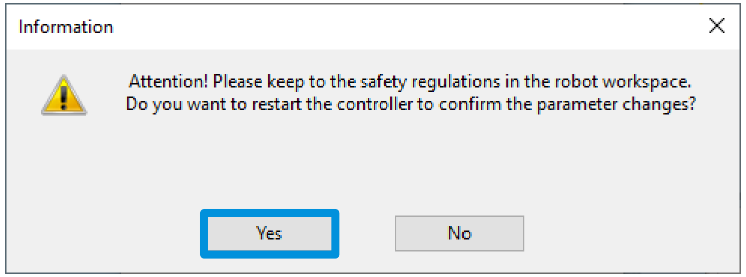
Only CIROS Studio

Load Backup

1. Right click on the robot and select **Load backup › All**.



2. Select backup folder.
3. Restart the controller after loading complete.

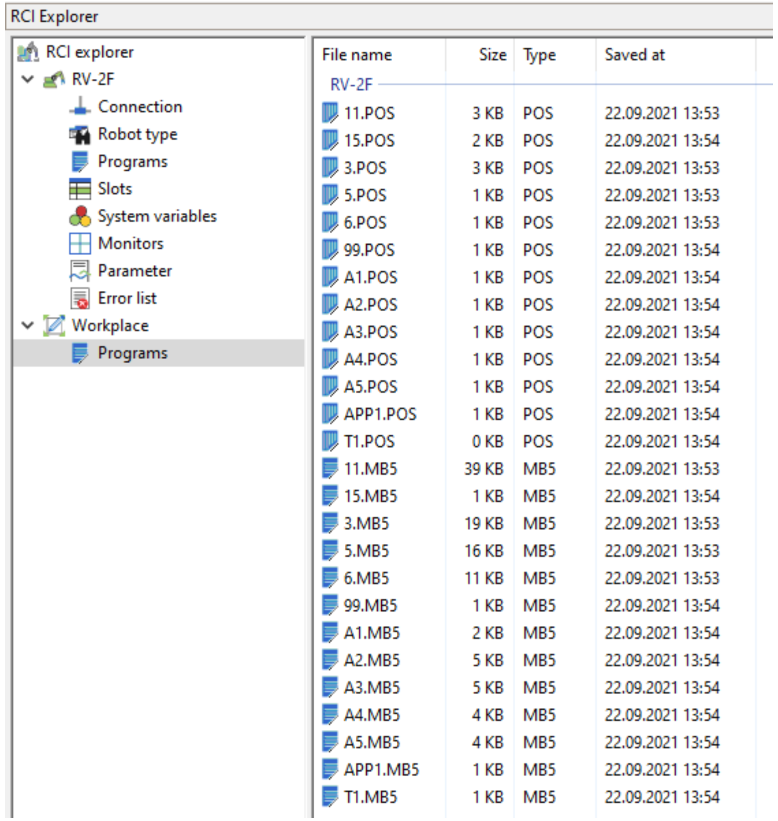


FESTO

Only CIROS Studio

Upload Robot Programs

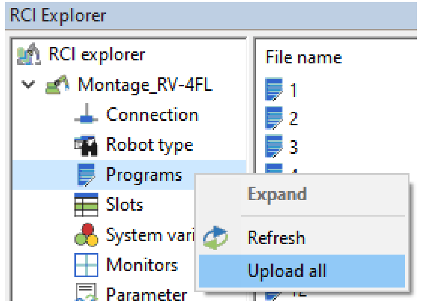
- Robot programs can be uploaded to the computer.
- Uploaded programs are listed in Workplace\Programs.
- The programs can then be edited.



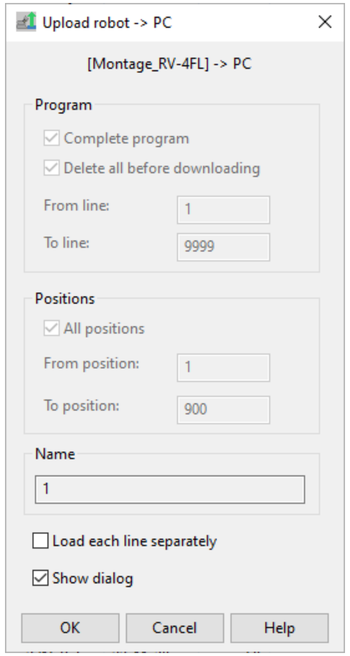
Only CIROS Studio

Upload Robot Programs

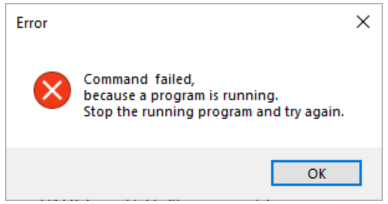
1. In RCI Explorer, right click on Programs, select Upload all.

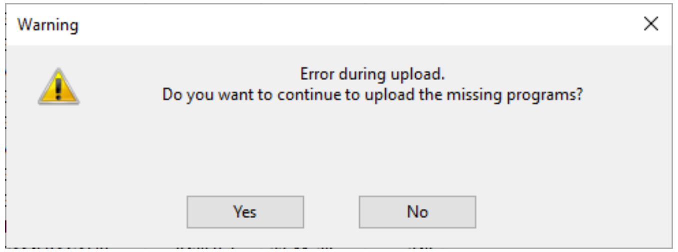


2. When this window pops out, click ok.



3. Programs which are running in slots cannot be uploaded. Click ok and continue uploading the missing programs.





Video tutorial: 60_UploadRobotPrograms.mp4

Only CIROS Studio

Robot Program Folder

- By default, uploaded robot programs are located in ‹project folder›\‹robot controller name›.
  - For example,
    - CIROS project : RASS2_717
    - Robot name : Montage_RV-4FL
  - Uploaded robot programs are located in ..\RASS2_717\Montage_RV-4FL.

Name	Date modified	Type	Size
Backup	30/11/2022 16:36	File folder	
CF	30/11/2022 16:23	File folder	
Montage_RV-4FL	30/11/2022 16:40	File folder	
Textures	30/11/2022 16:24	File folder	
RASS2_717.err	30/11/2022 16:33	ERR File	6 KB
RASS2_717.ini	30/11/2022 16:24	Configuration sett...	24 KB
RASS2_717.modx	30/11/2022 16:24	CIROS Model	35.471 KB
RASS2_717.par	30/11/2022 16:32	PAR File	1 KB
translate.py	30/11/2022 15:10	Python File	9 KB

This PC › Documents › CIROS › Projects › RASS2_717 › Montage_RV-4FL

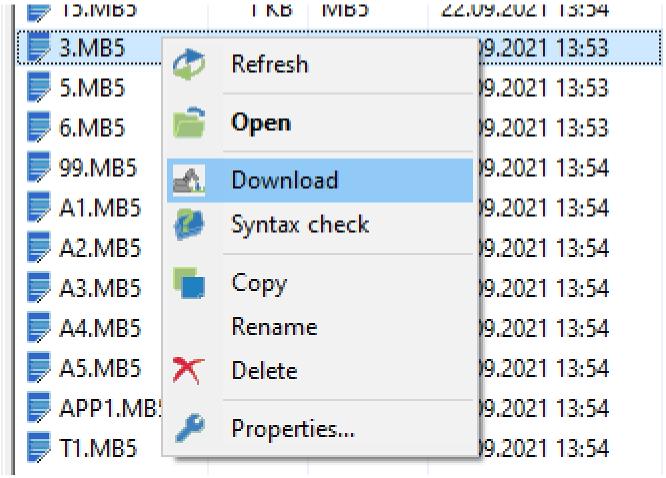
Name	Date modified	Type	Size
1.MB5	30/11/2022 16:39	MB5 File	3 KB
1.POS	30/11/2022 16:39	POS File	1 KB
2.MB5	30/11/2022 16:39	MB5 File	4 KB
2.POS	30/11/2022 16:39	POS File	1 KB
3.MB5	30/11/2022 16:39	MB5 File	4 KB
3.POS	30/11/2022 16:39	POS File	1 KB
4.MB5	30/11/2022 16:39	MB5 File	4 KB
4.POS	30/11/2022 16:39	POS File	1 KB
5.MB5	30/11/2022 16:39	MB5 File	3 KB
5.POS	30/11/2022 16:39	POS File	1 KB
6.MB5	30/11/2022 16:39	MB5 File	2 KB
6.POS	30/11/2022 16:39	POS File	0 KB
11.MB5	30/11/2022 16:39	MB5 File	1 KB
11.POS	30/11/2022 16:39	POS File	0 KB
12.MB5	30/11/2022 16:39	MB5 File	1 KB
12.POS	30/11/2022 16:39	POS File	1 KB
13.MB5	30/11/2022 16:39	MB5 File	1 KB
13.POS	30/11/2022 16:39	POS File	1 KB
14.MB5	30/11/2022 16:39	MB5 File	1 KB
14.POS	30/11/2022 16:39	POS File	1 KB
123.MB5	30/11/2022 16:39	MB5 File	5 KB
123.POS	30/11/2022 16:39	POS File	1 KB
234.MB5	30/11/2022 16:39	MB5 File	3 KB
234.POS	30/11/2022 16:39	POS File	1 KB
255.MB5	30/11/2022 16:39	MB5 File	1 KB
255.POS	30/11/2022 16:39	POS File	1 KB

Only CIROS Studio

Download Program

- Programs in Workspace can be downloaded into robot controller.

1. Select the MB5 program in workspace.
2. Right click and select Download.

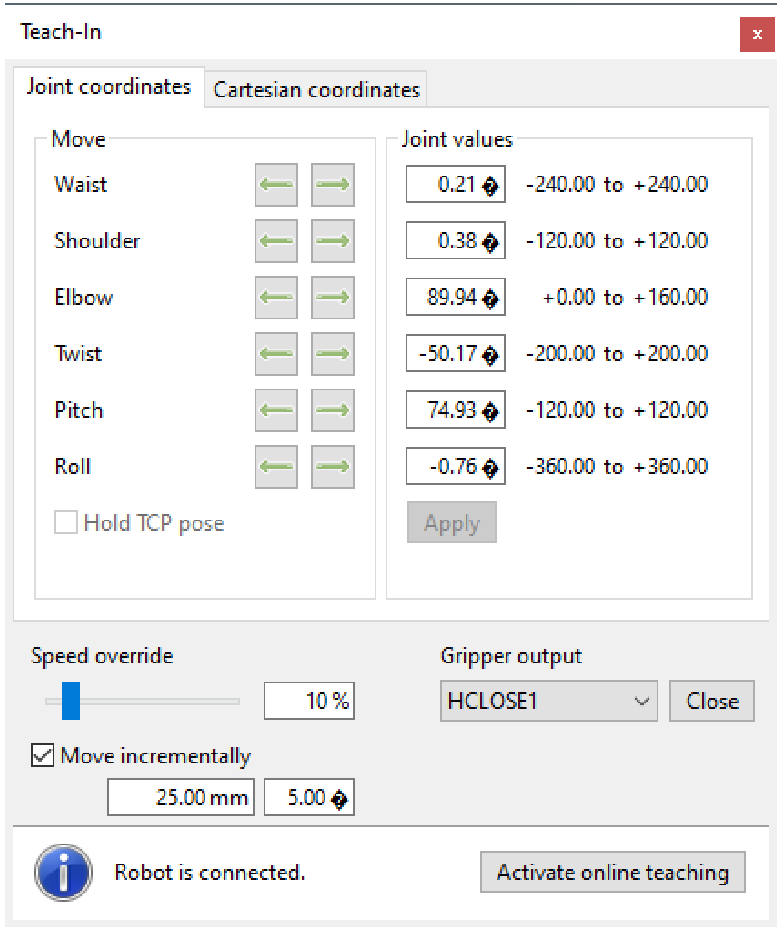


Only CIROS Studio

Online Teach-In

- Once connected, it is possible to activate online teaching in Teach-In panel.
- Online teaching mode allows
  - Simulation of real time robot position in CIROS
  - Move the real robot in CIROS
  - Track the real robot coordinates in CIROS

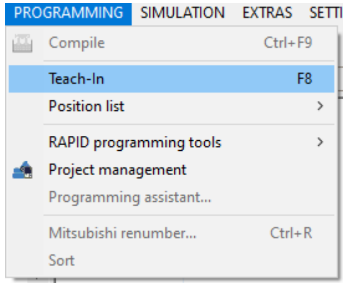
1. Activate online teaching.
2. Observe the change in model window.
3. Move Roll coordinate of the robot incrementally 5°.
4. Deactivate online teaching.
5. Reset the model.



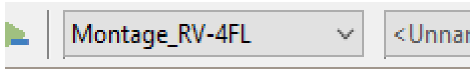
Only CIROS Studio

Online Teach-In

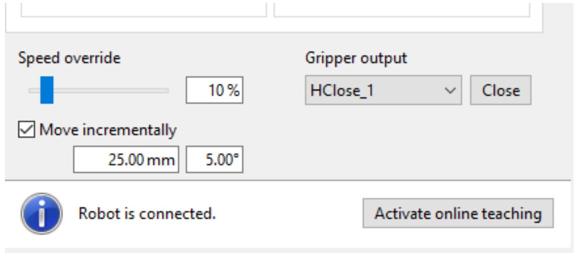
1. Make sure robot controller is online.
2. Open Teach-In panel.
   Shortcut: F8



3. Select the robot as active controller.



4. At the bottom of Teach-In panel, click on Activate online teaching.



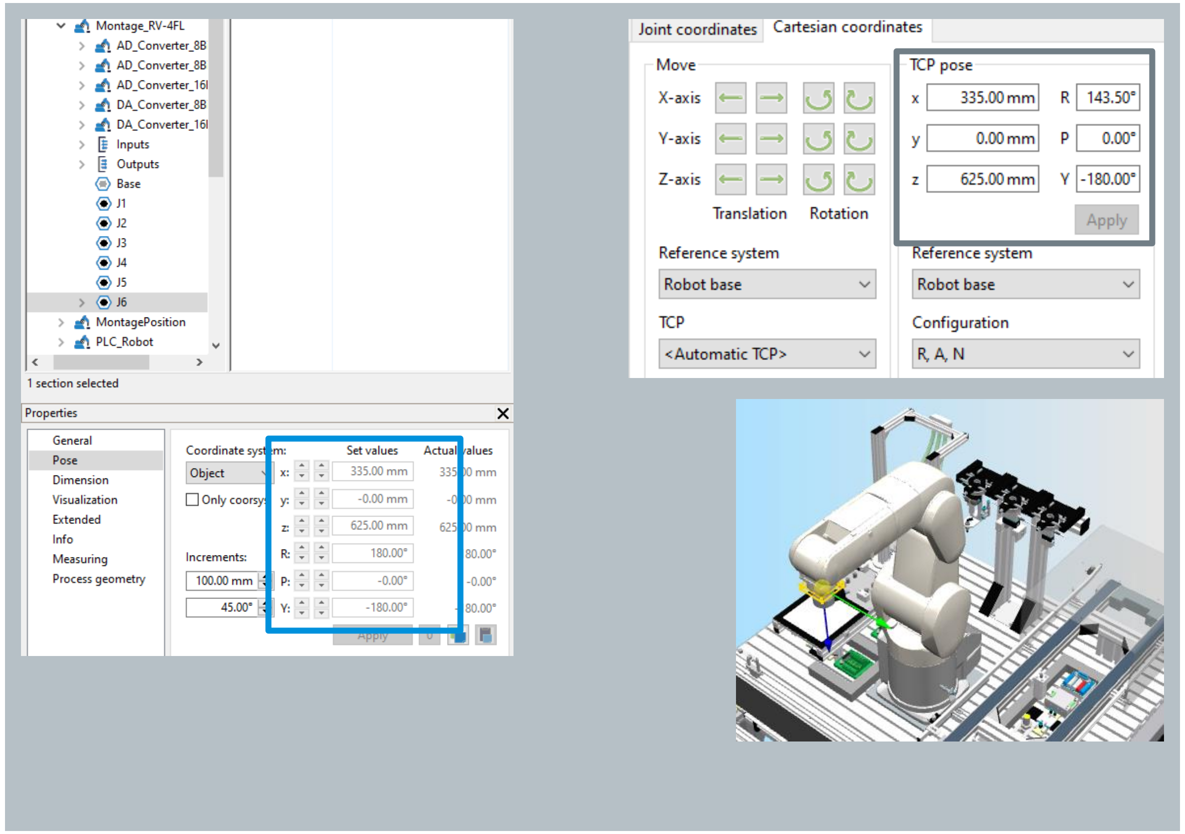
5. Click ok on pop-up window.

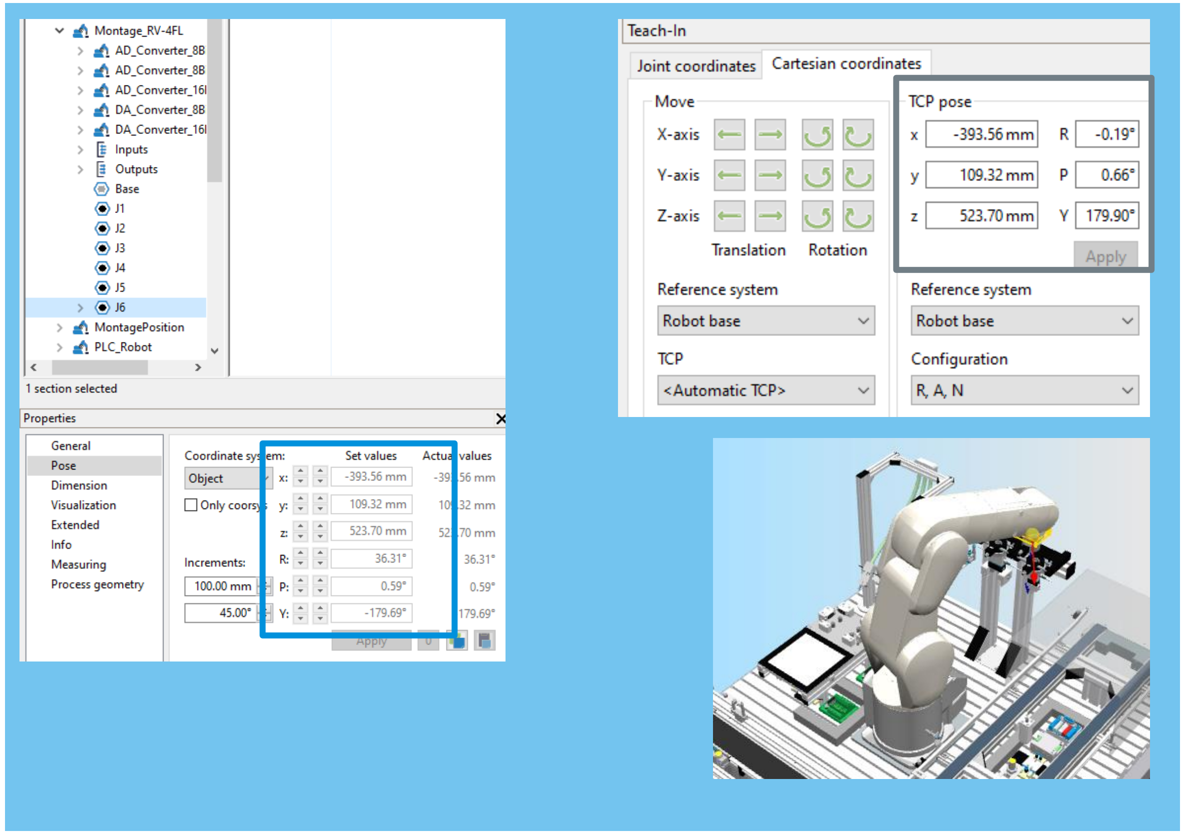
Video tutorial: 61_RobotTeachIn.mp4

Only CIROS Studio

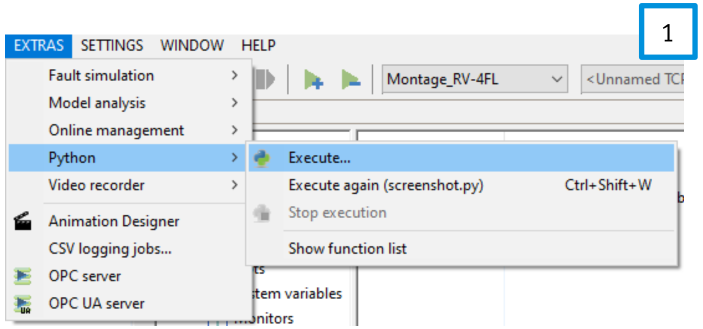
Online Teach-In

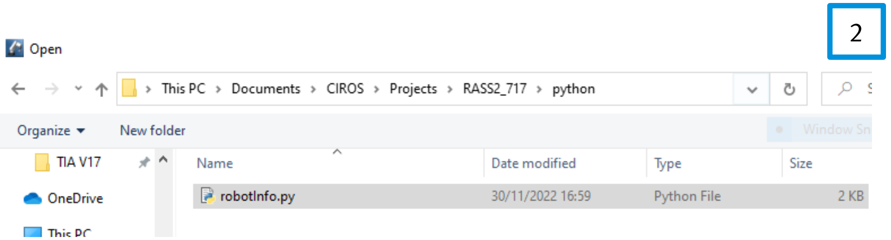
Notice the difference between before and after teach in is activated.

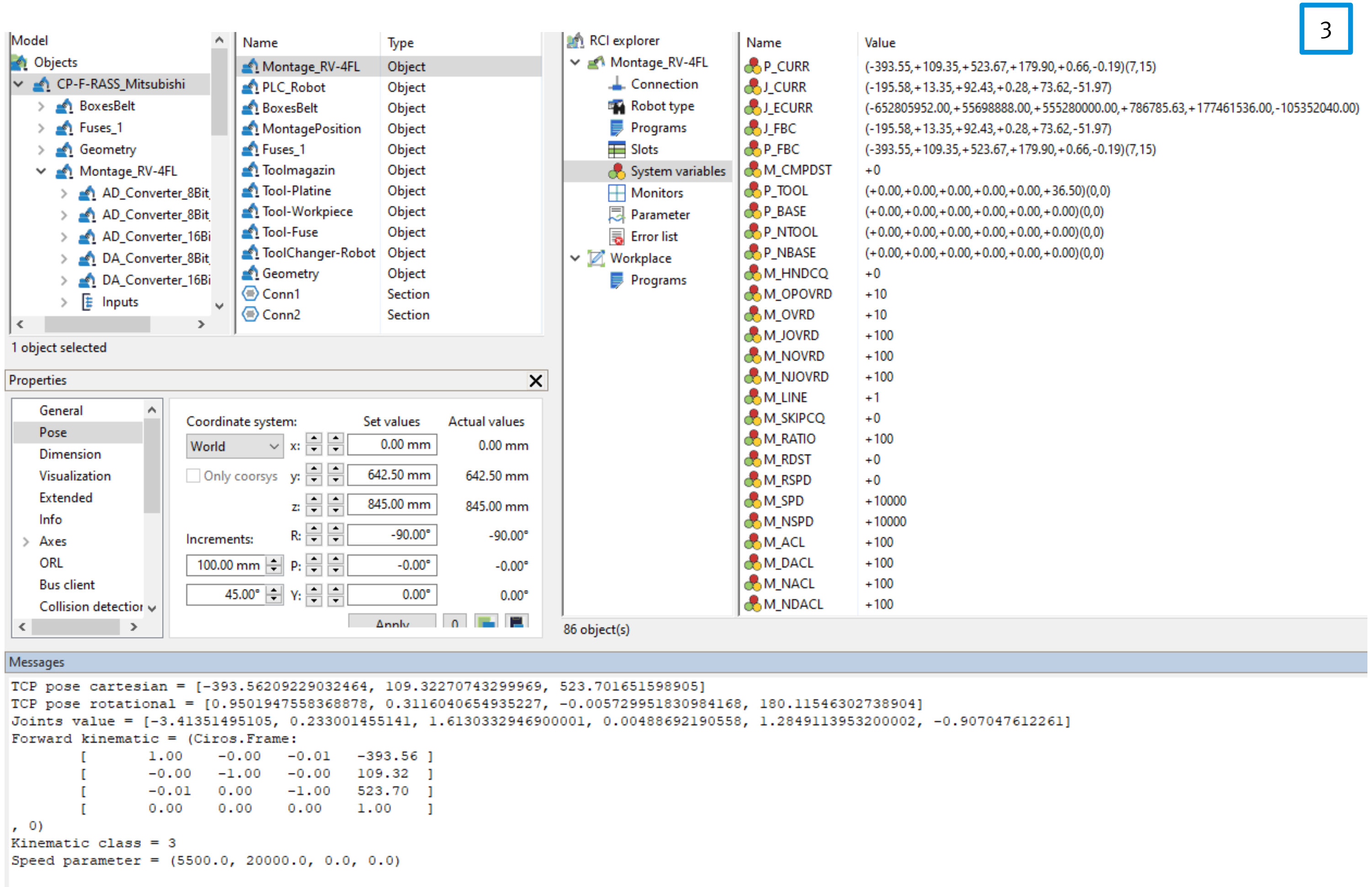




Get Actual Robot Data with Built-In Python Function







Ready to use project: robotInfo.py

Video tutorial: 62_RobotInfoPython.mp4

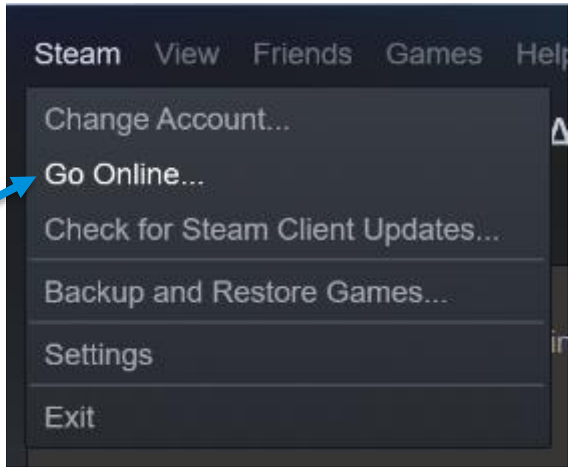
Virtual Reality

Setting up VR Glasses

- CIROS can be directly connected to VR glasses like HTC Vive and Oculus Rift.
- Typical procedure
  1. Install Steam / SteamVR and create a user account (https://store.steampowered.com/)
  2. Start Steam and switch to "offline mode" in case that Steam should be used without internet access
  3. Start SteamVR and run room setup
  4. Configure the VR mode of CIROS
  5. Activate VR mode of CIROS and start the simulation

Note: HTC Vive is the only officially supported VR glasses.

Switching between Steam's online/offline mode

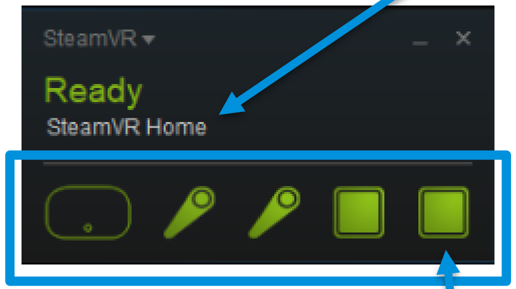


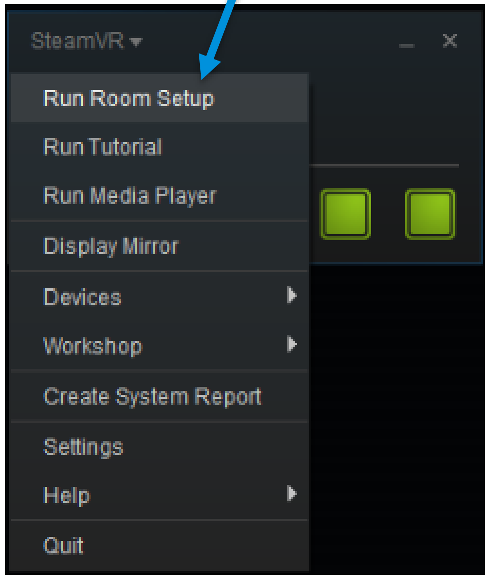
Setting up VR Glasses

Start SteamVR and run Room Setup

Configuration of the (virtual) room

Name of the application currently using SteamVR





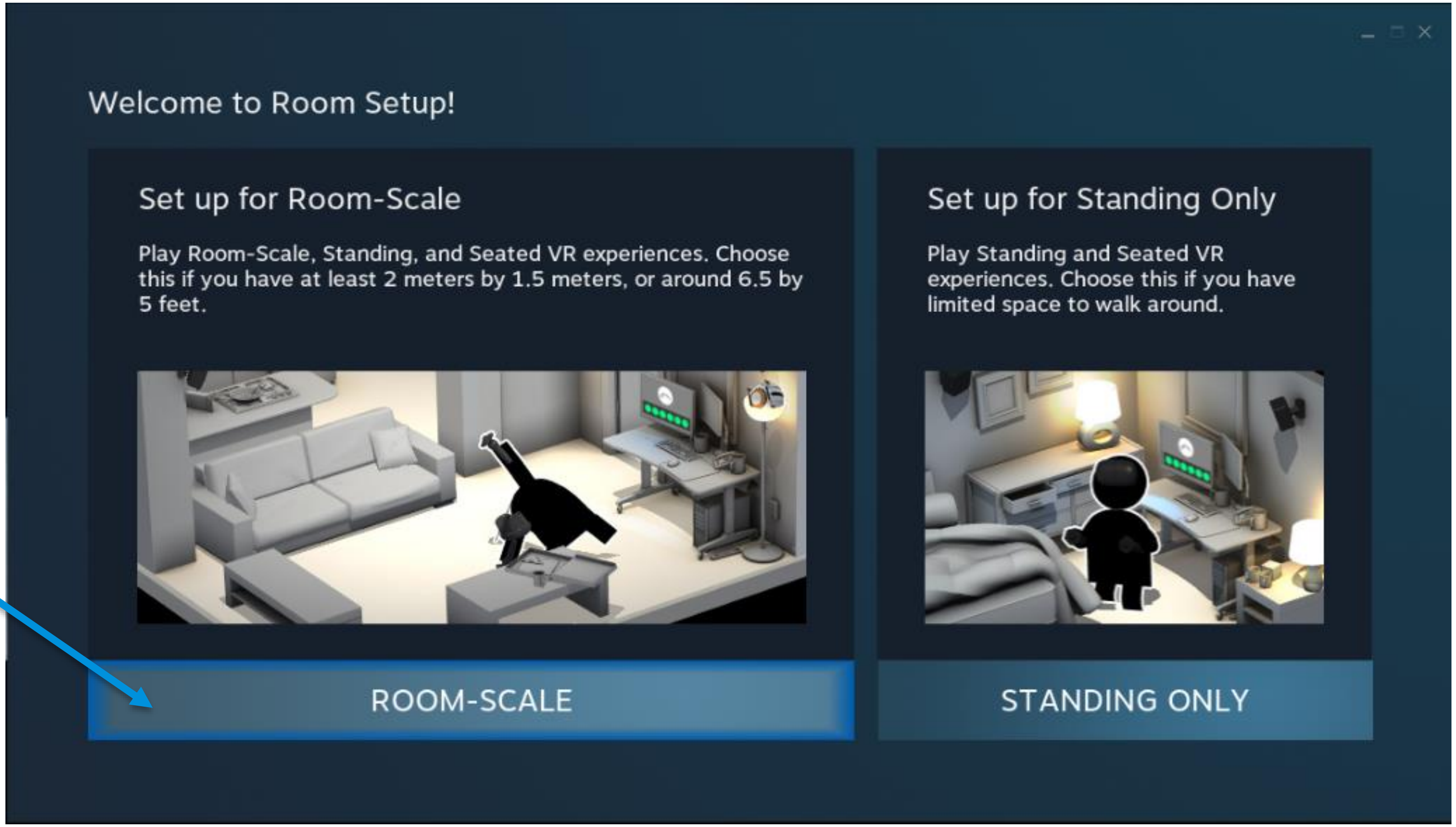
Filled icons represent active components

List of available HTC Vive components (from left to right):
Glasses, controller 1 & 2, left & right base station

Setting up VR Glasses

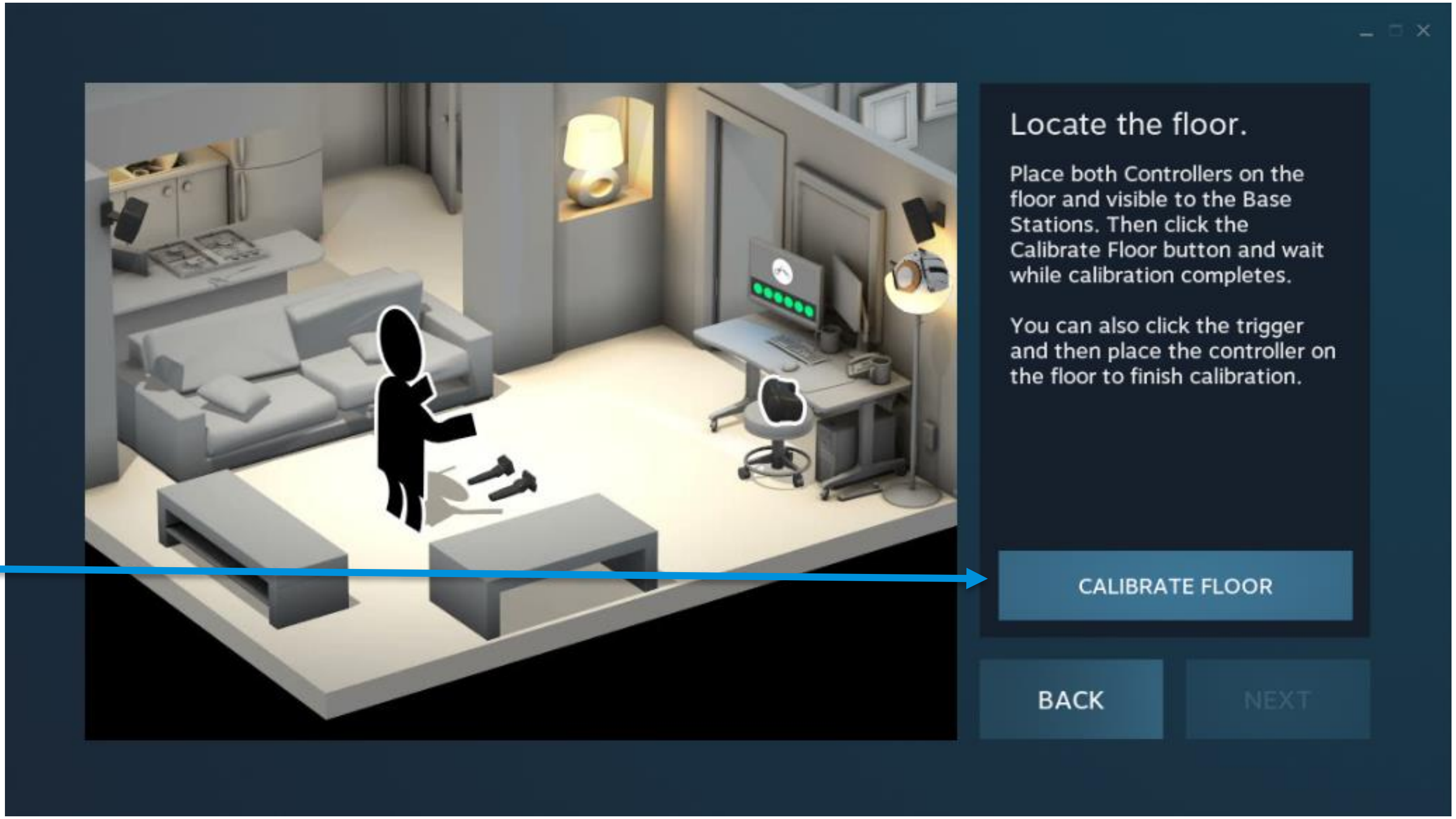
Start SteamVR and run Room Setup

Choose “Room-Scale” to be able to go around in the real and virtual room



Setting up VR Glasses

Start SteamVR and run Room Setup

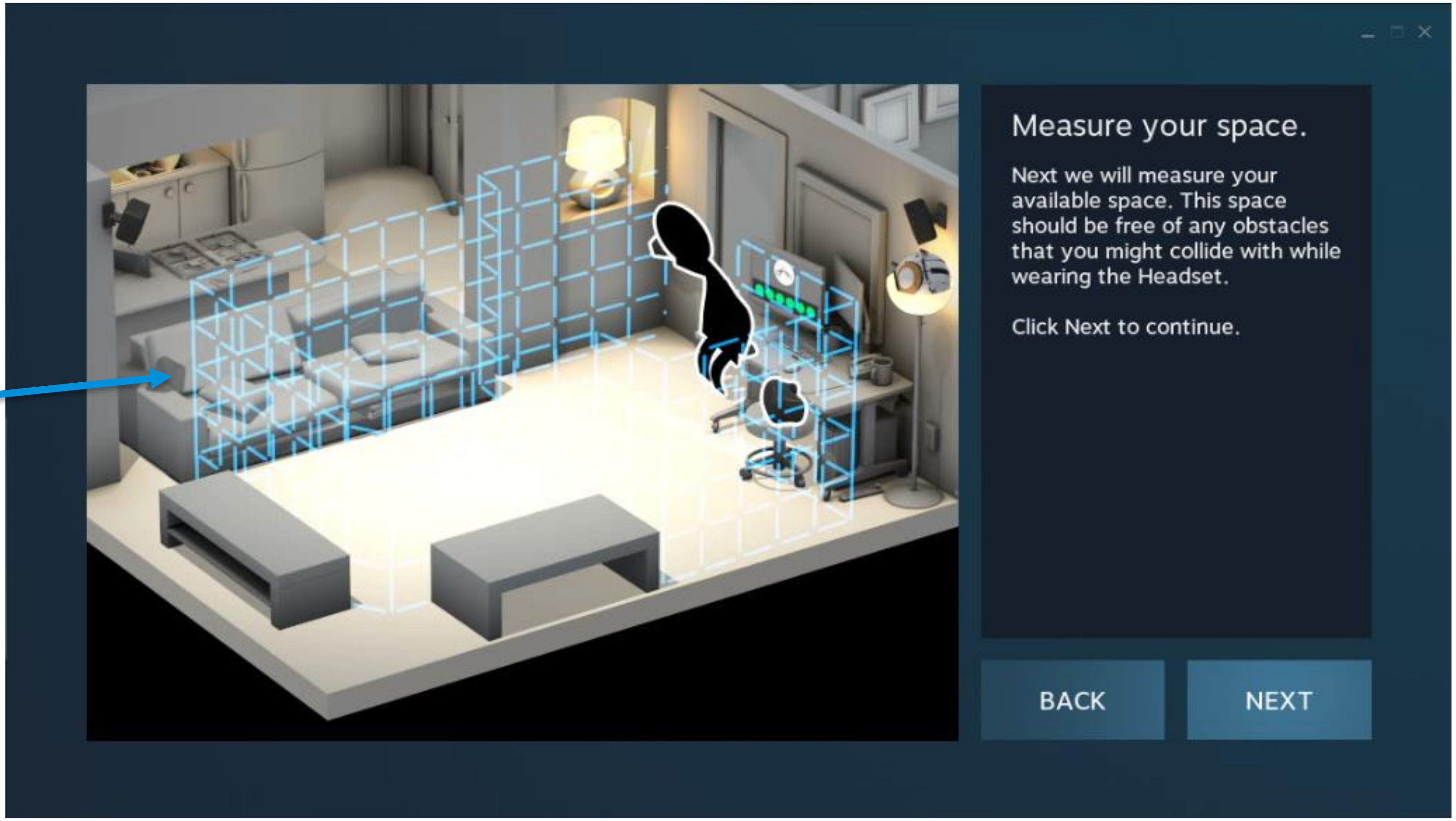


Calibrate real and virtual floor
(ground level wrt. to z-axis)

Setting up VR Glasses

Start SteamVR and run Room Setup

Configuration of the available physical space to be matched with the virtual room

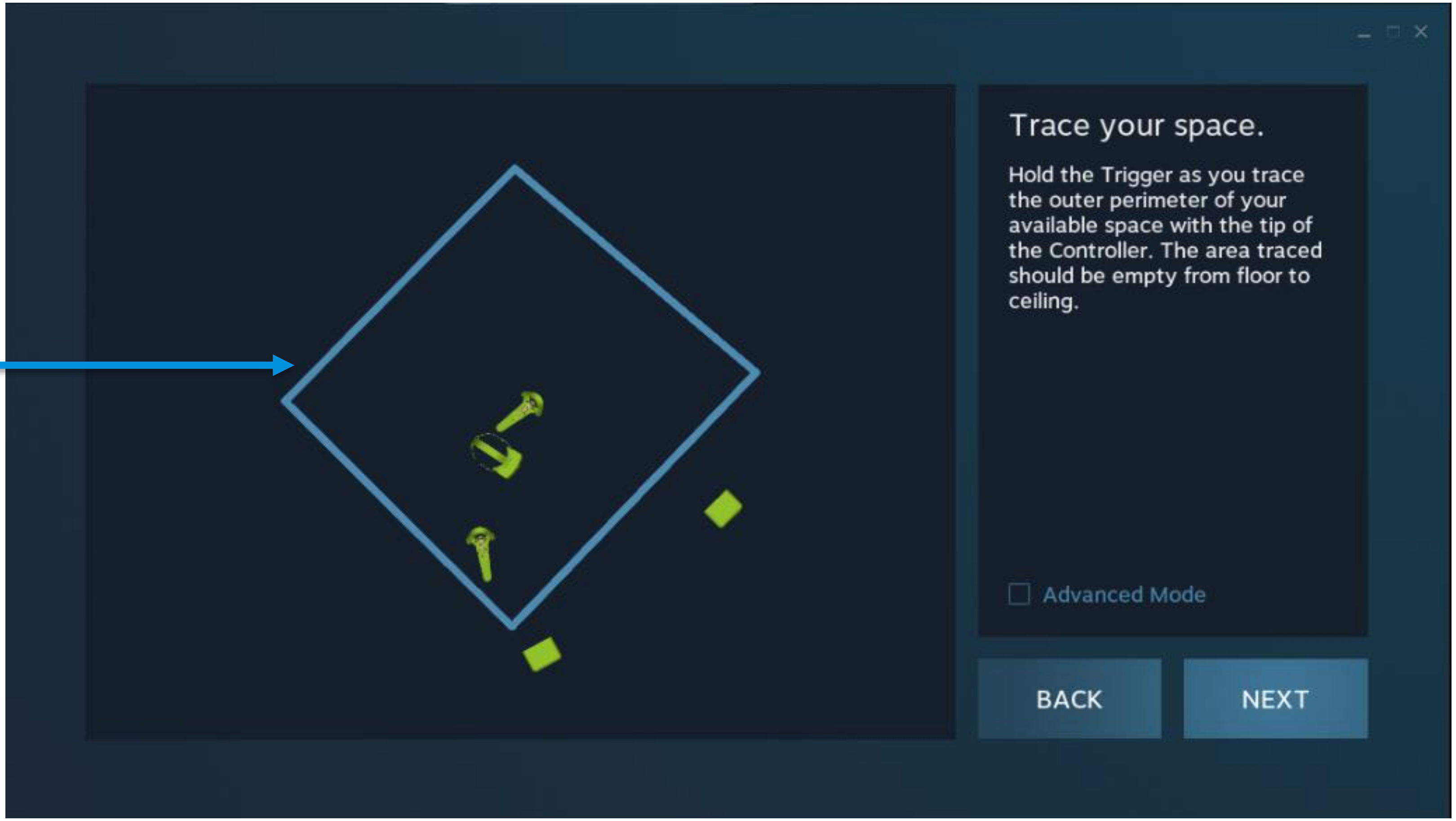


FESTO

Setting up VR Glasses

Start SteamVR and run Room Setup

Specify the available physical space (its outer perimeter) by holding the trigger button of one of the two controllers



FESTO

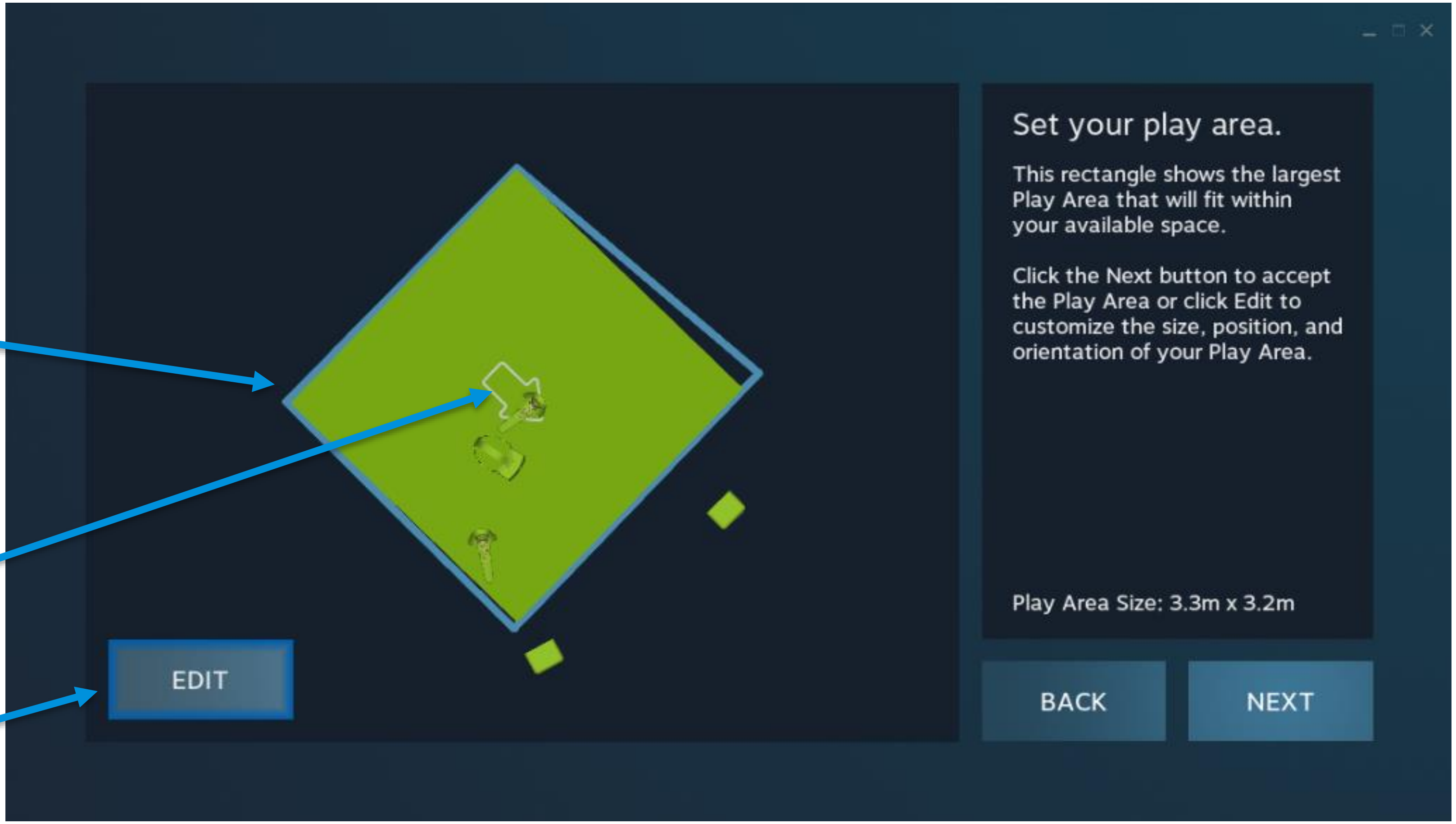
Setting up VR Glasses

Start SteamVR and run Room Setup

If the available space defined before fits the “Play Area” it will be highlighted in green

Direction of the user perspective onto the SteamVR scene

“EDIT” allows for optimizing position, size, and user perspective manually

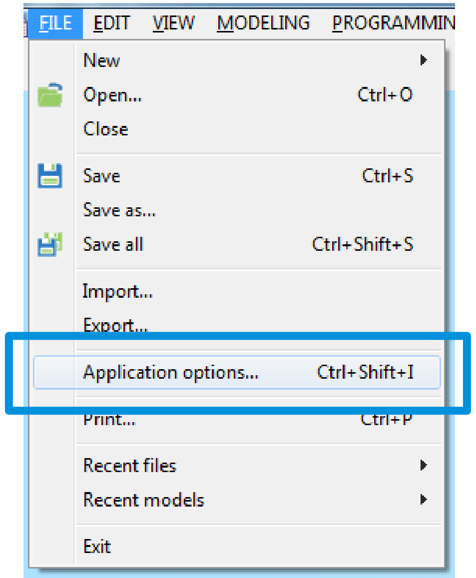


FESTO

Setting up VR Glasses

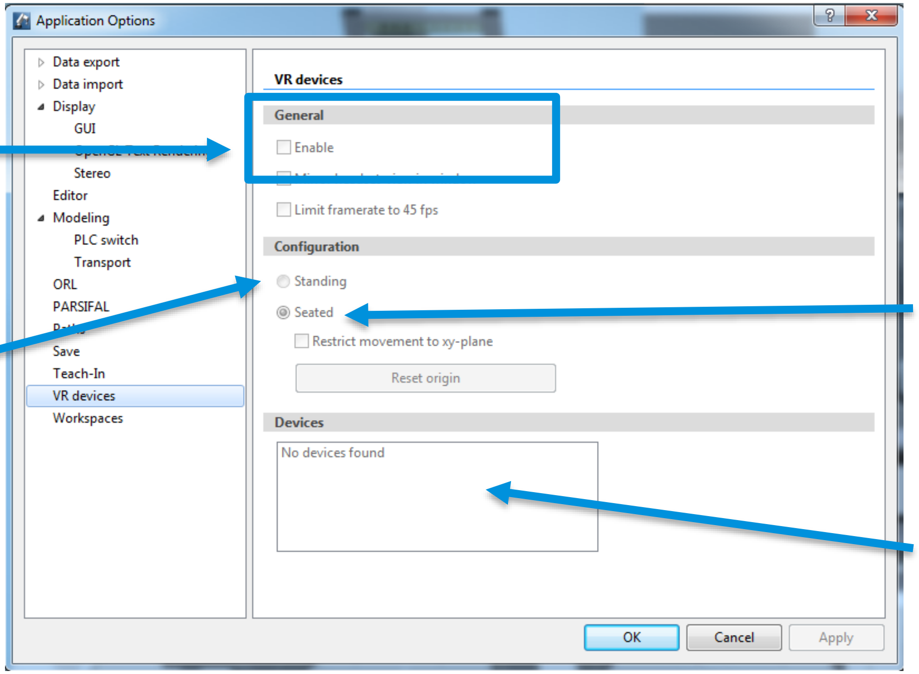
Configure VR mode of CIROS

1. Select File → Application options.
2. Configure VR devices.



Enable VR glasses

"Standing" corresponds to SteamVR's "Room Scale"



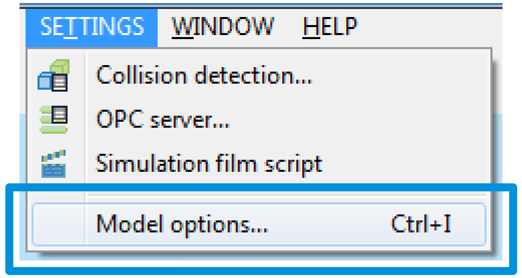
"Seated" should be used in cases in which "Standing Only" has been selected during SteamVR's room setup

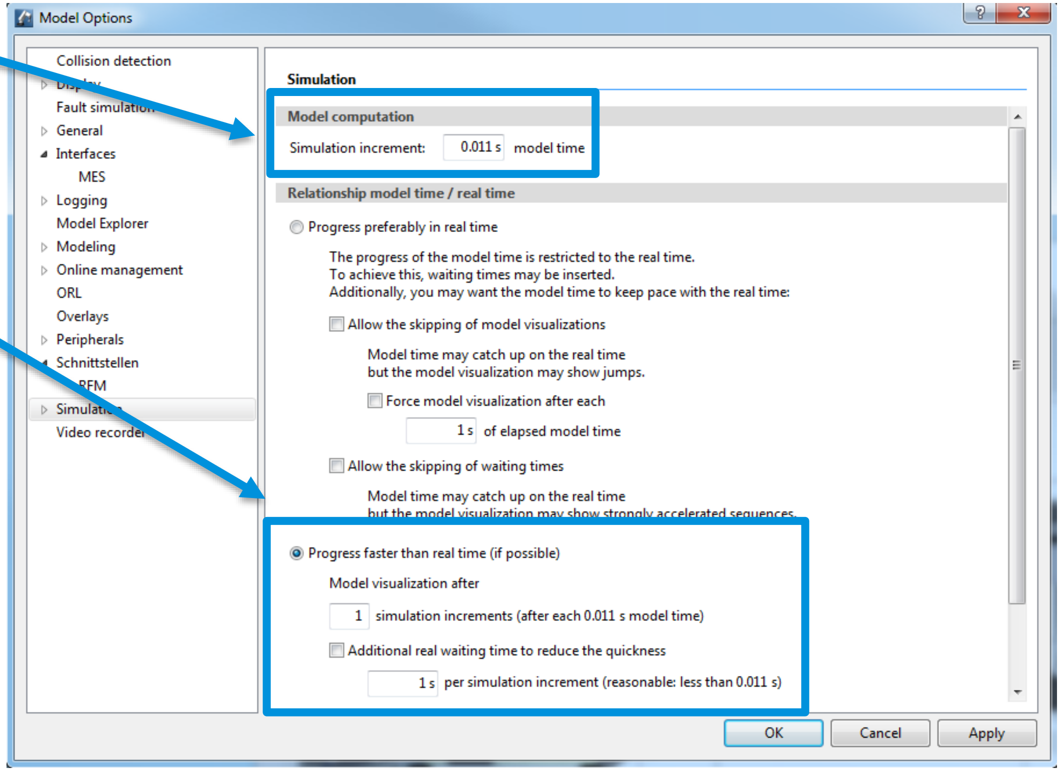
List of available/connected VR sets

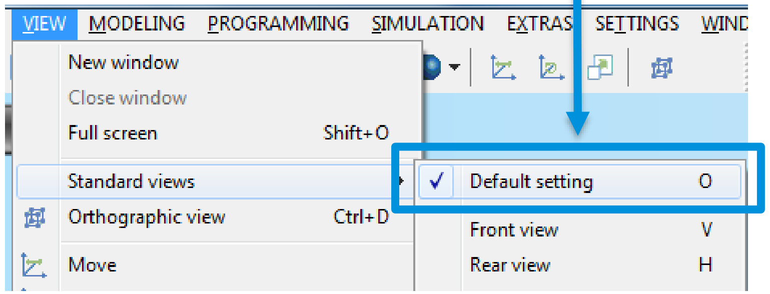
Setting up VR Glasses

Configure VR mode of CIROS

3. Select Settings → Model options
4. Set the simulation increment to 0.011s, since the HTC Vive can perform 90fps
5. Let the simulation run as fast as possible
6. Switch the standard views to default setting



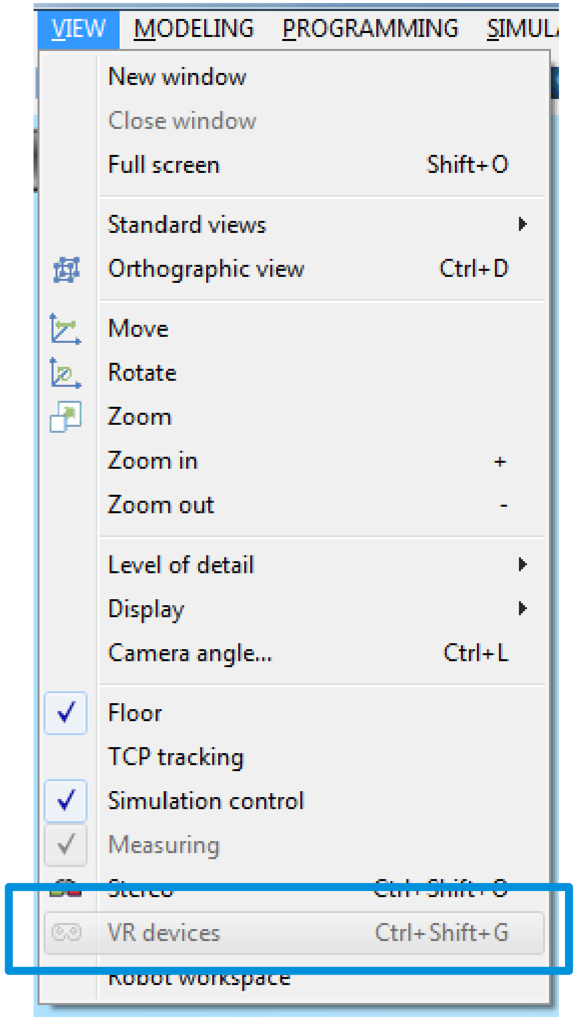




Setting up VR Glasses

Activate VR mode of CIROS and start the simulation

- Enable the connected VR device and start the simulation
- **Remark: It is highly recommended to disable all shadow calculations to have enough computational power for a "smooth" visualization!**

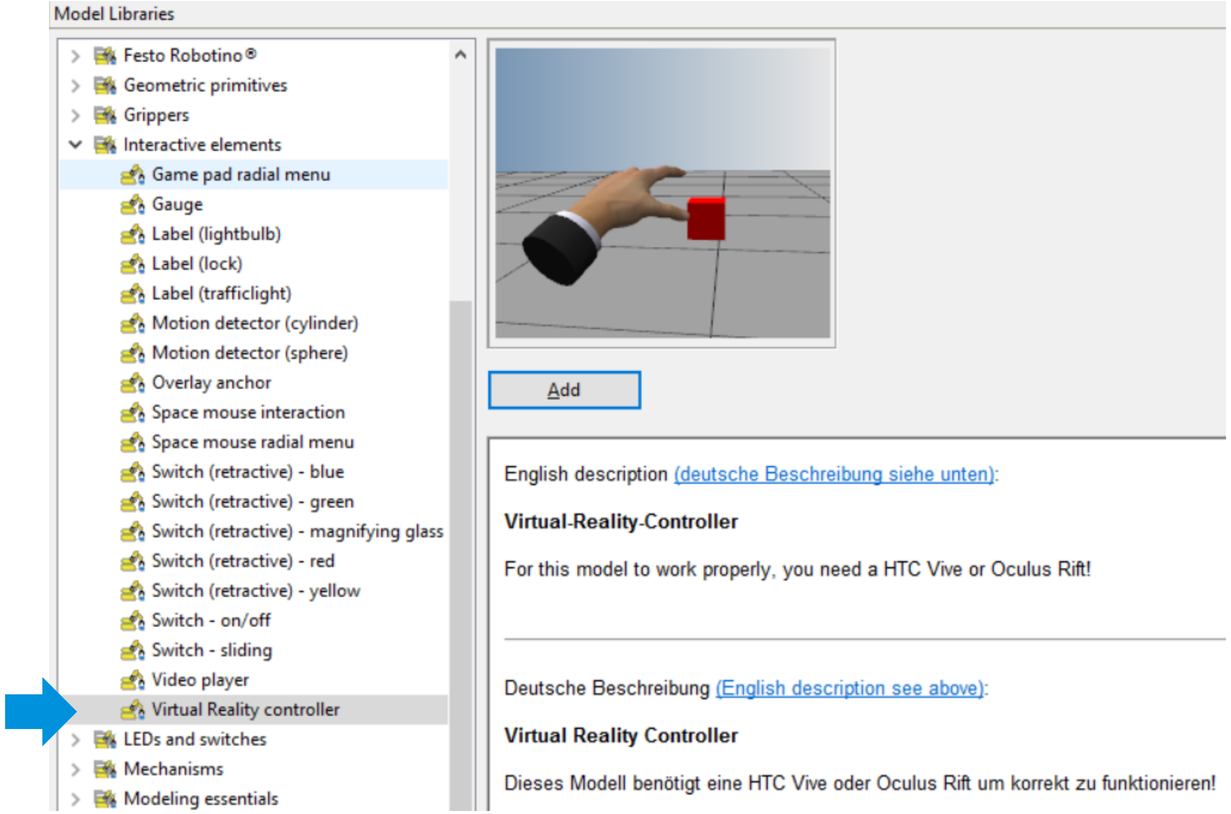


FESTO

Interact with Model

- To interact with the model, for example, click on the buttons, start and stop simulation, etc. An interactive element has to be added.

1. Go to “Modelling → Model libraries”.
2. Expand “Interactive elements”.
3. Add “Virtual Reality controller”.



- **Note:** “Simulation Control” has to be shown in “Model Window” to be able to control simulation in VR.

Advanced

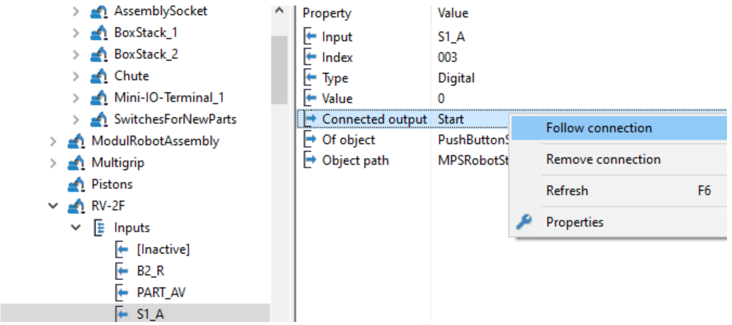
FESTO

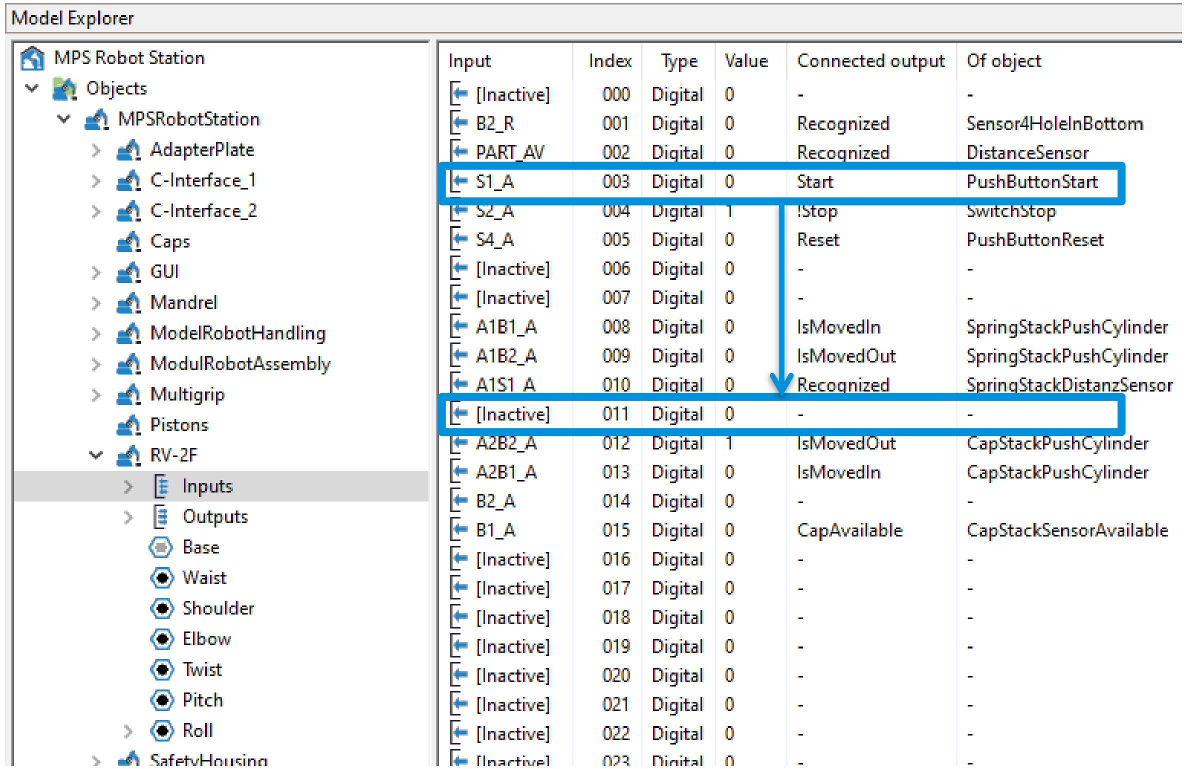
Move I/O Address

Example: Moving digital input S1_A from bit 3 to 11 (1)

1. Open “Model Explorer”.
2. Go to the input to be moved.
   MPS Robot Station → Objects → MPSRobotStation → RV-2F → Inputs
3. Right click on destination address, and select “Rename”.
4. Assign a name, note that names of the cannot repeat in the whole model, add an extension such as “_1” at the end if needed.

5. If the input is connected to an output, it has to be reconnected. Double click on original address to open it, right click on “Connected output” and select “Follow connection”.





Move I/O Address

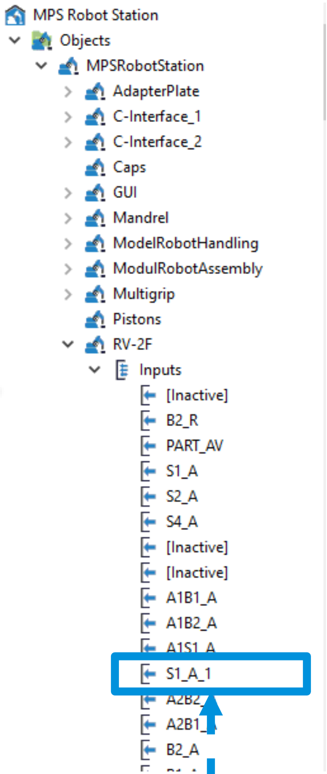
Example: Moving digital input S1_A from bit 3 to 11 (2)

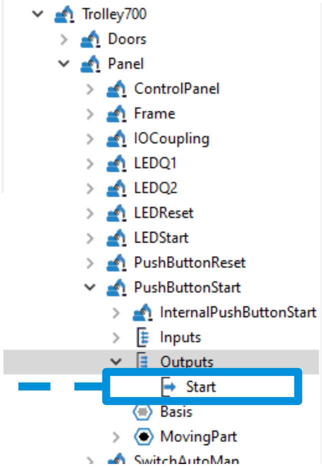
6. Now the connected output is shown, select the output, drag and drop it to the new address to connect them.
   In Model Explorer, drag "MPS Robot Station → MPSRobotStation → Trolley700 → Panel → PushButtonStart → Outputs → Start" and drop to "MPS Robot Station → MPSRobotStation → RV-2F → Inputs → S1_A_1"
7. Deactivate the old address by right click, select Edit → Deactivate.

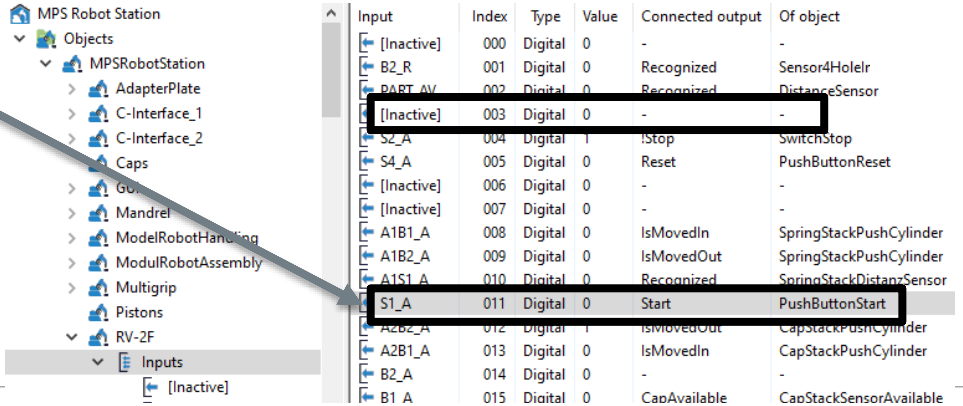
[Inactive]	003	Digital	0	-	-

8. Rename the destination input back.
9. Update the programs which use the bit.

```
Def Io SS1     = Bit,3  Bit,11
```







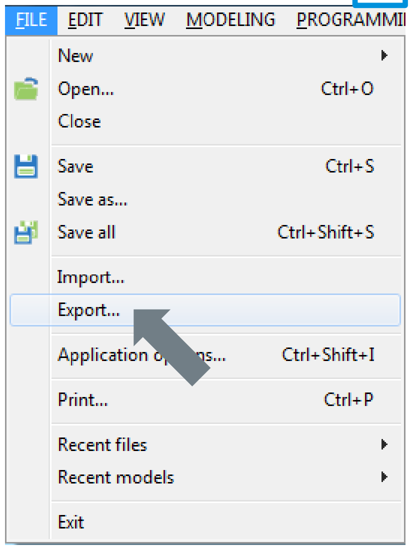
Video tutorial: 43_MoveIOs.mp4

FESTO

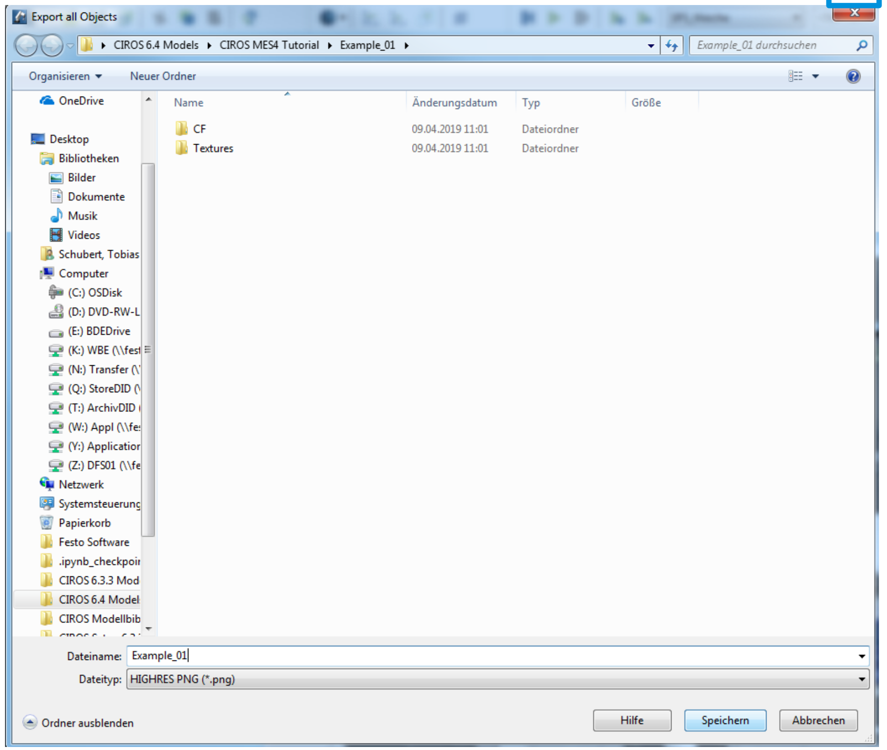
Export as High-Resolution Images

- File → Export enables the user to save the content of the so-called “first” view window as a high-resolution PNG file.

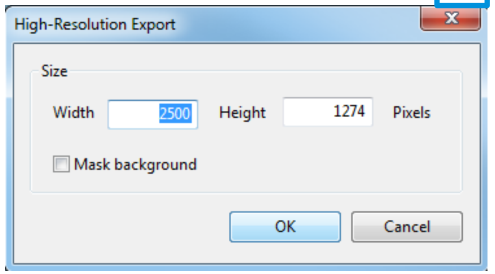
1



2



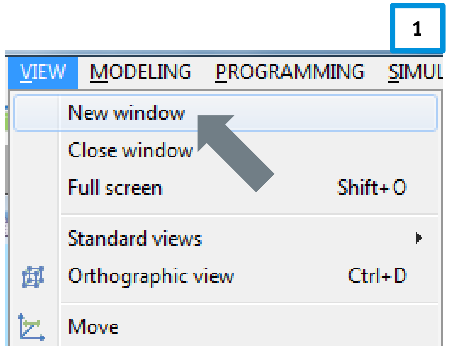
3



4

Multiple View Windows

- CIROS is not restricted to a single view window but offers the possibility to show a couple of view windows concurrently.
- To open a new window, select View → New window.
- Each view might have a different perspective onto the scene.



First / main view window

Additional view windows

CIROS Starter

Calling CIROS model with URL

Possible action	Parameter	Explanation
Start simulation	StartSimulation	Starts the simulation.
Activate full screen	ActivateFullScreen	Runs CIROS in in full screen mode.
Use single instance	UseSingleInstance	CIROS uses an existing instance to open the new model. A new instance is only started if no instance of CIROS is running yet.

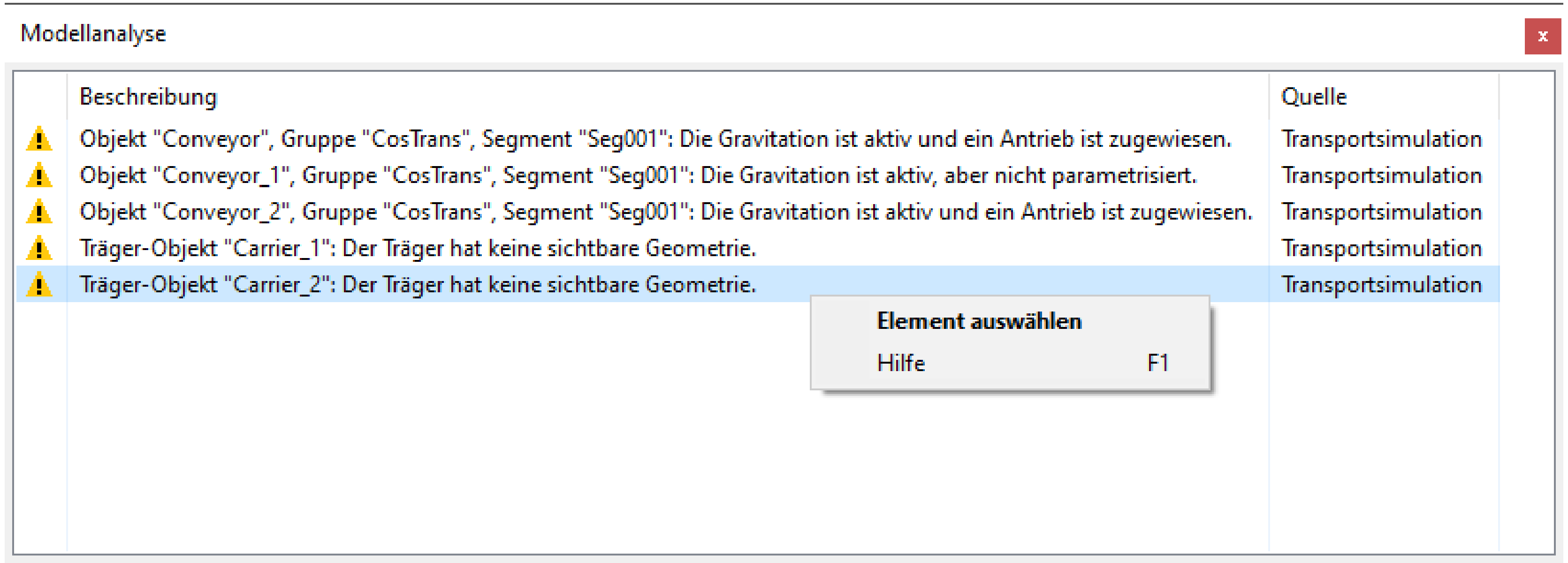
URL format:
“StartCIROS: ‹path to model›?Parameter1&Parameter2”

Example:
"StartCIROS:Example Models\Small Demos for Learning\Robot Assembly with UR5\Robot Assembly with UR5.modx?ActivateFullScreen&UseSingleInstance"

Model Analysis

Extras → Model analysis → Whole model

- Help in problem diagnosis and finding a solution

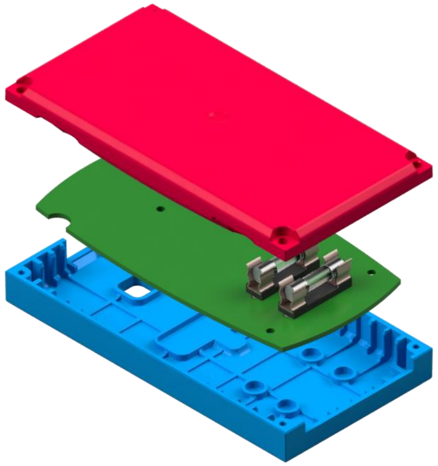


Note: Picture in German.

CIROS Part Number for CP System

“PartNrReplikator.py” replicates the part in simulation. “PartNrTranslation.py” translates MES4 part number to CIROS part number.

Digit 6 Special parts		Digit 5 – 3 Parts		Digit 2 Fuses		Digit 1 Front cover colours		Digit 0 Back cover colours	
Pallet	1	Unprocessed front cover	1	Front fuse	1	Black	0	Black	0
Front cover, raw block	2	Front cover	2	Rear fuse	2	Grey	1	Grey	1
Back cover, not pressed on front cover	3	PCB	4			Blue	2	Blue	2
Turn part	4	Back cover	8			Red	3	Red	3
Boxes	5	Label on front cover	16			Not used	4	Not used	4
Not used	6	Label on back cover	32			Not used	5	Not used	5
Not used	7	Not used	64			Not used	6	Not used	6
Not used	8	Not used	128			Not used	7	Not used	7
Not used	9	Not used	256			Not used	8	Not used	8



<u>Example:</u> see picture

0 | 2+4+8 | 1+2 | 2 | 3 = 0014323 = **14323**

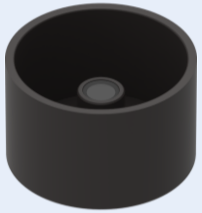
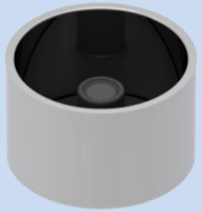
Create Own Part

Task: Add a MPS Part from CP-L-SOURCE

- There are some parts supported from CP-System model libraries. It is possible to create own part to work in CP environment in CIROS.
- User can import the CAD drawing of their parts and test it in virtual environment before implementing it on the real system.
- Steps to create own part will be explained with an example of adding a Festo MPS red housing with RFID and black MPS cap to a CP-L-CONVEYOR via a CP-L-SOURCE.

Create Own Part

MPS part numbers according to FactoryViews[1]

Part number	Part	Image
3001	Red MPS housing with RFID	
3002	Black MPS housing with RFID	
3003	Silver MPS housing with RFID	

Part number	Part	Image
3020	Black cap	
3080	Cap with integrated micro-controller	

[1] FactoryViews is the new software bundle for MES and web based services. It contains MES4 v3.

Create Own Part

Steps to create an own part (1)

1. Prepare CAD drawing of the parts in supported format. (→)
2. Define MES4 / FactoryViews part numbers. (→)
3. Define CIROS part numbers. (→)
4. In CIROS, create a new project and import the CAD drawing. (→)
5. If needed, optimize the geometry of the CAD drawing to reduce its size and rendering requirement in simulation. (→)
6. Rename the object and configure its properties, such as object type, gripper points and grip points. (→) (→)
7. Define coordinates of gripper points and grip points in the model, so the parts are replicated and snapped in correct place in simulation. (→)
8. Move the configured object to Templates by drag and drop. (→)
9. Insert following CP-System Models from Model Libraries and snap them to place.
   - CP-L-CONVEYOR, CP-L-SOURCE and CP-L-SINK
10. Modify following files to add new part numbers.
    ‹project›\CF\py\PartNrTranslator.py (→) and ‹project›\CF\py\PartNrReplikator.py (→)
11. Modify template "Palette" to allow it to take along new parts during transportation. (→)
12. Add following objects to "Objects → Werkstuecke". These are the meta objects.
    - Housing, Pistons and Caps
13. Save all, close CIROS and restart CIROS. Then, open the project and test the new parts added.

Create Own Part

Prepare CAD drawing of the parts in supported format.

- CIROS supported formats are as follow:
  - 3ds Max
  - AutoCAD DXF
  - Autodesk
  - Blender
  - Collada
  - IGES
  - PointCloud
  - STEP
  - STL
  - VRML
  - Wavefront Object

Create Own Part

Define MES4 / FactoryViews part numbers.

Part number	Part
1	MPS housing red
2	MPS housing black
3	MPS housing silver
3001	Red MPS housing with piston containing RFID chip
3002	Black MPS housing with piston containing RFID chip
3003	Silver MPS housing with piston containing RFID chip
3020	MPS cap
13001	Red MPS housing + piston + cap
13002	Black MPS housing + piston + cap
13003	Silver MPS housing + piston + cap

Create Own Part

CIROS part number as defined in “PartNrTranslation.py”

Digit 6 Special parts		Digit 5 – 3 Parts		Digit 2 Fuses		Digit 1 Front cover / housing colours		Digit 0 Back cover/cap colours	
Pallet	1	Unprocessed front cover	1	Front fuse	1	Black	0	Black	0
Front cover, raw block	2	Front cover	2	Rear fuse	2	Grey	1	Grey	1
Back cover, not pressed on front cover	3	PCB	4			Blue	2	Blue	2
Turn part	4	Back cover	8			Red	3	Red	3
Boxes	5	Label on front cover	16			Not used	4	Not used	4
Not used	6	Label on back cover	32			Not used	5	Not used	5
Not used	7	**MPS housing**	**64**			Not used	6	Not used	6
Not used	8	**MPS piston**	**128**			Not used	7	Not used	7
Not used	9	**MPS cap**	**256**			Not used	8	Not used	8

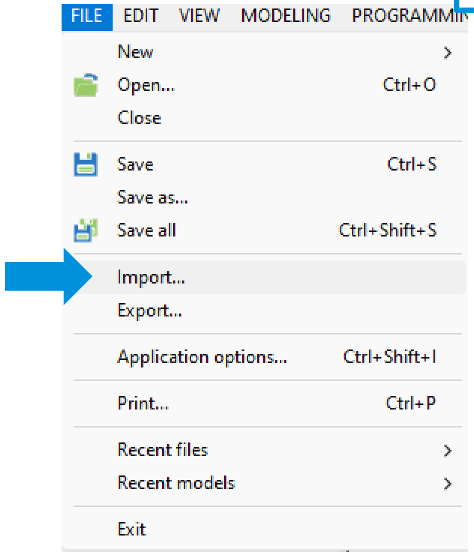
Example: see picture

0	64+128+256	0	3	0	= 0 448 0 3 0 = **448030**

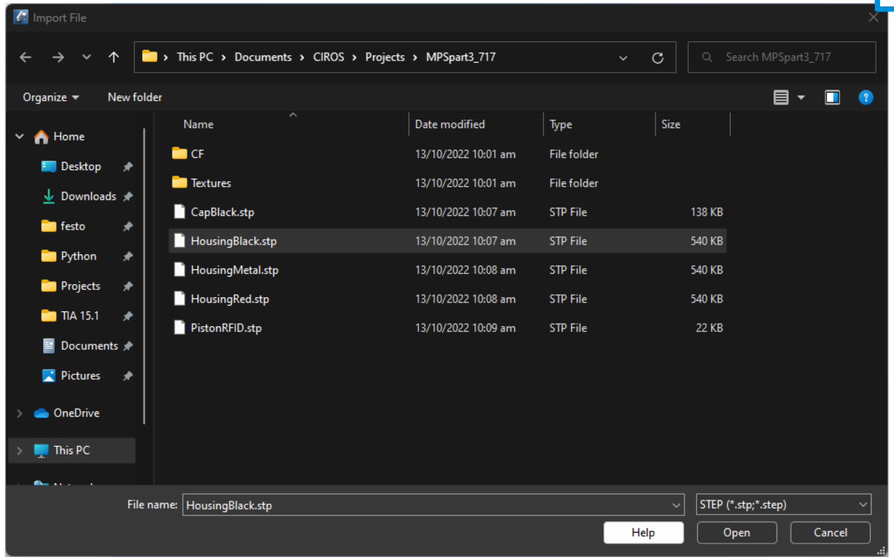
Create Own Part

Import CAD drawing

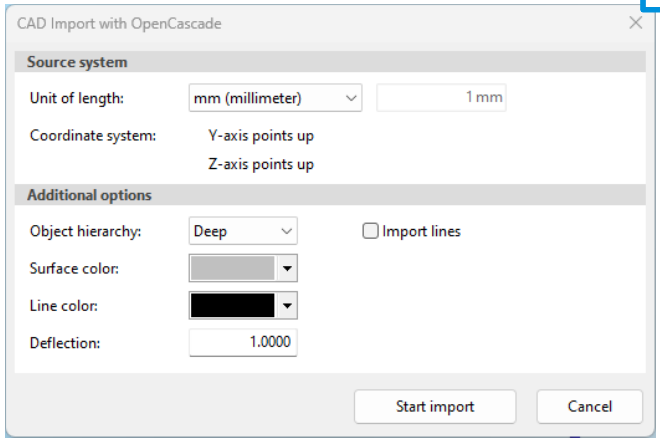
1



2



3

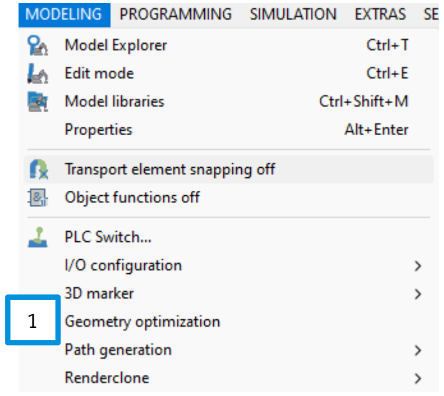


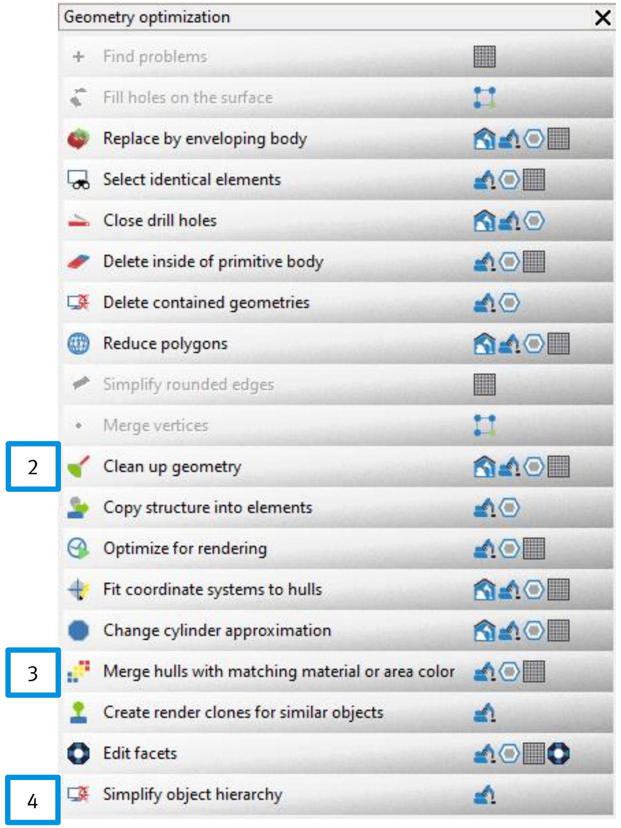
Create Own Part

Optimize geometry

1. Select the imported CAD object in Model Explorer.
2. Click Modeling → Geometry optimization [1].
3. Run the required optimizations.

- In the example, following optimizations were carried out.
  - Clean up geometry [2]
  - Merge hulls with matching material or area color [3]
  - Simplify object hierarchy [4]





Create Own Part

Objects' properties defined in example.

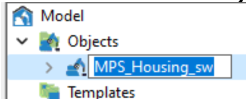
Part name	Meta object	Object type	Gripper points	Grip points
MPS_Housing_rt [1]	Housing	WS	GripPistonRed GripCapRed	HousingRed
MPS_Housing_sw [2]	Housing	WS	GripPistonBlack GripCapBlack	HousingBlack
MPS_Housing_gr [3]	Housing	WS	GripPistonSilver GripCapSilver	HousingSilver
MPS_Piston	Pistons	WS		PPiston
MPS_Cap	Caps	WS		CapStackBlackCap

[1] rt = red (German: rot)
[2] sw = black (German: schwarz)
[3] gr = grey, in this case silver (German: grau)

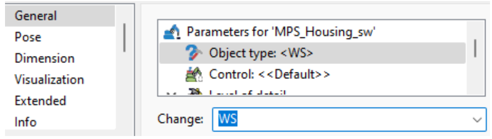
Create Own Part

Configure object’s properties in CIROS (1)

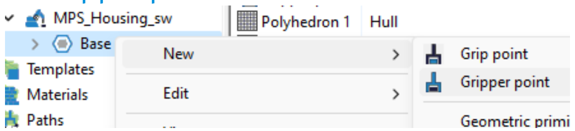
1. Rename the object in Model Explorer.



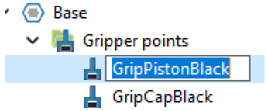
2. Configure object type.
   1. In Properties, select General → Object type.
   2. In Change field, type in ‘WS’.



3. Add gripper points.
   1. In Model Explorer, right click on MPS_Housing_sw → Base and select New → Gripper point.



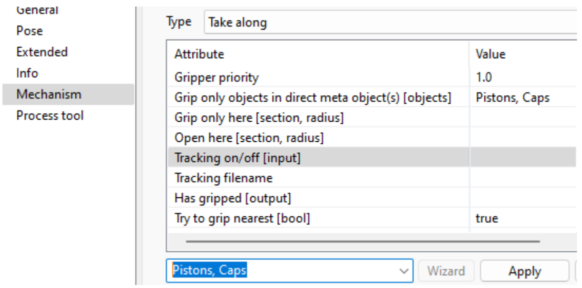
   2. Rename the gripper points.



4. Configure gripper points.
   1. Select gripper point. In Properties → Mechanism, select Type as Take along.
   2. Set value of Grip only objects in direct meta object(s) to the object laying in or on top of the part. Click Apply.

   Example: Black MPS Housing, the value is Pistons, Caps. Because these parts lay in and on top of the housing.

   Hint: The value can be written directly in the field below.

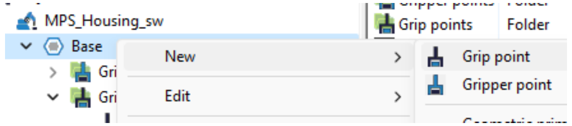


FESTO

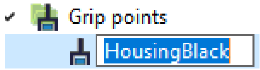
Create Own Part

Configure object's properties in CIROS (2)

5. Add grip point.
    1. Right click on Base, select New → Grip point.



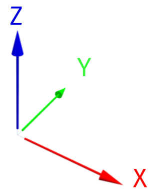
    2. Rename the grip point.



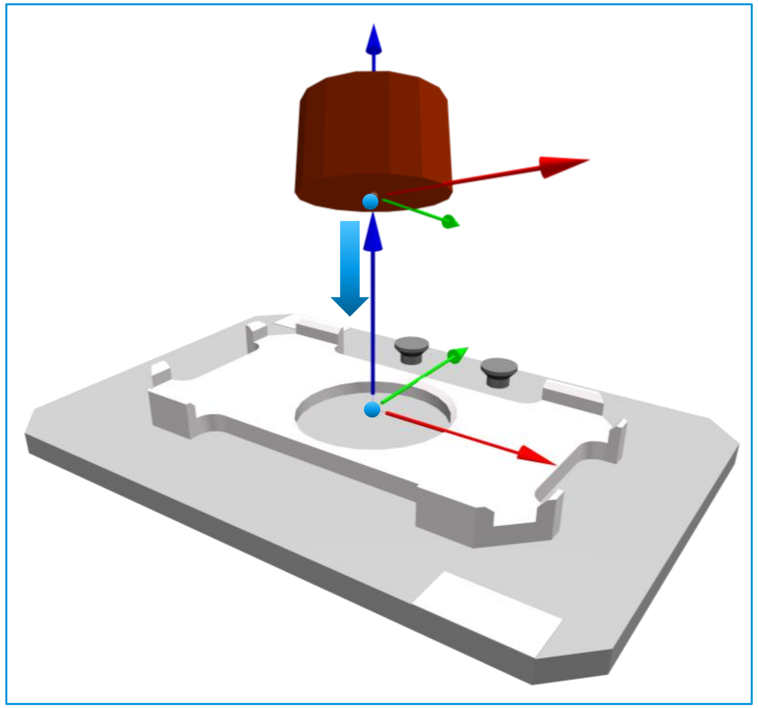
Create Own Part

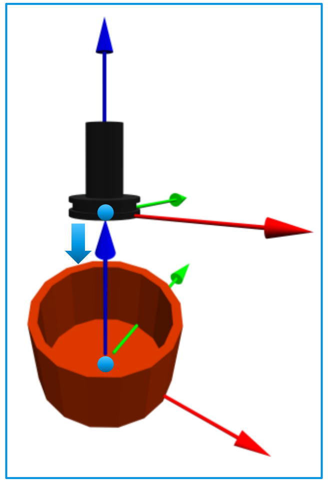
Define coordinates of gripper points and grip points.

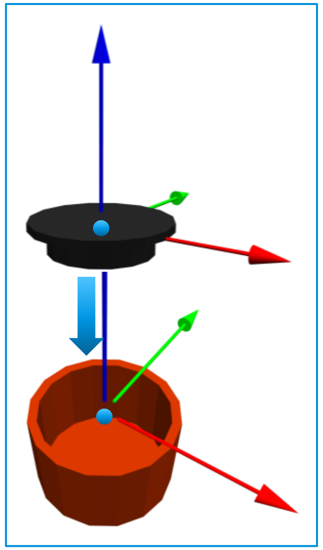
- The x, y, z-coordinates of the parts have to match, so that the parts replicated can be snapped in the right place.
- X, y, z-coordinates in CIROS are highlighted in different colours.



- Coordinates of gripper and grip points can be moved in Properties → Pose.





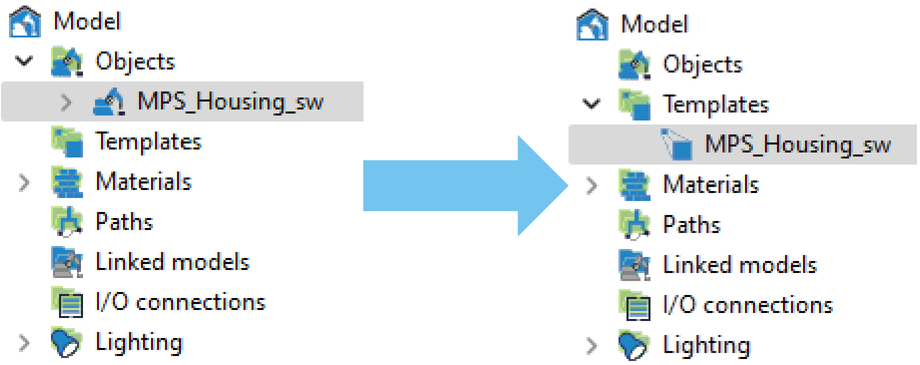


FESTO

Create Own Part

Move configured object to Templates.

- Objects can be moved between 'Objects' and 'Templates' in Model Explorer by drag and drop.



FESTO

Create Own Part

"PartNrTranslation.py" description

- "PartNrTranslation.py" translate MES4 part number to CIROS part number.
- 2D array "PN_by_color" sorts the part by colour.

```
# MES Part number sorted by colour [black, grey, blue, red]
PN_by_color = [
    [10, 0, 0, 0],              #cylinder
    [25, 0, 0, 0],              #palette
    [90, 91, 92, 93],           #back cover unpressed
    [101, 102, 103, 104],       #raw block
    [110, 109, 108, 107],       #undrilled front cover
    [111, 112, 113, 114],       #back cover pressed
    [120, 0, 0, 0],             #PCB
    [121, 0, 0, 0],             #broken, do not use
    [122, 0, 0, 0],             #broken, do not use
    [123, 0, 0, 0],             #broken, do not use
    [210, 310, 410, 510],       #front cover
    [211, 311, 411, 511],       #front cover with PCB
    [212, 312, 412, 512],       #front cover with PCB and front fuse
    [213, 313, 413, 513],       #front cover with PCB and rear fuse
    [214, 314, 414, 514],       #front cover with PCB and both fuses
    [1210, 1310, 1410, 1510],   #pressed front and back cover
    [1211, 1311, 1411, 1511],   #pressed front and back cover with PCB
    [1212, 1312, 1412, 1512],   #pressed front and back cover with PCB and front fuse
    [1213, 1313, 1413, 1513],   #pressed front and back cover with PCB and rear fuse
    [1214, 1314, 1414, 1514],   #pressed front and back cover with PCB and both fuses
    ]
```

- List "CirosPN_by_PNsw" translates MES part number to CIROS part number. Only part number of black part is needed here.

```
# Translate MES part number of black parts to CIROS part number
CirosPN_by_PNsw = {
      10: 4000000,
      25: 1000000,
      90: 3000000,
     101: 2000000,
     110: 1000,
     111: 8000,
     120: 4000,
     121: 4100,
     122: 4200,
     123: 4300,
     210: 2000,
     211: 6000,
     212: 6100,
     213: 6200,
     214: 6300,
    1210: 10000,
    1211: 14000,
    1212: 14100,
    1213: 14200,
    1214: 14300
    }
```

- **Attention:** Part row of "PN_by_color" must match with "CirosPN_by_PNsw" for the script to find the correct part!

Create Own Part

Add new parts to “<project>\CF\py\PartNrTranslation.py”

1. Add MES part numbers to “PN_by_color”.

```
# MES Part number sorted by colour [black, grey, blue, red]
PN_by_color = [
    [2, 3, 0, 1],               ''' NEW! ''' #MPS housing
    [10, 0, 0, 0],              #cylinder
    [25, 0, 0, 0],              #palette
    [90, 91, 92, 93],           #back cover unpressed
    [101, 102, 103, 104],       #raw block
    [110, 109, 108, 107],       #undrilled front cover
    [111, 112, 113, 114],       #back cover pressed
    [120, 0, 0, 0],             #PCB
    [121, 0, 0, 0],             #broken, do not use
    [122, 0, 0, 0],             #broken, do not use
    [123, 0, 0, 0],             #broken, do not use
    [210, 310, 410, 510],       #front cover
    [211, 311, 411, 511],       #front cover with PCB
    [212, 312, 412, 512],       #front cover with PCB and front fuse
    [213, 313, 413, 513],       #front cover with PCB and rear fuse
    [214, 314, 414, 514],       #front cover with PCB and both fuses
    [1210, 1310, 1410, 1510],   #pressed front and back cover
    [1211, 1311, 1411, 1511],   #pressed front and back cover with PCB
    [1212, 1312, 1412, 1512],   #pressed front and back cover with PCB and front fuse
    [1213, 1313, 1413, 1513],   #pressed front and back cover with PCB and rear fuse
    [1214, 1314, 1414, 1514],   #pressed front and back cover with PCB and both fuses
    [3002, 3003, 0, 3001],      ''' NEW! ''' #MPS housing with RFID piston
    [13002, 13003, 0, 13001],   ''' NEW! ''' #MPS housing with piston and cap
    ]
```

2. Add part numbers to “CirosPN_by_PNsw”.

```
# Translate MES part number of black parts to CIROS part number
CirosPN_by_PNsw = {
        2: 64000,     ''' NEW! '''
       10: 4000000,
       25: 1000000,
       90: 3000000,
      101: 2000000,
      110: 1000,
      111: 8000,
      120: 4000,
      121: 4100,
      122: 4200,
      123: 4300,
      210: 2000,
      211: 6000,
      212: 6100,
      213: 6200,
      214: 6300,
     1210: 10000,
     1211: 14000,
     1212: 14100,
     1213: 14200,
     1214: 14300,
     3002: 192000,    ''' NEW! '''
    13002: 448000     ''' NEW! '''
    }
```

- Note that the part rows of “PN_by_color” matches “CirosPN_by_PNsw”.

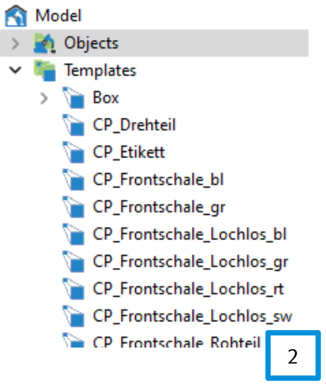
Create Own Part

“PartNrReplikator.py” description

- “PartNrReplikator.py” takes the translated CIROS part number and replicates the part in the model from templates.
- The script also configures gripper points and grip points.

- List “farbe_ID” matches colour ID[1] in “PartNrTranslator.py” with colour add on at the end of templates name[2].

```
farbe_ID = {
            0: '_sw',    #black
            1: '_gr',    #grey
            2: '_bl',    #blue
            3: '_rt'     #red
           }
```



- “bauteil_ID” contains detailed information of each CIROS ID.

```
# Lists below contain CIROS ID with
#                      (Template name without    , Meta object       ,GPP-Name in case none is given ,gripped by)
#                        colour addition                                                              which part
bauteil_ID = {
                  1: ('CP_Frontschale_Lochlos' , 'Frontschalen'    ,'Bauchlage'                     ,[]     ),  #undrilled front cover
                  2: ('CP_Frontschale'         , 'Frontschalen'    ,'Bauchlage'                     ,[]     ),  #front cover
                  4: ('CP_Platine'             , 'Platinen'        ,'GPP_Platine'                   ,[1,2]  ),  #PCB
                  8: ('CP_Rueckschale'         , 'Rueckschalen'    ,'GPP_Rueckschale'               ,[1,2]  ),  #Back cover
                 16: ('CP_Etikett'             , 'Etiketten'       ,'GPP_Etikett'                   ,[1,2]  ),  #Label, sticked on front cover
                 32: ('CP_Etikett'             , 'Etiketten'       ,'GPP_Etikett'                   ,[8]    ),  #Label, sticked on back cover
                 64: ('', '', '', ()),
                128: ('', '', '', ()),
                256: ('', '', '', ()),
                512: ('CP_Schmelzsicherung'    , 'Sicherungen'     ,'GPP_SicherungA'                ,[4]    ),  #Front fuse
               1024: ('CP_Schmelzsicherung'    , 'Sicherungen'     ,'GPP_SicherungB'                ,[4]    ),  #Rear fuse
            }
```

- “sonderteil_ID” contains detailed information of CIROS ID which are declared as special parts.

```
sonderteil_ID = {
                  1: ('Palette'                , 'Paletten'        ,'Bauchlage'                     ,[]     ),  #empty palette
                  2: ('CP_Frontschale_Rohteil' , 'Frontschalen'    ,'Bauchlage'                     ,[]     ),  #raw block
                  3: ('CP_Rueckschale'         , 'Rueckschalen'    ,'Rueckschale_aufgelegt_Palette',[]     ),  #back cover, unpressed
                  4: ('CP_Drehteil'            , 'Drehteile'       ,'Bauchlage'                     ,[]     ),  #cylinder
                  5: ('Box'                    , 'Boxen'           ,'Box'                           ,[]     ),  #box
            }
```

Create Own Part

Add new parts to “<project>\CF\py\PartNrReplikator.py”

1. Add new parts to list “bauteil_ID”.

```
bauteil_ID = {
        1: ('CP_Frontschale_Lochlos' , 'Frontschalen'    ,'Bauchlage'          ,[]     ), #undrilled front cover
        2: ('CP_Frontschale'         , 'Frontschalen'    ,'Bauchlage'          ,[]     ), #front cover
        4: ('CP_Platine'             , 'Platinen'        ,'GPP_Platine'        ,[1,2]  ), #PCB
        8: ('CP_Rueckschale'         , 'Rueckschalen'    ,'GPP_Rueckschale'    ,[1,2]  ), #Back cover
       16: ('CP_Etikett'             , 'Etiketten'       ,'GPP_Etikett'        ,[1,2]  ), #Label, sticked on front cover
       32: ('CP_Etikett'             , 'Etiketten'       ,'GPP_Etikett'        ,[8]    ), #Label, sticked on back cover
       64: ('MPS_Housing'            , 'Housing'         ,''                   ,[]     ), #MPS housing
      128: ('MPS_Piston'             , 'Pistons'         ,'GripPiston'         ,[64]   ), #MPS piston with RFID chip
      256: ('MPS_Cap'                , 'Caps'            ,'GripCap'            ,[64]   ), #MPS cap
      512: ('CP_Schmelzsicherung'    , 'Sicherungen'     ,'GPP_SicherungA'     ,[4]    ), #Front fuse
     1024: ('CP_Schmelzsicherung'    , 'Sicherungen'     ,'GPP_SicherungB'     ,[4]    ), #Rear fuse
    }
```

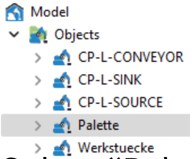
2. In function “replicatePartAt”, under “elif bauteile:”, add “or id == 64”, to allow colour allocation of MPS housings. At default v7.1 script it is at line 123.

```
 97     elif bauteile:
 98         # Schleife über alle Bauteil IDs
 99         # Die IDs sind in der bauteil_ID Liste in aufk
100         for i in range( 0, 11 ):
101             id = 2**i
102 
103             # Falls das Bauteil enthalten ist
104             if bauteile & id:
105                 # Schauen ob übergebenes Objekt gültig
106                 if not repl_Obj:
107                     for nr in bauteil_ID[id][3]:
108                         try:
109                             repl_Obj = teile[nr]
110                             break
111                         except:
112                             repl_Obj = None
113                 if not repl_Obj:
114                     print( "DE: Kein Replikator-Objekt
115                     return
116 
117                 # Namen für GPP an aktuellem repl_Obj
118                 repl_GPP = findGPPStartingWith( repl_C
119                 if repl_GPP == '':
120                     print( "DE: Kein Greiferpunkt für
121 
122                 # Template + Farbe bestimmen
123                 template = bauteil_ID[id][0]
124                 if id == 1 or id == 2 or id == 64:
125                     template += farbe_ID[farbe_front]
126                 if id == 8:
127                     template += farbe_ID[farbe_rueck]
```

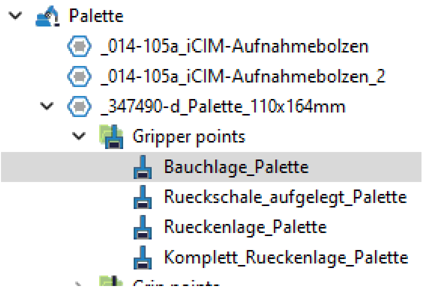
Create Own Part

Modify template “Palette”

1. Move “Palette” from “Templates” to “Objects”.

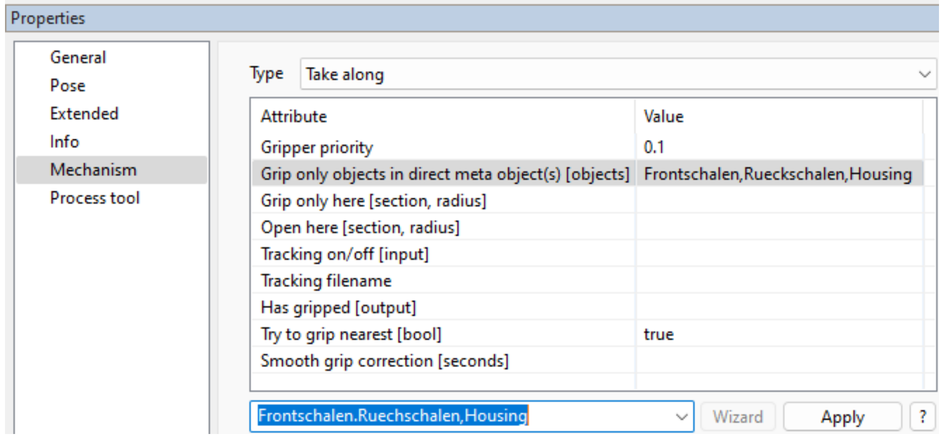


2. Select “Palette → _347490-d_Palette_110x164mm → Gripper points → Bauchlage_Palette”.



3. In Properties window, change the value of “Grip only objects in direct meta object” to “Frontschalen,Ruechschalen,Housing” and click Apply.

Note: The value can be typed directly in the field below.



4. Move “Palette” back to “Templates”.

Create Own Model Library

- There are model libraries with a huge variety of models by default. However, these models are encrypted and not editable.
- It is possible to create own libraries, either from scratch or from modified standard model libraries. These libraries created can be edited from time to time and can also be used as a building block to build own models, just like the standard model libraries.
- In coming tutorials, steps to create own model libraries from modified standard model libraries containing a modified CP-L-CONVEYOR, CP-L-SOURCE and CP-L-SINK which supports MPS workpiece will be shown. Steps to create MPS workpiece in CP environment is shown in previous chapter.

- **Important!** If the library is made from modified standard library, avoid mixing new model library with standard library for an error free simulation.

Create Own Model Library

Elements of a model library

- CF.7z
- Textures.7z
- Model
  - Model.modx
  - Model.png
  - Model.ini
  - Model.html
- InitModBib.modx
- InitModBib.ini
- InitModBib.fin

Create Own Model Library

Steps to create own model library

1. Create CF.7z and Textures.7z.
2. Create InitModBib (*.fin, *.ini and *.modx).
3. Create the models.
4. Link the model library in CIROS.

Create Own Model Library

Create CP.7z and Textures.7z

1. In a chosen window’s path, create an empty folder. This will be the new model library folder.
2. In CIROS Studio, create a CIROS model in the folder created above. Name it InitModBib.modx.
3. Add any model from Festo CP System model library.
4. Delete all Materials in Model Explorer.
5. In ‹project folder›\CF, replace any modified python or irl scripts.
6. Compress following folders with 7-Zip:
    1. ‹project folder›\CF
    2. ‹project folder›\Textures

FESTO

Create Own Model Library

Create InitModBib (*.modx, *.ini and *.fin)

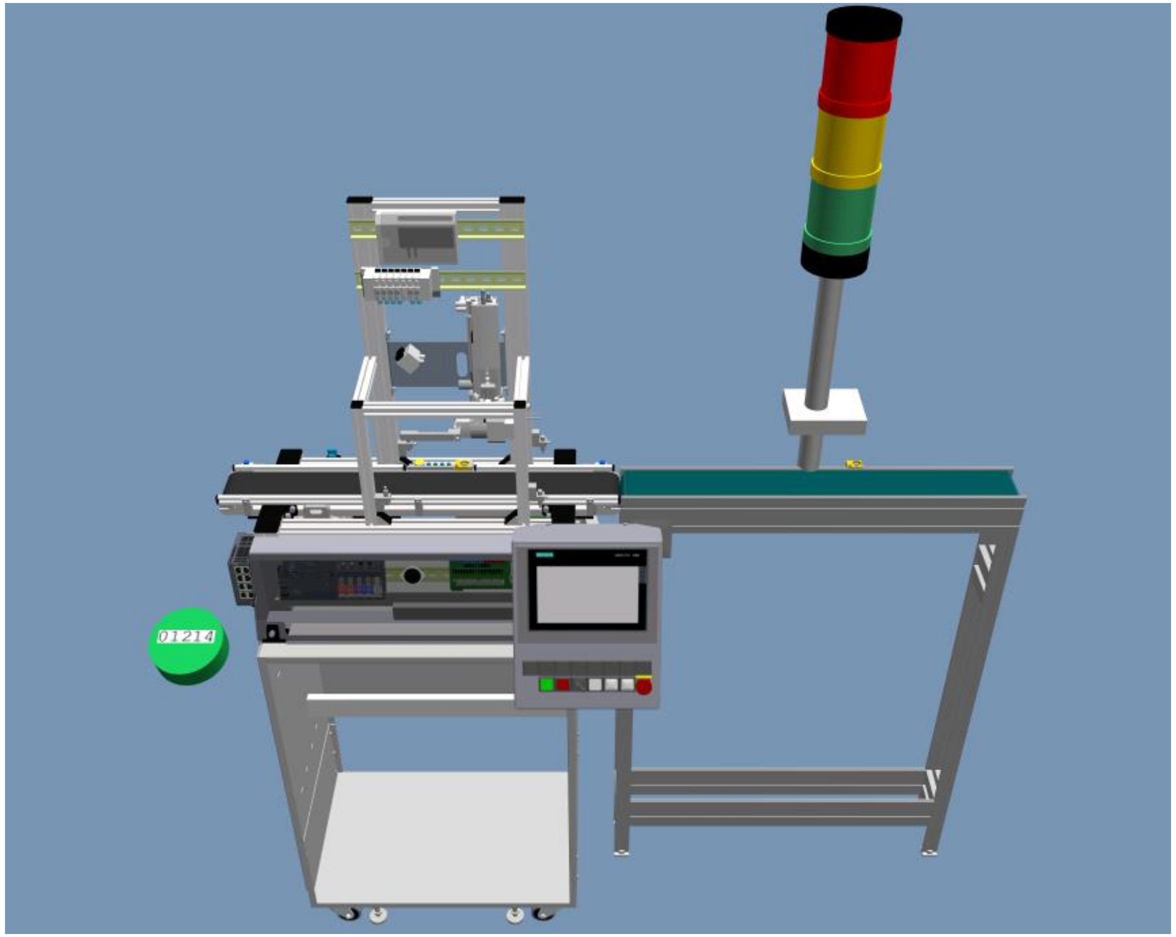
1. Go to CIROS project created (InitModBib.modx).
2. Make sure Model Explorer → Templates contains all the parts required in the new model library.

   Note: Import the part as an object to CIROS model and convert it into template if required.
3. Make sure Model Explorer → Objects → Werkstuecke contains all the meta objects required in new model library.
4. In Model Explorer, delete all objects except “Werkstuecke”.
5. In Model Explorer, delete all materials.
6. Save all.

Steps to Create own Virtual Machine Communicating with MES4

- At the end of this tutorial, a resource consists of a belt with application module traffic light is created (see picture on the right) and is able to work together with a CP-L-CONVEYOR.

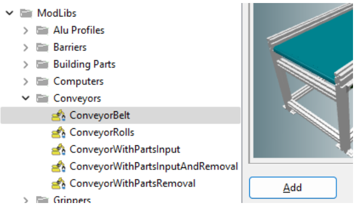
- The module can work in default mode and MES mode.
- In default mode, red, yellow and green LEDs will light up one after another.
- In MES mode, the LED configured in MES will light up.



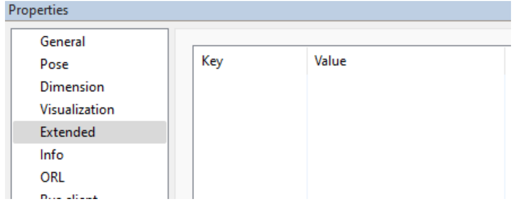
Steps to Create own Virtual Machine Communicating with MES4

Steps to create base module.

1. Insert CP-L-CONVEYOR from Festo CP System model library.
2. Insert ModLibs\Conveyors\ConveyorBelt.

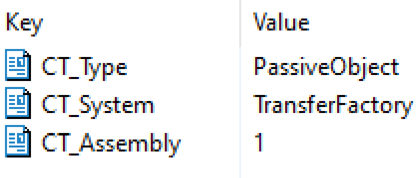


3. Delete all Extended properties in Properties\Extended.



4. Add following extended properties.

Key	Value
CT_Type	PassiveObject
CT_System	TransferFactory
CT_Assembly	1

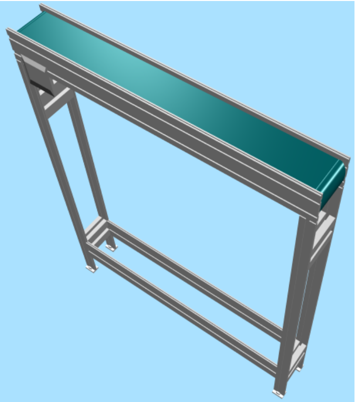


5. Delete following components in ConveyorBelt.
   - LedBackwards
   - LedForwards
   - LedStopped
   - ProfilesTopX
   - All inputs
   - All outputs
6. Go to Properties\General, select object type as Inactive object.
7. Go to Modelling\Geometry optimization.
8. Select Merge hulls with matching material or area color to optimize ConveyorBelt.

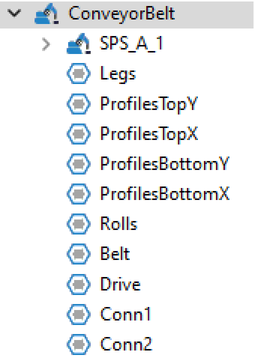
Steps to Create own Virtual Machine Communicating with MES4

Steps to create base module.

7. Change the size of ConveyorBelt in Properties\Dimension.
   - X = 700 mm
   - Y = 120 mm
   - Z = 974.50 mm



10. Copy CP-L-CONVEYOR\SPS_A and paste it under object ConveyorBelt. The object is renamed to SPS_A_1.
11. Change position of SPS_A_1 in world coordinate.
    - X = 350 mm
    - Y = 54.55 mm
    - Z = 975 mm
12. Copy CP-L-CONVEYOR\Conn1 and Conn2 and paste them under ConveyorBelt.



Hint: Use prepared CAD model “ConveyorBelt.stp” to skip step 2, 3, 5 and 7. Please request the CAD model from Festo Didactic.

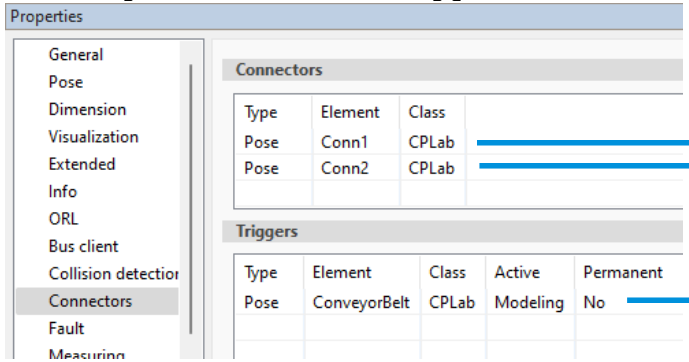
Steps to Create own Virtual Machine Communicating with MES4

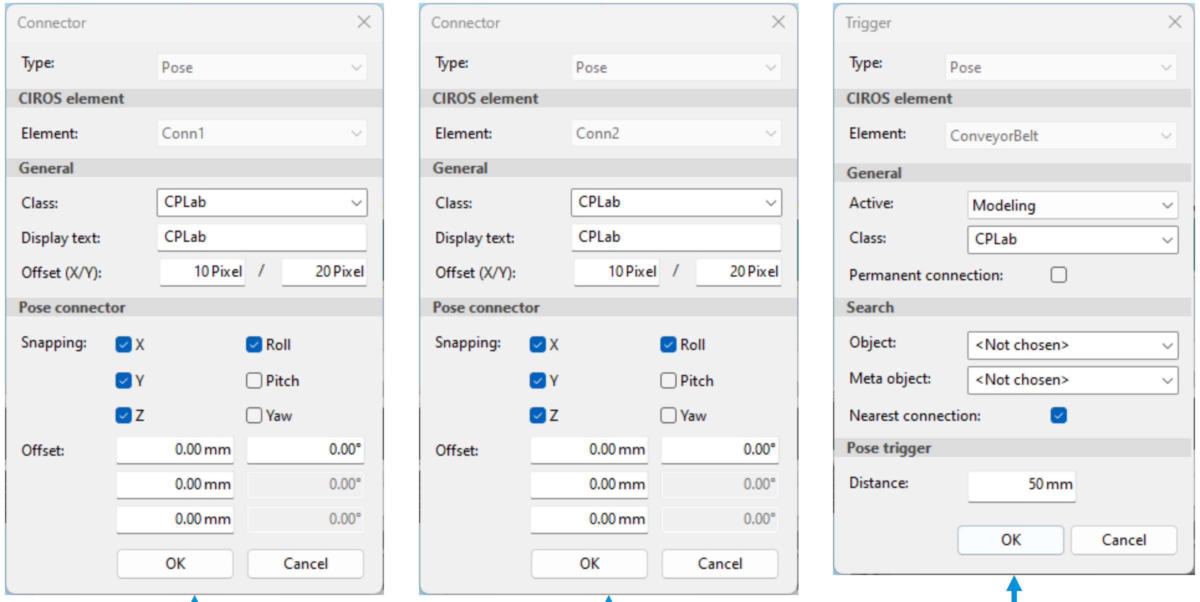
Steps to create base module.

13. Move ConveyorBelt\Conn1 and Conn2 to right and left end of ConveyorBelt respectively.

Object coordinate	X [mm]	Y [mm]	Z [mm]
Conn1	0	54.55	975
Conn2	700	54.55	975

14. Select object ConveyorBelt. Go to Properties\Connectors. Add following connectors and triggers.

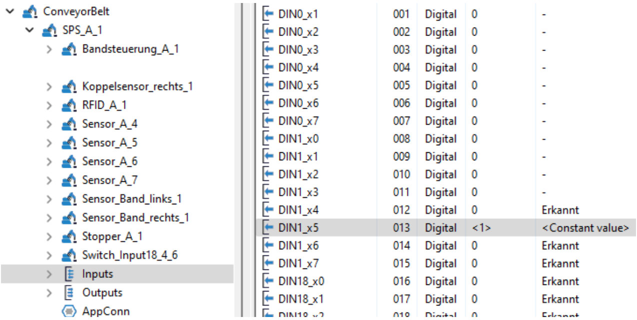




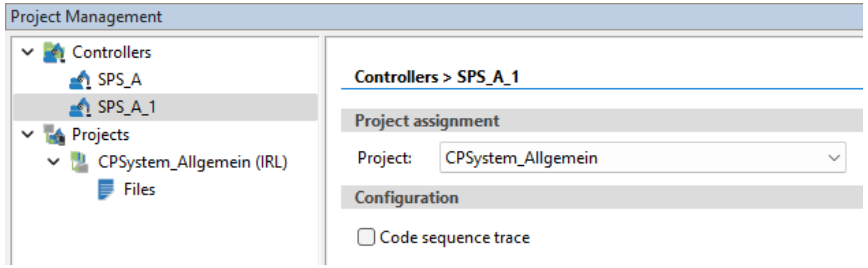
Steps to Create own Virtual Machine Communicating with MES4

Steps to create base module.

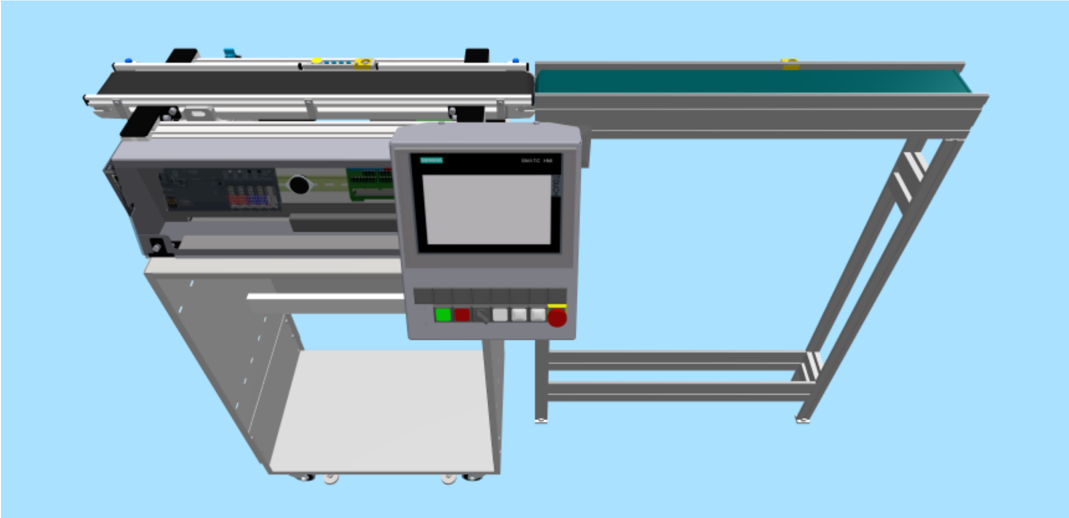
15. Go to SPS_A_1\Inputs, connect DIN1_x5 to 1.



16. Open Project Management.
17. Open project ‹project folder›\CF\CPSystems\CPSystem_Allgemein.prjx.
18. In Controllers\SPS_A_1, assign CPSystem_Allgemein. Do the same for SPS_A.



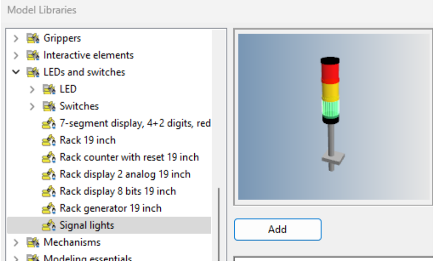
19. Right click on Projects\CPSystem_Allgemein(IRL) and compile the project. Make sure that all projects are compiled successfully.
20. Drag and snap ConveyorBelt to CP-L-CONVEYOR.
21. Base module is created.



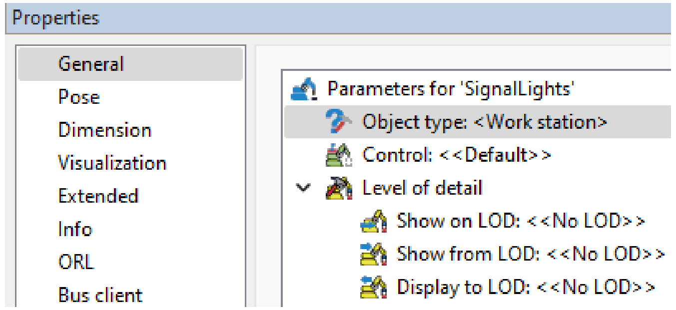
Steps to Create own Virtual Machine Communicating with MES4

Steps to create application module.

1. In model libraries, insert LEDs and switches\Signal lights.

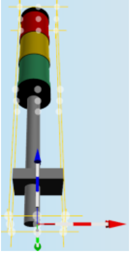


2. Select SignalLights, in Properties\General, change object type to Work station.

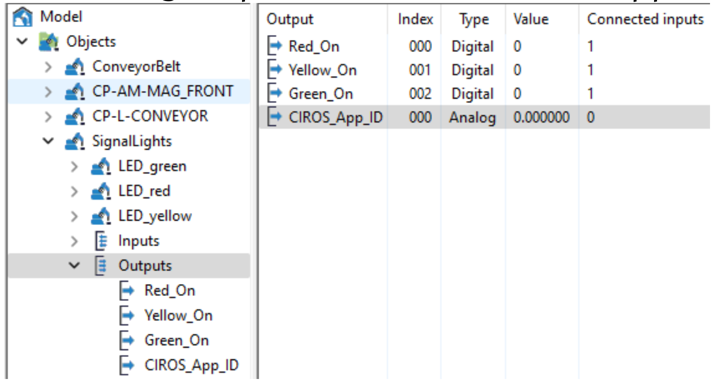


3. Move origin of SignalLights to middle bottom of the object.

   Hint: With help of 3D marker.



4. Change z-coordinate of SignalLights in world coordinate to 975 mm.
5. Add analog output and name it CIROS_App_ID.

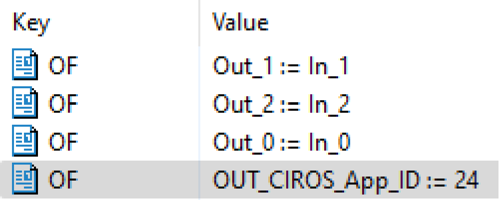


Steps to Create own Virtual Machine Communicating with MES4

Steps to create application module.

6. Select SignalLights, add following extended property.

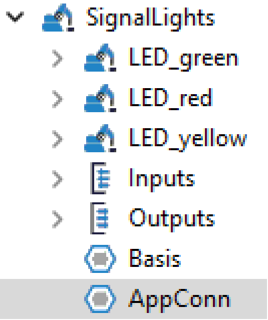
Key	Value
OF	OUT_CIROS_App_ID := 24



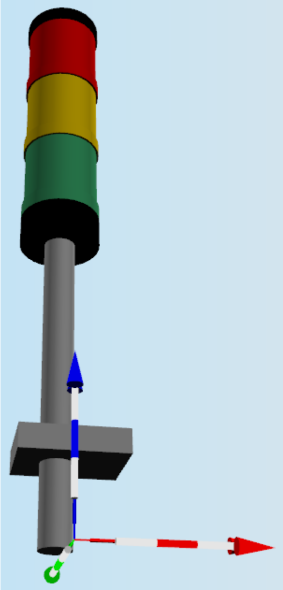
Key	Value
OF	Out_1 := In_1
OF	Out_2 := In_2
OF	Out_0 := In_0
OF	OUT_CIROS_App_ID := 24

7. In CP System model library, add any application module, for example, CP-AM-MAG_FRONT.

8. Copy CP-AM-MAG_FRONT\AppConn and paste it under object SignalLights.



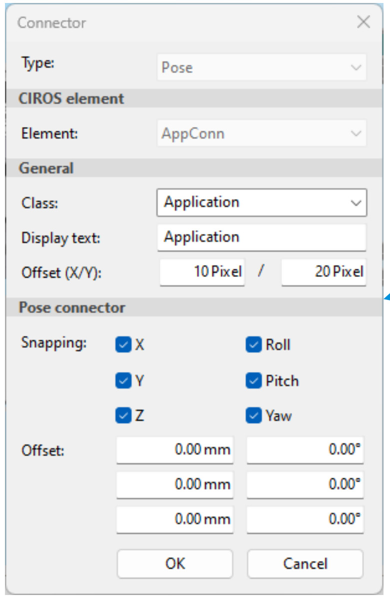
9. Move AppConn pose to front middle of object's bottom plane.

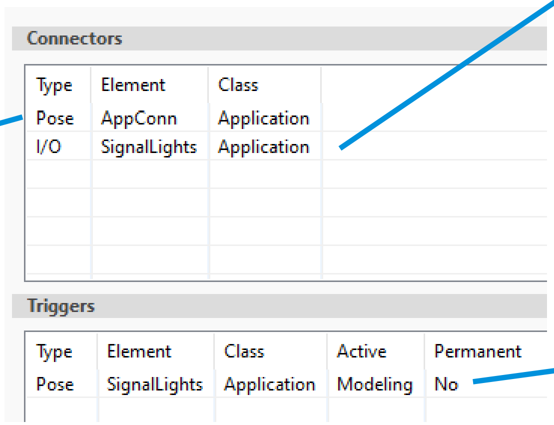


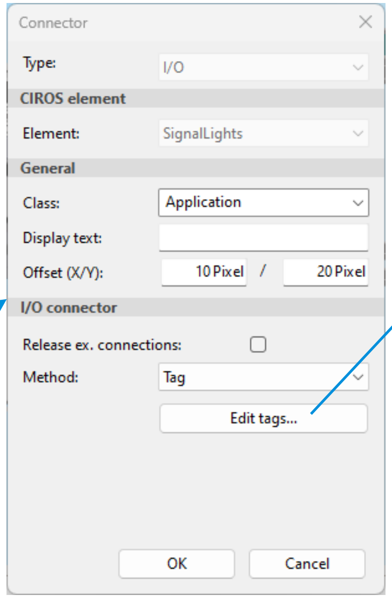
Steps to Create own Virtual Machine Communicating with MES4

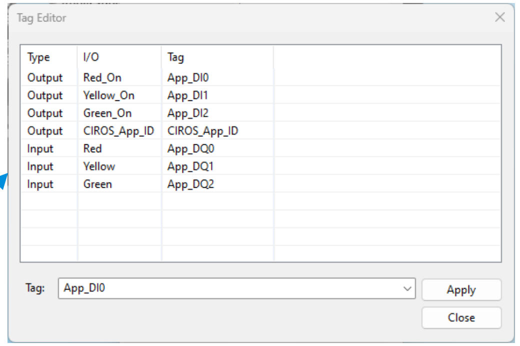
Steps to create application module.

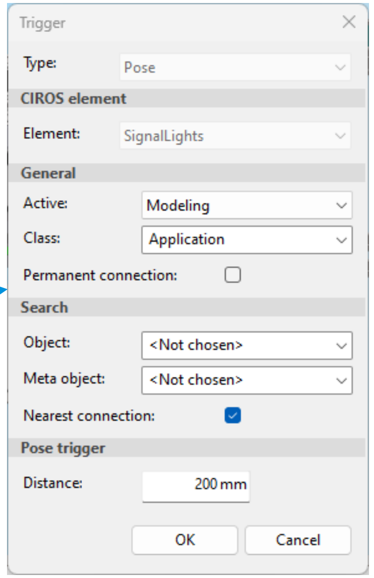
10. Select SignalLights, in Properties\Connectors, configure connectors and triggers as follow.











Steps to Create own Virtual Machine Communicating with MES4

Steps to program virtual machine in IRL

1. Add app ID to "Applikationen.irl".
   1. In "Project Management\Projects\CPSystem_Allgemein(IRL)\Files", open "Applikationen.irl".
   2. In Function "App()", add following line,

      if(applicationID = 24) then return(appSignalLights(mes.Parameter)); endif;

```
FUNCTION App( IN INT: applicationID; IN MESFields : mes ) : int;
BEGIN
        if( applicationID = 0 ) then return( 0 ); endif;
        if( applicationID = 1 ) then return( appBohren( mes.Parameter ) ); endif;
        if( applicationID = 2 ) then return( appWenden( mes.Parameter ) ); endif;
        if( applicationID = 3 ) then return( appPresse( mes.Parameter ) ); endif;
        if( applicationID = 4 ) then return( appMuskelpresse( mes.Parameter ) ); endif;
{       if( applicationID = 5 ) then return( appMagazin( mes.Parameter ) ); endif;}
        if( applicationID = 6 ) then return( appKamera( mes.Parameter ) ); endif;
        if( applicationID = 7 ) then return( appTunnelofen( mes.Parameter ) ); endif;
{       if( applicationID = 8 ) then return( appEntnahme( mes.Parameter ) ); endif;
        if( applicationID = 9 ) then return( appNacharbeit( mes.Parameter ) ); endif;}
        if( applicationID = 10 ) then return( appWerkstueckausgabe( mes.Parameter ) ); endif;
        if( applicationID = 11 ) then return( appEtikettieren( mes.Parameter ) ); endif;
        if( applicationID = 12 ) then return( appMessenAnalog( mes ) ); endif;
        if( applicationID = 13 ) then return( appLager( mes.Parameter ) ); endif;
        if( applicationID = 14 ) then return( appRobMontage(  mes.Parameter ) ); endif;
        if( applicationID = 15 ) then return( appRobCNCBeladen(  mes.Parameter ) ); endif;
        if( applicationID = 16 ) then return( appLagerLab( mes.Parameter ) ); endif;
        if( applicationID = 17 ) then return( appManual( mes.Parameter ) ); endif;
        if( applicationID = 18 ) then return( appPickByLight( mes.Parameter ) ); endif;
        if( applicationID = 19 ) then return( appMagazinFront( mes.Parameter ) ); endif;
        if( applicationID = 20 ) then return( appMagazinBack( mes.Parameter ) ); endif;
        if( applicationID = 21 ) then return( appBoxBufferManual( mes ) ); endif;
        if( applicationID = 22 ) then return( appBoxRobotBypass( mes ) ); endif;
{       if( applicationID = 23 ) then return( appBoxRobotMill105( mes ) ); endif;        }
        if( applicationID = 24 ) then return( appSignalLights( mes.Parameter ) ); endif;
        return( 0 );
ENDFCT;
```

Steps to Create own Virtual Machine Communicating with MES4

Steps to program virtual machine in IRL

2. Add function “appSignalLights()” to Applikationen.irl.

```
FUNCTION appSignalLights( IN ARRAY [1..7] OF REAL: Parameter ) : int;
VAR
        INPUT BOOL : RedIsOn            AT 0;
        INPUT BOOL : YellowIsOn         AT 1;
        INPUT BOOL : GreenIsOn          AT 2;
        OUTPUT BOOL : RedOn             AT -1;
        OUTPUT BOOL : YellowOn          AT 0;
        OUTPUT BOOL : GreenOn           AT 1;
        REAL : redMES;
        REAL : yellowMES;
        REAL : greenMES;
BEGIN
        RedOn := false;
        YellowOn := false;
        GreenOn := false;
        if( MyResourceId = 0 ) then
                {Standardmode}
                RedOn := true;
                WAIT 1.0 SEC;
                RedOn := false;
                YellowOn := true;
                WAIT 1.0 SEC;
                YellowOn := false;
                GreenOn := true;
        else
                redMES := Parameter[1];
                yellowMES := Parameter[2];
                greenMES := Parameter[3];
                IF redMES = 1 THEN
                        RedOn := true;
                ENDIF;
                IF yellowMES = 1 THEN
                        YellowOn := true;
                ENDIF;
                IF greenMES = 1 THEN
                        GreenOn := true;
                ENDIF;
        endif;

        WAIT 1.0 SEC;
        RedOn := false;
        YellowOn := false;
        GreenOn := false;

        return (0);
ENDFCT;
```

Steps to Create own Virtual Machine Communicating with MES4

Steps to program virtual machine in IRL

3. Save the program.
4. Compile the project and make sure that all projects are compiled successfully.
5. A resource with base and application module is created.

Troubleshoot in External Document

CIROS-CP_Troubleshoot_EN_v7.1_xxxxxx.pdf