EcoStruxure™ Cobot Expert

Software Guide

Original instructions

EIO0000004780.05
10/2025

www.se.com

Legal Information

The information provided in this document contains general descriptions, technical characteristics and/or recommendations related to products/solutions.

This document is not intended as a substitute for a detailed study or operational and site-specific development or schematic plan. It is not to be used for determining suitability or reliability of the products/solutions for specific user applications. It is the duty of any such user to perform or have any professional expert of its choice (integrator, specifier or the like) perform the appropriate and comprehensive risk analysis, evaluation and testing of the products/solutions with respect to the relevant specific application or use thereof.

The Schneider Electric brand and any trademarks of Schneider Electric SE and its subsidiaries referred to in this document are the property of Schneider Electric SE or its subsidiaries. All other brands may be trademarks of their respective owner.

Schneider Electric does not grant any right or license for commercial use of the document or its content, except for a non-exclusive and personal license to consult it on an "as is" basis.

Schneider Electric reserves the right to make changes or updates with respect to or in the content of this document or the format thereof, at any time without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this document, as well as any non-intended use or misuse of the content thereof.

Table of Contents

Safety Information

Important Information

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Document

Document Scope

This document describes the functionalities contained in EcoStruxure Cobot Expert.

Validity Note

This document has been created for the release of EcoStruxure Cobot Expert version 1.7.

Product Related Information

⚠WARNING

LOSS OF CONTROL

- Perform a Failure Mode and Effects Analysis (FMEA), or equivalent risk analysis, of your application, and apply preventive and detective controls before implementation.
- Provide a fallback state for undesired control events or sequences.
- Provide separate or redundant control paths wherever required.
- Supply appropriate parameters, particularly for limits.
- Review the implications of transmission delays and take actions to mitigate them.
- Review the implications of communication link interruptions and take actions to mitigate them.
- Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and error conditions) according to your risk assessment, and applicable codes and regulations.
- Apply local accident prevention and safety regulations and guidelines.[1]
- Test each implementation of a system for proper operation before placing it into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), *Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control* and to NEMA ICS 7.1 (latest edition), *Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems* or their equivalent governing your particular location.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Update your application program as required, paying particular attention to I/O address adjustments, whenever you modify the hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Incomplete file transfers, such as data files, application files and/or firmware files, may have serious consequences for your machine or controller. If you remove power, or if there is a power outage or communication interruption during a file transfer, your machine may become inoperative, or your application may attempt to operate on a corrupted data file. If an interruption occurs, reattempt the transfer. Be sure to include in your risk analysis the impact of corrupted data files.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION, DATA LOSS, OR FILE CORRUPTION

- Do not interrupt an ongoing data transfer.
- If the transfer is interrupted for any reason, re-initiate the transfer.
- Do not place your machine into service until the file transfer has completed successfully, unless you have accounted for corrupted files in your risk analysis and have taken appropriate steps to prevent any potentially serious consequences due to unsuccessful file transfers.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

⚠WARNING

UNINTENDED MOVEMENT OF THE LEXIUM COBOT ARM

- Ensure the proper functioning of the functional safety equipment before commissioning.
- Ensure that you can stop Lexium Cobot Arm movements at any time using functional safety equipment (limit switch, emergency stop) before and during commissioning.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The Lexium Cobot systems are calibrated before delivery. Under certain conditions, the correspondence between the hardware position and its representation in the software may be lost during the life cycle of the products, for example by moving the Lexium Cobot Arm without drive energy or by overtwisting the joints. In such a case, the verification of the mechanical position in relation to the software representation is required.

If you have any doubts about the correspondence between the position of the hardware and its representation in the software, contact your local Schneider Electric service representative.

⚠WARNING

INCORRECT REFERENCE TO MECHANICAL SYSTEM

- Ensure that a valid mechanical position reference exists by performing commissioning tests for all operating modes.
- Verify the mechanical position reference before operating the Lexium Cobot if the Lexium Cobot Arm has been moved without drive energy or if at least one joint may have been overtwisted, as described in the Lexium Cobot Hardware Guide.
- Verify that the mechanical positions of the joints correspond to the representation in the software.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

For further information, refer to the Lexium Cobot Hardware Guide, chapter *Verification of Mechanical Position*.

General Cybersecurity Information

In recent years, the growing number of networked machines and production plants has seen a corresponding increase in the potential for cyber threats, such as unauthorized access, data breaches, and operational disruptions. You must, therefore, consider all possible cybersecurity measures to help protect assets and systems against such threats.

To help keep your Schneider Electric products secure and protected, it is in your best interest to implement the cybersecurity best practices as described in the Cybersecurity Best Practices document.

Schneider Electric provides additional information and assistance:

- Subscribe to the Schneider Electric security newsletter.
- Visit the Cybersecurity Support Portal web page to:
  - Find Security Notifications.
  - Report vulnerabilities and incidents.
- Visit the Schneider Electric Cybersecurity and Data Protection Posture web page to:
  - Access the cybersecurity posture.
  - Learn more about cybersecurity in the cybersecurity academy.
  - Explore the cybersecurity services from Schneider Electric.

Product Related Cybersecurity Information

For detailed information, refer to Cybersecurity in EcoStruxure Cobot Expert, page 19.

Available Languages of the Document

The document is available in these languages:

- English (EIO0000004780)
- German (EIO0000004781)
- French (EIO0000004782)
- Spanish (EIO0000005392)
- Italian (EIO0000005393)
- Chinese (EIO0000005394)

Related Documents

Document title	Reference
Lexium Cobot, Hardware Guide	EIO0000004783 (EN) EIO0000004785 (FR) EIO0000004784 (DE) EIO0000005395 (ES) EIO0000005396 (IT) EIO0000005397 (ZH)
Lexium Cobot Release Notes	RN0000000035 (EN)
EcoStruxure Machine Expert, LexiumCobotCommunication, Library Guide	EIO0000005112 (EN)
Schneider Electric Cybersecurity Support Portal	www.se.com/en/work/support/cybersecurity/overview.jsp
Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment, User Guide	EIO0000004242 (EN)
Cybersecurity Best Practices	CS-Best-Practices-2019-340 (EN)

To find documents online, visit the Schneider Electric download center (www.se.com/ww/en/download/).

Information on Non-Inclusive or Insensitive Terminology

As a responsible, inclusive company, Schneider Electric is constantly updating its communications and products that contain non-inclusive or insensitive terminology. However, despite these efforts, our content may still contain terms that are deemed inappropriate by some customers.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/ programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
EN ISO 10218-1:2011	Robots and robotic devices- Safety requirements for industrial robots - Part 1: Robots
EN ISO 10218-2:2011	Robots and robotic devices- Safety requirements for industrial robots - Part 2: Robot systems and integration
ISO/TS 15066:2016-02	Robots and robotic devices - Collaborative robots
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive
2014/53/EU	Radio Emission Directive
IEC 62443	Industrial communication networks - Network and system security

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Figures

Unless otherwise specified, the various references of the Lexium Cobot Arm are represented in the figures as LXMRL03S0•••.

Hazard Information

What's in This Chapter

Proper Use

This product is a software to be used together with the Lexium Cobot system intended solely for the purposes as described in the present documentation as applied in the industrial environment for the civilian end-use case.

Always observe the applicable safety-related instructions, the specified conditions, and the technical data.

Perform a risk assessment concerning the specific use before using the product. Take protective measures according to the result.

Since the product is used as a part of an overall system, you must ensure the safety of the personnel by means of the design of this overall system (for example, machine design).

Any other use is not intended and may be hazardous.

Qualification of Personnel

Target Audience for This Manual

This documentation is intended for users having the following knowledge:

- Skills and knowledge related to the construction and operation of electrical equipment and the installation
- Knowledge and experience in industrial control programming
- Received safety-related training to recognize and avoid the hazards involved

Qualified Person

Aside from skills and knowledge, qualified personnel must be able to detect possible hazards that may arise from parametrization, changing parameter values and generally from mechanical, electrical, or electronic equipment. The qualified personnel must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when working on the Lexium Cobot system.

Before You Begin

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

Start-Up and Test

Before using electrical control and collaborative robotic equipment for regular operation after installation, the Lexium Cobot system must be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check are made and that enough time is allowed to perform complete and satisfactory testing.

⚠WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

NOTE: Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Perform all start-up tests.

Operation and Adjustments

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer’s instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer’s instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

Cybersecurity in EcoStruxure Cobot Expert

Our Position on Cybersecurity

Products and solutions of Schneider Electric are being used in a wide range of infrastructures as well as in manufacturing plants. The demands of cloud computing, the Internet of Things (IoT), and increasing threats against critical infrastructure have elevated cybersecurity as a priority. For Schneider Electric, cybersecurity and data privacy encompasses the measures, actions, and practices employed to help protect digital offerings and solutions from cyber threats.

Schneider Electric provides a document that gives guidelines on how to help improve the cybersecurity posture of customer systems.

If you have a cybersecurity question or issue, contact your local Schneider Electric service representative.

EcoStruxure

EcoStruxure is the open, interoperable, IoT-enabled system architecture and platform of Schneider Electric. EcoStruxure leverages advancements in IoT, mobility, sensing, cloud, analytics, and cybersecurity to deliver innovation at every level. This includes connected products and edge control as well as apps, analytics and services.

Cybersecurity Key Aspects

The cybersecurity position of Schneider Electric focuses on key apects:

- Protecting strategic IT systems, assets, and internal activities
- Leading the digital transformation of energy management and automation
- Designing and developing new solutions and products within a cybersecurity framework

Best Practices for Reinforcing Cybersecurity of Lexium Cobot

To operate the Lexium Cobot, follow these industry cybersecurity best practices:

- Locate the Lexium Cobot behind a firewall and isolate it from the business network.
- Install physical controls so that only authorized personnel can access the zone where the Lexium Cobot is located.
- Change the default passwords used for the WiFi Service Access Point, the WiFi Service Webserver and the Lexium Cobot on first use to help ensure that only authorized users can gain access.

  NOTE: This is enforced at the first connection or after a firmware update for each component.
- Minimize network exposure and ensure that the Lexium Cobot is not accessible from the Internet.
- The Lexium Cobot supports Industrial Protocols such as Modbus TCP, Ethernet/IP and Profinet which do not support user authentication. So, enable only the protocols which are required for the application and implement a firewall to block unauthorized access.
- When remote access is required, use secured methods, such as Virtual Private Networks (VPNs).

- The Lexium Cobot follow the principles of Least Privilege by providing user levels with different level of permissions. For further information, refer to Connecting the Lexium Cobot, page 39.

Getting Started

What's in This Chapter

General Information on EcoStruxure Cobot Expert

Overview

EcoStruxure Cobot Expert is the graphical control software for operating the Lexium Cobot (**co**llaborative ro**bot**). The software provides integrated functions for manual operation, program implementation, parameter configuration and information monitoring of the Lexium Cobot. EcoStruxure Cobot Expert offers an alternative to cumbersome robot handheld programming devices and integrates the same functionality into software that can be installed on Android mobile devices or Windows PCs.

Software Functions

EcoStruxure Cobot Expert provides the following functions:

- Configuration of system, operation, safety functions, hardware and communication parameters of the Lexium Cobot system
- Manual operation of the Lexium Cobot Arm
- Editing and setting of I/O configuration of the Lexium Cobot system
- Programming the Lexium Cobot system
- Displaying of general information of the Lexium Cobot system

System Requirements

The following table presents the minimum requirements of the device for using EcoStruxure Cobot Expert.

NOTE: EcoStruxure Cobot Expert is intended to be used either on Android or Windows devices.

Terminal type	Android device	Windows device
Operating System	Android 8.0	Windows 10 64 bit
Processor	Kirin 659 or Snapdragon 660	Intel Core i3
Storage capacity	32 GB	32 GB
System memory	4 GB	4 GB
Screen size / graphics	8.0 inches	Intel HD Graphics 4000
Network	WiFi standard: 802.11 b/g/n	WiFi standard 802.11 b/g/n or cable bound network card

Refer to *WiFi Connection Considerations* in the Lexium Cobot Hardware Guide, for more information concerning the use of a wireless connection.

Installing EcoStruxure Cobot Expert on Android

Overview

Since EcoStruxure Cobot Expert is included as an .apk file in the Lexium Cobot software package that you can download directly from the Schneider Electric website to your device and cannot be obtained from the Google Play Store, the software is identified on the Android system as software from an unknown source.

To install EcoStruxure Cobot Expert on your Android device, you must first enable applications from unknown sources to be installed on your device. Then you can install EcoStruxure Cobot Expert. See the procedures hereafter.

Prerequisites

Verify that your system meets the minimum system requirements, page 22, to install and run EcoStruxure Cobot Expert.

Allowing to Install Unknown Source Apps on Android

Step	Action
1	On your Android device, go to **Settings** > **Application** > **Special app access** > **Install unknown apps** or **Settings** > **Security and privacy** > **Install unknown apps**. **NOTE:** The procedure may differ from device to device or between different Android versions. If the following procedure does not apply to your device, refer to the documentation of your device.
2	Select your file explorer app from the list.
3	Select **Allow app installs**.

Installing EcoStruxure Cobot Expert on Android

Step	Action
1	Download the latest software package from the Lexium Cobot page on the Schneider Electric website.
2	Locate the .apk file in the downloaded package in your file explorer app and tap it. **Result:** The installation confirmation prompt is displayed.
3	Select **INSTALL**. **Result:** EcoStruxure Cobot Expert is installed on your Android device. When the installation is complete a confirmation prompt is displayed.

Installing EcoStruxure Cobot Expert on Windows

Prerequisites

- Verify that your system meets the minimum system requirements, page 22 to install and run EcoStruxure Cobot Expert.
- To install EcoStruxure Cobot Expert, you must have administrator privileges on the device.

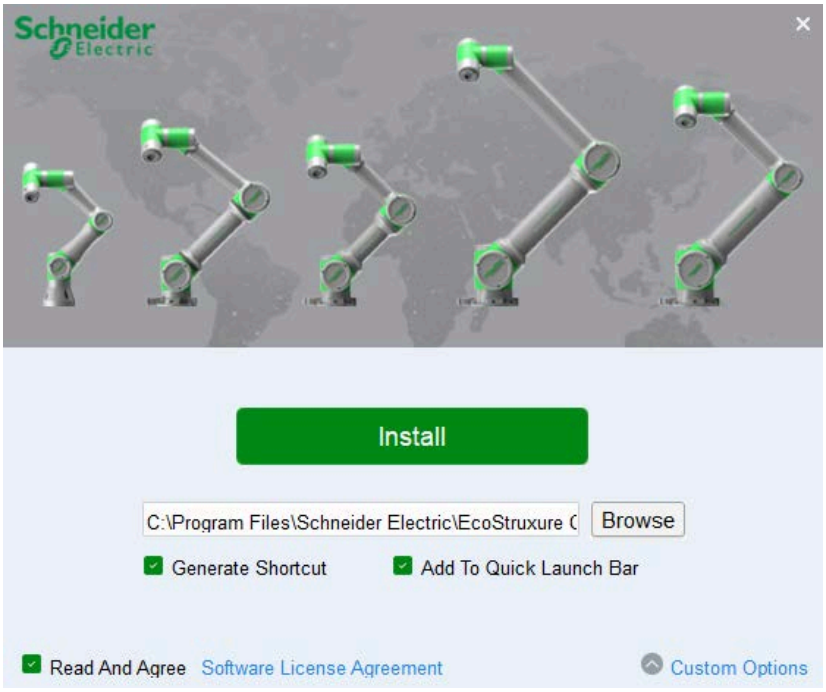
  NOTE: EcoStruxure Cobot Expert is installed for the other users of the device.

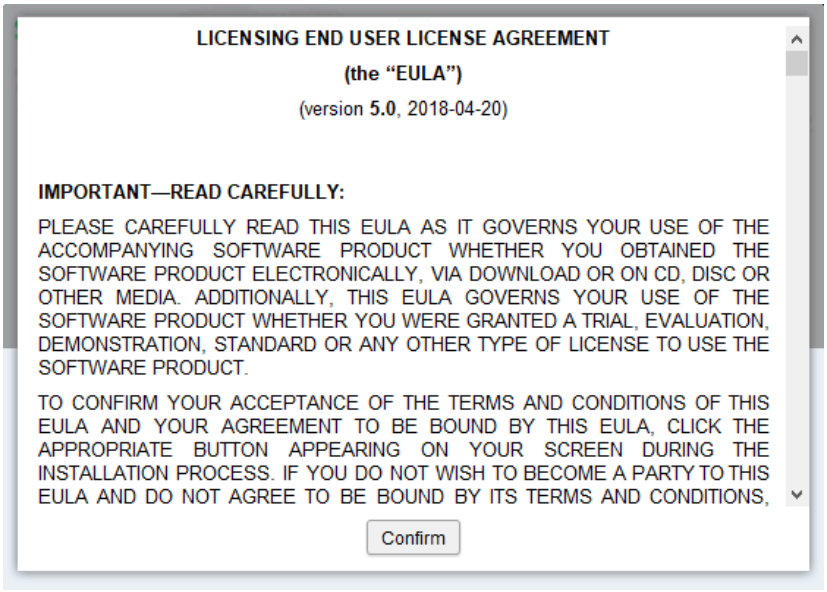
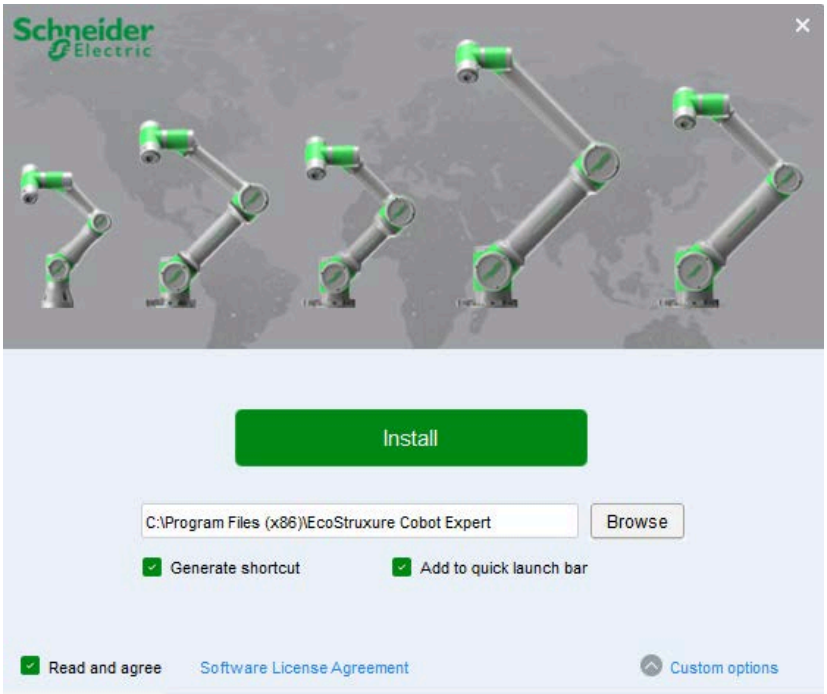
Default Directories

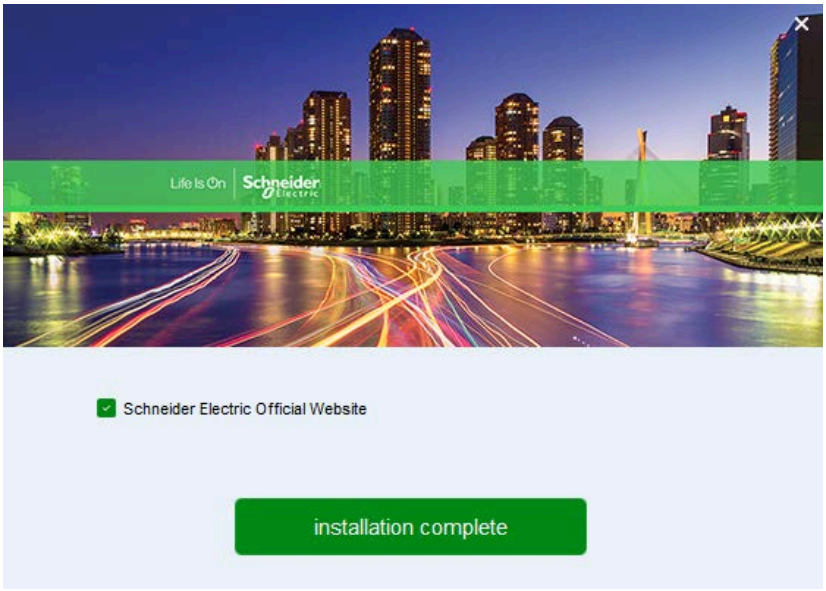
The default destination directory of the EcoStruxure Cobot Expert software installation is:

C:\Program Files\Schneider Electric\EcoStruxure Cobot Expert\V<*Version Number*>

Installing EcoStruxure Cobot Expert on Windows

Step	Action
1	Download the latest software package from the Lexium Cobot page on the Schneider Electric website.
2	Locate the file **EcoStruxure Cobot Expert.exe** in the downloaded package and execute it. **Result:** The EcoStruxure Cobot Expert installation window is displayed. 

Step	Action
3	Click **Software License Agreement** to read the agreement and, if you agree, click **Confirm**. 
4	Select **Read and agree** to confirm you read and agreed to the **Software License Agreement**.
5	Optionally, to configure the installation settings, click **Custom options**. You can configure the following installation settings: • Installation path • Desktop shortcut • Quick launch bar shortcut 

Step	Action
6	Click **Install**. **Result:** EcoStruxure Cobot Expert is installed on your computer. When the installation is complete a confirmation prompt is displayed. 
7	When the installation process is complete, click **installation complete**. **NOTE:** Optionally, select **Schneider Electric Official Website** to open the Schneider Electric website in your browser.  **Result:** EcoStruxure Cobot Expert is ready for use.

WiFi Service Webserver Connection

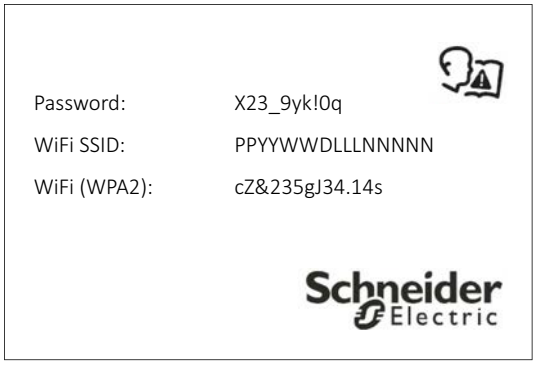
Overview

The Lexium Cobot system provides an embedded WiFi service. The webserver is password protected. Each Lexium Cobot Controller has a unique password, which can be found on the WiFi access label attached at the inner side of the front door of the Lexium Cobot Cabinet Controller or on the back of the Lexium Cobot Compact Controller.

Refer to *WiFi Connection Considerations* in the *Lexium Cobot Hardware Guide*, for more information concerning the use of a wireless connection.

WiFi Access Data

The following figure represents the WiFi access label:



Password: The password is required to get access to the webserver. The password on the label is unique to the device.

WiFi SSID: The SSID of the WiFi is a unique 15-digit serial number, which corresponds to the serial number on the type plate. The WiFi SSID is linked to the serial number of the device and cannot be modified.

WiFi (WPA2): The WiFi connection is based on WPA2 standard. The WPA2 password on the label is unique to the device.

First Connection to the WiFi Service

When you connect for the first time or after a reset to factory settings, the web server user interface opens in your browser and you need to change the passwords for the web server and the WiFi connection.

If the web server user interface does not open automatically, connect manually by typing 10.5.5.1 in the address bar of your browser.

For guidance on creating strong passwords, refer to Creating Strong Passwords, page 27.

NOTE: Changing these passwords is mandatory to log into the Lexium Cobot Controller.

Creating Strong Passwords

Creating strong passwords helps protect devices and equipment from unauthorized access. Passwords should be unique. The same password should never be used for duplicated purposes and the same passwords should not be used on different devices.

Acceptance criteria for entering new passwords on the Lexium Cobot Arm system is a length of 8...20 characters. Create a password that contains at least one character from each of the following categories:

- Uppercase letter (A, B, C… X, Y, Z)
- Lowercase letter (a, b, c…x, y, z)
- Number (0..9)
- Special character (for example, “?”, “#”, “!”)

Reset to Factory Settings

Reset WiFi Service to Factory Settings

If the webserver password and/or the WiFi (WPA2) password have been forgotten, the WiFi service can be reset to the factory settings using a hardware reset button on the Lexium Cobot Controllers. To reset the WiFi service to the factory settings, perform the following steps:

For Lexium Cobot Cabinet Controller:

Step	Action
1	Open the front door of the Lexium Cobot Cabinet Controller.
2	Press and hold the **Reset WiFi** button (1) located on the connector plate inside the Lexium Cobot Cabinet Controller for at least 10 seconds.  **Result:** • The Control Stick status indicator turns off and flashes blue when the reset is complete. • The WiFi service has been reset to factory settings. Use the webserver password and the WiFi (WPA2) password written on the WiFi access label inside the front door of the Lexium Cobot Cabinet Controller. To connect and set new user-defined passwords, refer to First Connection to the Lexium Cobot Controller, page 40.

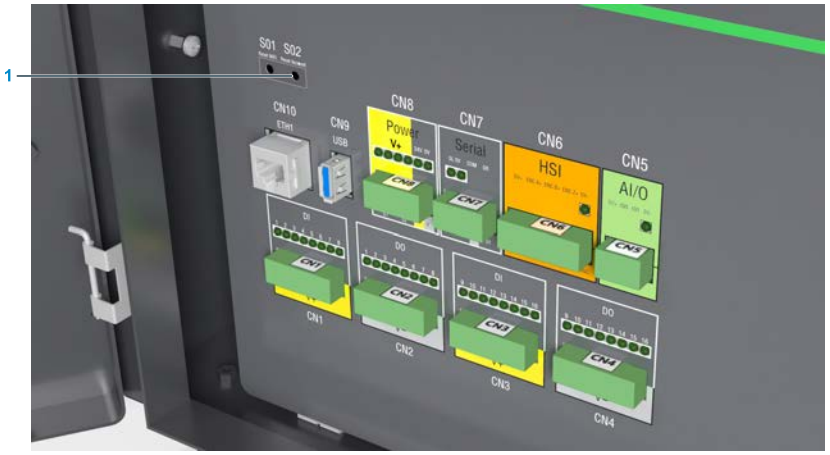
For Lexium Cobot Compact Controller:

Step	Action
1	Press and hold the **Reset WiFi** button (1) located on the antenna side of the Lexium Cobot Compact Controller for at least 10 seconds.  **Result:** • The Control Stick status indicator turns yellow and flashes blue again when the reset is complete. • The WiFi service is reset to factory settings. Use the webserver password and the WiFi (WPA2) password written on the WiFi access label on the back of the Lexium Cobot Compact Controller. To connect and set new user-defined passwords, refer to First Connection to the Lexium Cobot Controller, page 40.

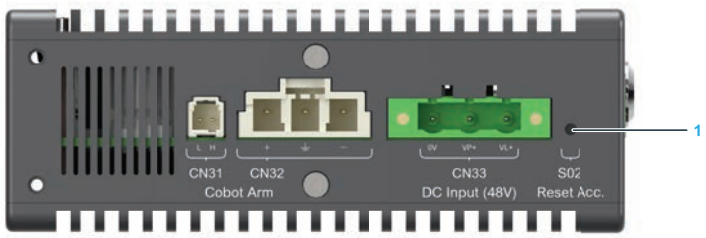
Reset Account to Factory Settings

If the passwords for the user levels have been forgotten, the account setting can be reset to the factory settings using a hardware reset button on the Lexium Cobot Controllers. To reset the user level passwords to the factory settings, perform the following steps:

For Lexium Cobot Cabinet Controller:

Step	Action
1	Open the front door of the Lexium Cobot Cabinet Controller.
2	Press and hold the **Reset Account** button (1) located on the connector plate inside the Lexium Cobot Cabinet Controller for at least 10 seconds.  **Result:** • The Control Stick status indicator turns off and flashes blue when the reset is complete. • The user level passwords have been reset to factory settings. To connect and set new user level passwords, refer to First Connection to the Lexium Cobot Controller, page 40.

For Lexium Cobot Compact Controller:

Step	Action
1	Press and hold the **Reset Account** button (1) located on the power connection side of the Lexium Cobot Compact Controller for at least 10 seconds.  **Result:** • The Control Stick status indicator turns yellow and flashes blue again when the reset is complete. • The user level passwords are reset to factory settings. To connect and set new user level passwords, refer to First Connection to the Lexium Cobot Controller, page 40.

Uninstalling EcoStruxure Cobot Expert

Uninstalling EcoStruxure Cobot Expert on Android

Step	Action
1	Tap and hold on the EcoStruxure Cobot Expert icon located in the app drawer or on the **Home** screen.
2	Depending on the device, tap **Uninstall** or drag the app to the **Uninstall** section that appears on the screen.
3	In the confirmation prompt, tap **OK**. **Result:** EcoStruxure Cobot Expert is uninstalled from your Android device.

Uninstalling EcoStruxure Cobot Expert on Windows

Step	Action
1	Close EcoStruxure Cobot Expert if it is opened: In the top menu, click **More** > **Sign out**.
2	Click the Windows **Start** button or press the Windows key.
3	In the Start menu select **Settings**.
4	In the Settings window select **Applications** and then **Applications and features**.
5	Select EcoStruxure Cobot Expert from the list of installed programs.
6	Click **Uninstall** and follow the instructions on the screen. **Result:** EcoStruxure Cobot Expert is uninstalled from your computer.

Operator Risk Estimation and Reduction

In some applications, additional operator protection such as point-of-operation guarding must be provided and/or technical measures must be taken to help avoid or at least limit any possible impact forces caused by the overall system to the operator. Depending on your risk assessment, take into account the applicable values for impact forces in accordance with ISO/TS 15066. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas where injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safety measures and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safety measures related to point-of-operation protection must be coordinated with the related collaborative robotic equipment and software programming.

⚠WARNING

CRUSHING, SHEARING, CUTTING AND IMPACT INJURY

- Avoid contact exposure to sensitive areas of the body, including the skull, forehead, larynx, eyes, ears or face.
- Define the clearance distance to the collaboration zone of operation of the Lexium Cobot Arm to be within the mechanical limits such that the operational staff do not have access to, nor can be enclosed between, the Lexium Cobot Arm user-defined collaboration zone and the mechanical limits of operation.
- Ensure that movement of the Lexium Cobot Arm is in accordance to the user-defined limits as soon as a person enters the collaboration zone of operation.
- All barriers, protective doors, contact mats, light barriers, visual protection system, and other protective equipment must be connected, configured correctly and enabled whenever the robot mechanics are under power.
- The Lexium Cobot Arm must always be considered active even though the Lexium Cobot Arm has reached an intermediate stop position waiting for a run command.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTE: The configuration of the robot mechanics, the Tool Center Point (TCP) velocity, as well as the additional payload have an effect on the total energy, which can potentially be a source of damage and injury.

As inertia and payload of the Lexium Cobot increases, so do the physical requirements for controlling and reducing forces and pressures.

⚠WARNING

UNINTENDED MACHINE OPERATION

- Use appropriate risk mitigation measures taking into account the inertia and the payload of the Lexium Cobot.
- Coordination of safety measures and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the software or other implementation referenced in this documentation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

EcoStruxure Cobot Expert Basics

What's in This Chapter

User Interface

Home Screen

This is the **Home** screen of EcoStruxure Cobot Expert:

The main interface consists of:

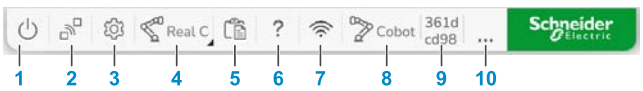
1 Top menu

2 Switch buttons

3 Feature bar

Top Menu

The top menu contains features for setting and managing the Lexium Cobot system and EcoStruxure Cobot Expert.



Number	Description	Function
1	Off(1)	Power off the Lexium Cobot Controller, page 44
2	Remote control	Delegate the control to a remote source(2), page 50.
3	Settings	Configure the system and the Lexium Cobot parameters, page 67
4	Operation mode	Switch between real Lexium Cobot and simulation, page 53
5	Log	Display the **Log information**, page 55
6	User interface description	Display the user interface description, page 36
7	Signal(3)	Signal strength indicator
8	Lexium Cobot connection	Connect or disconnect to the Lexium Cobot Controller, page 39
9	Checksum	Checksum for safety-related parameters, page 57
10	More	Window size control and sign out
1 You can only power off the Lexium Cobot Controllers in EcoStruxure Cobot Expert. To power on the Lexium Cobot Controllers, use the Control Stick. **2** While the Lexium Cobot system is controlled by the remote source, EcoStruxure Cobot Expert is locked. **3** If the system is connected with cables, the signal strength indicator displays full signal strength.		

NOTE: The features available in the top menu vary between the different interfaces of the software. The specific features for the individual interfaces are described in the respective sections in this document.

Switch Buttons

Use the switch buttons to power on or off the Lexium Cobot Arm and to enable or disable it.

Number	Description	Function
1	**Power on the robot** / **Power off the robot**	Power the Lexium Cobot Arm on or off, page 44
2	**Enable robot** / **Disable robot**	Enable or disable the Lexium Cobot Arm, page 44

NOTE: The Lexium Cobot Arm can only be enabled when it is powered on as it needs to be disabled to be powered off.

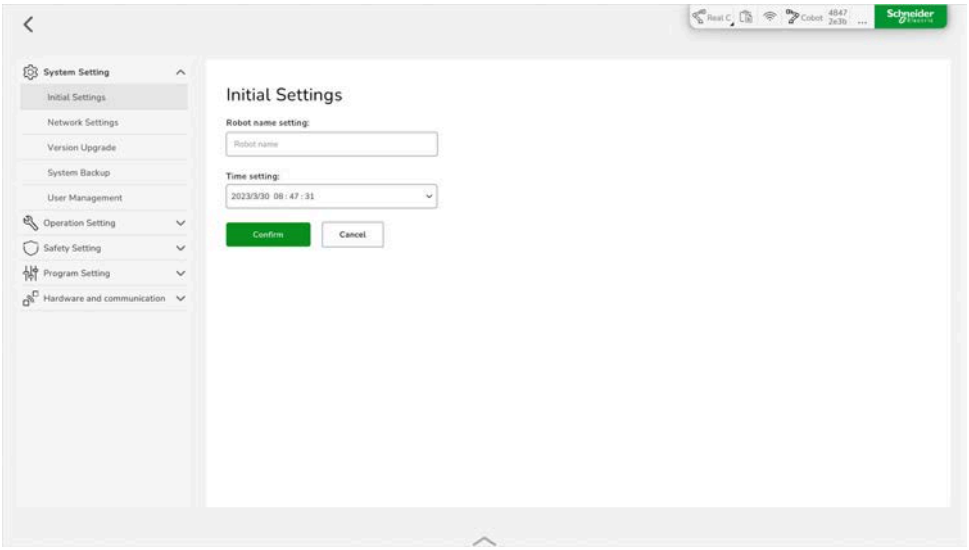
Feature Bar

In the feature bar, you will find the main features for operating the Lexium Cobot system (for example the **Programming Control**, Lexium Cobot motion control, I/O control and **Monitoring Information**).

Number	Description	Function
1	**Home**	Return to the **Home** screen
2	**Programming Control**	Display the **Programming Control** interface of EcoStruxure Cobot Expert, page 177
3	**Manual Operation**	Operate the Lexium Cobot Arm manually, page 60
4	**I/O Panel**	Set the I/O parameters, page 147
5	**Monitoring Information**	Monitor the Lexium Cobot status, page 59
6	**About**	• Switch the language • Adjust the sound volume • Disable and enable the **Soft Keyboard** (only Windows) • Information about the Lexium Cobot Controller and Lexium Cobot Arm (for example, commercial reference and serial number) • Information about the software and firmware versions

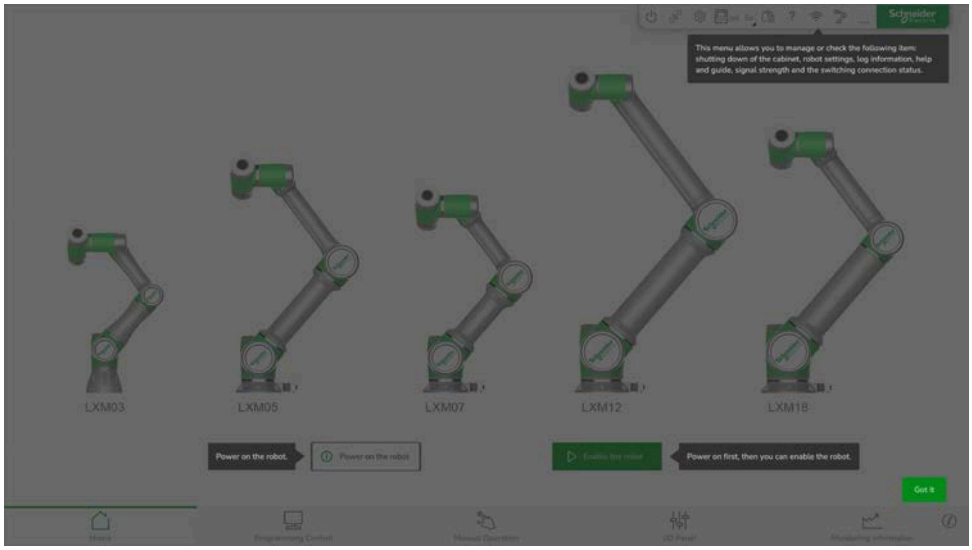
NOTE: On the **Home** screen, the feature bar is constantly displayed at the bottom. In the other windows of the software, the feature bar is being minimized. To display the feature bar, click the arrow icon at the bottom of the screen:

Example:



User Interface Description

You can display an overlay help in the user interface, which describes the particular icons and buttons of the **Home** screen.



To display the overlay, click the **Help** icon in the menu bar.

To close the overlay, click **Got it**.

Software Settings

Overview

To change software settings, for example, the language or the sound volume, or to view the version information, click the **About** icon in the feature bar.

In this dialog box, you will find the following settings and information:

- **Language settings**
- **Sound volume**
- **Soft keyboard** (only Windows)
- **User level**
- **App version**
- **Servo version**
- **SCB/PSCB version**(1)
- **Controller version**
- **CR Arm**(2)
- **SN Arm**(3)
- **CR Controller**(2)
- **SN Controller**(3)
- **Official website** (Lexium Cobot page on the Schneider Electric Website)

(1) SCB: Safety-related Control Board / PSCB: Portable Safety-related Control Board

(2) CR: Commercial Reference

(3) SN: Serial Number

NOTE: The version information of the Lexium Cobot Arm is only available when the equipment is connected and under power.

Language Settings

EcoStruxure Cobot Expert provides the following languages:

- Chinese
- English
- Japanese

To switch the language, click the respective language to be displayed.

NOTE: When **Auto** is selected, EcoStruxure Cobot Expert detects the equipment system language and applies it to the user interface, when available. Otherwise, English is applied.

Soft Keyboard

If you want to use only the hardware keyboard when using the Windows operating system, and do not want the soft keyboard to be displayed, select **Disable**.

If you want to use the soft keyboard, select **Enable**.

Connecting the Lexium Cobot

Overview

EcoStruxure Cobot Expert can connect to any Lexium Cobot Controller. There are two software installation versions available that provide the following connection options:

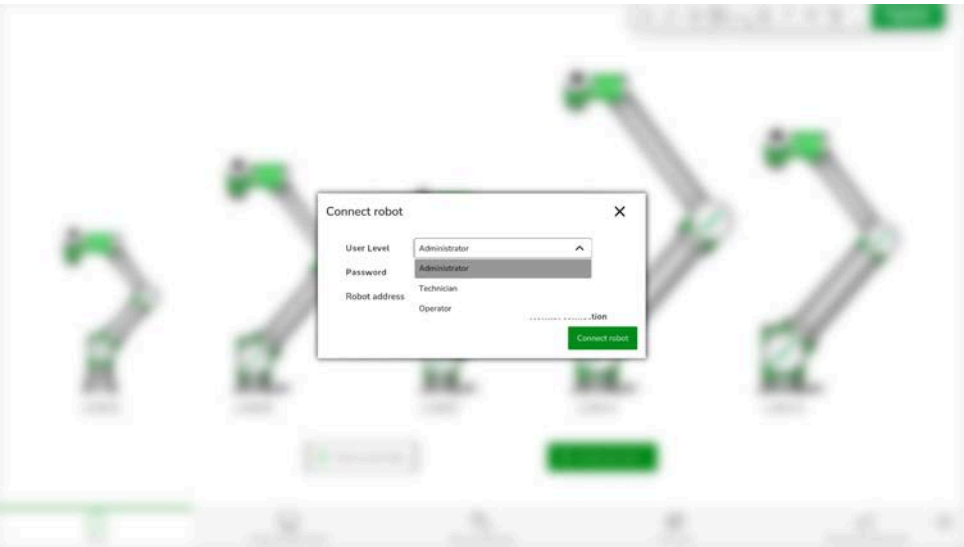
Software installation version	Connection options
EcoStruxure Cobot Expert for Windows PC	LAN or WiFi connection
EcoStruxure Cobot Expert for Android device	WiFi connection

NOTE: To connect to the Lexium Cobot system, the device with EcoStruxure Cobot Expert installed must be connected to the same network as the Lexium Cobot Controller.

User Levels

The Lexium Cobot system provides three different user levels for:

- Setting up the system (**Administrator**)
- Editing of the application program (**Technician**)
- Operating the system (**Operator**).



The following table presents in detail which user level has which permissions:

User level	Is authorized for
Operator	• Opening and browsing the program stored in the Lexium Cobot Controller • Starting, pausing and stopping the program • Adjusting the program speed • Starting and stopping the Lexium Cobot Arm • Displaying **Log information** and state of the Lexium Cobot needed for program execution and observation
Technician	• Performing the operations listed in the **Operator** user level • Unlocking programs • Creating, editing, saving programs • Debugging programs • Adjusting of settings required for program editing • Modifying non-safety-related content • Delegating the control source to remote control and requesting it back
Administrator	• Performing the operations listed in the **Operator** and the **Technician** user levels • Accessing, editing and modifying the user management page • Displaying and modifying the pages • Modifying the safety-related settings

First Connection to the Lexium Cobot Controller

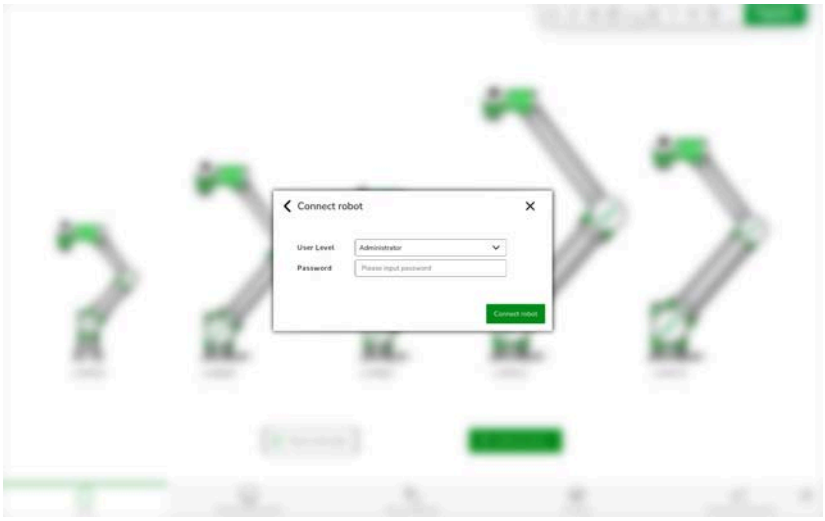
When you connect for the first time, after a reset to factory settings, or after a firmware upgrade of the Lexium Cobot Controller, the password for the **Administrator** user level is `Administrator`. With the first login you will need to change the password for the **Administrator** user level. For guidance on creating strong passwords, refer to Creating Strong Passwords, page 27.

NOTE: Changing the password is mandatory, otherwise it will not be possible to log into the Lexium Cobot Controller.

The user levels **Technician** and **Operator** can be activated in User Management, page 79.

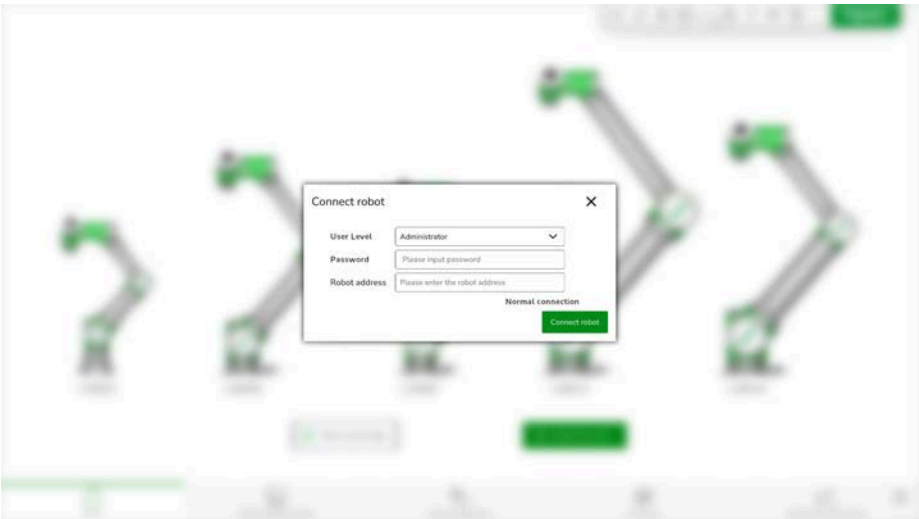
Connecting the Lexium Cobot

To connect the Lexium Cobot, perform the following steps:

Step	Action
1	Power on the Lexium Cobot Controller as described in Powering On the Lexium Cobot Controller, page 44.
2	Connect your device with the installed EcoStruxure Cobot Expert to the network created by the Lexium Cobot Controller.
3	In the top menu, click the **Cobot connection** icon. **Result:** The dialog box **Connect robot** is displayed. 
4	Select the Lexium Cobot to be connected from the list. **NOTE:** The network connection is established to the Lexium Cobot Controller. **Result:** The confirmation prompt is displayed.
5	Click **Confirm**. **Result:** The login dialog box is displayed. 
6	On first connection, in **User Level**, select **Administrator**. Otherwise, on subsequent connections select your user level.
7	On first connection, in **Password**, type in `Administrator`. Otherwise, on subsequent connections, type in the password for the selected user level. **NOTE:** The number of unsuccessful password attempts within 10 minutes when logging in to the Lexium Cobot Controller using the Windows or Android versions of EcoStruxure Cobot Expert is limited to five, after which a lockout period of 10 minutes is triggered before a new login attempt is permitted.

Step	Action
8	On first connection, type in a new password and confirm it.
9	Click **Connect robot**. **Result:** The Lexium Cobot system is connected. An established connection is indicated by the status column of the robot list. Furthermore, the **Cobot connection** icon in the top menu displays the name of the connected Lexium Cobot.  **NOTE:** Observe the network status when the Lexium Cobot Controller is connected through WiFi to detect disconnection due to a weak network signal.

NOTE: Alternatively, you have the option to manually connect a Lexium Cobot system by typing its IP address. Therefore, click **Offline connection** in the **Connect robot** dialog box and follow the instructions.



Displaying the Connection Information

In the top menu, click the **Cobot connection** icon.

Result: The dialog box **Connect robot** is displayed and presents the connection information.

Description of the Connection information

The connection information displays the following data:

Name	Description
Current connection	Name of the connected Lexium Cobot Controller.
Commercial reference	Commercial reference of the Lexium Cobot Arm (only updated after the Lexium Cobot Arm has been powered on).
Serial number	Serial number of the Lexium Cobot Arm.
Robot name	Configurable name of the Lexium Cobot Controller. For further information, refer to Initial Settings, page 67.
Robot IP	IP address of the Lexium Cobot Controller.
Control version	Firmware version of the Lexium Cobot Controller.
Status	Displays the connection status of the Lexium Cobot (connected, disconnected or occupied).
Cloud icon	Navigates to the **Version Upgrade** section, page 69.

Starting Up the Lexium Cobot System

Start-Up Sequence

To start up the Lexium Cobot system, perform the following tasks:

1. Power on the Lexium Cobot Cabinet Controller or Lexium Cobot Compact Controller
2. Power on the Lexium Cobot Arm
3. Enable the Lexium Cobot Arm

Powering On the Lexium Cobot Controller

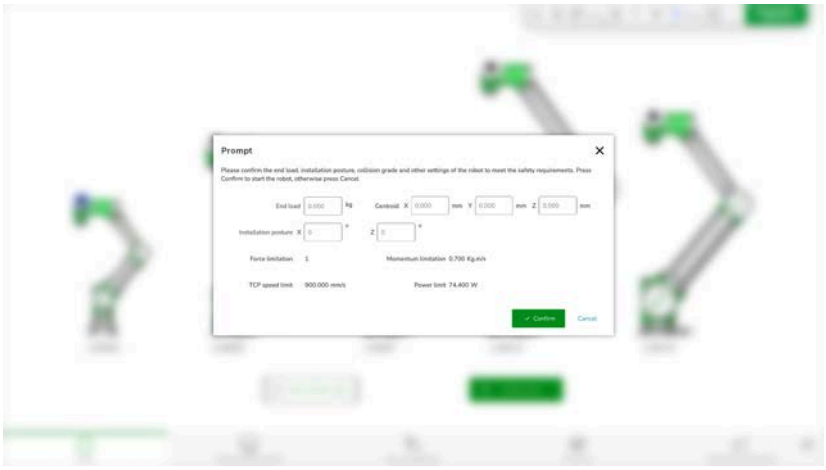
To power on the Lexium Cobot Controller, perform the following step:

Step	Action
1	At the Control Stick, press and hold the **On/Off** button (1) for 1 second.  **Result:** When the status indicator (2) on the Control Stick is blue, the Lexium Cobot Controller is powered on.

NOTE: When using the Lexium Cobot Compact Controller, you can alternatively press and hold the power button ⏻ (1) for 3 seconds.

Powering on and Enabling the Lexium Cobot Arm

To power on and enable the Lexium Cobot Arm from EcoStruxure Cobot Expert, perform the following steps:

Step	Action
1	Verify that EcoStruxure Cobot Expert is connected to the Lexium Cobot Controller.
2	On the **Home** screen, click **Power on the robot**. **Result:** The illuminated ring of the Lexium Cobot Arm (in the software and at the hardware) and the **Cobot connection** icon turn blue: the Lexium Cobot Arm is powered on. 
3	Click **Enable robot**. **Result:** The following confirmation prompt is displayed. 

Step	Action
4	Verify and adjust, if necessary, the system parameters.
5	Click **Confirm**. **Result:** The illuminated ring of the Lexium Cobot Arm (in the software and at the hardware) and the **Cobot connection** icon turn green: the Lexium Cobot Arm is enabled. 

NOTE: Alternatively, you can use the Control Stick to power on and enable the Lexium Cobot Arm. For further information, refer to chapter *Lexium Cobot Control Stick Details* in the *Lexium Cobot Hardware Guide*.

Shutting Down the Lexium Cobot System

Overview

Removing power from the Lexium Cobot Controller during operation of the Lexium Cobot Arm may cause a loss of control of the end-effector. The Lexium Cobot Arm must be disabled and powered off before removing power from the Lexium Cobot Controller.

⚠WARNING

UNCONTROLLED EQUIPMENT OPERATION

Ensure that the Lexium Cobot Arm is disabled and powered off before removing power from the Lexium Cobot Controller.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Disabling and Powering off the Lexium Cobot Arm

To disable and power off the Lexium Cobot Arm, perform the following steps:

Step	Action
1	Verify that no program is being executed.
2	Click **Disable robot**. **Result:** The indicator lights of the Lexium Cobot Arm (in the software and at the hardware) and the **Cobot connection** icon turn blue. 
3	Click **Power off the robot**. **Result:** The indicator lights of the Lexium Cobot Arm (in the software and at the hardware) and the **Cobot connection** icon turn off. 

NOTE: Alternatively, you can use the Control Stick to disable and power off the Lexium Cobot Arm. For further information, refer to chapter *Lexium Cobot Control Stick Details* in the *Lexium Cobot Hardware Guide*.

Power Off the Lexium Cobot Controller

To power off the Lexium Cobot Controller, perform the following steps:

Step	Action
1	Verify that the Lexium Cobot Arm is disabled and powered off.
2	At the Control Stick, press and hold the **On/Off** button (1) for 3 seconds.  **NOTE:** Alternatively, click the **Power Off** icon in the top menu of EcoStruxure Cobot Expert:  **Result:** The status indicator (2) on the Control Stick turns off and the Lexium Cobot Controller is powered off.

NOTE: When using the Lexium Cobot Compact Controller, you can alternatively press and hold the power button ⏻ (1) for 5 seconds.

Delegating the Control of the Lexium Cobot

Overview

Lexium Cobot provides three control sources for sending commands:

- EcoStruxure Cobot Expert
- Remote

  Includes LexiumCobotCommunication library and Digital Input Functions. For further information, refer to Function Settings, page 148.
- Control Stick

  For further information, refer to chapter *Lexium Cobot Control Stick Details* in the *Lexium Cobot Hardware Guide*.

Except for specific commands (see Delegating the Control Source to Remote Control, page 51), only one control source is active at a time, the others are locked. Commands received from the locked sources are not executed.

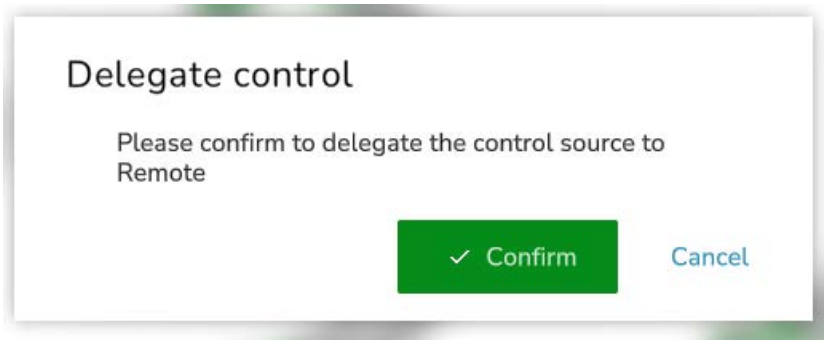
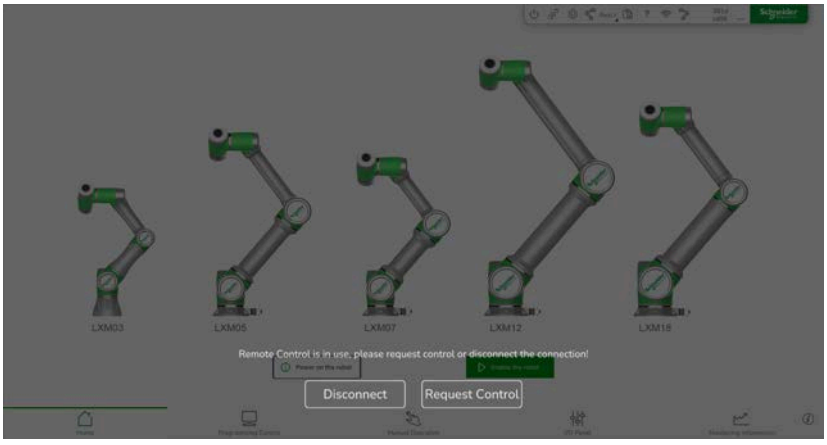
NOTE: The following commands are always performed regardless of the control source:

- All special safety-related inputs (refer to Special Safety IO, page 121)
- DI function Protective Stop
- DI functions Level 1 and Level 2 Override Mode

Switching the control source always results in a paused program or stopped movement.

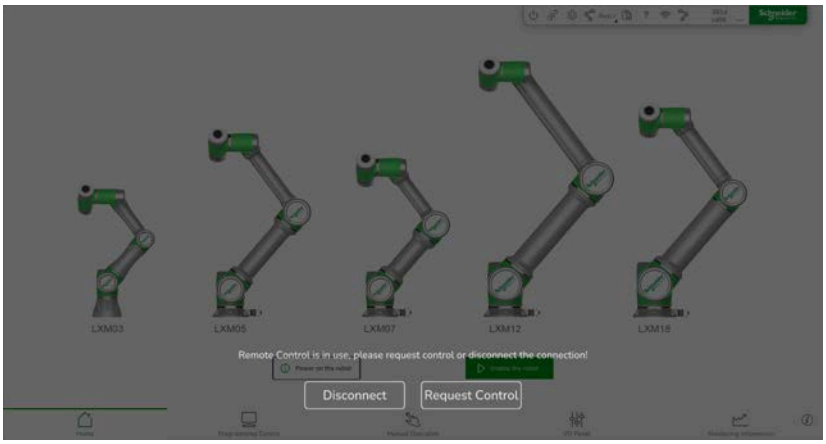
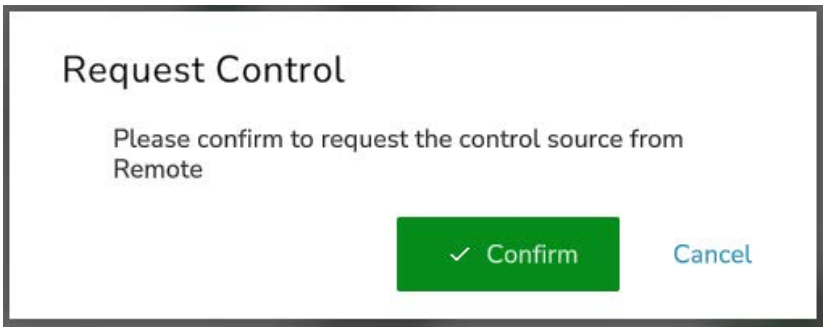
Delegating the Control Source to Remote Control

To delegate the control source to **Remote Control**, perform the following steps:

Step	Action
1	Verify that EcoStruxure Cobot Expert is connected to the Lexium Cobot Controller.
2	Click the **Remote Control** button in the top menu.  **Result:** The confirmation dialog is displayed. 
3	Click **Confirm**. **Result:** The control source is delegated to remote and the **Remote Control** overlay is displayed. 

Requesting the Control Source from Remote Control

To request the control source from **Remote Control**, perform the following steps:

Step	Action
1	Verify that EcoStruxure Cobot Expert is connected to the Lexium Cobot Controller.
2	Click **Request Control**.  **Result:** The confirmation dialog is displayed. 
3	Click **Confirm**. **Result:** The control source is delegated to EcoStruxure Cobot Expert.

Switching between a Physical Lexium Cobot and a Simulation

Overview

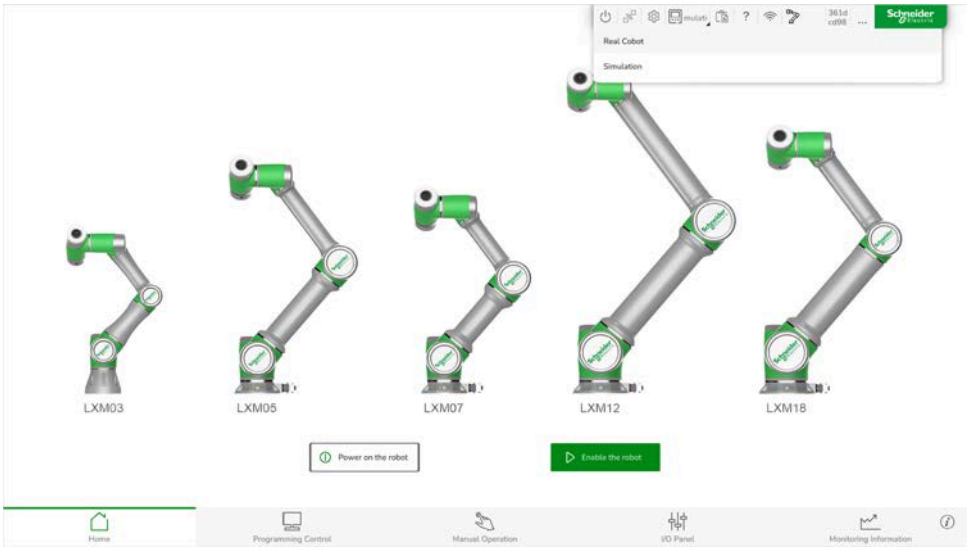
EcoStruxure Cobot Expert can either work with a physical Lexium Cobot Arm or with a digital representation of different sizes of the arm in combination with a physical Lexium Cobot Controller.

In **Simulation** operation mode you can control the digital representation of the Lexium Cobot Arm, access inputs and outputs, including the physical I/O, and program offline.

The active operation mode (**Real Cobot** or **Simulation**) is displayed in the top menu. You can switch the operation mode by clicking the status button.

NOTE:

- To switch between operation modes, the Lexium Cobot Arm must be disabled and powered off.
- The display of the serial number, commercial reference, and servo version of the Lexium Cobot Arm is updated only when the physical Lexium Cobot Arm is powered on after switching from **Simulation** to **Real Cobot**.



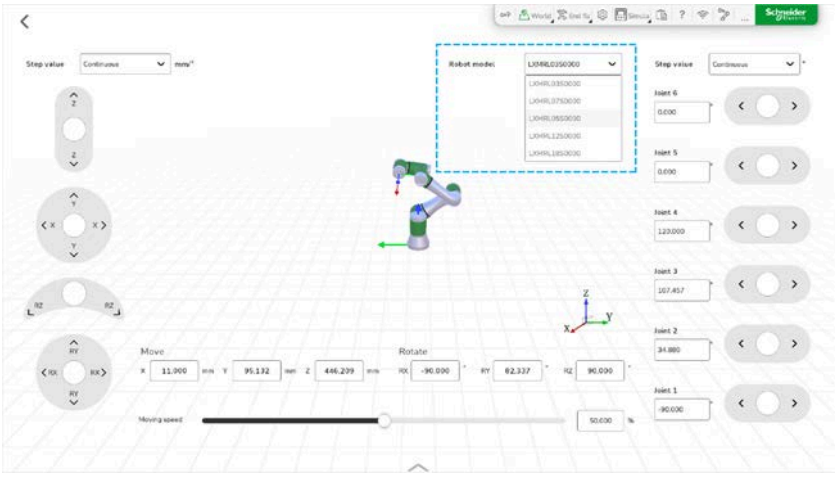
Activating the Simulation Operation Mode

To activate the **Simulation** operation mode, perform the following steps:

Step	Action
1	Disable and power off the Lexium Cobot Arm.
2	Click the **Operation Mode** button in the top menu and select **Simulation**. **Result:** The confirmation prompt is displayed.
3	Click **Confirm**. **Result:** The **Simulation** operation mode is active.

Selecting the Size of the Digital Representation

To select a specific size of the digital representation, go to **Manual Operation** in the feature bar and select the size from the **Robot model** dropdown list.



For further information on the interface, refer to Operating the Lexium Cobot Manually, page 60.

Activating the Real Cobot Operation Mode

To activate the **Real Cobot** operation mode, perform the following steps:

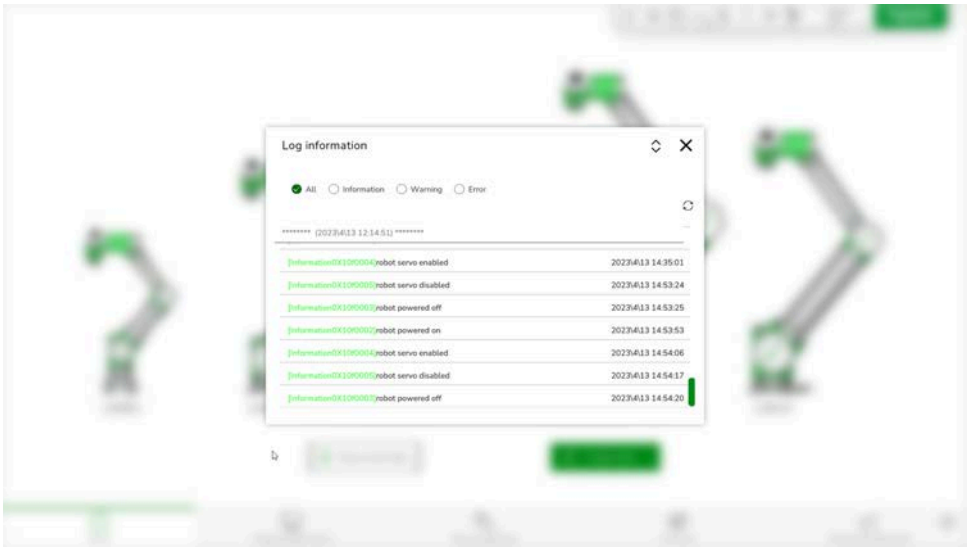
Step	Action
1	Disable and power off the digital representation of the Lexium Cobot Arm.
2	Click the **Operation Mode** button in the top menu and select **Real Cobot**. **Result:** The confirmation prompt is displayed.
3	Click **Confirm**. **Result:** The physical Lexium Cobot Arm is active.

NOTE: Verify your payload parameters after using the simulation mode.

Displaying the Log Information

Overview

All operation information, advisory, and errors are automatically recorded in the **Log information**.



To display the **Log information**, click the **Log** icon in the top menu:

You can use these **Log information** for data analysis and processing, event tracking and solving issues.

If an error is detected during Lexium Cobot operation, display the **Log information** to find out the cause of the detected error and perform a self-inspection if possible. If you could not resolve the error, contact your local Schneider Electric service representative.

When you select the option **All**, the three categories of entries (**Information**, **Warning**, and **Error**) are displayed, in chronological order from the newest to the oldest.

You can filter the list by **Information**, **Warning**, and **Error** by activating the corresponding option.

Information List

When the state of the Lexium Cobot changes, the change is recorded in the **Log Information**. The **Information** list displays status changes of the Lexium Cobot.

Warning List

A **Warning** is displayed in case of abnormal movements or abnormal status of the Lexium Cobot.

This information is stored in the **Log Information** in the **Warning** list for troubleshooting and monitoring.

Error List

Detected errors are displayed in case of incorrect motions or when the Lexium Cobot is in an error state.

This error information is stored in the **Log Information** in the **Error** list for troubleshooting and monitoring.

Checksum for Safety-Related Parameters

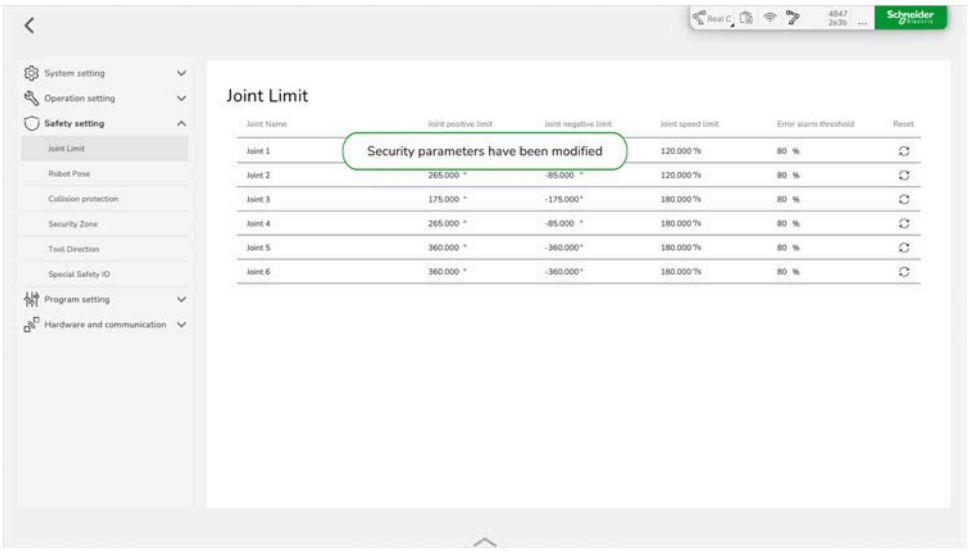
Overview

The checksum for safety-related parameters is an 8-digit hexadecimal number that is displayed in the top menu after connecting the Lexium Cobot.

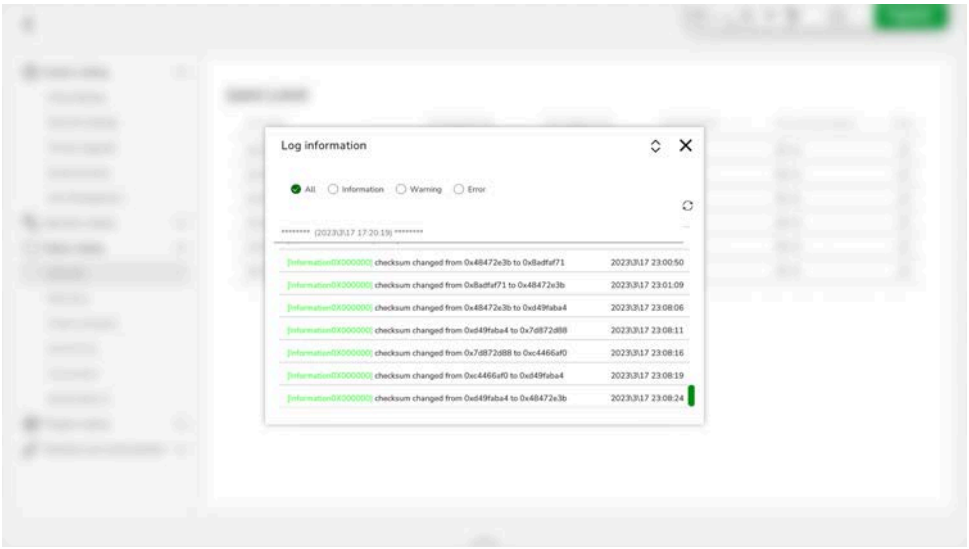
The checksum is used to represent the configuration of the safety-related parameters of the Lexium Cobot in the form of letters and numbers. If the safety-related parameters are changed, the checksum changes accordingly.

Checksum Changes

When you change any safety-related parameters, the message **Security parameters have been modified** is displayed and the checksum code is updated.



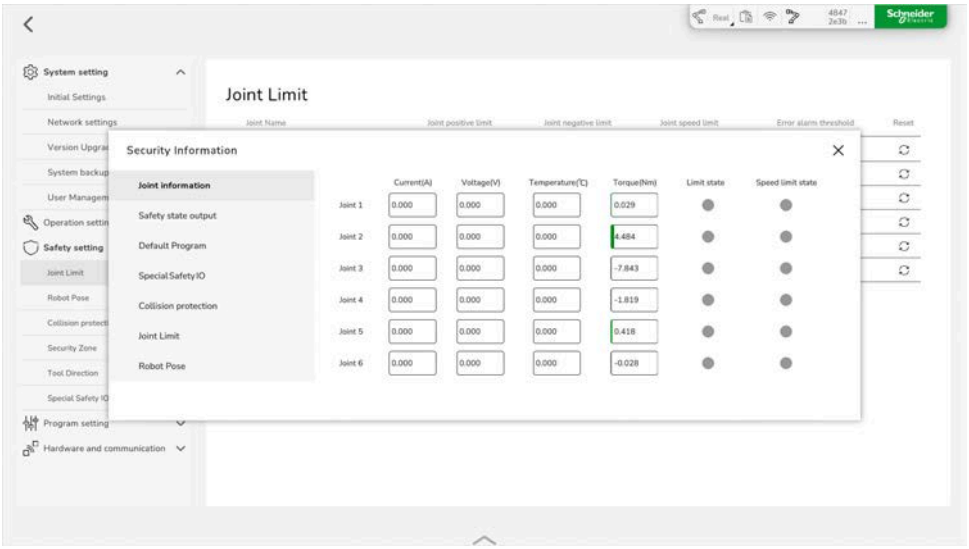
Changes to the checksum appear in the **Log information**.



Security Information Window

You can verify the checksum code and the safety-related parameters of the Lexium Cobot by clicking the checksum in the top menu.

The **Security Information** window is displayed. This window provides an overview of the safety-related parameters.



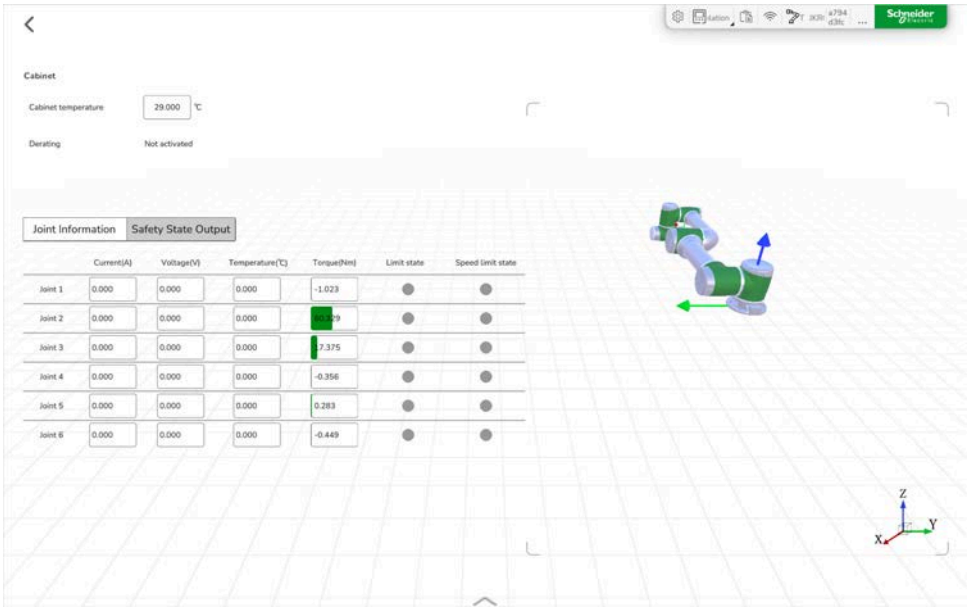
To display a specific category, click the respective menu item.

NOTE:

- The parameters are read-only and cannot be edited in this window.
- The three user levels have the permission to display this page.

Displaying the Monitoring Information

To display a visual representation of the Lexium Cobot Arm and information about the Lexium Cobot system and its joints, select **Monitoring Information** in the feature bar.

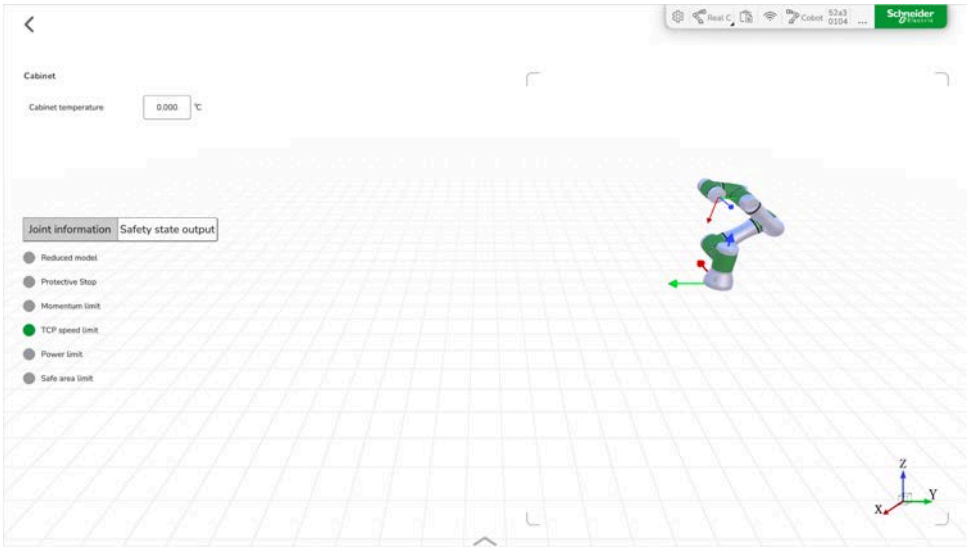


The **Cabinet** section presents the temperature of both Lexium Cobot Controllers and, for the Lexium Cobot Compact Controller, the status of the derating feature.

> **NOTE:** The temperature of the Lexium Cobot Compact Controller housing may exceed the temperature displayed in EcoStruxure Cobot Expert. The displayed value presents the temperature inside the housing and depends on the application. For further information on heat dissipation, refer to section *Hot Surfaces* in chapter *Residual Risks* in the *Lexium Cobot Hardware Guide*.

The **Joint Information** tab presents the current, voltage, temperature, torque, limit state and speed limit state of each joint.

The **Safety state output** tab presents the active limit modes of the Lexium Cobot. When a limit mode is active, the indicator lights up green.



> **NOTE:** Enable the Lexium Cobot Arm to read the values. If the Lexium Cobot Arm is disabled, the values are set to `0` and the visualization therefore does not represent actual values.

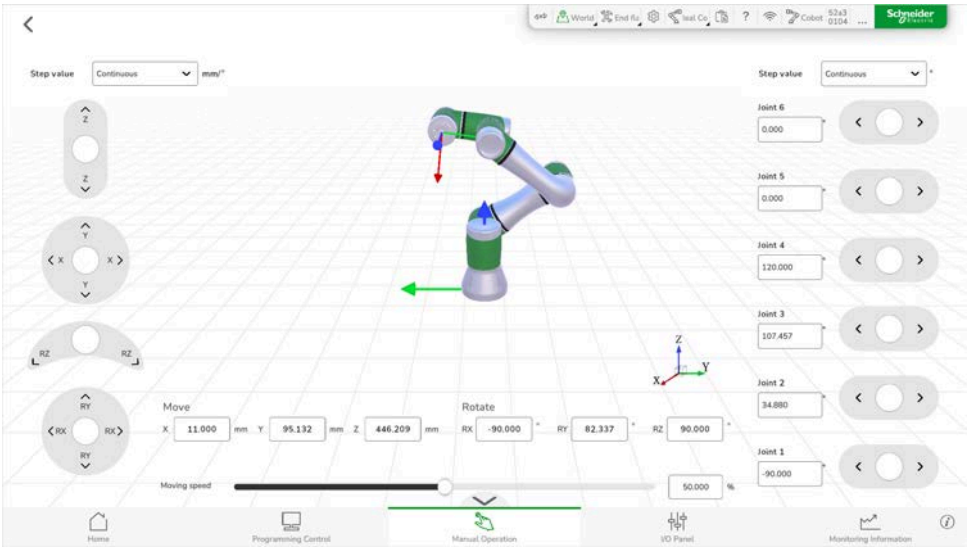
Operating the Lexium Cobot Manually

What's in This Chapter

Manual Operation Interface

Overview

To operate the Lexium Cobot manually, select **Manual Operation** in the feature bar.

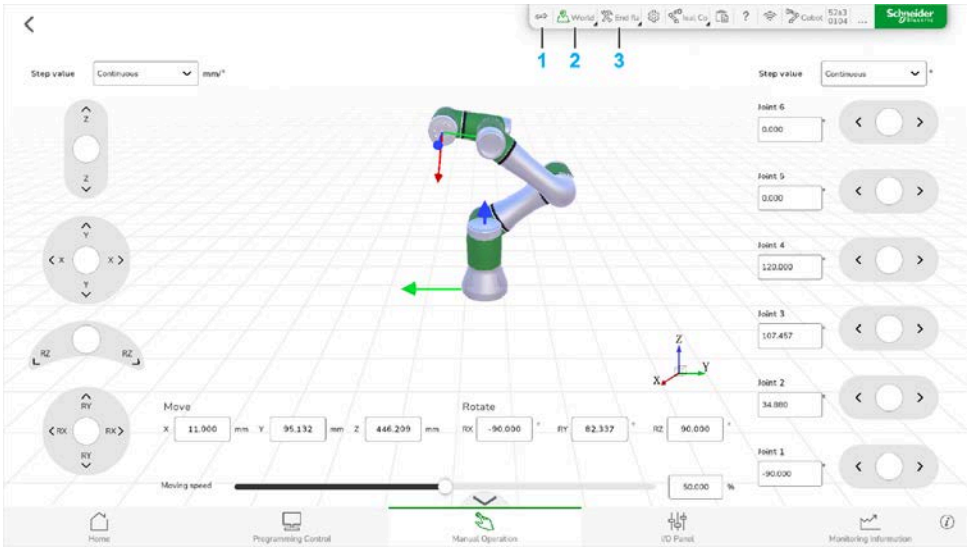


Coordinate Systems

The Lexium Cobot system supports the customization of the following two coordinate systems in EcoStruxure Cobot Expert:

- Tool coordinate system (TCS)
- User coordinate system (UCS)

Both coordinate systems can be adjusted in the settings (refer to TCP Settings, page 81, and User Coordinate System, page 96 for this purpose). The digital representation displays these two coordinate systems in the **Manual Operation** interface.



1 Switch Coordinate System for switching between tool coordinate system and user coordinate system. The active system is displayed green.

2 User Coordinate System dropdown list for selecting the UCS to be used when active.

3 Tool Coordinate System dropdown list for selecting the TCS to be used when active.

NOTE: After switching the coordinate system, the model remains in the same position but the coordinates are now displayed in the new coordinate system.

Switching the Motion Reference Coordinate System

To switch the motion reference coordinate system, click the **Switch Coordinate System** button in the top menu.

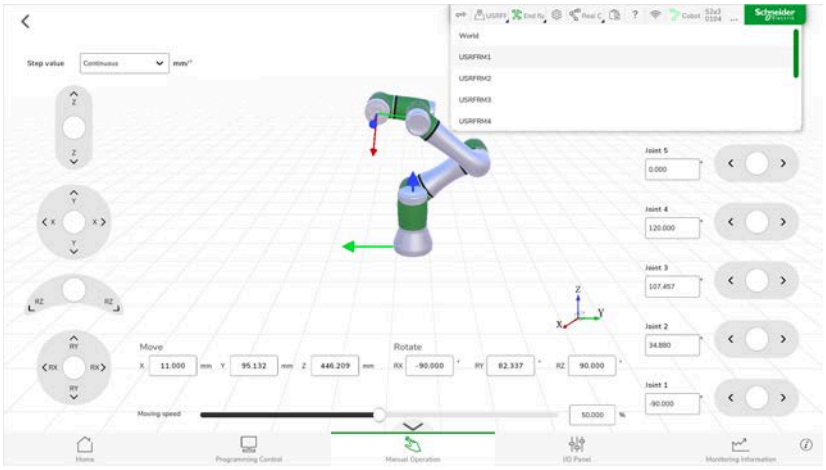
When the **User Coordinate System** icon color turns green, it means that the user coordinate system is being used.

NOTE:

- The tool coordinate system refers to the end flange coordinate system by default. The origin of the flange coordinate system is the center of the flange end. For more information, refer to TCP Settings, page 81
- The default user coordinate system of the Lexium Cobot Arm is the world coordinate system with the base center of the Lexium Cobot Arm as the origin. For further information, refer to User Coordinate System, page 96.

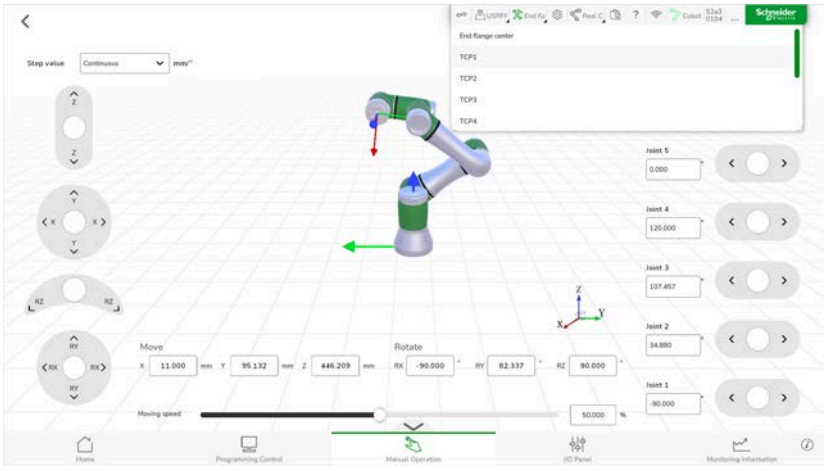
Switching the User Coordinate System

To switch the user coordinate system, perform the following steps:

Step	Action
1	In the top menu, click the small triangle at the lower right corner of the **User Coordinate System** icon. **Result:** The user coordinate system list is displayed. 
2	Select the user coordinate system to be set from the list. **Result:** The user coordinate system switches to the selected one.

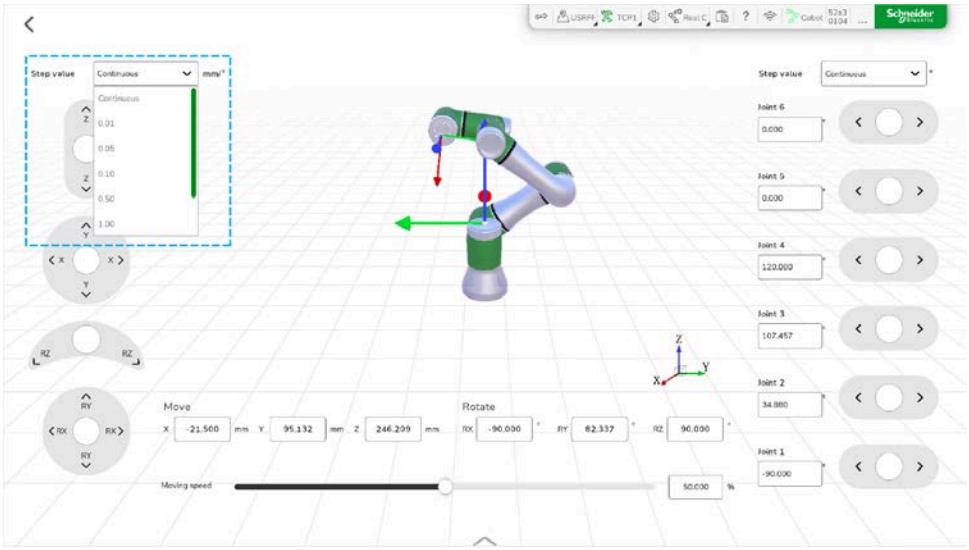
Switching the Tool Coordinate System

To switch the tool coordinate system, perform the following steps:

Step	Action
1	In the top menu, click the small triangle at the lower right corner of the **Tool Coordinate System** icon. **Result:** The tool coordinate system list is displayed. 
2	Select the tool coordinate system to be set from the list. **Result:** The tool coordinate system switches to the selected one.

Adjusting the Step Value

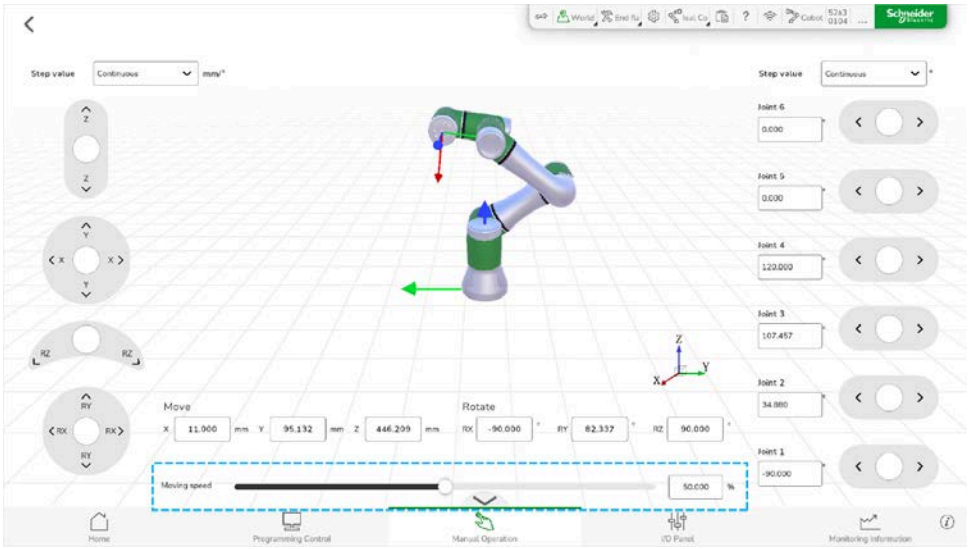
At the top of both sides of the **Manual Operation** interface are step value options that you can use to control the step values of the lower virtual jog buttons. The movement distance or angle of the manual operation is controlled by changing the step value. The smaller the step value, the more precise the Lexium Cobot Arm motion will be.



Controlling the Moving Speed

The moving speed of the Lexium Cobot Arm can be set in two ways:

- By dragging the movement speed bar at the bottom of the **Manual Operation** interface.
- By clicking on the percentage value next to the movement speed bar and entering the specific speed as a percentage of maximum speed.



NOTE: 100% equals 250 mm/s (9.8 in/s), which is the maximum possible speed for manual movement.

Lexium Cobot Motion Types

Hand-Guided Mode

In hand-guided mode, you can manually guide the Lexium Cobot Arm to a position by hand.

To enter the hand-guided mode, press and hold the **FREE** button at the Lexium Cobot Arm.

> **NOTE:** You can also use the **play/pause** button to enter the hand-guided mode if you configure the buttons accordingly. For more information on configuring the buttons of the Lexium Cobot Arm, refer to Auxiliary Hardware Settings, page 138.

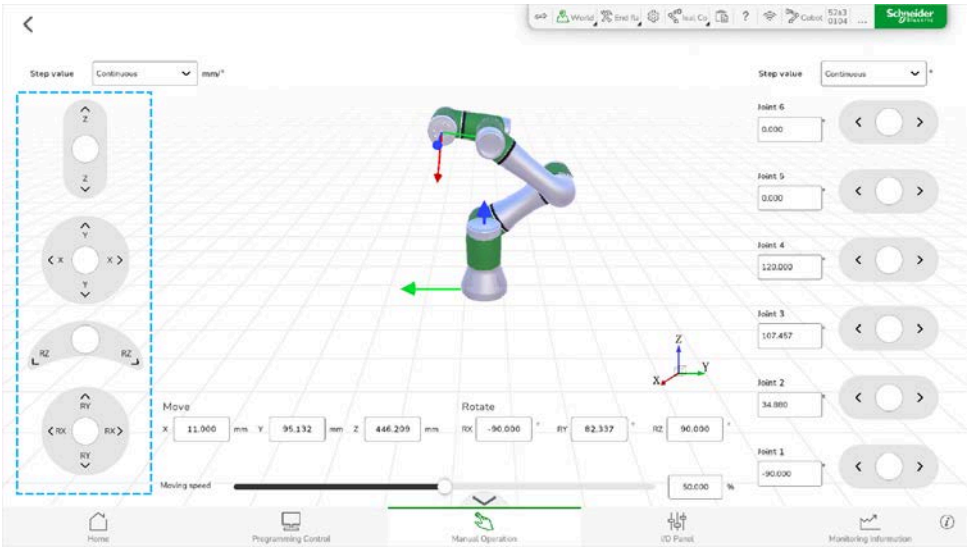
The maximum dragging speed is limited; the default value is 250 mm/s. If this limit is exceeded, the Lexium Cobot Arm leaves the hand-guided mode. For further information on configuring the dragging speed limit, refer to Hand-Guided Mode, page 114.

Spatial Motion

Spatial movement means that the origin of the tool coordinate system of the Lexium Cobot Arm moves in the Cartesian space.

You can choose to move in the user coordinate system or in the tool coordinate system.

As presented in the following figure, the spatial motion refers to the motion of each joint, and the **Manual Operation** of the spatial motion of the Lexium Cobot Arm is displayed as follows.

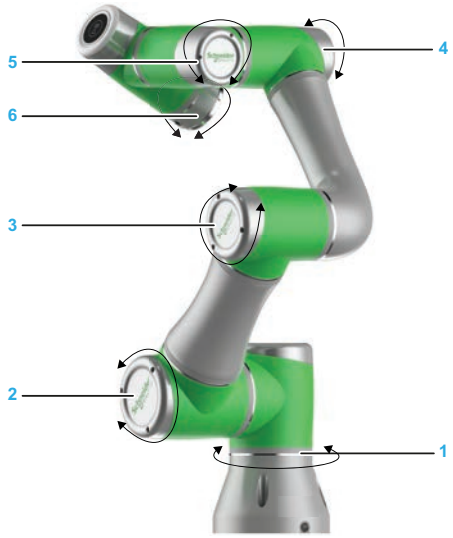


Slide and hold the virtual jog buttons on the left side of the **Manual Operation** interface. The origin of the tool coordinate system performs the corresponding spatial movement in the user coordinate system.

When you release the virtual jog button, it automatically returns to its original position and the Lexium Cobot Arm stops moving.

Joint Motion

The Lexium Cobot Arm consists of six joints.



1 Joint 1

2 Joint 2

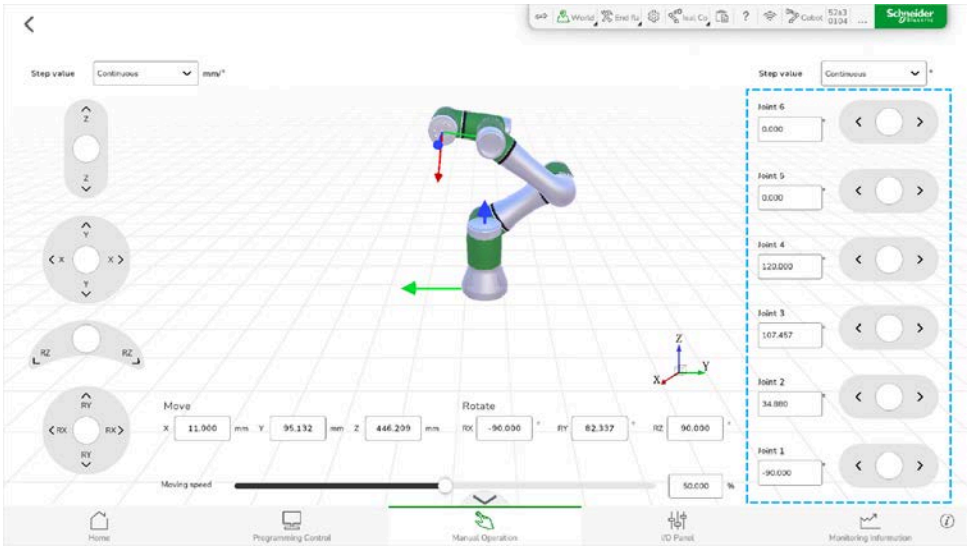
3 Joint 3

4 Joint 4

5 Joint 5

6 Joint 6

Joint motion describes the independent movement of a single joint.



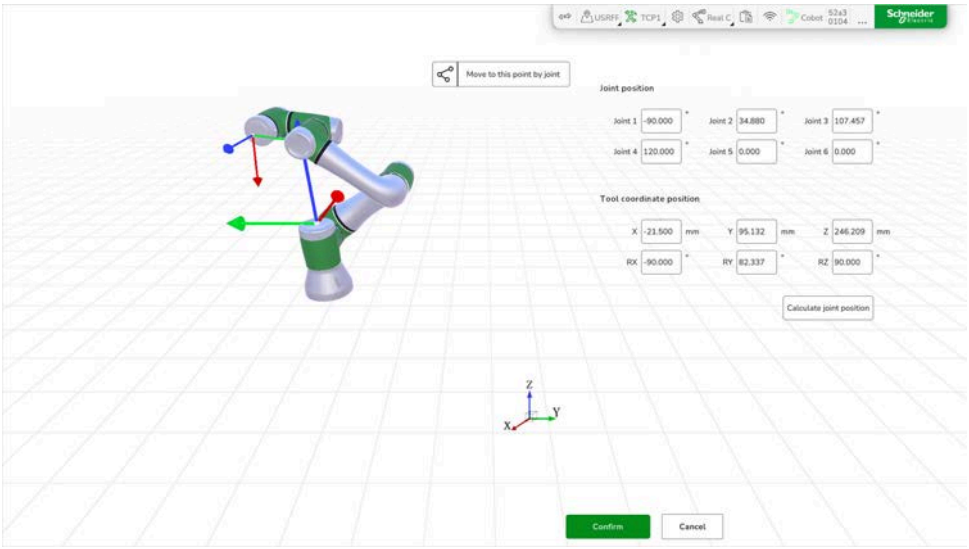
If you slide and hold the virtual jog buttons at the right side of the **Manual Operation** interface, the corresponding joint immediately rotates in the corresponding direction.

When released, the virtual jog button automatically returns to the origin, and the Lexium Cobot Arm stops moving.

Position Motion

With position motion, you can operate the Lexium Cobot Arm manually to the designated position.

You can define the joint position of the Lexium Cobot Arm and the spatial position of the origin of the tool coordinate system in the user coordinate system.



Moving the Lexium Cobot Arm via Position Motion

To move the Lexium Cobot Arm via position motion, perform the following steps:

Step	Action
1	In the **Manual Operation** interface, click the joint information or spatial position information box to be modified. **Result:** The position motion interface is displayed and the **Manual Operation** interface is closed.

To move the Lexium Cobot Arm to a specific joint position, proceed as follows:

Step	Action
1	Under **Joint Position**, type in the end positions of the six joints.
2	To move the Lexium Cobot Arm to the designated position, press and hold the **Move to this point by joint** button until it is reached.
3	Click **Confirm**. **Result:** The Lexium Cobot Arm is in the designated position and the position motion interface is closed.

To move the Lexium Cobot Arm to a specific spatial position, proceed as follows:

Step	Action
1	Under **Tool Coordinate Position**, type in the spatial position of the end point.
2	Click **Calculate Joint Position**.
3	To move the Lexium Cobot Arm to the designated position, press and hold the **Move to this point by joint** button until it is reached.
4	Click **Confirm**. **Result:** The Lexium Cobot Arm is in the designated position and the position motion interface is closed.

To close the position motion interface in case the Lexium Cobot Arm has not reached the designated position, click **Cancel**.

Settings

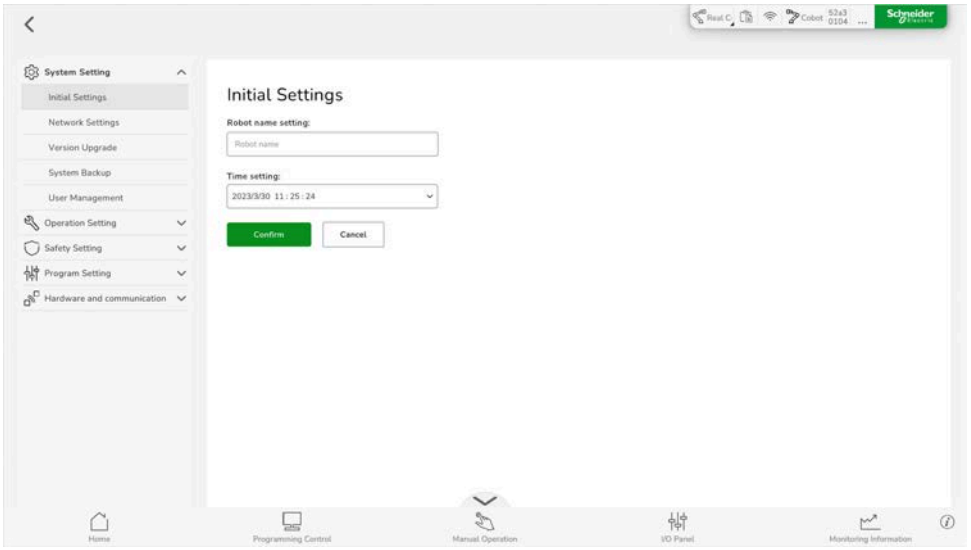
What's in This Chapter

System Setting

Initial Settings

Overview

To configure the Lexium Cobot name and the system time, go to **Settings > System Setting > Initial Settings**.



Robot Name Setting

If you assign a unique Lexium Cobot name under **Robot name setting**, this is used as the name for the Lexium Cobot in EcoStruxure Cobot Expert. Assigning a unique name helps identifying the Lexium Cobot when connecting or switching between Lexium Cobot systems.

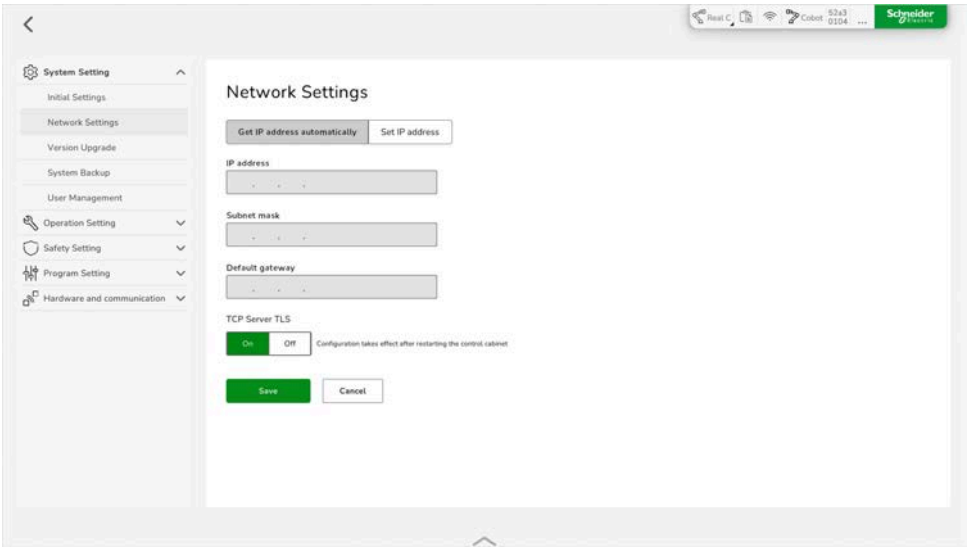
Time Setting

Set the system time so that the time in the Lexium Cobot Controller is consistent with your local time.

Network Settings

Overview

To define how the Lexium Cobot should acquire the IP address, go to **Settings > System Setting > Network Settings**. By default, the Lexium Cobot gets the IP address automatically.



In case the IP address of the Lexium Cobot needs to be set, ensure that the devices communicating with the Lexium Cobot over the network are on the same subnet as the Lexium Cobot.

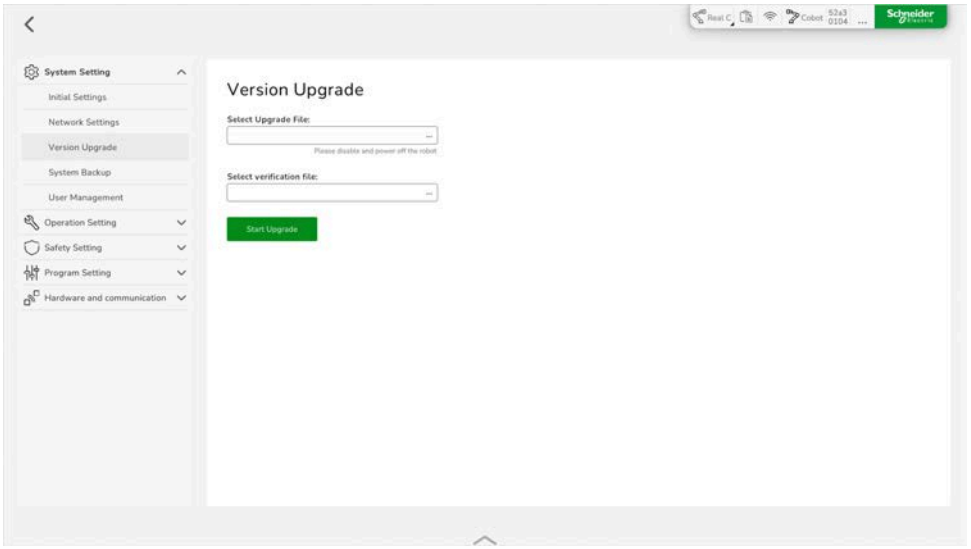
NOTE:

- Disable and power-off the Lexium Cobot Arm when setting the IP address.
- After confirmation of the address change, a network restart is performed automatically by the software.
- Ensure that the IP address is not configured in the 10.5.5.x range.

Version Upgrade

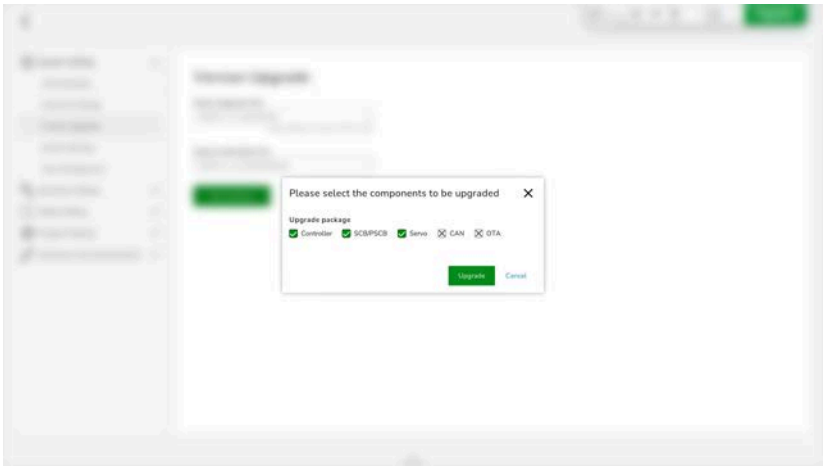
Overview

To upgrade the Lexium Cobot firmware, go to **Settings > System Setting > Version Upgrade**.



Upgrading the Components

To upgrade the components, perform the following steps:

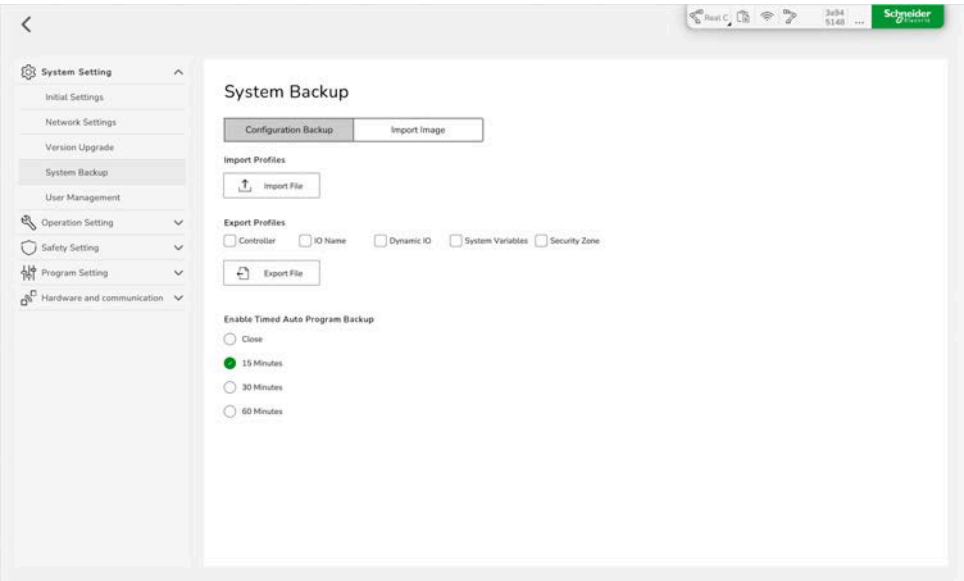
Step	Action
1	Download the firmware package from the Lexium Cobot page on the Schneider Electric website.
2	Connect the Lexium Cobot, page 39.
3	Ensure that the Lexium Cobot Arm is disabled and powered off.
4	In **Settings** > **System Setting** > **Version Upgrade**, click the **Select Upgrade File** box.
5	Select the upgrade package and click **Confirm**. **Result:** The upgrade package is selected. **NOTE:** The original file name must remain unchanged.
6	Click the **Select verification file** box.
7	Select the signature file and click **Confirm**.
8	Click **Start Upgrade**. **Result:** The component selection dialog box is displayed. 
9	Select the components to be upgraded.
10	Click **Upgrade**. **Result:** The upgrade package is being uploaded and installed.
11	Wait until the Lexium Cobot Controller is automatically powered off and then power on the Lexium Cobot Controller with the Control Stick. **NOTE:** The Control Stick has no power while the SCB is in update mode and the status indicator is off.
12	Wait for the start-up and then connect the Lexium Cobot in EcoStruxure Cobot Expert. **Result:** The upgrade is installed.
13	Verify the version by clicking the **About** icon in the feature bar.

System Backup

Configuration Backup

Overview

To import and export profiles or to set the automatic program backup, go to **Settings > System Setting > System Backup**.



Export Profiles

Export profiles are configuration files that contain the parameters that are set in the software.

An export profile can include parameters for:

- **Controller**
- **IO name**
- **Dynamic IO**
- **System Variables**
- **Security Zone**

The particular profile files are packed in an archive file named `lxmcsettings.tar.gz`.

Exporting Profiles

Step	Action
1	In **Settings > System Setting > System Backup > Export profiles**, select the profiles to be exported.
2	Click **Export File**.
3	In the dialog box, choose a location for the file. **NOTE:** Already existing profile archives are overwritten.
4	Click **Confirm**. **Result:** The export is finished. The filename of your exported profile archive is `lxmcsettings.tar.gz`.

Importing Profiles

After exporting a profile archive, you can import the file named `lxmcsettings.tar.gz` to the same Lexium Cobot Controller or to another one.

NOTE: Importing profile files for different Lexium Cobot Controller versions can cause controller errors. When you import profile files, you must keep the versions of the controllers consistent with the profile files.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Be sure to develop a method that uniquely identifies the profile and the version of the controller firmware.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Step	Action
1	In **Settings** > **System Setting** > **System Backup**, click **Import File**. **Result:** The **File Selection** dialog box is displayed.
2	Select the profile to be imported.
3	Click **Confirm**. **Result:** The **Import File** prompt is displayed. If you proceed, the existing profile will be overwritten with the one imported.
4	Click **Confirm**. **Result:** The profile is imported.
5	Restart the Lexium Cobot Controller. **Result:** The import is finished.

Timed Auto Program Backup

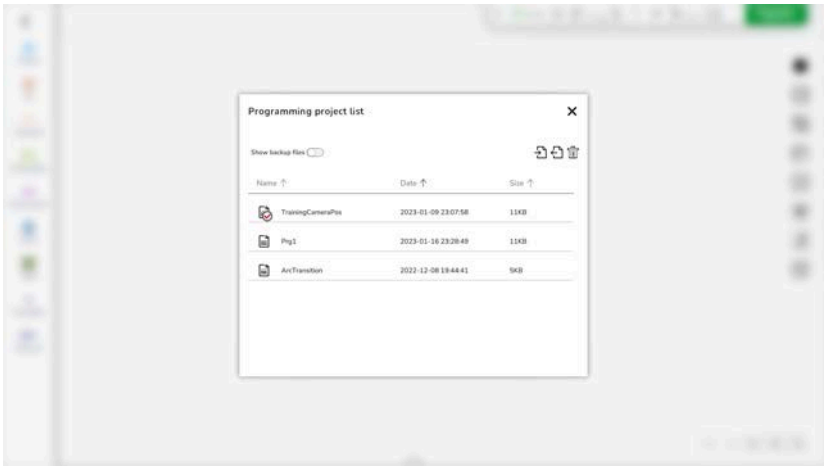
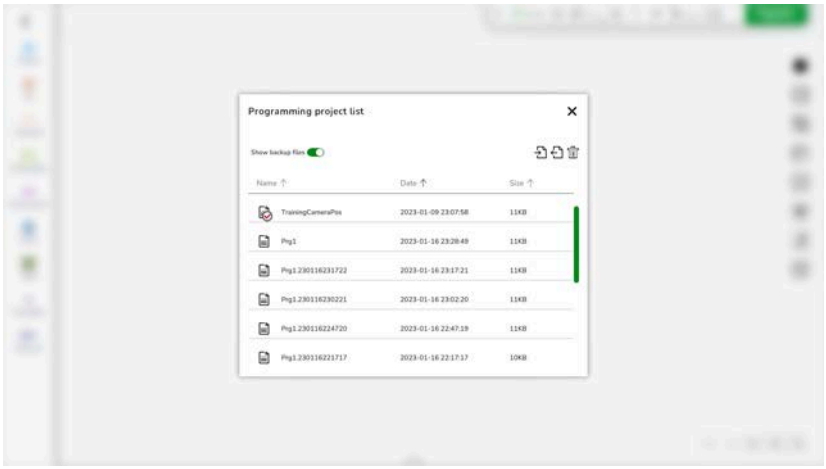
If you activate the automatic program backup, EcoStruxure Cobot Expert automatically saves a backup of the program according to the set interval.

To enable the timed auto program backup, select an interval and then modify the program to start the interval counter.

The saved program backup is named as follows `ProgramName.SystemTime`.

For example: `ArcTransition.221207142506`

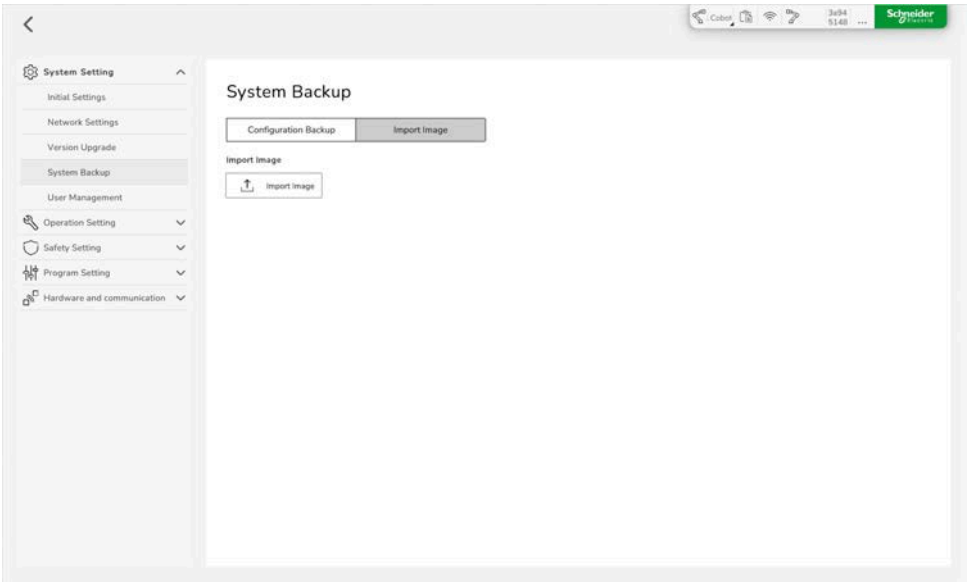
Open an Auto Program Backup

Step	Action
1	In the feature bar, select **Programming Control**. **Result:** The **Programming Control** interface is displayed.
2	In the **Programming Control** menu bar, click the **Open** icon:  **Result:** The **Programming project list** dialog box is displayed. 
3	Activate the **Show backup files** toggle. **Result:** The backup files are displayed. 
4	Click the name of the backup file to be opened. **Result:** The backup file is displayed.

Import Image

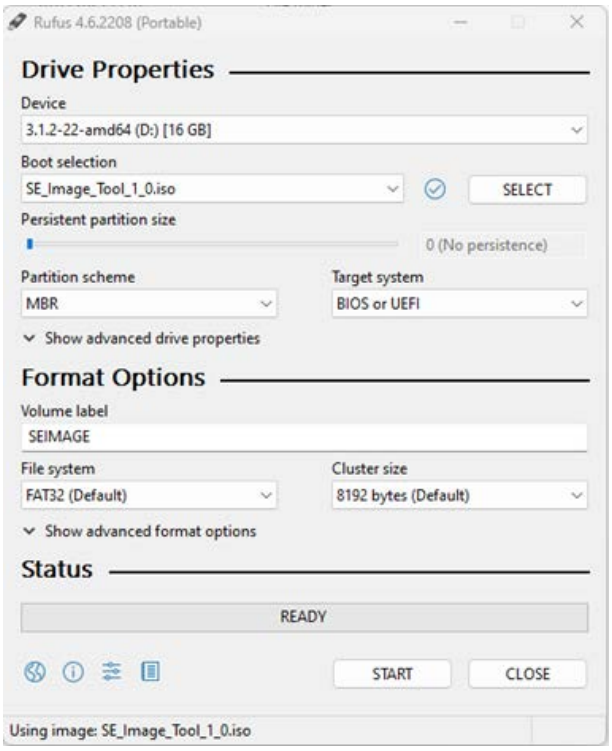
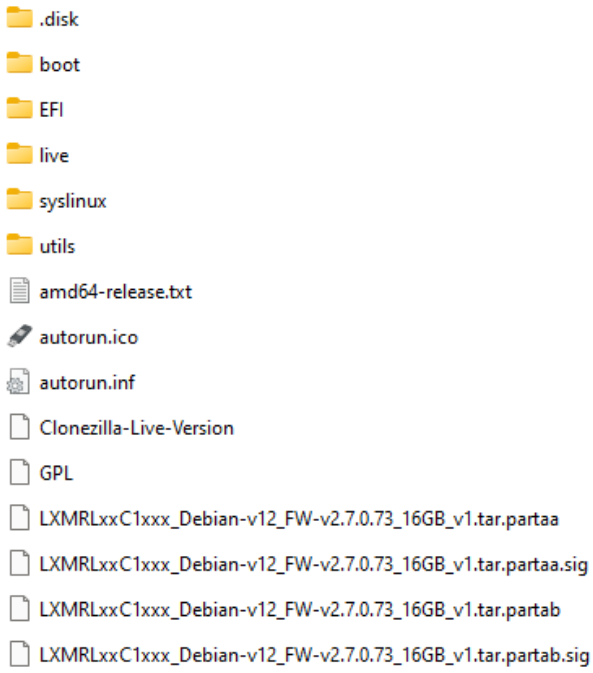
Overview

To perform a system upgrade of the Lexium Cobot Controller operating system (**Import Image**), go to **Settings > System Setting > System Backup > Import Image**.



Preparing the USB Flash Drive

Prepare the USB flash drive in advance before proceeding with the system upgrade. Use a USB 2.0 flash drive for this purpose.

Step	Action
1	Burn the provided image tool to the USB flash drive using a burning tool, for example Rufus, with the following properties. 
2	Copy the provided image and signature files to the root directory of the USB flash drive. • LXMRLxxC1xxx for Lexium Cobot Cabinet Controller • LXMRL00C2xxx for Lexium Cobot Compact Controller 

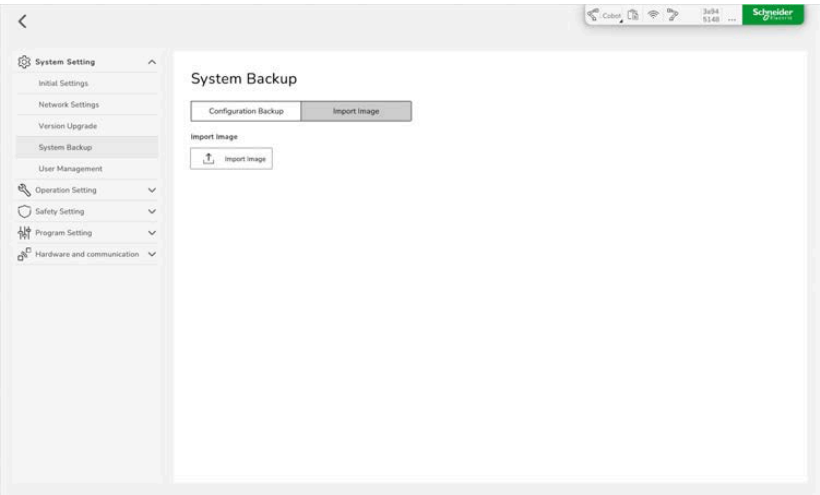
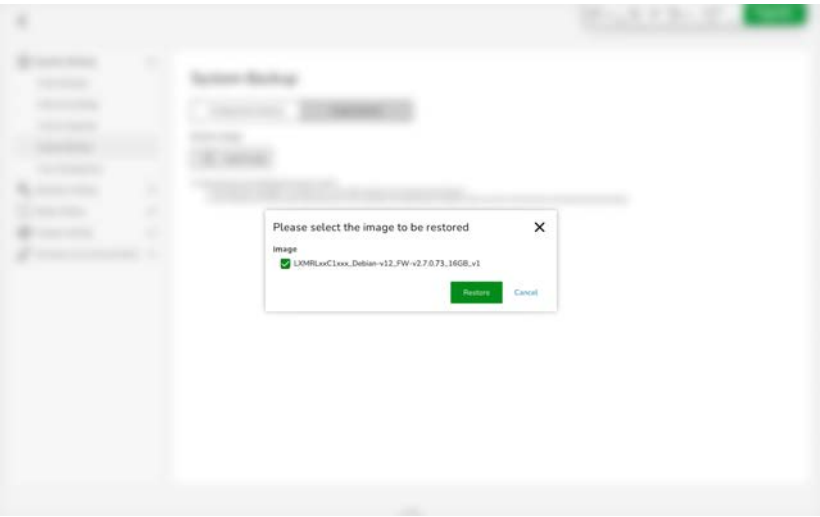
Preparing the Lexium Cobot Controller

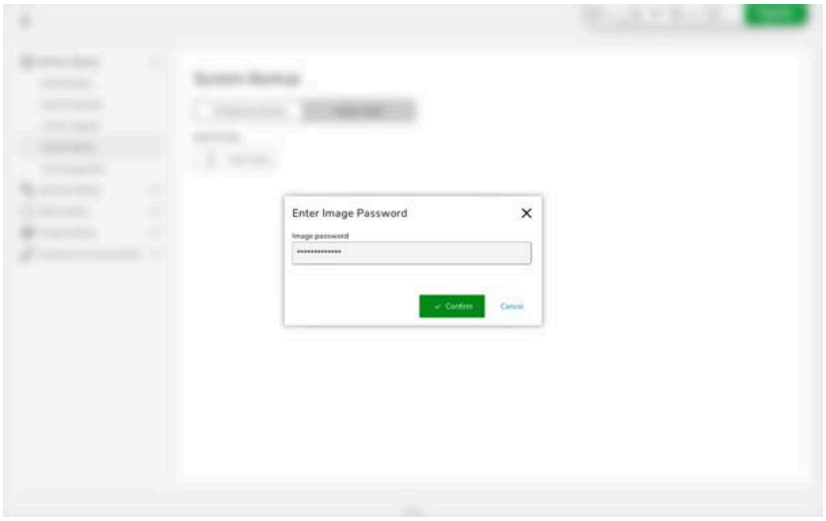
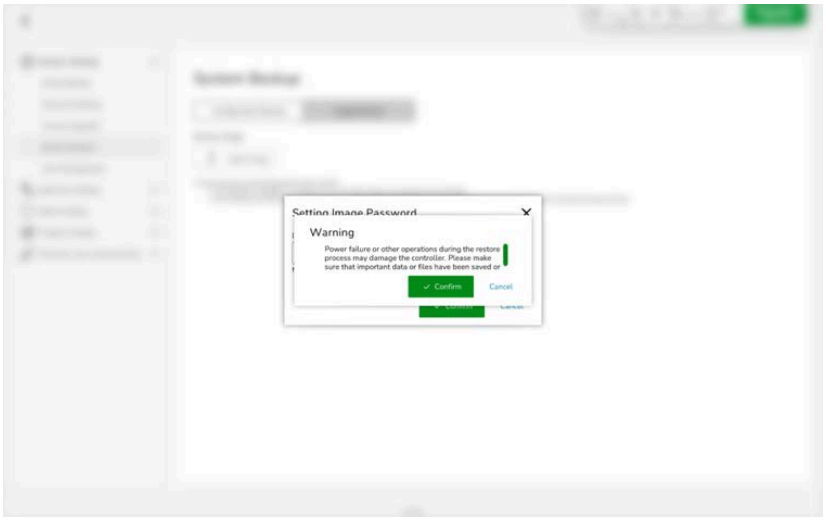
Consider the following points before performing the system upgrade:

- Ensure that firmware version 1.7.0.70_SE is installed on the Lexium Cobot Controller.
- Backup the data of the Lexium Cobot Controller including user configurations and programs. The present system will be overwritten, so unsaved data will be lost.
- Do not remove power to the Lexium Cobot Controller during the upgrade. Interruption of power may damage the Lexium Cobot Controller.
- Ensure that the network connection is stable so that you can monitor the status of the Lexium Cobot Controller during the upgrade.

Performing the System Upgrade

Consider using EcoStruxure Cobot Expert V1.7.0.61 software to ensure the **Import Image** dialog is displayed.

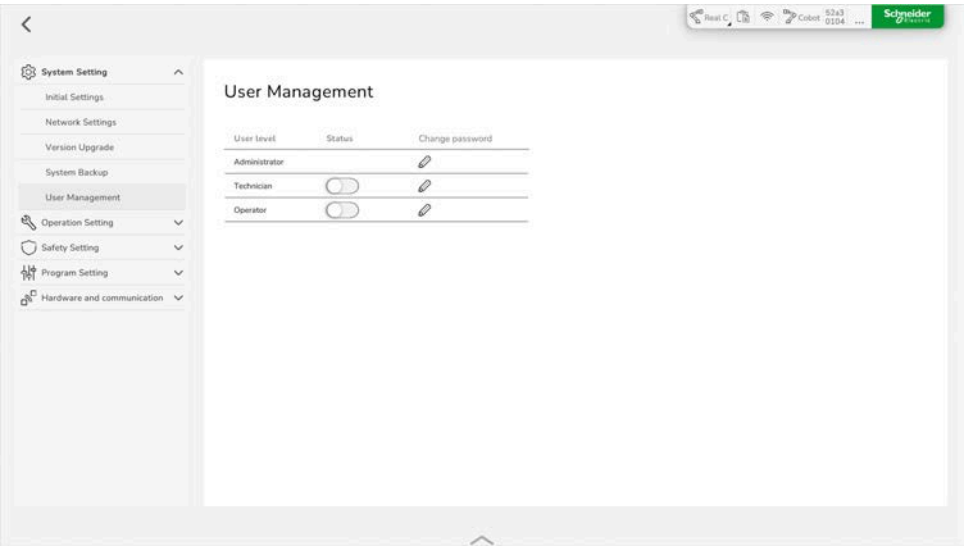
Step	Action
1	Login to the Lexium Cobot Controller using EcoStruxure Cobot Expert.
2	Connect the prepared USB flash drive to the appropriate USB port. • Lexium Cobot Cabinet Controller: Connector **CN9 USB** • Lexium Cobot Compact Controller: Connector **CN22/23 USB 2.0** **NOTE:** Ensure that you are using a USB 2.0 flash drive.
3	In **Settings** > **System Setting** > **System Backup** > **Import Image**, click **Import Image**.  **Result:** The system verifies the connected USB flash drive and displays the detected image. 
4	Verify the image and click **Import**.

Step	Action
5	Enter the image password `Administrator` and click **Confirm**. 
6	Read the message and click **Confirm**.  **Result:** The upgrade process is started.
7	Monitor the upgrade procedure in the dialog box **Connect robot**. **NOTE:** After starting the upgrade, it may take a few minutes before the Lexium Cobot Controller appears in the list.
8	Verify the upgrade in the dialog box **Connect robot**. **NOTE:** The controller firmware version 2.7.0.73_SE_X64 must be shown as Control version.

User Management

Overview

To manage the different user levels, go to **Settings > System Setting > User Management**.



In this section, you can activate or deactivate the user levels **Technician** and **Operator** and change the passwords for all defined user levels.

For further information on the user levels, refer to Connecting the Lexium Cobot, page 39.

NOTE:

- This section is only accessible for logged-in administrators.
- For guidance on creating strong passwords, refer to Creating Strong Passwords, page 27.
- For resetting the passwords of the user levels to factory settings, refer to Reset User Levels to Factory Settings, page 29.

Activating or Deactivating the User Levels

To activate a user level, activate the **Status** toggle.

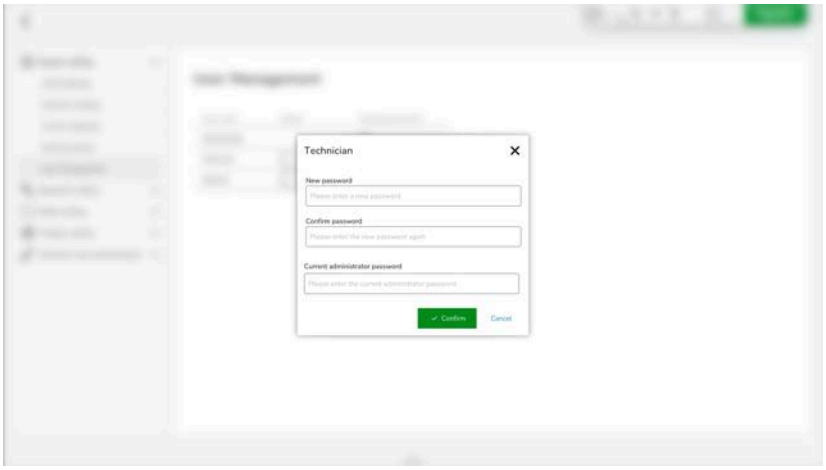
To deactivate a user level, deactivate the **Status** toggle.

NOTE:

- The user level **Administrator** cannot be deactivated.
- The administrator has the option to set the passwords for the technician or operator at first activation. For further information, refer to Connecting the Lexium Cobot, page 39

Changing Passwords

To change a password, perform the following steps:

Step	Action
1	In **Setting** > **System Setting** > **User Management**, in the row of the user level for which you want to change the password, click the **Change password** icon. **Result:** The password settings dialog box for the selected user level is displayed. 
2	Type in the new password, confirm the new password, and type in the **Administrator** password.
3	Click **Confirm**. **Result:** The new password is set.

Operation Setting

TCP Settings

Overview

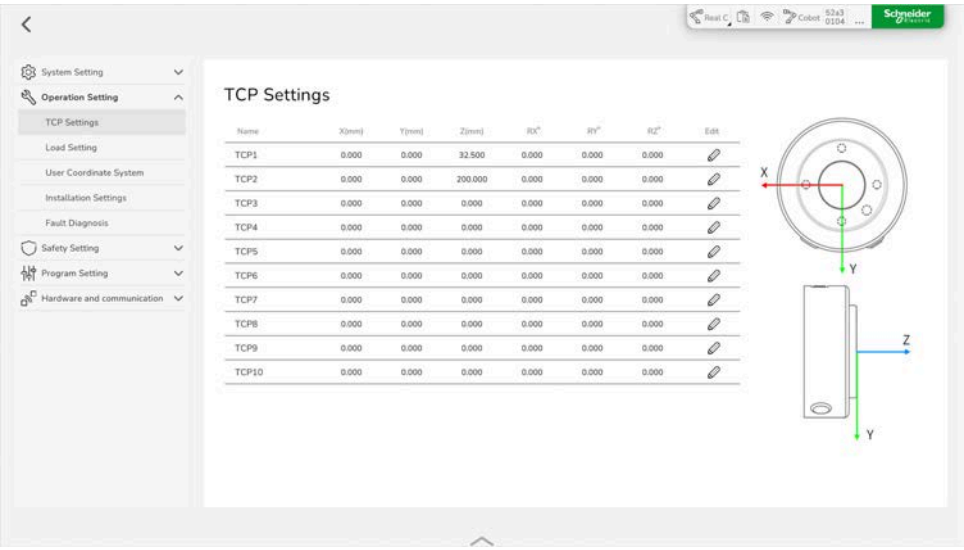
The Lexium Cobot Arm has a default tool coordinate system which is the flange coordinate system.

The origin of the flange coordinate system is the center of the flange end. The positive direction of the Z-axis is defined by the outward direction of the flange end. The negative direction of the Y axis is defined by the line connecting the center of the flange end and the Tool IO connector. The positive direction of the X-axis is defined by the right-hand screw rule.

The parameters of the flange coordinate system cannot be changed.

In addition to the default tool coordinate system, the Lexium Cobot provides 10 additional configurable TCP settings.

To view the TCP settings, click **Settings > Operation Setting > TCP Settings**.



Setting Methods

In EcoStruxure Cobot Expert three methods are available to edit the TCP parameters:

- Input settings, page 82

  When using input settings you must first calculate the required pose offset of the tool coordinate system relative to the flange coordinate system as needed. Then you can type in these data into the data fields of the dialog box.

  When using this method, the Lexium Cobot Arm can be disabled.

- Four-point setting, page 83

  A fixed reference point in the working space is defined. You control the Lexium Cobot Arm so that the TCP endpoint reaches the fixed point from four different poses. Then, the expected pose offset of the tool coordinate system relative to the end flange center coordinate system is calculated automatically.

  When using this method, the Lexium Cobot Arm must be enabled.

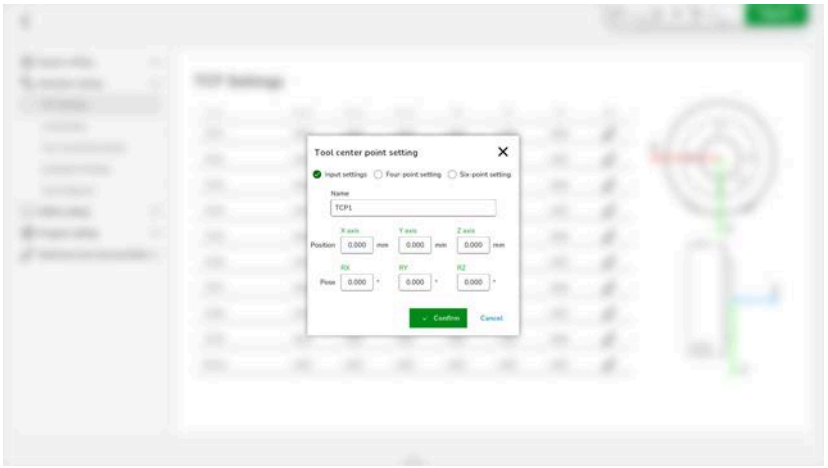
- Six-point setting, page 85

  Use the six-point setting when the tool axes at the Lexium Cobot Arm end are not perpendicular or parallel to the Lexium Cobot Arm end flange.

  When using this method, the Lexium Cobot Arm must be enabled.

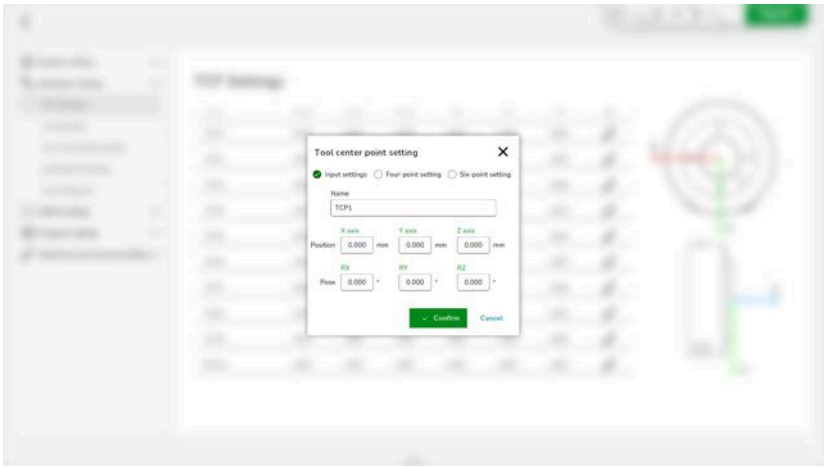
Setting the TCP Parameters Manually (Input Settings)

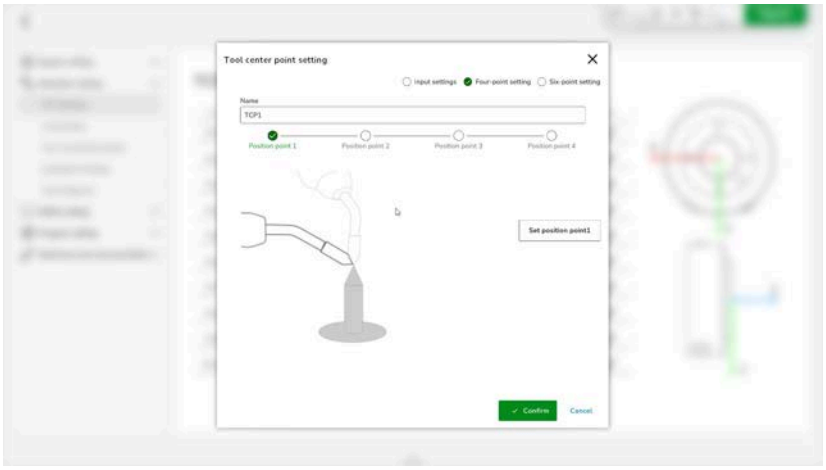
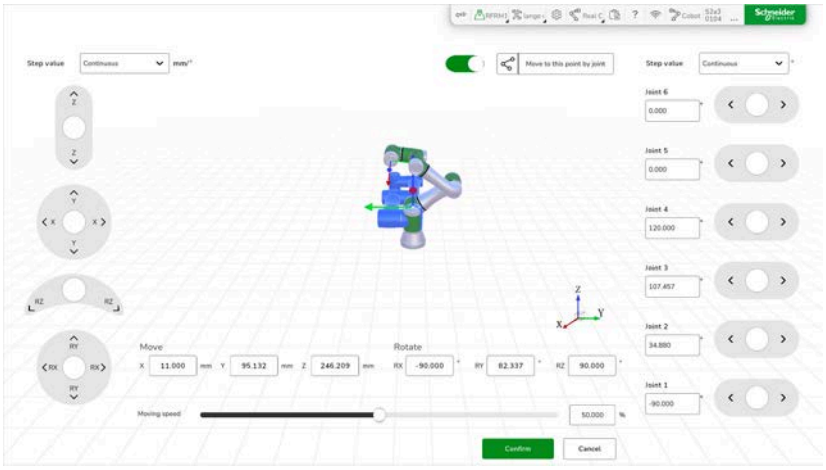
To set the TCP parameters manually, perform the following steps:

Step	Action
1	Calculate the required pose offset of the tool coordinate system relative to the end flange center coordinate system.
2	In **Settings > Operation Setting > TCP Settings**, in the row of the TCP you want to edit, click the **Edit** icon. **Result:** The dialog box **Tool center point setting** is displayed. 
3	Verify that **Input settings** is selected.
3	Optionally, edit the name of the TCP.
4	Type in the appropriate values.
5	Click **Confirm**. **Result:** The TCP is set.

Setting the TCP Parameters via Four-Point Setting

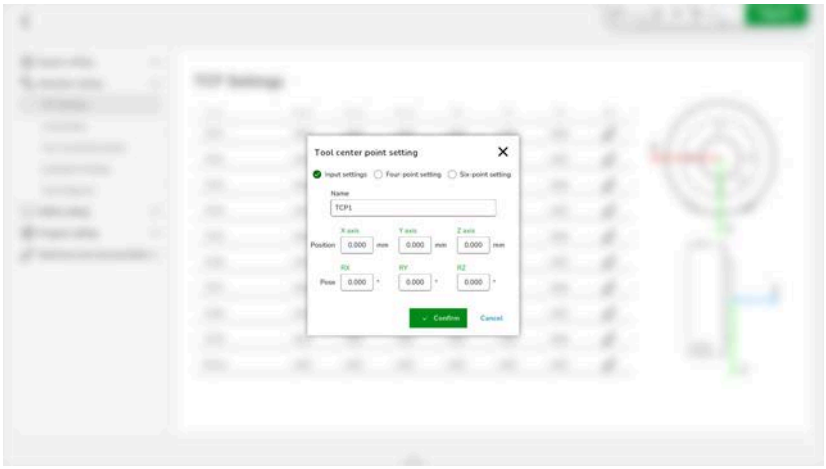
To set the TCP parameters with the four-point setting method, perform the following steps:

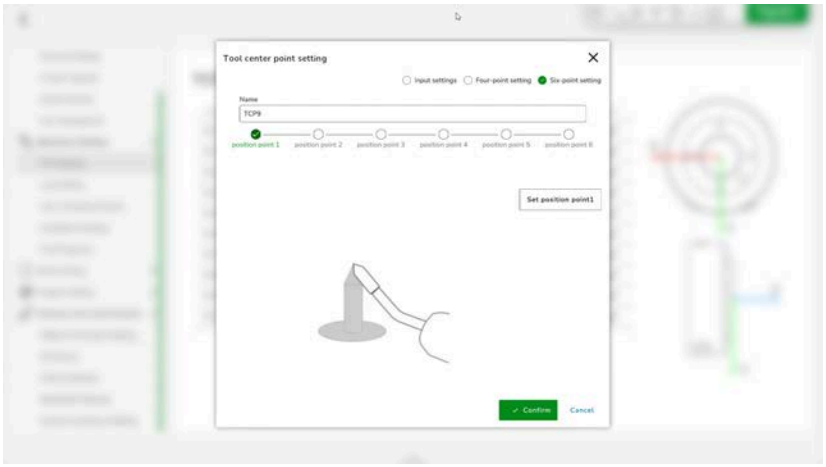
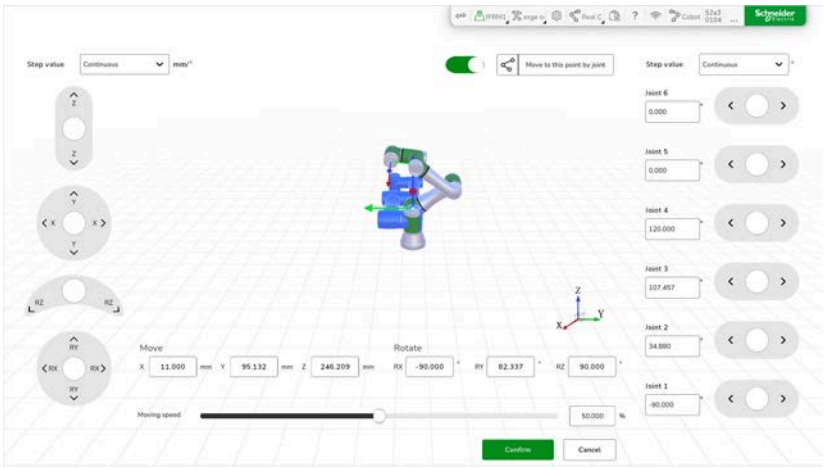
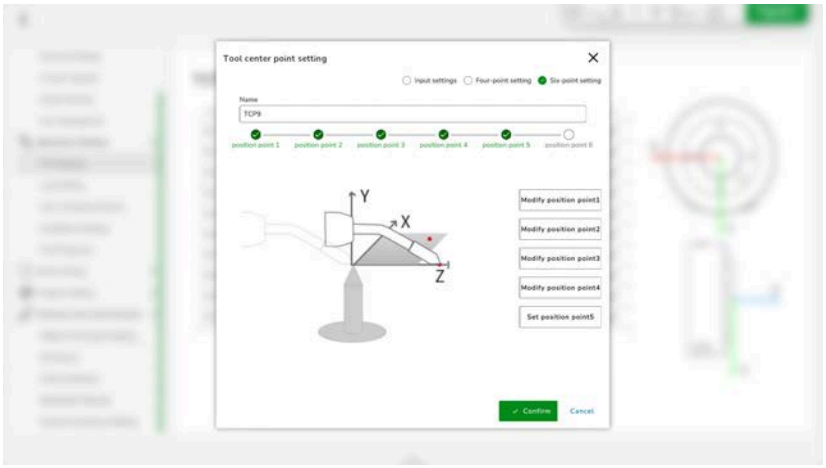
Step	Action
1	Power on and enable the Lexium Cobot Arm.
2	Click **Settings** > **Operation Setting** > **TCP Settings**.
3	In the row of the TCP you want to edit, click the **Edit** icon. **Result:** The configuration dialog box is displayed. 
4	Select **Four-point setting**.
5	Optionally, edit the name of the TCP.
6	Define a fixed reference point in the working space. For example, the vertex of a pointed cone.

Step	Action
7	Click **Set position point1**.  **Result:** The **Manual Operation** interface is displayed. 
8	Manually move the Lexium Cobot Arm by hand-guiding or **Manual Operation** so that the end effector reaches the fixed reference point and then click **Confirm**.
9	Repeat steps 7 and 8 for the other three positions.
10	Click **Confirm**. **Result:** The TCP is set.

Setting the TCP Parameters via Six-Point Setting

To set the TCP parameters with the six-point setting method, perform the following steps:

Step	Action
1	Power on and enable the Lexium Cobot Arm.
2	Click **Settings** > **Operation Setting** > **TCP Settings**.
3	In the row of the TCP you want to edit, click the **Edit** icon. **Result:** The configuration dialog box is displayed. 
4	Select **Six-point setting**.
5	Optionally, edit the name of the TCP.
6	Define a fixed reference point in the working space. For example, the vertex of a pointed cone.

Step	Action
7	Click **Set position point1**.  **Result:** The **Manual Operation** interface is displayed. 
8	Manually move the Lexium Cobot Arm by hand-guiding or **Manual Operation** so that the end effector reaches the fixed reference point and then click **Confirm**.
9	Repeat steps 7 and 8 for the next three positions.
10	Click **Set position point5**.  **NOTE: position point 5** and **position point 6** are used to define the working plane of the end effector.
11	Leave the position of point 4 unchanged. Move forward along the positive direction of the Z-axis of the respective tool coordinate system to get the position of point 5 and then click **Confirm**.
12	Click **Set position point6**.

Step	Action
13	Leave the position of point 5 unchanged. Move forward in the respective XZ plane to get the position of point 6 and then click **Confirm**.
14	Click **Confirm**. **Result:** The TCP is set.

Load Setting

Overview

If the payload information is set correctly, the working state of the Lexium Cobot can be calculated correctly by the Lexium Cobot Controllers.

If the set payload information deviates from the physical situation, the Lexium Cobot Controllers can incorrectly detect a collision during the movement of the Lexium Cobot Arm. As a result, the Lexium Cobot Arm movement is stopped. Furthermore, the Lexium Cobot Controllers try to compensate for gravity, which also may cause unanticipated behavior in hand-guided mode.

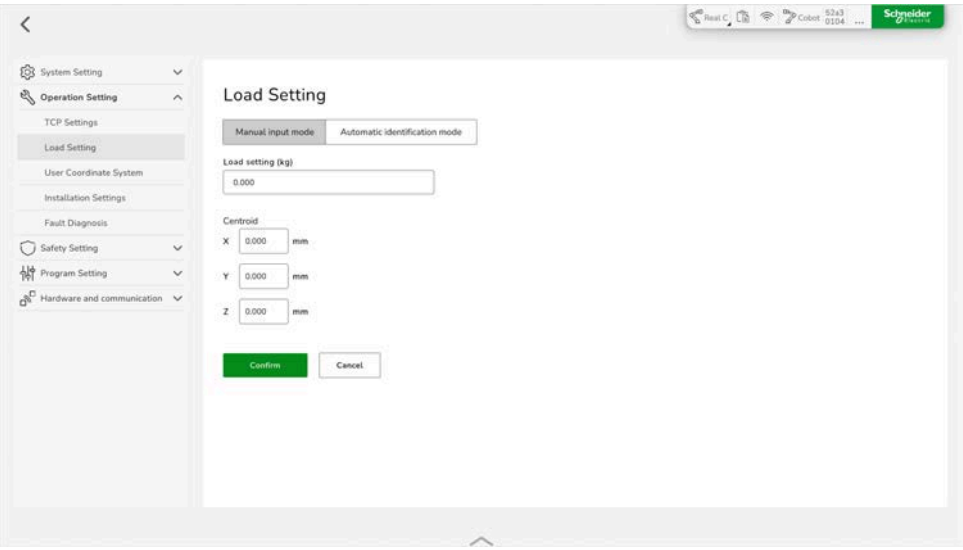
⚠WARNING

UNINTENDED EQUIPMENT OPERATION

- Ensure that the payload is mounted at the Tool Center Point.
- Ensure that the load setting is configured properly.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

To edit the load settings, go to **Settings > Operation Setting > Load Setting**.



There are two methods available for setting the payload:

- **Manual input mode**

  In manual input mode, you can manually enter the measured or calculated payload information.

- **Automatic identification mode**

  The **Automatic identification mode** can identify and calculate the mass and centroid position of the payload through the Lexium Cobot Arm motion using predefined positions.

  Using this mode, the payload must be mounted and the Lexium Cobot Arm must be powered on and enabled.

Manual Input Mode

To type in the measured or calculated information manually, perform the following steps:

Step	Action
1	In **Settings** > **Operation Setting** > **Load Setting**, select **Manual input mode**.
2	Type in the load setting and the centroid data.
3	Click **Confirm**.

NOTE:

- The position of the mass center is relative to the Lexium Cobot Arm end flange center, and X, Y and Z of the position of the mass center are also the spatial values in the flange coordinate system.
- Use a 3D design software, for example PTC Creo, AutoDESK Inventor, or SolidWorks, for an accurate calculation.

Prerequisites for Automatic Identification Mode

The following prerequisites must be met before using the **Automatic identification mode**:

- If the joints have been replaced or the joints have been overtwisted, verify the position and direction of the joint. For further information, refer to chapter *Verification of Mechanical Position* in the *Lexium Cobot Hardware Guide*.
- The payload is installed properly.

Identification Phases of Automatic Identification Mode

The process for identification follows two phases:

- On-load
- No-load

NOTE: The no-load state is identified only after the on-load identification.

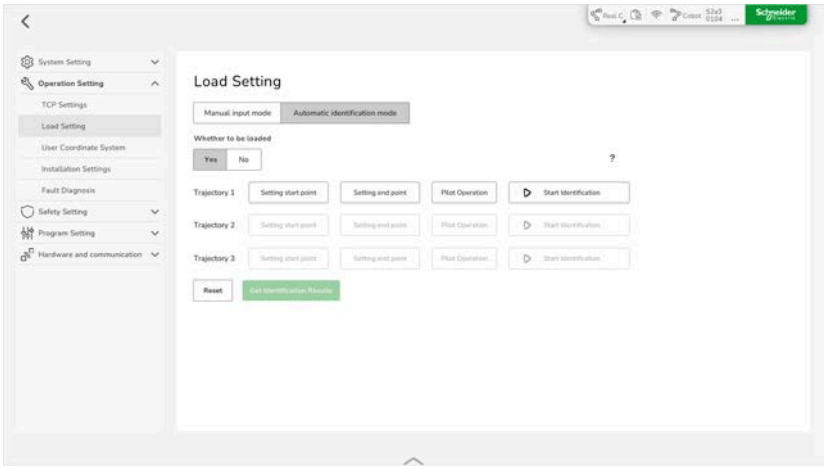
Start Point and End Point Configuration

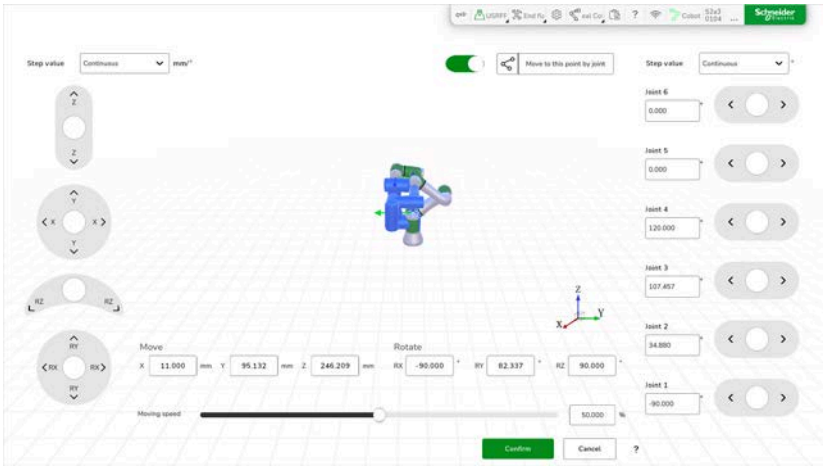
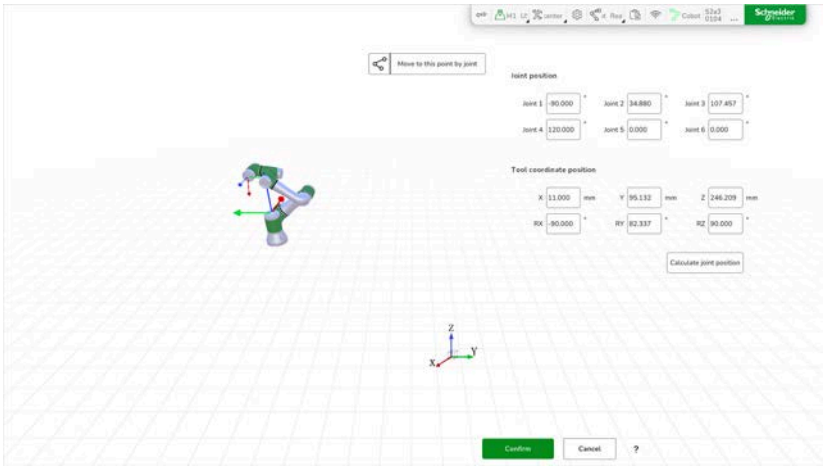
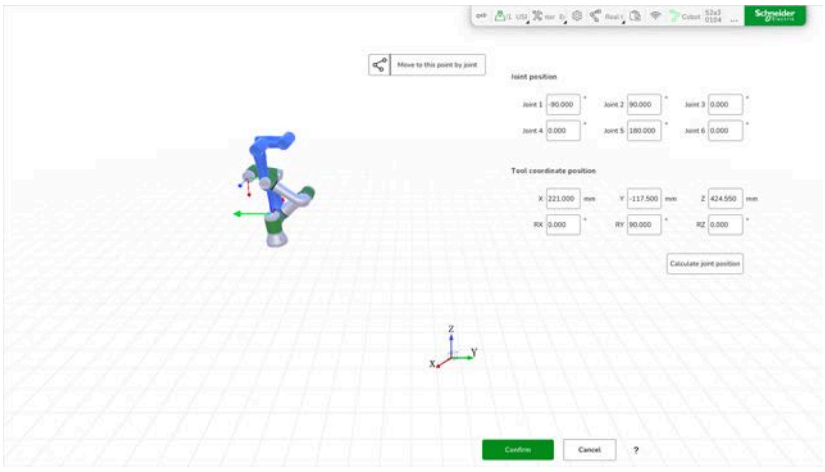
	Start Point Configuration	End Point Configuration
Trajectory 1	Joint 2 = 90° Joint 3 = 0° Joint 4 = -60°...60° Joint 5 = 180° Joint 6 = -60°...60° **NOTE:** Use the same angle for joints 4 and 6.	Joint 2 = 90° Joint 3 = 0° Joint 4 = -60°...60° Joint 5 = 180° Joint 6 = -60°...60° **NOTE:** Use the same angle for joints 4 and 6. However, the angles of joints 4 and 6 must differ from the angles of the starting point by at least 10°.
Trajectory 2	Joint 2 = 90° Joint 3 = 0° Joint 4 = -60°...60° Joint 5 = 180° Joint 6 = angle of joint 4 + 90° **NOTE:** The angle of Joint 6 is 90° greater than the angle of Joint 4.	Joint 2 = 90° Joint 3 = 0° Joint 4 = -60°...60° Joint 5 = 180° Joint 6 = angle of joint 4 + 90° **NOTE:** The angle of joints 4 must differ from the angles of the starting point by at least 10°. The angle of Joint 6 is 90° greater than the angle of Joint 4.
Trajectory 3	Joint 2 = 90° Joint 3 = 0° Joint 4 = 0° Joint 5 = 170°...175° Joint 6 = 0° **NOTE:** Use the same angle for joints 4 and 6.	Joint 2 = 90° Joint 3 = 0° Joint 4 = 0° Joint 5 = 185°...190° Joint 6 = 0° **NOTE:** Use the same angle for joints 4 and 6.

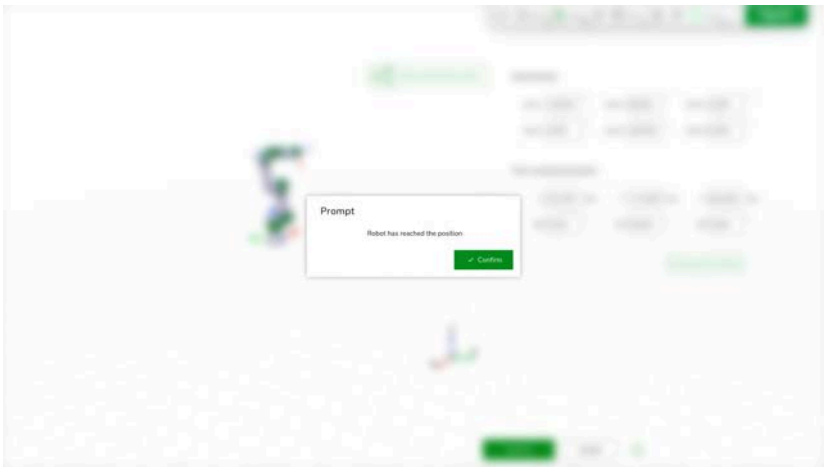
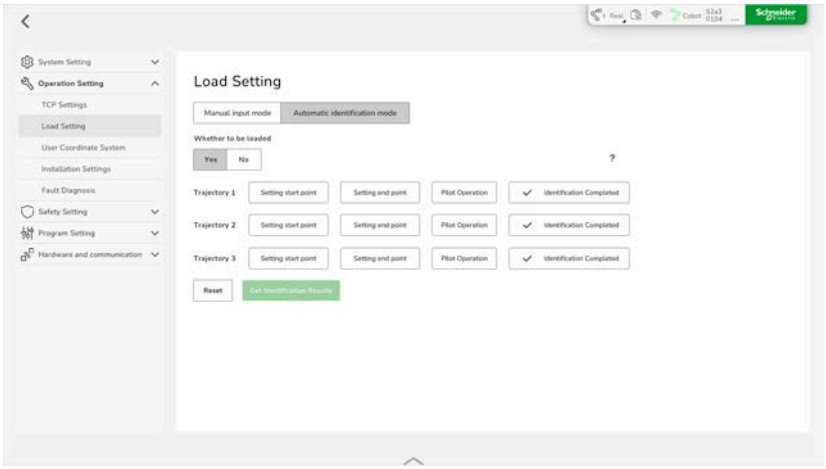
NOTE: If the start points are not set correctly, EcoStruxure Cobot Expert displays a notification.

Automatic Identification Mode - Phase 1: On-Load

To use the **Automatic identification mode**, perform the following steps:

Step	Action
1	Verify that the payload is mounted, power on and enable the Lexium Cobot Arm.
2	In **Settings > Operation Setting > Load Setting**, in **Manual input mode**, set the approximate payload parameters of the payload to be detected.
3	Select **Automatic identification mode**. **Result:** The following confirmation prompt is displayed. 
4	Click **Confirm**. **Result:** The **Automatic identification mode** pane is displayed. 
5	Click **Yes** in **Whether to be loaded**.

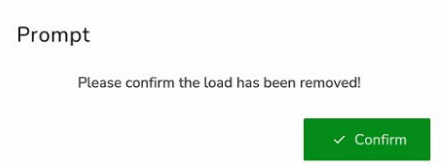
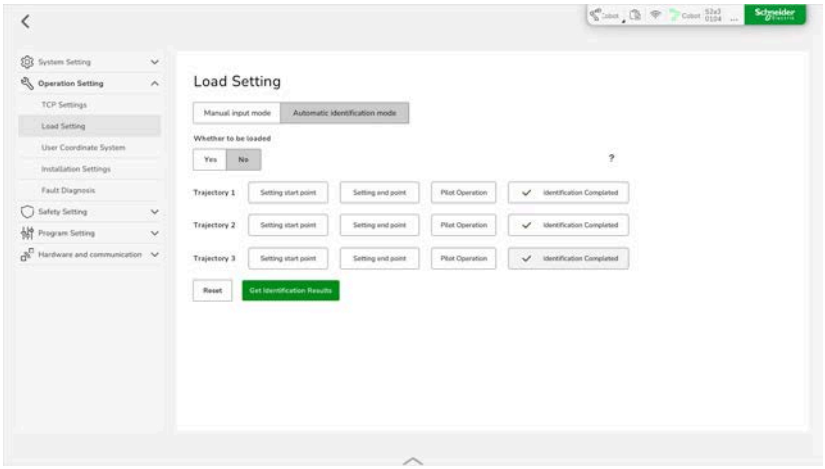
Step	Action
6	Click **Setting start point** at **Trajectory 1**. **Result:** The **Manual Operation** interface is displayed. 
7	Click one of the joint angles. **Result:** The editing interface is displayed. 
8	Set the angles according to the table Start Point Configuration, page 90: **NOTE:** Use the same angle for joints 4 and 6. 

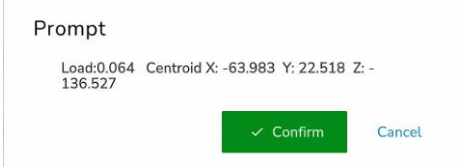
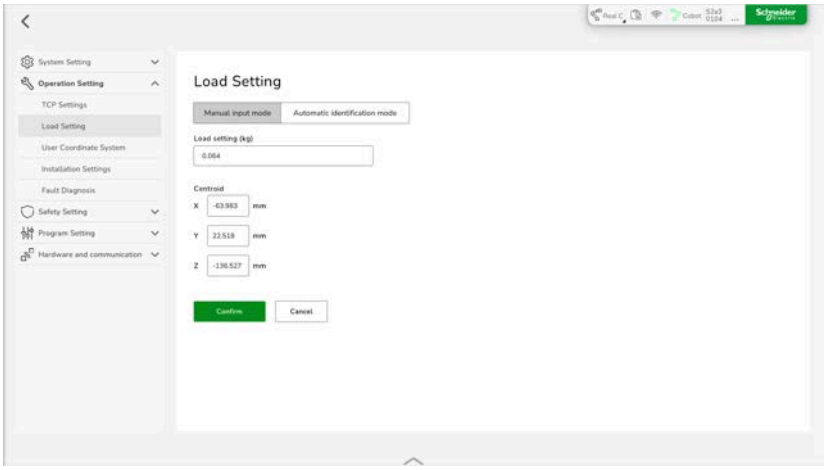
Step	Action
9	Move the joints to the defined angles by pressing **Move to this point by joint** until the prompt **Robot has reached the position** is displayed. 
10	Click **Confirm** in the prompt, the editing interface, and the **Manual Operation** interface to get back to the **Load Setting** pane.
11	Click **Setting end point** at **Trajectory 1** and repeat steps 5, 6, 7, and 8 for the end position. The angles are the same as for the starting point, but angles 0 and 6 must be at least 10° different between the starting point and the end point.
12	Click and hold **Setting start point** until the prompt **Robot has reached the position** is displayed.
13	Optionally, to verify that the identification movement is collision free: Click and hold **Pilot Operation** until the prompt **Robot has reached the position** is displayed. **Result:** The joint rotates from the start point to the end point. • If an interference occurred during the pilot operation, adjust the angles. • If no interference occurred, click and hold **Setting start point** until the prompt **Robot has reached the position** is displayed to return to the starting point.
14	When the Lexium Cobot Arm is in its initial position, click **Start identification**. **Result:** The identification process is running. When it is completed, the **Start Identification** button changes to **Identification completed**.
15	Repeat steps 6 to 14 for **Trajectory 2** and **Trajectory 3**.  **Result:** The on-load setting is completed. Continue with the no-load setting.

NOTE: To clear identification results, click **Reset**.

Automatic Identification Mode - Phase 2: No-Load

To use the **Automatic identification mode**, perform the following steps:

Step	Action
1	Remove the payload.
2	In **Settings** > **Operation Setting** > **Load Setting**, select **Automatic identification mode** and select **No** from **Whether to be loaded**. **Result:** The following confirmation prompt is displayed. 
3	Click **Confirm**.
4	At **Trajectory 1**, click and hold **Setting start point** until the prompt **Robot has reached the position** is displayed.
5	Click **Start identification**. **Result:** The identification is done.
6	Repeat steps 4 and 5 for the other trajectories. 

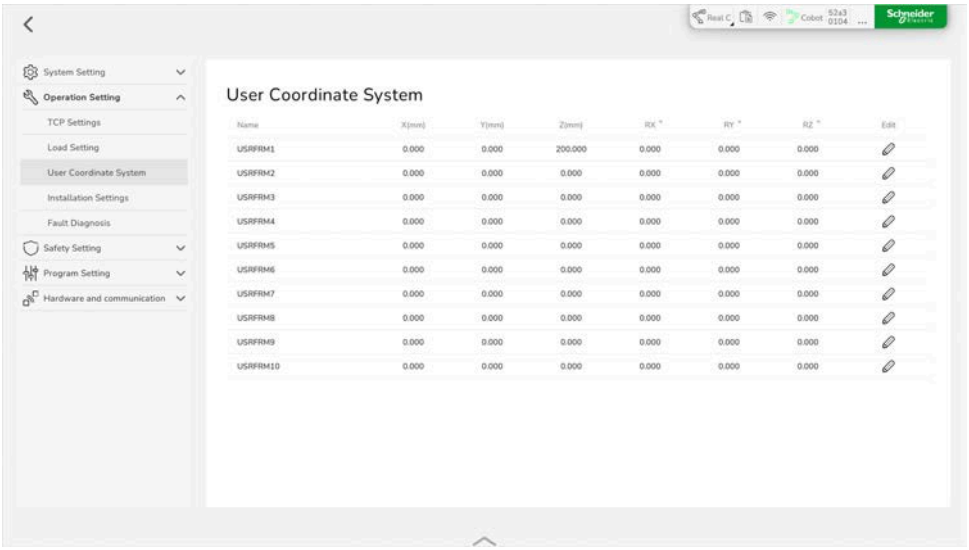
Step	Action
7	Click **Get identification results**. **Result:** The parameters of the payload mass and the centroid are displayed. **Example:** 
8	Click **Confirm**. **Result:** The automatic identification is completed. The **Manual input mode** parameters are overwritten with the calculated parameters. **Example:** 

NOTE: To clear identification results, click **Reset**.

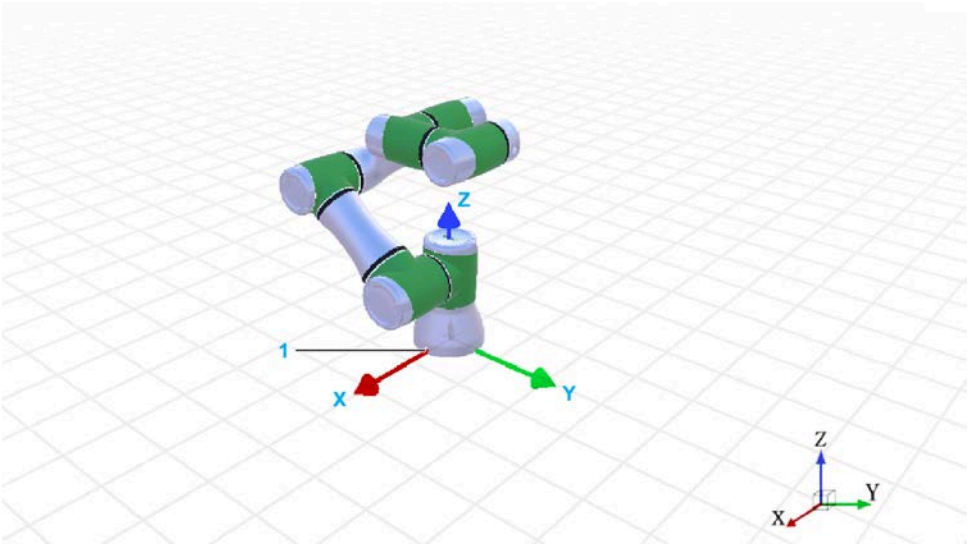
User Coordinate System

Overview

To edit user coordinate systems, go to **Settings > Operation Setting > User Coordinate System**



The default user coordinate system of the Lexium Cobot Arm is the world coordinate system with the base center of the Lexium Cobot Arm as the origin.



1 Cable connector

- +X at the Lexium Cobot Arm is in the direction of the power cable at the base.
- +Y is determined based on the right-hand screw rule.
- +Z is the direction in which the base points vertically to the Lexium Cobot Arm.

In addition to the world coordinate system, the Lexium Cobot Arm has 10 user coordinate systems with editable parameters.

There are two methods for setting the parameters of a user coordinate system:

- **Input settings**

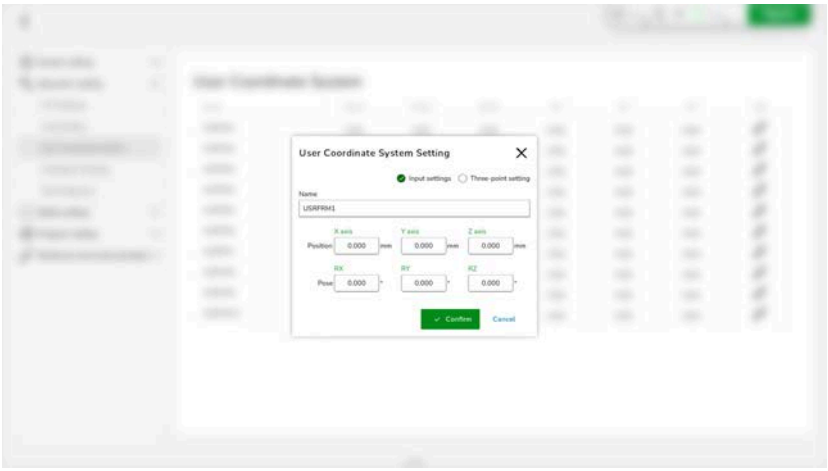
  Edit the parameters manually.
- **Three-point Setting**

  Set three points and calculate the directions X, Y, Z and the orientations RX, RY, and RZ automatically.

  To use this method, the Lexium Cobot Arm must be powered on and enabled.

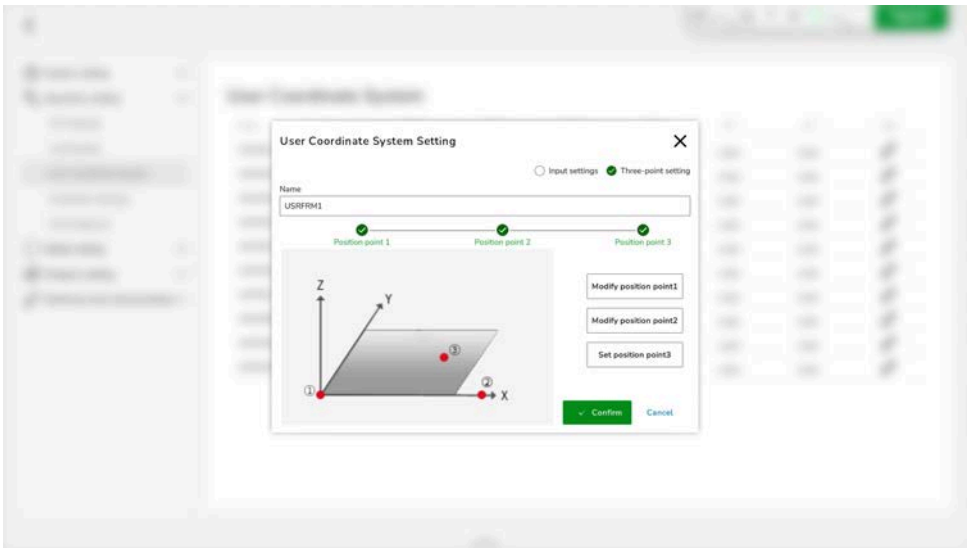
Using Input Settings

To edit the parameters of the user coordinate system manually, perform the following steps:

Step	Action
1	Calculate the required pose offset of the user coordinate system relative to the world coordinate system.
2	In **Settings** > **Operation Setting** > **User Coordinate System**, click the **Edit** icon in the row of the user coordinate system to be edited. **Result:** The dialog box **User Coordinate System Setting** is displayed. 
3	Optionally, edit the name of the user coordinate system.
4	Select **Input settings**.
5	Type in the calculated data from step 1.
6	Click **Confirm**. **Result:** The parameters of the coordinate system are updated.

Three-Point Setting Definitions

The parameters of the axis directions X, Y, and Z of the respective user coordinate system are automatically calculated from three-position points. The axis directions X, Y, and Z of the user coordinate system generated by the three-point setting are aligned with those of the world coordinate system.

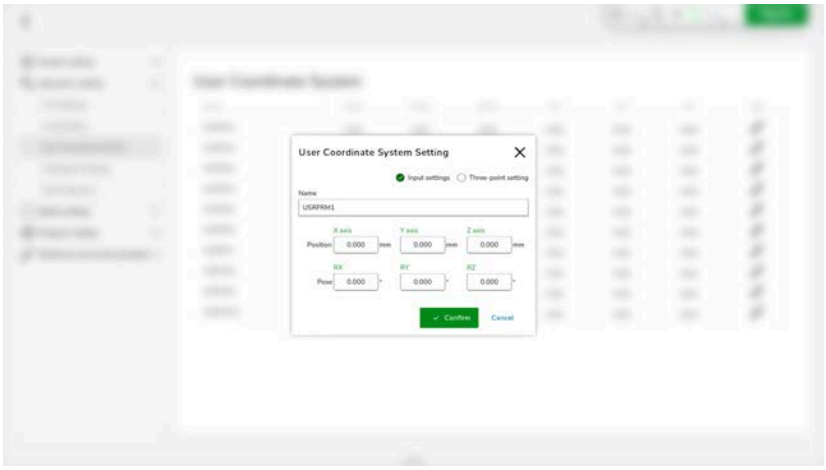
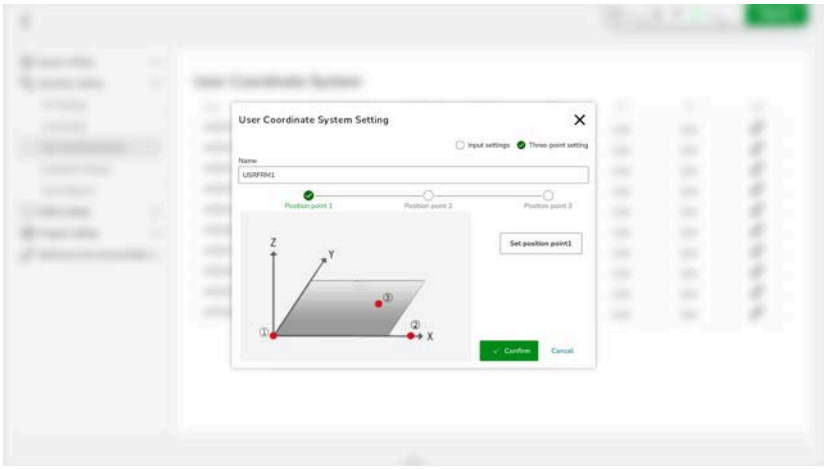
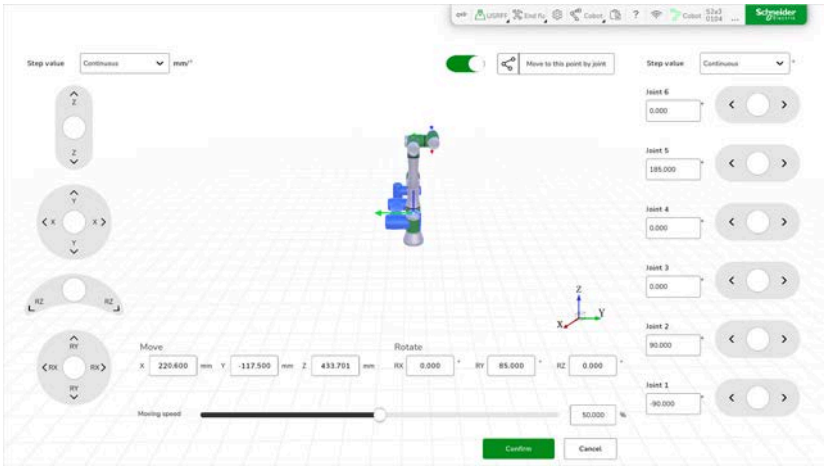


The following table presents the definition of the three points.

Position Point	Description
1	Origin of the user coordinate system.
2	Any point in the forward direction of the X-axis of the user coordinate system.
3	Any point in the first quadrant of the XY-plane of the user coordinate system.

Using the Three-Point Setting

To calculate the parameters of the user coordinate system with the three-point setting, perform the following steps:

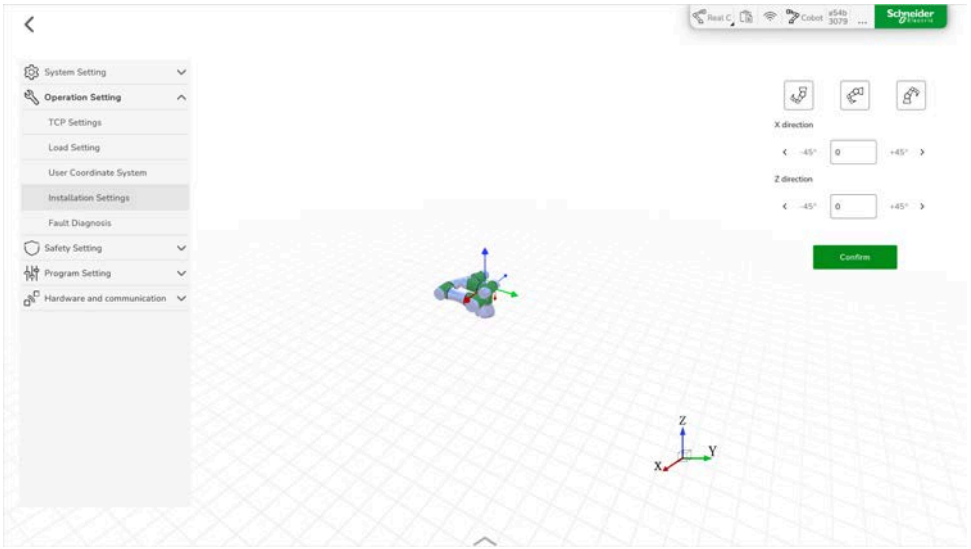
Step	Action
1	Verify that the Lexium Cobot Arm is powered on and enabled.
2	In **Settings** > **Operation Setting** > **User Coordinate System**, click the **Edit** icon in the row of the user coordinate system to be edited. **Result:** The dialog box **User Coordinate System Setting** is displayed. 
3	Select **Three-point setting**. 
4	Click **Set position point1**. **Result:** The **Manual Operation** interface is displayed. 
5	Set the point and click **Confirm**.

Step	Action
6	Repeat steps 4 and 5 for the other two points.
7	Click **Confirm**. **Result:** The directions X, Y, and Z are calculated.

Installation Settings

Overview

To define the installation position of the Lexium Cobot Arm, go to **Settings > Operation Setting > Installation Settings**.



The Lexium Cobot Arm supports installation at any position and angle. After installing the Lexium Cobot Arm, define the information about the installation position and angle of the Lexium Cobot Arm in EcoStruxure Cobot Expert to help ensure the correct representation of the Lexium Cobot Arm by the software and proper functionality of the Lexium Cobot.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that the installation setting is configured properly.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Setting the Installation

To adjust the installation position of the Lexium Cobot Arm in EcoStruxure Cobot Expert, perform the following steps:

Step	Action
1	Disable the Lexium Cobot Arm.
2	In **Settings** > **Operation Setting** > **Installation Settings**, click on the predefined buttons and/or use the triangles to set the installation angles of the mounting surface. 
3	Click **Confirm**. **Result:** The installation position is set.

Error Diagnosis

Overview

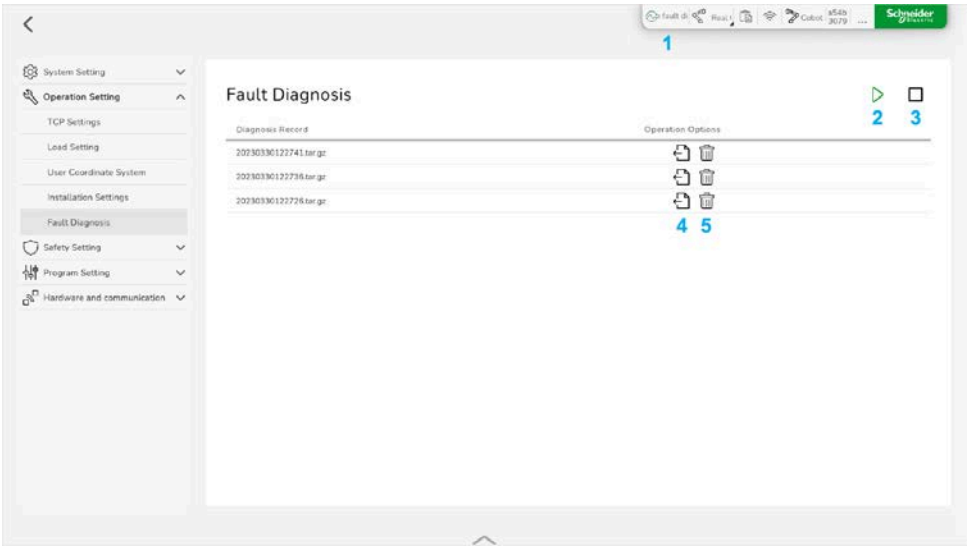
To display the error diagnosis, go to **Settings > Operation Setting > Fault Diagnosis**.

In case of a Lexium Cobot detected error, the Lexium Cobot Controller automatically saves the information in a compressed file named after the system time and displays it in the **Fault Diagnosis** pane.

In **Operation Options**, you can download or delete the file.

Operation Options

You have the following operation options in the **Fault Diagnosis** pane:



1 Status indicator for an ongoing error diagnosis

2 Start diagnosis

3 Stop diagnosis

4 Download the diagnosis file

5 Delete the diagnosis file

Starting an Error Diagnosis

To perform an error diagnosis manually, in **Settings > Operation Setting > Fault Diagnosis**, click the **Start** button.

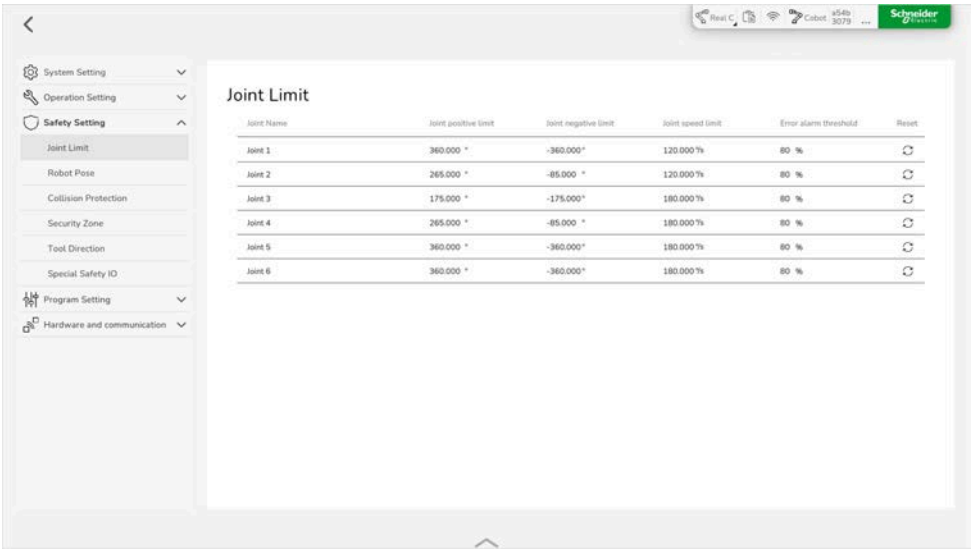
The error diagnosis is executed for 30 seconds. If you want to stop it earlier, click the **Stop** button that is displayed during recording.

Safety Setting

Joint Limit

Overview

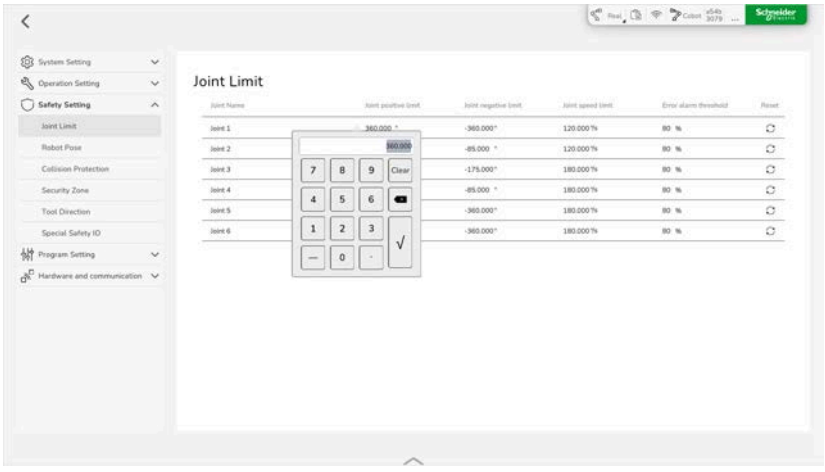
To set for each joint of the Lexium Cobot Arm the joint limit angles, the joint limit speed, and the error alarm threshold, go to **Settings > Safety Setting > Joint Limit**:



NOTE:

- **Error alarm threshold** means that the Lexium Cobot triggers an alarm when the motion displacement error of the mechanical arm is greater than the alarm threshold. 100% represents 1°.
- Since the default values for the positive limit, negative limit and speed limit of the joint are the maximum ranges, you can change them within the default range. The default ranges are presented in the preceding figure.

Editing the Joint Limit Parameters

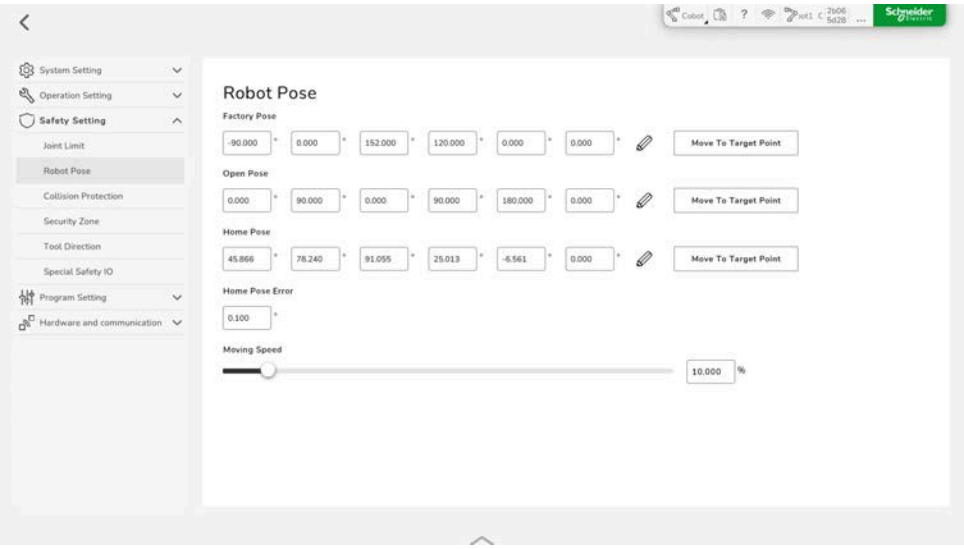
Step	Action
1	In **Settings** > **Safety Setting** > **Joint Limit**, click the value to be edited. **Result:** The editing interface is displayed. 
2	Type in the value and click ✓. **Result:** The new value is set.

NOTE: To reset the values of a joint to the default values, click the **Reset** icon.

Robot Pose

Overview

To define different poses for the Lexium Cobot Arm, go to **Settings > Safety Setting > Robot Pose**.

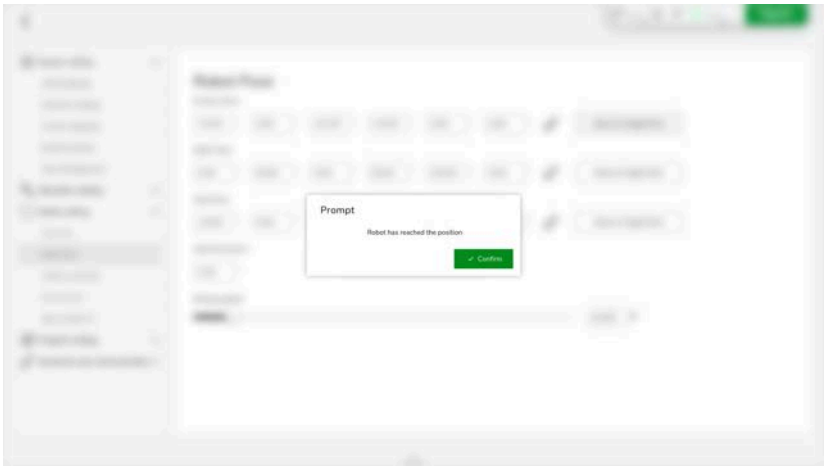


You can use and adjust three different poses for the Lexium Cobot Arm:

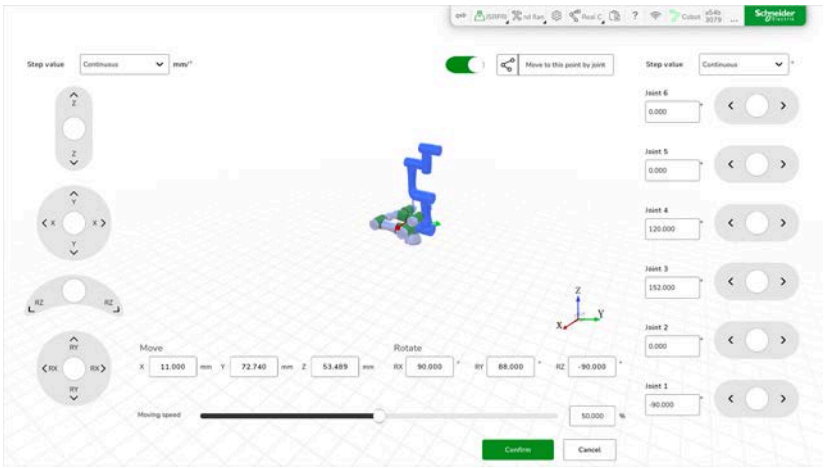
- **Factory Pose**: folding positions of the Lexium Cobot Arm in delivery condition.
- **Open Pose**: zero position for each joint.
- **Home Pose**: initial position of the Lexium Cobot Arm. You can define the **Home Pose** in the software and reach the home position through the **Home** button on the Control Stick.

Home Pose Error: Acceptable deviation from the designated **Home Pose**.

Moving to a Pose

Step	Action
1	Power on and enable the Lexium Cobot Arm.
2	Click and hold **Move To Target Point** at the corresponding pose until the prompt **Robot has reached the position** is displayed. **NOTE:** To adjust the moving speed of the Lexium Cobot Arm, use the **Moving Speed** slider. 100% equals 250 mm/s (9.8 in/s). 

Setting the Lexium Cobot Arm Pose

Step	Action
1	Power on and enable the Lexium Cobot Arm.
2	Click the **Edit** button in the **Robot Pose** dialog box of the pose to be edited: **Result:** The **Manual Operation** interface is displayed. 
3	Set the pose and click **Confirm**. **Result:** The pose definition is set.

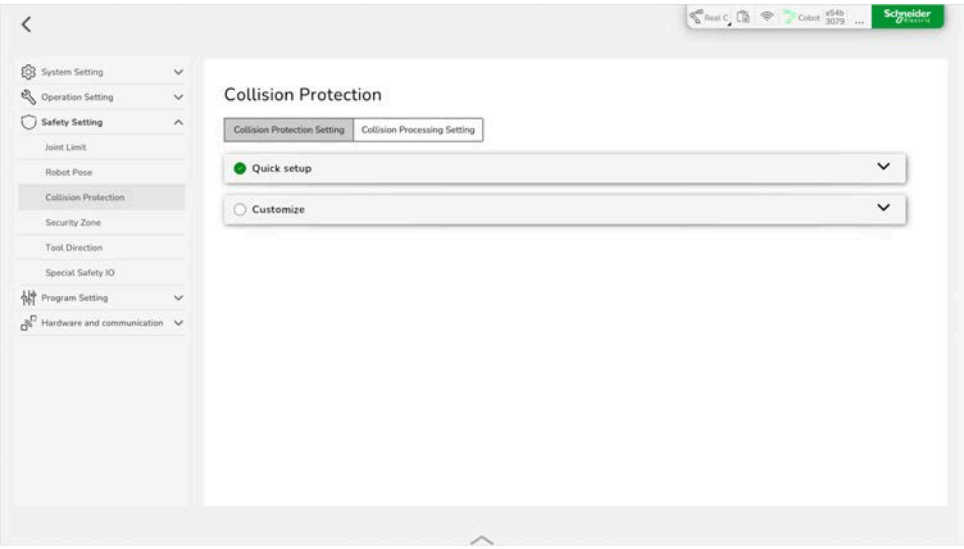
Collision Protection

Overview

In the event of a collision during manual operation, the Lexium Cobot Arm does not rebound, either in motion or at rest. The Lexium Cobot Arm can be pushed within a certain range if the external force continues to act.

In the event of collision during automatic operation, collision handling is performed as specified in **Collision Processing Setting**.

To set up the collision handling, go to **Settings > Safety Setting > Collision Protection**.



Two setting methods for the collision protection are available:

- **Quick setup** (automatic setting according to a selected protection sensitivity)
- **Customize** (user-defined setting)

⚠DANGER

INCORRECT COLLISION SETTINGS

- Ensure that the collision settings are appropriate for the intended operation and lifecycle based on your risk assessment.
- Wherever possible and required, apply all technical measures to help protect the operator from possible collisions when entering the zone of operation.
- Instruct and inform the operator if technical measures to protect from collisions are not possible.
- Ensure that the operator is aware of the active collision setting.

Failure to follow these instructions will result in death or serious injury.

NOTE: The factory setting is **Relax restriction**.

Collision Handling

You can define the following collision handling options in **Settings > Safety Setting > Collision Protection > Collision Processing Setting**:

- **Program pause**

  The program is stopped without any rebound. Subsequently, the program can be continued by using the **Resume program** function. For further information, refer to Function Settings, page 148.

- **Program terminated and rebounded**

  The program is terminated and a rebound is performed. You can set the rebound angle from 0° to 3°.

Bouncing Procedure

The bouncing procedure depends on the motion type of the Lexium Cobot Arm before the collision. The process is presented in the following figure.

Blue arrows: Original trajectory

Red dots: Bouncing end point

Red lines: Bouncing trajectory

Green dots: Collision point

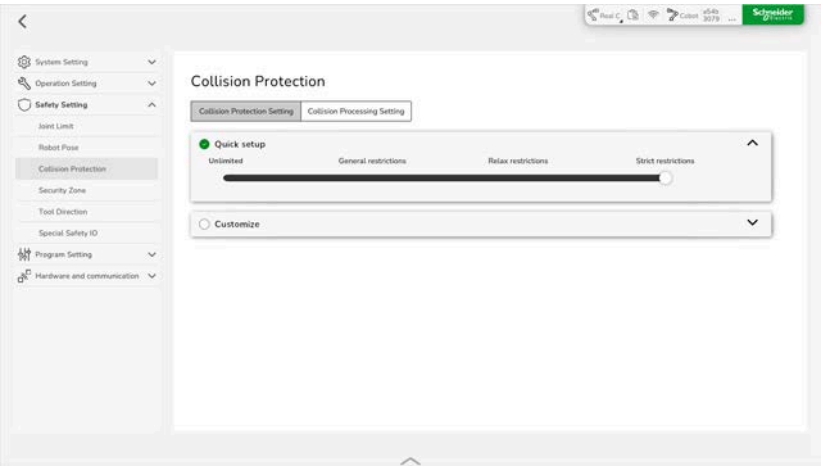
Example:

If the rebound angle value is 3, the Lexium Cobot Arm operates at the moment of a collision as follows:

- If the Lexium Cobot Arm is moving in Cartesian space (linear motion), it bounces back 3 cm (1.18 in) along the original trajectory (a, b).
- If the robot is moving in the joint space (joint motion), the joint with maximum speed at the moment of collision bounces back 3°, and the other joints bounce back correspondingly, so that the Lexium Cobot Arm moves back along the original trajectory.

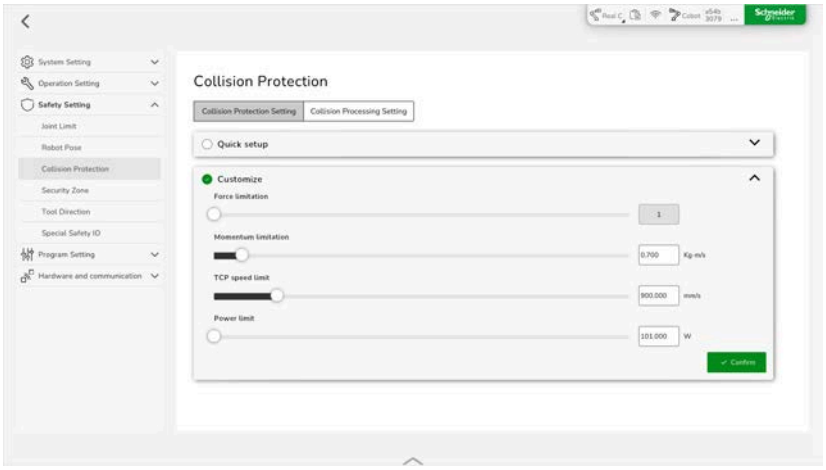
Setting Up Collision Protection via Quick Setup

To set up the collision protection with predefined settings, perform the following steps:

Step	Action
1	In **Settings** > **Safety Setting** > **Collision Protection**, select **Collision Protection Setting**.
2	Select **Quick setup**.
3	Move the slider in the **Quick setup** bar to the appropriate protection sensitivity. Four predefined settings are available • **Unlimited** • **General restrictions** • **Relax restrictions** • **Strict restrictions**  **Result:** The collision protection is set. To display the corresponding values for the **Quick setup**, expand **Customize** using the arrow icon.
4	Display the corresponding values for each **Quick setup** level by expanding **Customize** using the arrow icon on the right and verify if the parameters are applicable for your application. Change the settings as necessary. **Result:** The collision protection is set.

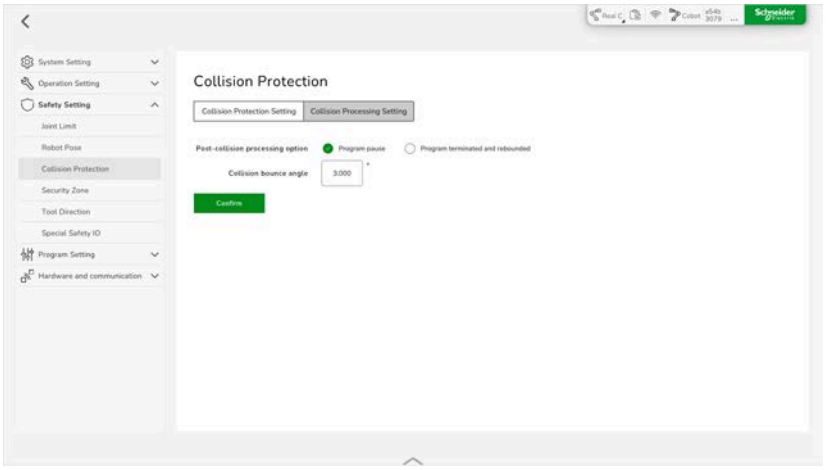
Using Customized Settings

To set up the collision protection with customized settings, perform the following steps:

Step	Action
1	In **Settings** > **Safety Setting** > **Collision Protection**, select **Collision Protection Setting**.
2	Select **Customize**. 
3	Enter the collision protection data either by moving the corresponding slider, or by clicking on the value and typing in the new value. **NOTE:** • **Force limitation** means the collision force limit level from 1...5 (high to low), while level 0 means unlimited. • The lower the **Momentum Limitation**, the **TCP speed limit**, and **Power limit** values are, the slower the Lexium Cobot Arm moves. Set the limit value according to your risk assessment. **Result:** The collision protection is set.

Setting the Collision Processing

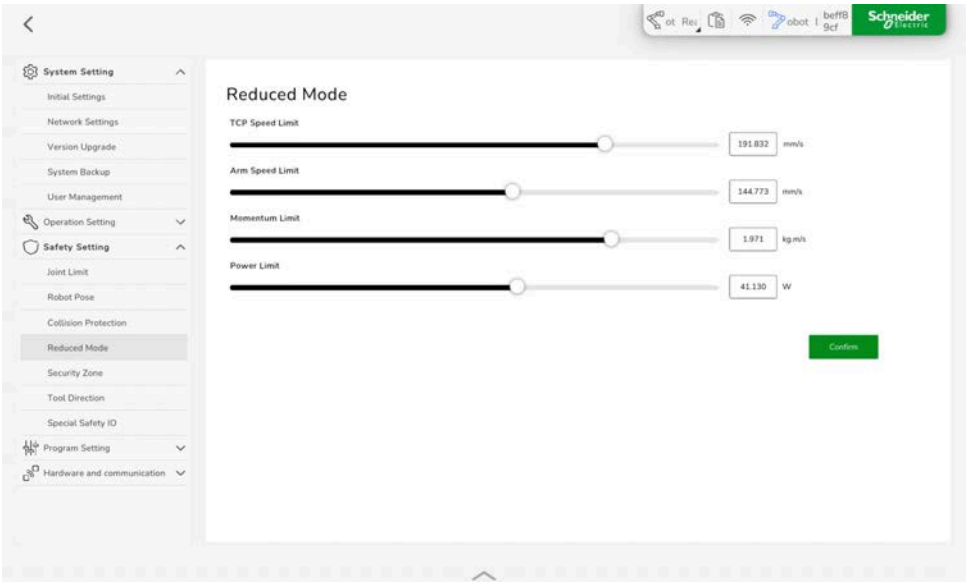
To set up the collision processing, perform the following steps:

Step	Action
1	In **Settings** > **Safety Setting** > **Collision Protection**, select **Collision Processing Setting**. 
2	Select the **Post-collision processing option** and type in the **Collision bounce angle**. **Result:** The collision processing is set.

Reduced Mode

Overview

To change the **Reduced Mode** values for **TCP Speed Limit**, **Arm Speed Limit**, **Momentum Limit**, and **Power Limit**, go to **Settings > Safety Setting > Reduced Mode**.



Setting Up Reduced Mode Values

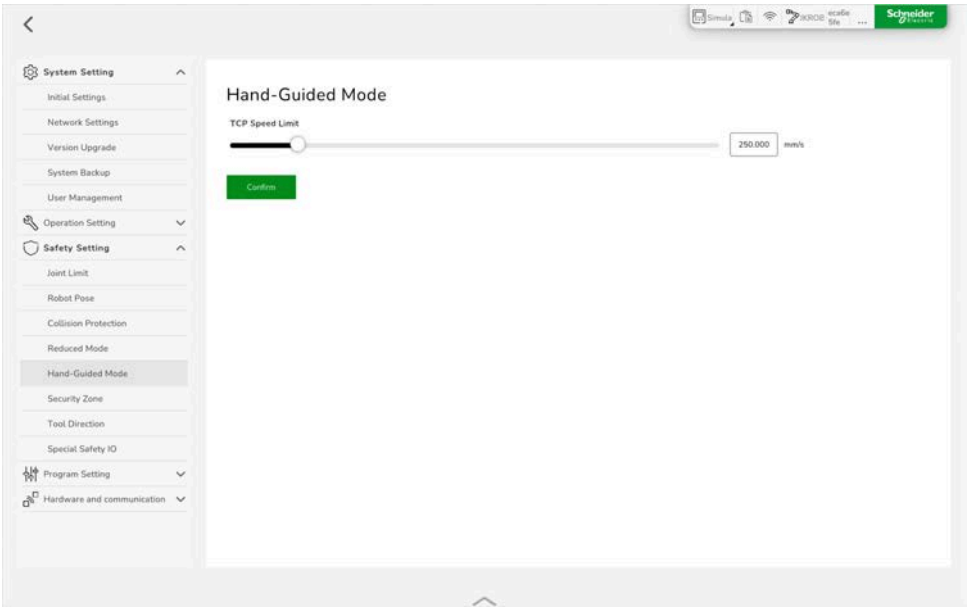
To set up the **Reduced Mode** values, perform the following steps:

Step	Action
1	Go to **Settings** > **Safety Setting** > **Reduced Mode**.
2	Enter the **Reduced Mode** values either by moving the corresponding slider, or by clicking on the value and typing in the new value. **NOTE:** The lower the **TCP Speed Limit**, **Arm Speed Limit**, **Momentum Limit**, and **Power Limit** values are, the slower the Lexium Cobot Arm moves. Set the limit value according to your risk assessment. **Result:** The **Reduced Mode** values are set.

Hand-Guided Mode

Overview

To change the TCP speed limit in hand-guided mode, go to **Settings > Safety Setting > Hand-Guided Mode**.



Setting Up the TCP Speed Limit for Hand-Guided Mode

To set up the TCP speed limit for hand-guided mode, perform the following steps:

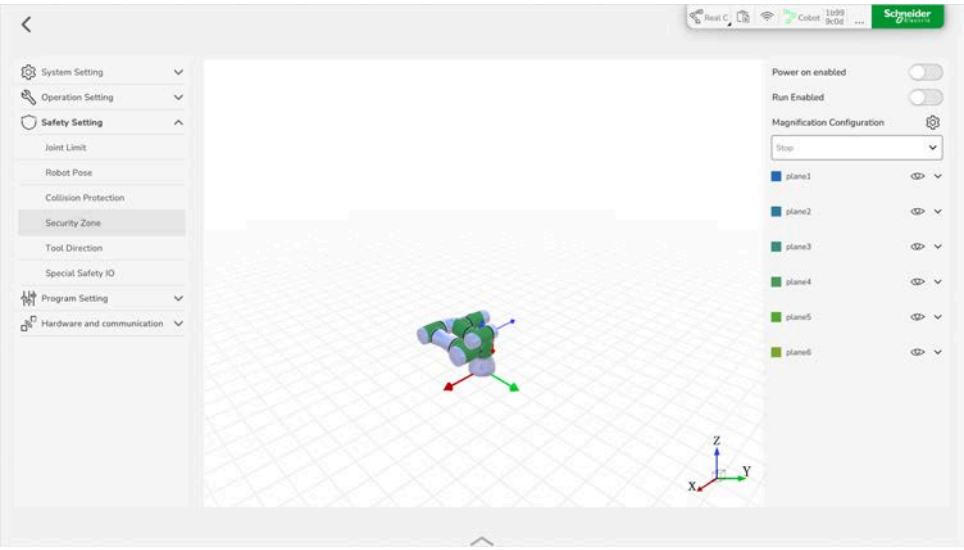
Step	Action
1	Go to **Settings > Safety Setting > Hand-Guided Mode**.
2	Enter the **TCP Speed Limit** value either by moving the corresponding slider, or by clicking on the value and typing in the new value. **NOTE:** The lower the **TCP Speed Limit** is, the slower the Lexium Cobot Arm moves. Set the limit value according to your risk assessment. **Result:** The TCP speed limit for hand-guided mode is set.

Security Zone

Overview

To help prevent the Lexium Cobot Arm from colliding with another object during movement, you can define up to six planes. Exceeding one or more of the planes results in a configured response mode: **Stop**, **Protective Stop** or **Reduced Mode**. For further information, refer to *Functional Safety* in the *Lexium Cobot Arm Hardware Guide*.

To set the planes, go to **Settings > Safety Setting > Security Zone**.



Activation of Planes

You have two options to activate the planes:

- **Power on enabled**

  The planes are active as soon as the Lexium Cobot Arm is powered on and enabled. The Lexium Cobot Arm will operate according to the configuration in response mode as soon as it crosses the defined plane into the restricted area.

- **Run Enabled**

  The planes are activated as soon as the Lexium Cobot Arm executes a program.

  NOTE: The planes are not active during hand-guiding and **Manual Operation**, if **Run Enabled** is configured.

Response Mode

When one or more planes are exceeded, the Lexium Cobot Arm operates according to the configured response:

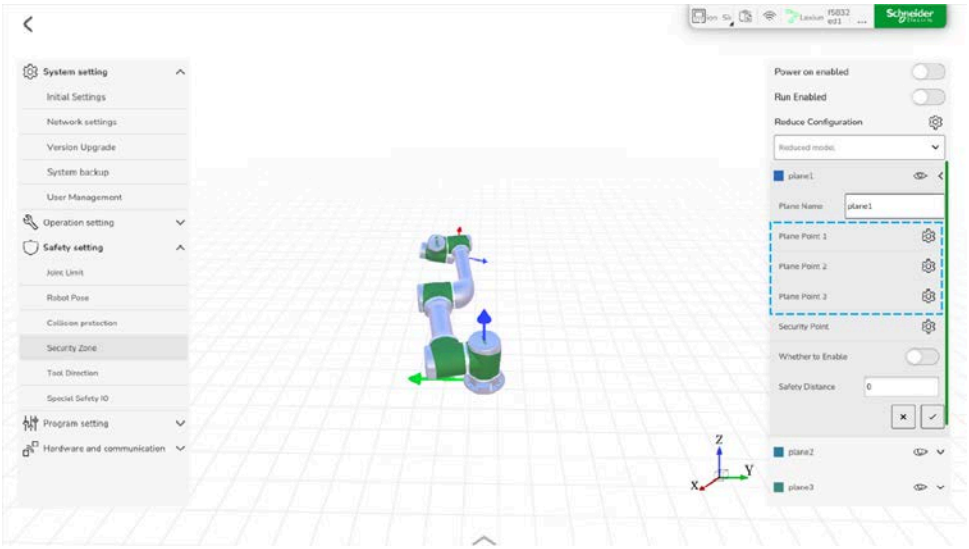
Response mode	Response action
Stop	• The program is stopped. • The Lexium Cobot system stops the Lexium Cobot Arm motion and disables the Lexium Cobot Arm.
Protective Stop	• The Lexium Cobot Arm decelerates to zero and reports an error message. • The program is paused. • The program is resumed after you confirm the status (the Lexium Cobot Arm continues to move outside of the defined zone of operation).
Reduced Mode	• The Lexium Cobot Arm enters reduced mode and reports a message. For further information about the reduced mode, refer to *Functional Safety* in the *Lexium Cobot Hardware Guide*. To configure the reduced mode, refer to Reduced Mode, page 113. • After the TCP is moved back to the zone of operation, the Lexium Cobot Arm movement exits the reduced mode.

For further information, refer to chapter *Position Monitoring Safety Functions* in the *Lexium Cobot Hardware Guide*.

Plane Points

The plane points 1 to 3 are set to determine the planes.

NOTE: The three points must not be on a straight line.



Security Point

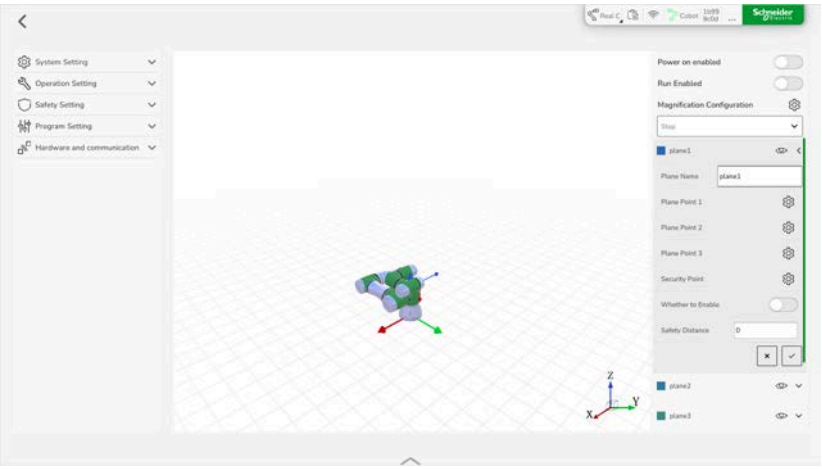
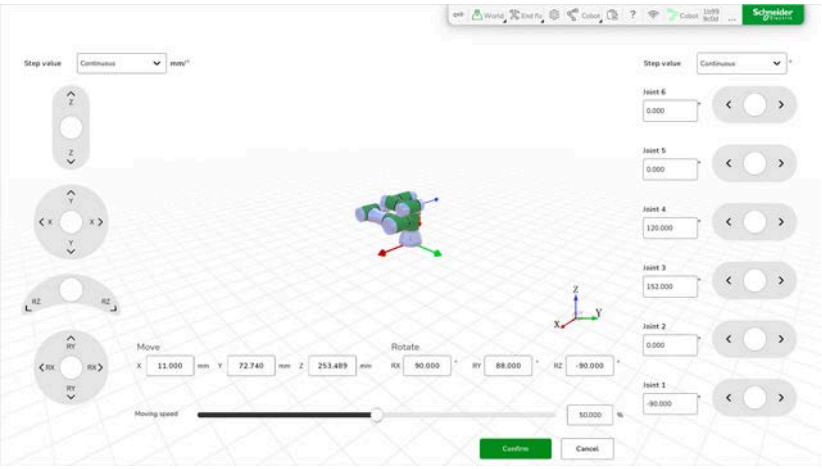
The security point defines which side of the space is allowed for the Lexium Cobot Arm and as such defines the zone of operation. This can be any point in space (except the plane itself).

Safety Distance

Safety Distance is the distance between the TCP of the Lexium Cobot Arm and the plane (unit: mm). When the safety distance is less than or equal to the user-specific value, the security zone is triggered, and the Lexium Cobot Arm reacts according to the configured response mode.

Setting a Plane

To set a plane, perform the following steps:

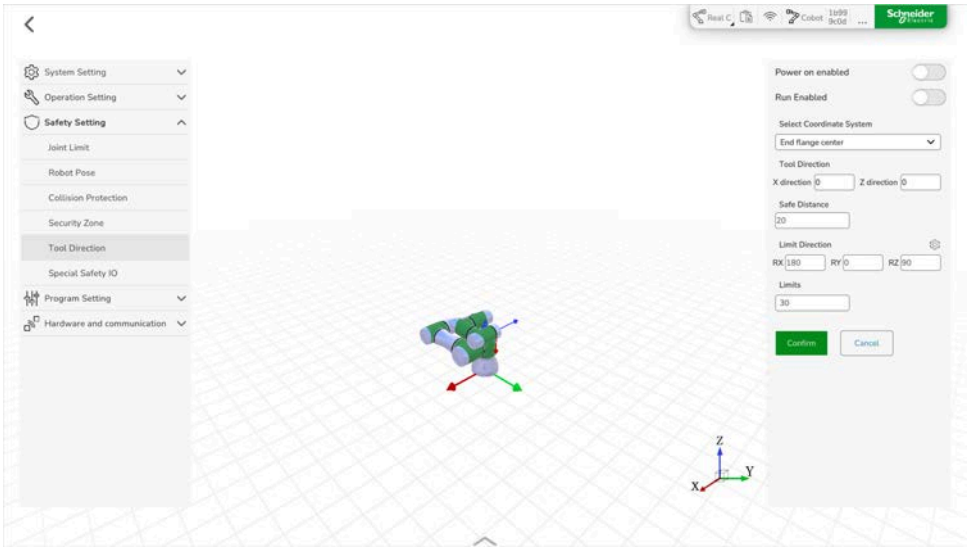
Step	Action
1	Power on and enable the Lexium Cobot Arm.
2	Expand the plane to be edited using the arrow on the right: **Result:** The editing menu is displayed. 
3	In **Plane Name**, edit the name of the plane.
4	Click the **Settings** icon for **Plane Point 1**: **Result:** The **Manual Operation** interface is displayed. 
5	Move the Lexium Cobot Arm to one point of the plane and click **Confirm**.
6	Repeat steps 4 and 5 for **Plane Point 2** and **Plane Point 3**.
7	Click the **Settings** icon for the **Security Point** and set the allowed side of the plane.
8	To enable the plane, activate **Whether to Enable**.
9	Optionally, in the **Safety Distance** field, determine a safety distance in mm. For details, refer to Safety Distance, page 116.
10	To confirm these settings, click ✓. **Result:** The Lexium Cobot Controller calculates the spatial position of the plane based on the three defined plane points. The plane is displayed in the digital representation on the left side. To cancel these changes, click **x**.

Tool Direction

Overview

In **Tool Direction**, you can restrict the angle at which the tool is pointing. The limit is defined by a cone that has a fixed orientation with respect to the base frame of the Lexium Cobot Arm. As the Lexium Cobot Arm moves in the working space, the tool direction is constrained to remain within the defined cone. The direction of the tool can be set to coincide with the Z-axis of the flange coordinate system or the tool coordinate system at the end of the Lexium Cobot Arm.

To set the tool direction, go to **Settings > Safety Setting > Tool Direction**.



Activation of Tool Direction

You have two options to activate the tool direction limit:

- **Power on enabled** (activation on start-up)
- **Run Enabled** (activation on operation)

Activation on start-up means that the tool direction limit is active as soon as the Lexium Cobot Arm is powered on and enabled.

Activation on operation means that the tool direction limit is activated as soon as the Lexium Cobot Arm executes the program or moves to the point by MoveJ/ MoveL.

> **NOTE:** The tool direction limit is not active in the operating modes hand-guiding and **Manual Operation**, if activation on operation is configured.

Safety Distance

Safety Distance represents the angle between the tool orientation and the cone boundary (unit: degree). When the safety distance is less than or equal to the user-specific value, a **Warning** message is displayed.

Limit Direction

There are two methods to set the limit direction:

- Typing in the parameters for the limit direction manually, or
- Setting the parameters for the limit direction by teaching it

If you enter the parameters manually, the cone centerline can be defined by the three angles RX, RY, and RZ.

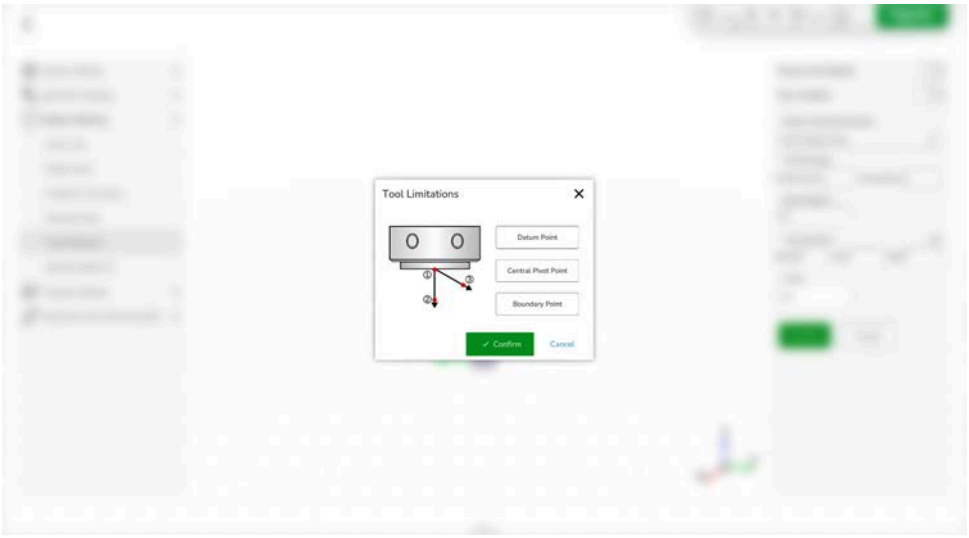
If you set the limit direction by teaching, the following three points are required:

- **Datum Point**

  This is the cone apex.
- **Central Pivot Point**

  The line between the **Central Pivot Point** and the **Datum Point** is the centerline of the cone.
- **Boundary Point**

  The line between the **Boundary Point** and the **Datum Point** defines the cone the tool must not exceed.



Setting the Tool Direction

Step	Action
1	From **Settings** > **Safety Setting** > **Tool Direction** > **Select coordinate system**, select the coordinate system to define the tool orientation. It corresponds either to the Z axis of the flange end (end flange center) or to a configured tool coordinate system (**Settings** > **Operation Setting** > **TCP Settings**).
2	Optionally, adjust the tool orientation setting the two angles in **Tool Direction**: • **X direction**: The angle of the limited tool orientation around the X axis of the reference coordinate system. • **Z direction**: The angle of the limited tool orientation around the Z axis of the reference coordinate system.
3	In **Safety Distance**, optionally determine a distance in degrees. For details, refer to Safety Distance, page 119.
4	In **Limit Direction**, set the cone centerline direction either manually by typing in the values for the three angles RX, RY, and RZ or by clicking **Settings** to set the parameters by teaching. For further information, refer to Limit Direction, page 119.
5	Click **Confirm**. **Result:** The tool orientation limit is configured.

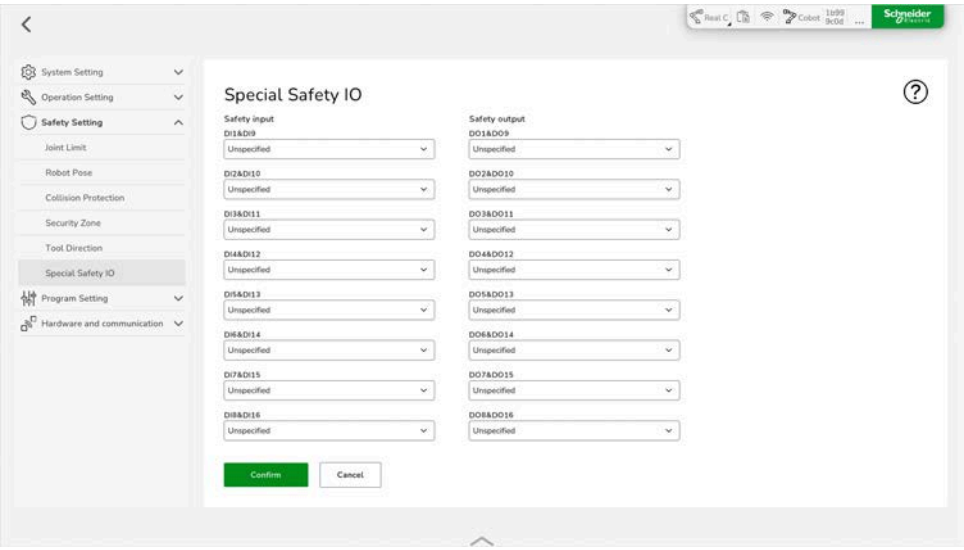
Special Safety IO

Overview

The digital inputs and outputs of the Lexium Cobot Controllers can be configured as a **Special Safety IO** to control the safety-related functions and monitor the safety-related status of the Lexium Cobot. This **Special Safety IO** signal is a two-channel signal.

NOTE: Due to the limited number of digital I/O connections available for the Lexium Cobot Compact Controller, the number of safety-related inputs and outputs used simultaneously is restricted to two.

To set special safety-related inputs and outputs, go to **Settings > Safety Setting > Special Safety IO**.



Setting the Special Safety IO

To set a **Special Safety IO**, perform the following steps:

Step	Action
1	Disable and power off the Lexium Cobot Arm.
2	Go to **Settings > Safety Setting > Special Safety IO**.
3	At the safety-related input or output to be set, use the dropdown menu to select the required configuration. For further information on the different configurations, refer to Description of the Safety-Related Signals, page 121.
4	Click **Confirm**. **Result:** The **Special Safety IO** is set.

Description of the Safety-Related Signals

The following table presents the available safety-related signal configurations that you can select. For further information, refer to *Functional Safety* in the *Lexium Cobot Hardware Guide*.

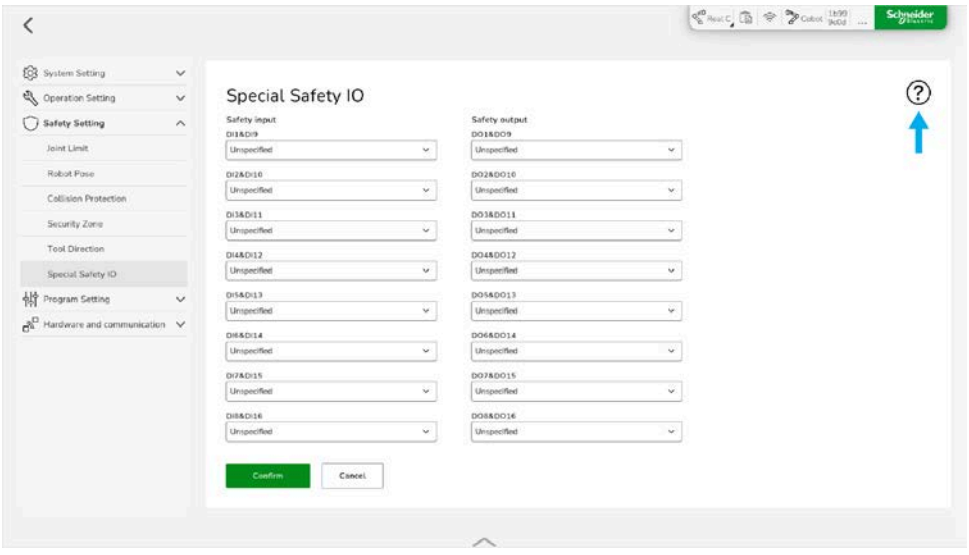
Name	Action logic	Direction
Additional Emergency Stop Input	When the input signal is FALSE, the Lexium Cobot Arm performs an emergency stop.	Input
Additional Protective Stop Input	When the input signal is FALSE, the Lexium Cobot Arm performs a protective stop.	Input

Name	Action logic	Direction
Protective Stop Resetting Input	The protective stop is reset by the rising edge (FALSE > TRUE) of this signal.	Input
Reduced Mode Input	When the input signal is FALSE, the Lexium Cobot Arm switches to reduced mode.	Input
Three Position Enable Input	When the input signal is TRUE, the Lexium Cobot Arm can be moved in manual mode.	Input
Emergency Stop Button State Output	When the emergency stop pushbutton on the Control Stick is pressed, the output signal is FALSE.	Output
System Emergency Stop State Output	When the Lexium Cobot system is in emergency stop state, the output signal is FALSE.	Output
System Protective Stop State Output	When the Lexium Cobot system is in protective stop state, the output signal is FALSE.	Output
Robot Motion State Output	When the Lexium Cobot Arm is in motion, the output signal is FALSE.	Output
Robot Not-Stopping State Output	When an emergency stop or protective stop is triggered, causing the Lexium Cobot Arm to stop or to decelerate to a full stop, the output signal is TRUE.	Output
Robot Reduced Mode Output	When the Lexium Cobot Arm is in reduced mode, the output signal is FALSE.	Output
Robot Not in Reduced Mode	When the Lexium Cobot Arm is not in reduced mode, the output signal is FALSE.	Output

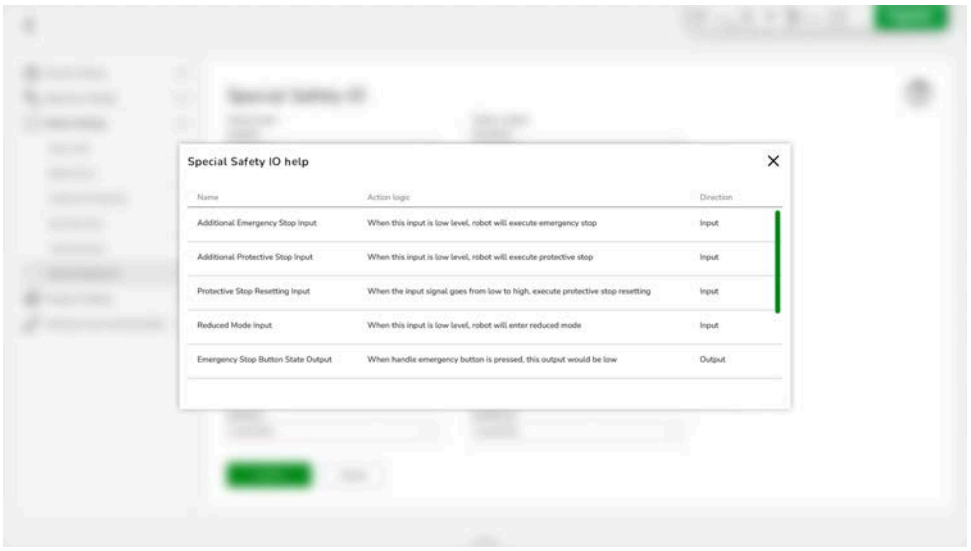
For further information on the safety-related signals, refer to *Functional Safety* in the *Lexium Cobot Hardware Guide*.

Viewing the Special Safety IO Help

To display a description of the safety-related signals in the software, click the **Help** icon.



Result: The **Special Safety IO help** is displayed.

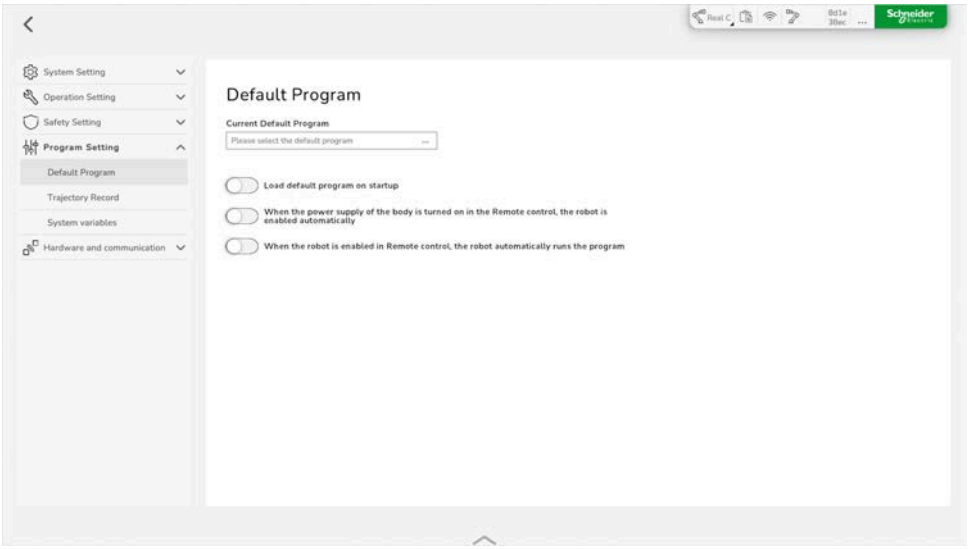


Program Setting

Default Program

Overview

To designate a program as default program, go to **Settings > Program Setting > Default Program**.



The Lexium Cobot can automatically load the default program when the Lexium Cobot Arm is started up, if you activate the toggle **Load default program on startup**.

This way, the program to be executed through EcoStruxure Cobot Expert at start-up is already selected.

Once the Lexium Cobot Arm is enabled, you can run the default robot program by pressing the **Start** button on the Control Stick.

The following options are available:

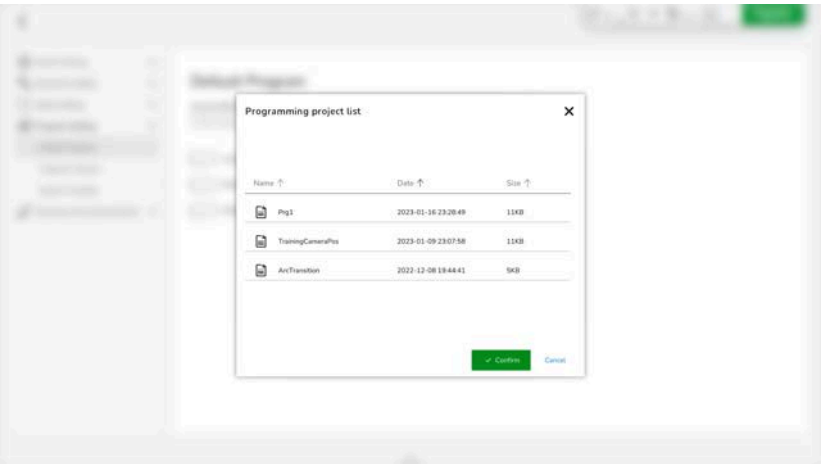
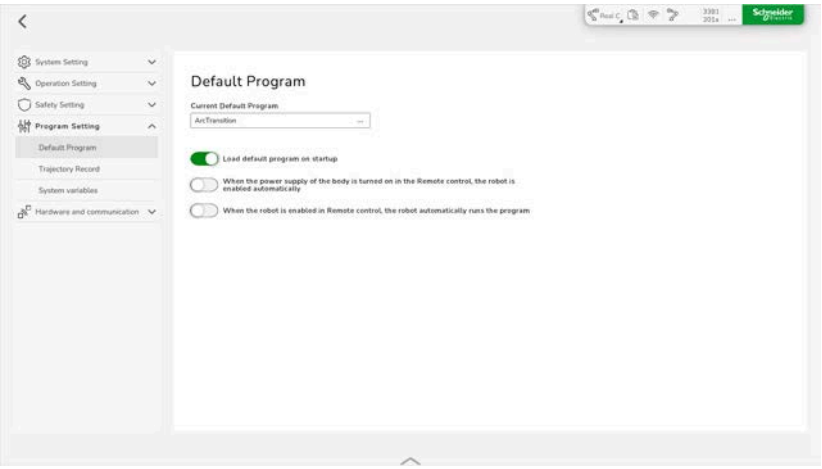
NOTE: These options are only available when the control source is set to **Remote Control**, page 50.

- When powered on, the Lexium Cobot Arm is also enabled automatically.
- When the Lexium Cobot Arm is enabled, the default program is started.

These options can be combined.

Designating a Default Program

To designate a default program, perform the following steps:

Step	Action
1	In **Settings** > **Program Setting** > **Default Program**, click the **Current default program** box. **Result:** The **Programming project list** is displayed: 
2	Select a program and click **Confirm**.
3	Optionally, select the following options by activating the respective toggles: • **Load default program on startup** **NOTE:** The default program is only loaded at startup if you activate the option **Load default program on startup**. • **When the power supply of the body is turned on, the robot is enabled automatically** • **When the robot is enabled, the robot automatically runs the program** **NOTE:** Automatic program execution is only available when the remote source is in control. **Result:** The default program is set with the selected options. 

⚠WARNING

AUTOMATIC START OF MOVEMENT

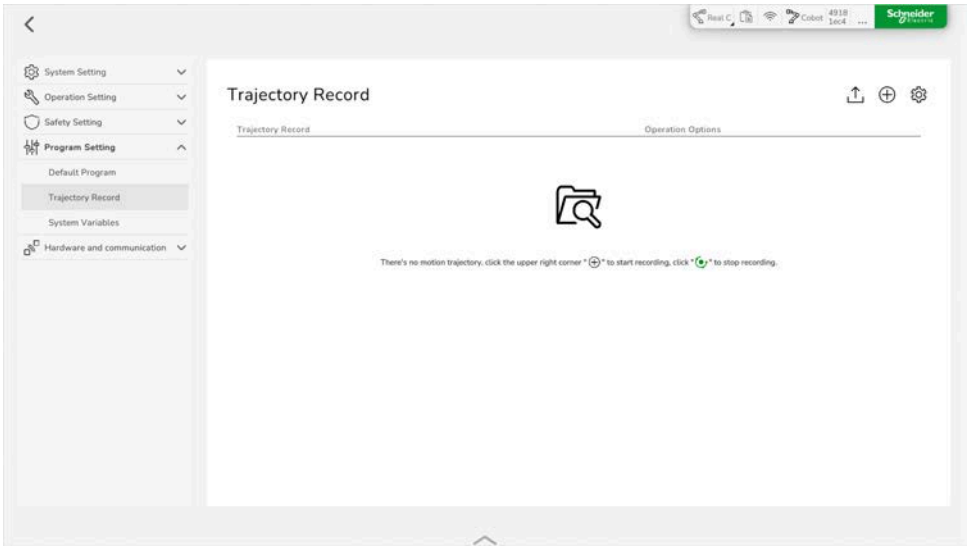
- Ensure that the operator is aware that selecting the options **When the power supply of the body is turned on, the robot is enabled automatically** and **When the robot is enabled, the robot automatically runs the program** may cause the Lexium Cobot Arm to start the movement after it is powered on.
- Ensure that the operator is aware that selecting the option **When the robot is enabled, the robot automatically runs the program** may cause the Lexium Cobot Arm to start the movement after it is enabled.
- Ensure that the operator or a person near the Lexium Cobot Arm cannot be endangered or trapped by powering on or enabling the Lexium Cobot Arm with the options described in the preceding statements.
- Ensure that a risk assessment is conducted and respected according to EN/ISO 12100 during the design of your machine.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Trajectory Record

Overview

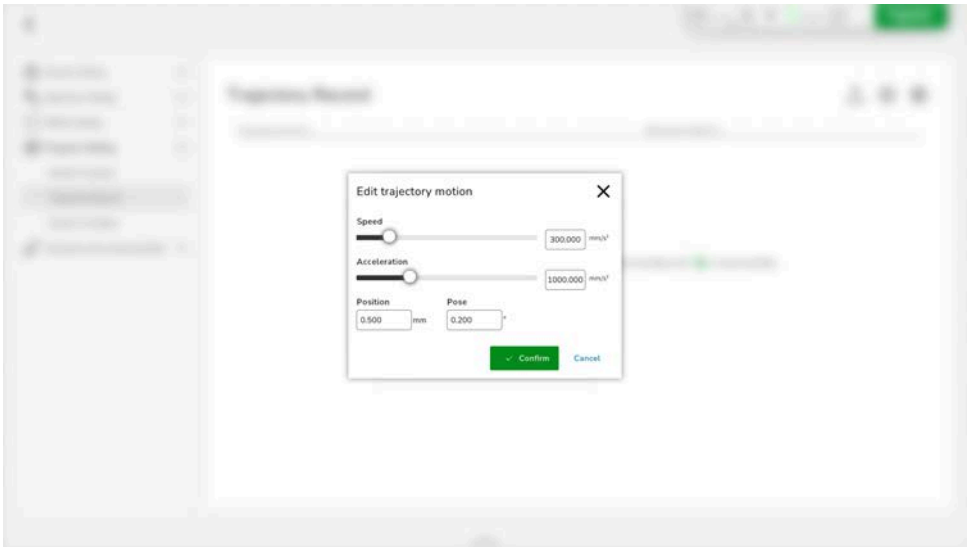
To record the trajectory of the Lexium Cobot Arm during hand-guiding and **Manual Operation**, go to **Settings > Program Setting > Trajectory Record**.



This trajectory file can be called during programming by the trajectory recording instruction to reproduce the recorded trajectory in the program.

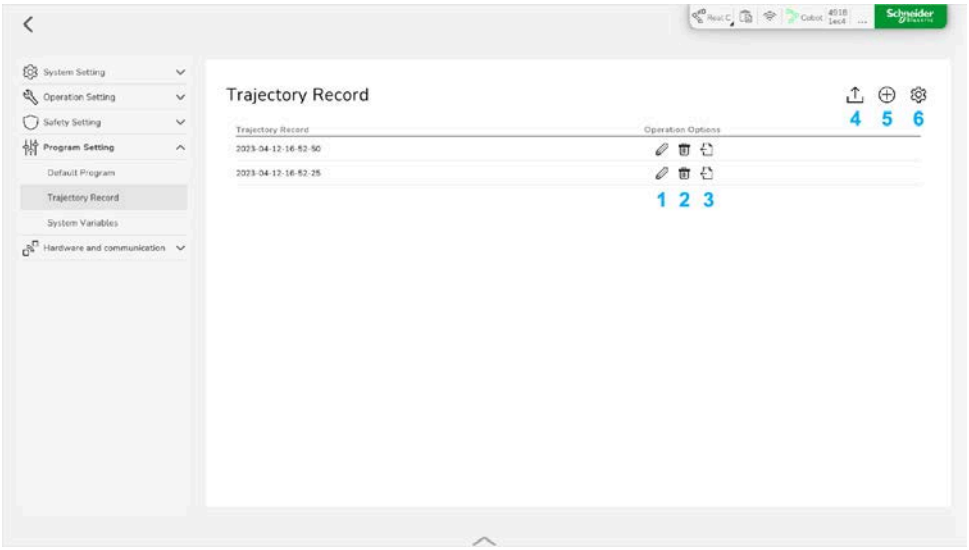
> **NOTE:** The trajectory recording function records only the path information and not the motion parameters.

In the settings, you can set the default speed and acceleration for this trajectory in the program. These parameters have no influence on the recording itself and can be adjusted later for specific program requirements. You can also set the sampling accuracy of position and pose.



Operation Options

You have the following operation options in the **Trajectory Record** pane:



1 Edit the name of the trajectory record

2 Delete the trajectory record

3 Export the trajectory record

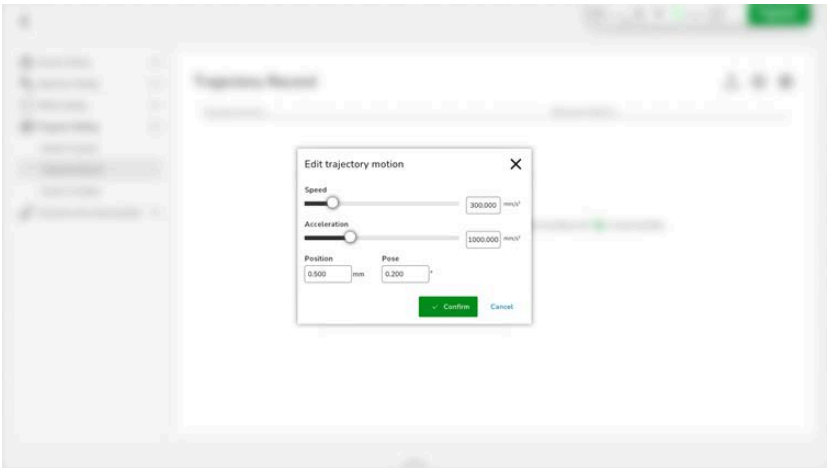
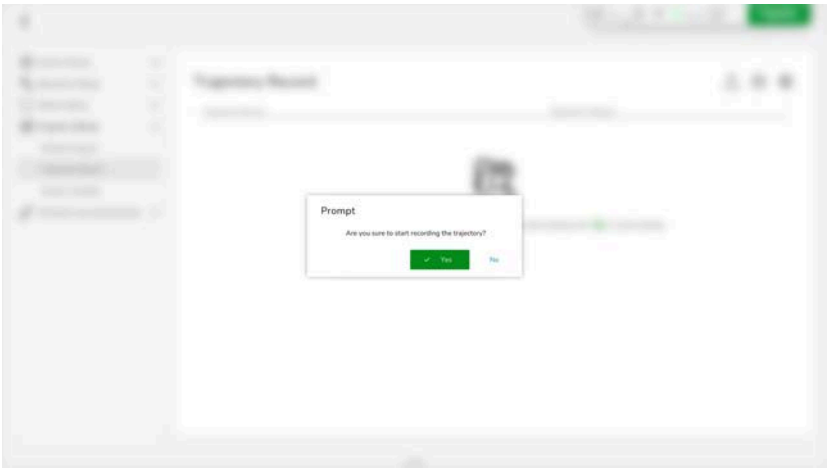
4 Import a trajectory record

5 Add a new recording

6 Settings for configuring the trajectory recording parameters

Recording the Trajectory

To record a trajectory, perform the following steps:

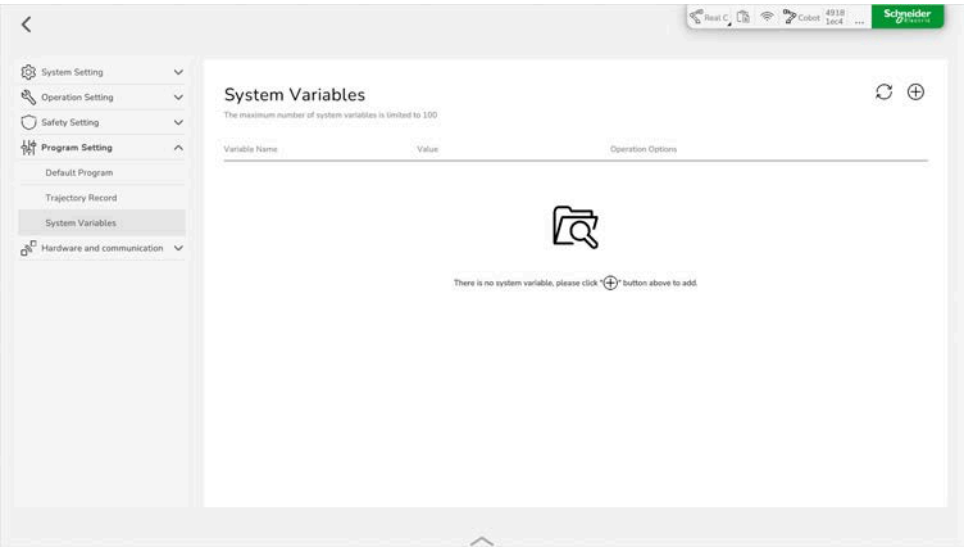
Step	Action
1	In **Settings** > **Program Setting** > **Trajectory Record**, click the **Settings** icon. **Result:** The dialog box **Edit trajectory motion** is displayed. 
2	Adjust the parameters according to your needs and click **Confirm**. **NOTE:** If the motion distance of the trajectory is short, the sampling deviation of **Position** and **Pose** may be as much as 0.1 mm or degree respectively.
3	Click the **Add** icon. **Result:** The following confirmation prompt is displayed. 

Step	Action
4	Click **Yes** to start a new record of the trajectory. **Result:** The recording starts. The record is added to the list and the **Trajectory recording** button is displayed in the top menu. 
5	Move the Lexium Cobot Arm by hand-guiding or **Manual Operation** to record the motion trajectory. **NOTE:** If no movement of the Lexium Cobot Arm has been performed, no recording is saved.
6	Click the **Trajectory recording** button in the top menu to stop the recording. **Result:** The trajectory is saved with the timestamp as a name.
7	Click the **Edit** icon to give a meaningful name to the record. **Result:** The trajectory record is ready to be used in a program.

System Variable

Overview

To add system variables, which can be called and modified in programs, go to **Settings > Program Setting > System Variables**



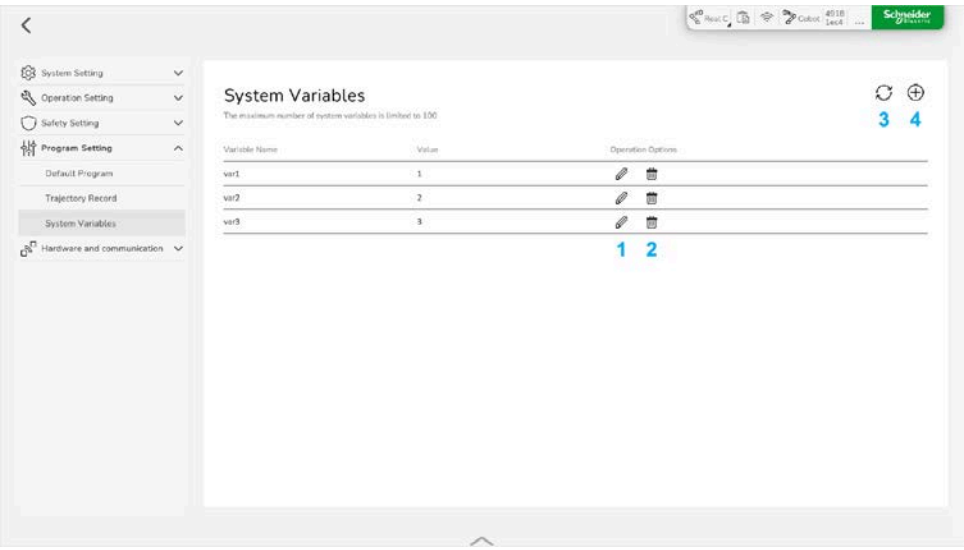
Another option to add variables is provided in the **Programming Control** interface. For further information, refer to the help in the instructions menu of the **Programming Control** interface, see Types of Instructions, page 191.

The variables are of numeric type only and are stored in the Lexium Cobot Controllers. The values of the variables are not modified or reset by starting and stopping the program or powering on and off the Lexium Cobot system.

NOTE: Up to 100 variables can be stored.

Operation Options

You have the following operation options in the **System Variables** pane:



1 Edit the name and the initial value of the variable

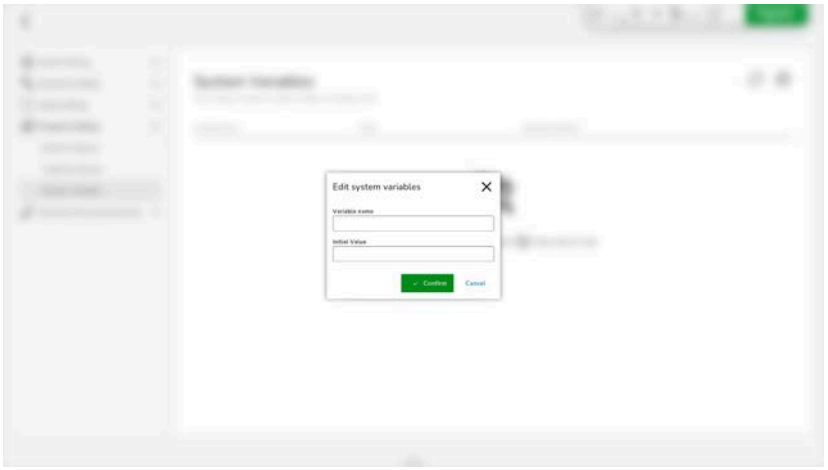
2 Delete the variable

3 Refresh the variable list

4 Add a new variable

Creating a System Variable

To add a new variable, perform the following steps:

Step	Action
1	In **Settings** > **Program Setting** > **System Variables**, click the **Add** icon on the right. **Result:** The **Edit system variables** dialog box is displayed. 
2	Type in a variable name and an initial value.
3	Click **Confirm**. **Result:** The variable is added to the **System Variable** list.

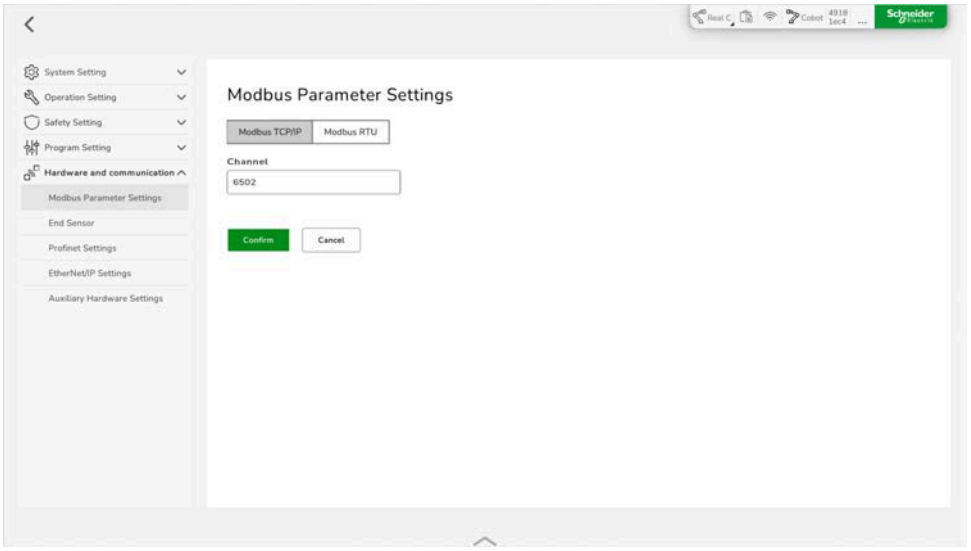
Hardware and Communication

Modbus Parameter Setting

Overview

The Lexium Cobot supports Modbus TCP/IP and Modbus RTU communication modes.

To set the parameters for a connection with the Modbus server, go to **Settings > Hardware and communication > Modbus Parameter Settings**.



After the connection is established on client side, you can read the Lexium Cobot state and control the Lexium Cobot I/O signal based on the register address and the function code program in the Modbus Address Table, page 267.

To edit the Modbus information, click **I/O Panel > Modbus** in the feature bar.

NOTE: Disable the Lexium Cobot Arm for setting the Modbus configuration.

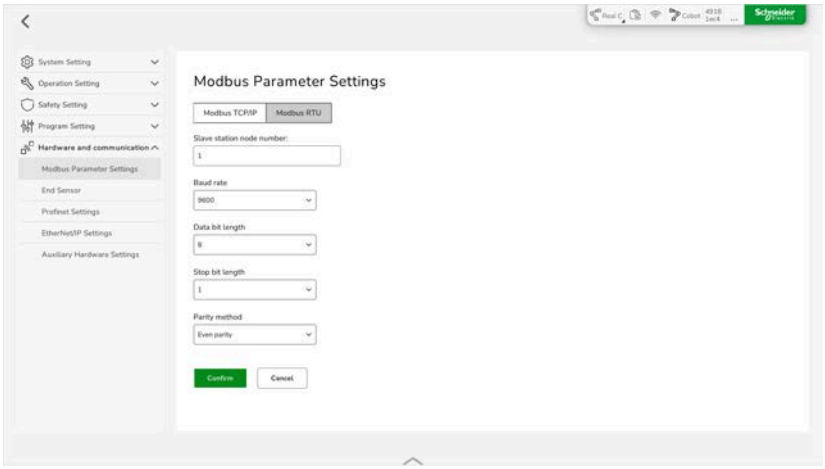
Setting the Modbus TCP/IP Mode

To set the Modbus TCP/IP mode, perform the following steps:

Step	Action
1	In **Settings** > **Hardware and communication** > **Modbus Parameter Settings**, select **Modbus TCP/IP**.
2	In **Channel**, edit the channel number according to your Modbus settings.
3	Click **Confirm**. **Result:** The new Modbus port is set.
4	Restart the Lexium Cobot Controller.

Setting the Modbus RTU Mode

To set the Modbus RTU mode, perform the following steps:

Step	Action
1	In **Settings** > **Hardware and communication** > **Modbus Parameter Settings**, select **Modbus RTU**. 
2	Set the parameters for the Modbus server. **NOTE:** PLC-side setting interface must be configured as it is for the Lexium Cobot. The default parameters are: • **Slave station node number**: – • **Baud rate**: 4800 • **Data bit length**: 8 bits • **Stop bit length**: 1 bit • **Parity method**: Even parity
3	Click **Confirm**. **Result:** The new Modbus configuration is set.
4	Restart the Lexium Cobot Controller.

End Sensor

Overview

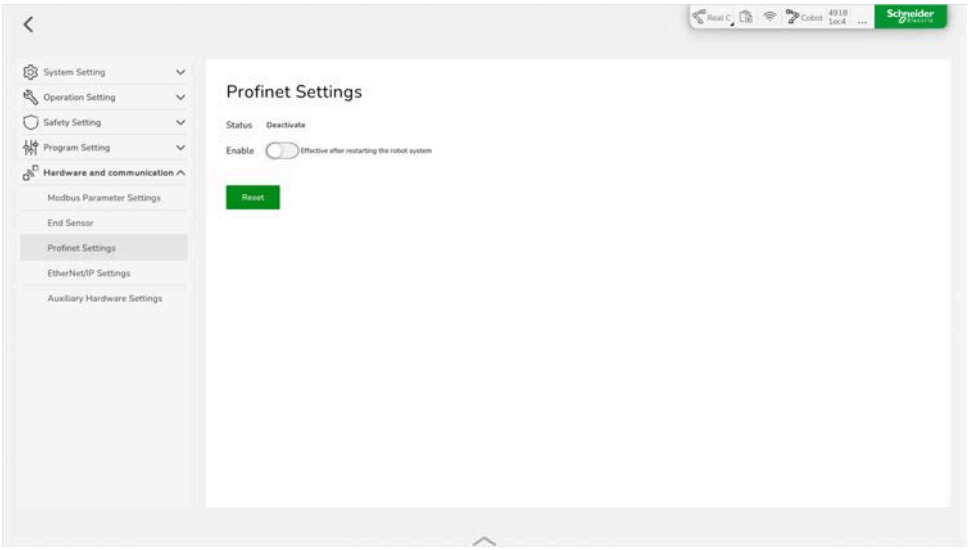
This feature is not supported.

Profinet Settings

Overview

Lexium Cobot supports the Profinet communication protocol and can be used as a Profinet server to connect to external devices.

To enable or disable Profinet, go to **Settings > Hardware and communication > Profinet Settings**.



The Profinet function can only interact with the external controller, with the Profinet I/O information being displayed in the **I/O Panel**.

For further information, refer to:

- Profinet, page 165
- Profinet Address Table, page 273

Enabling Profinet

To enable the Profinet communication protocol, perform the following steps:

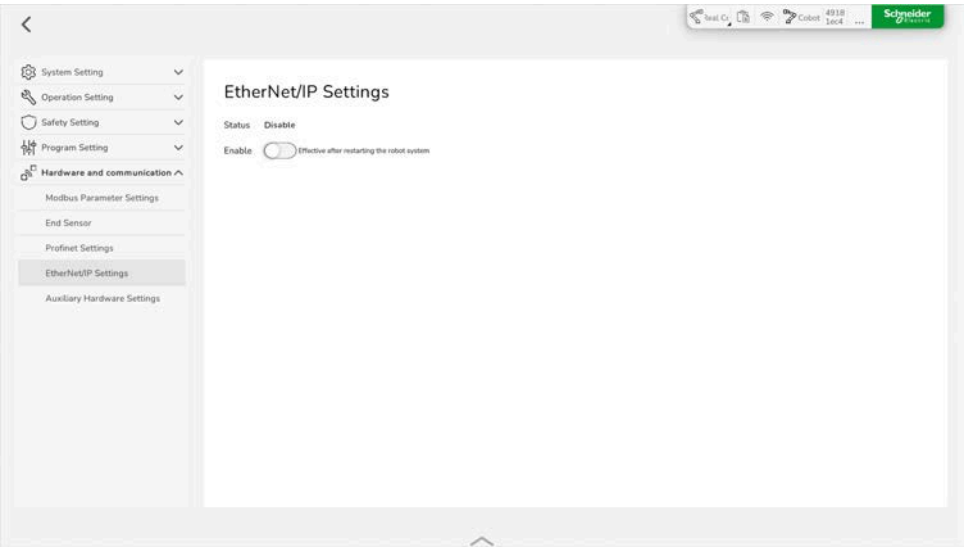
Step	Action
1	Disable the Lexium Cobot Arm.
2	In **Settings > Hardware and communication > Profinet Settings**, click **Enable**.
3	Restart the Lexium Cobot Controller by using the Control Stick. **Result:** The Profinet communication protocol is enabled on the Lexium Cobot Controller.

EtherNet/IP Settings

Overview

Lexium Cobot supports the EtherNet/IP communication protocol and can be used as an Ethernet/IP server for connection with external devices.

To set the EtherNet/IP configuration, go to **Settings > Hardware and communication > EtherNet/IP Settings**.



The EtherNet/IP function can be enabled or disabled (default) and can only interact with the external controller for Ethernet/IP communication when enabled. The EtherNet/IP I/O information is displayed in the IO interface.

For further information refer to:

- Ethernet/IP, page 170
- Ethernet/IP Address Table, page 277

Enabling EtherNet/IP

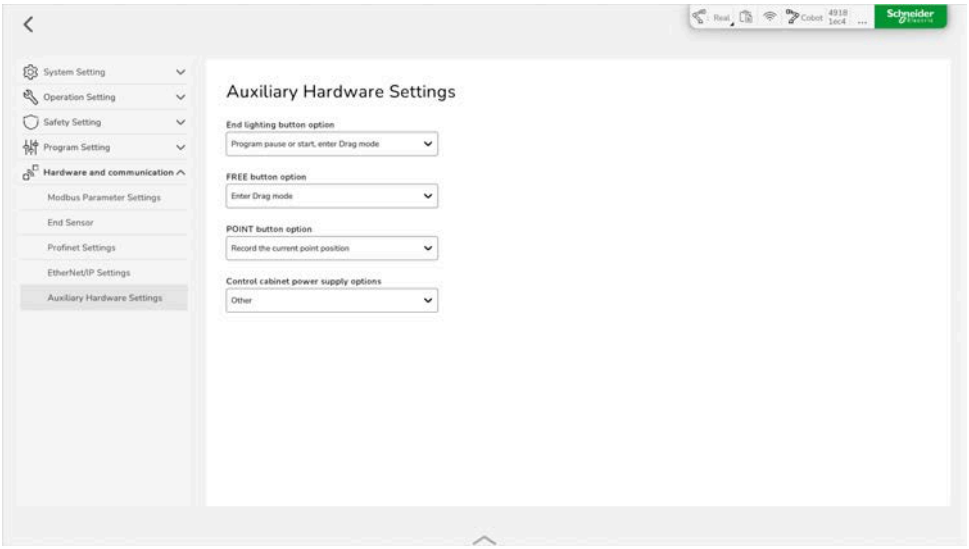
To enable the EtherNet/IP communication protocol, perform the following steps:

Step	Action
1	Disable the Lexium Cobot Arm.
2	In **Settings > Hardware and communication > EtherNet/IP Settings**, click **Enable**.
3	Restart the Lexium Cobot Controller by using the Control Stick. **Result:** The EtherNet/IP communication protocol is enabled on the Lexium Cobot Controller.

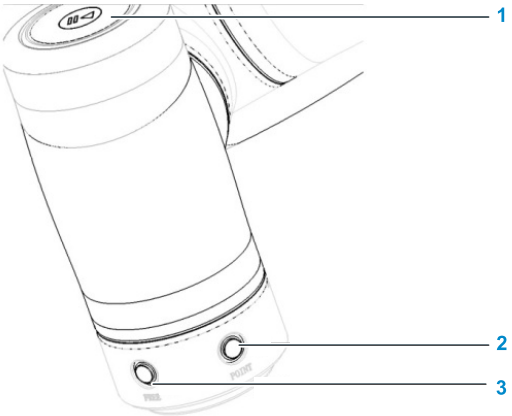
Auxiliary Hardware Settings

Overview

To configure the functions of the three buttons at the end of the Lexium Cobot Arm (**Play/pause** button, **FREE** button, **POINT** button) and the mains voltage of the Lexium Cobot Controller power supply, go to **Settings > Hardware and communication > Auxiliary Hardware Settings**.



The following graphic shows where the three buttons are located.



1 Play/pause button

2 POINT button

3 FREE button

For further information on the buttons, refer to *Lexium Cobot Arm Tool Flange Details* in the *Lexium Cobot Hardware Guide*.

Configuration Options

To set a function for one of the buttons or the Lexium Cobot Controller power supply, select the function from the dropdown list.

The following options are available.

End lighting button option for the **play/pause** button:

- **Prohibited**
- **Program pause or start**

- **Enter Drag mode** (hand-guided mode)
- **Program pause or start, enter Drag mode** (hand-guided mode)

FREE button option:

- **Prohibited**
- **Enter Drag mode** (hand-guided mode)

POINT button option:

- **Prohibited**
- **Record the current point position**

Control cabinet power supply option (only available for the Lexium Cobot Cabinet Controller):

- **Other** (represents 110 Vac)
- **220VAC** (represents 220 Vac)

Terminal IO

Overview

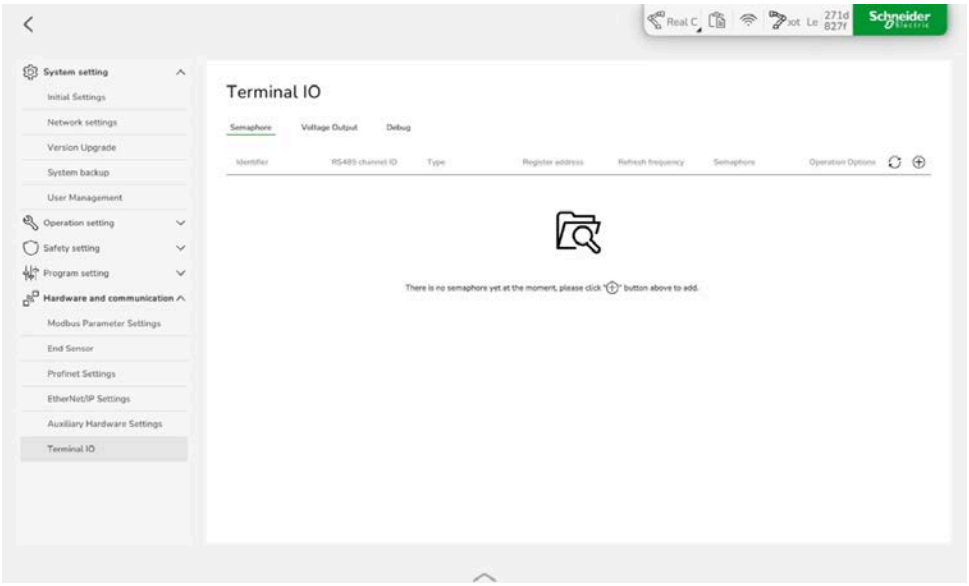
The Lexium Cobot Arm is equipped with a tool flange IO interface (TIO) that provides the following inputs and outputs:

- Two digital inputs (TDI)
- Two digital outputs (TDO)
- Two analog inputs (TAI)

The two digital outputs can be multiplexed as high-speed RS485 channels and the analog inputs can be multiplexed as low-speed RS485 channels.

The configurable voltage output (12V/24V/0V) supports the power supply of the external expansion devices.

To set the configuration of the TIO, go to **Settings > Hardware and Communication > Terminal IO**.

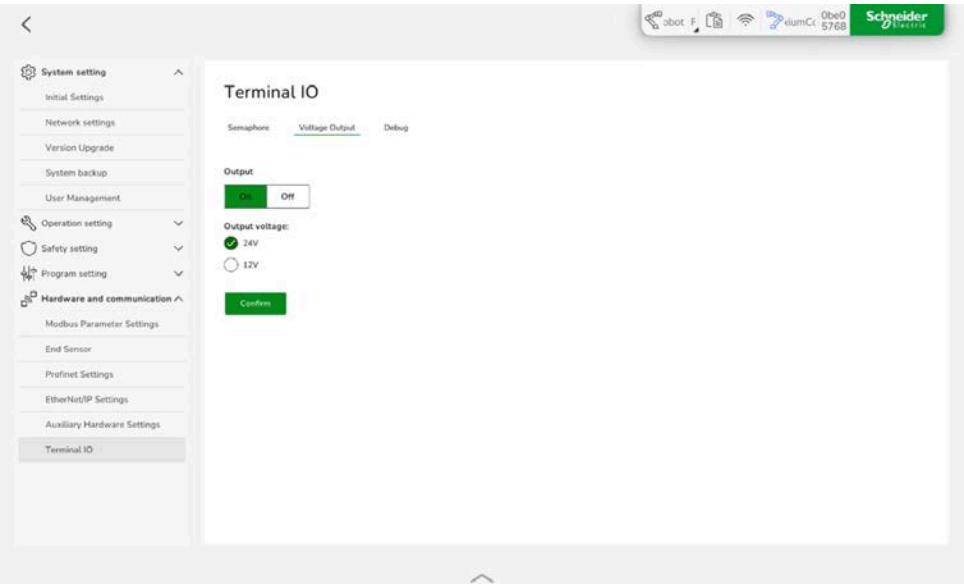


NOTE:

- The **Terminal IO** section is only displayed if the Lexium Cobot Arm is connected.
- When using the Lexium Cobot Arm for the first time, you must power on the Lexium Cobot Arm to activate the section.
- After restarting the Lexium Cobot Controller, it may be necessary to power on the Lexium Cobot Arm to display this section.

Voltage Output

In the **Voltage Output** tab, you can switch the output voltage on or off and select **12V** or **24V** as output.



NOTE: Select **12V** or **24V** to use the tool flange inputs and outputs.

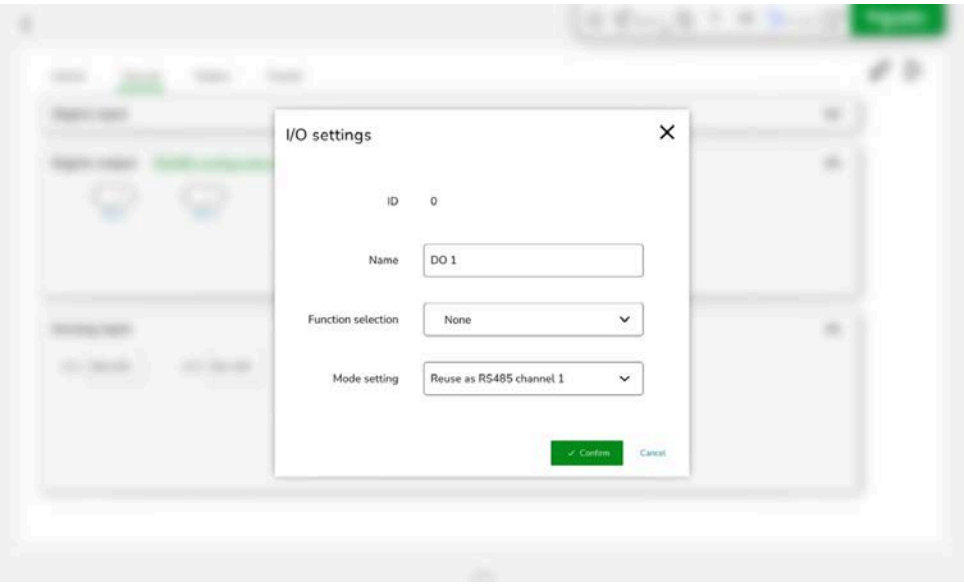
Configuring the Voltage Output

To set the voltage output at the **Terminal IO**, perform the following steps:

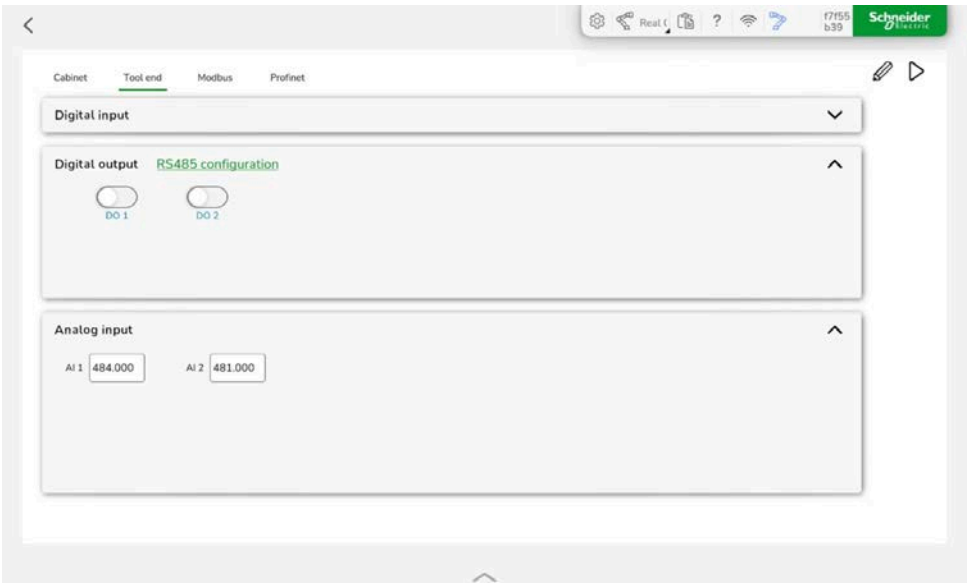
Step	Action
1	Click **Settings** > **Hardware and Communication** > **Terminal IO**.
2	Select the **Voltage Output** tab.
3	Select whether to enable or to disable the output.
4	If enabled, select the output voltage.
5	Click **Confirm**. **Result:** The voltage output is configured.

RS485 Configuration

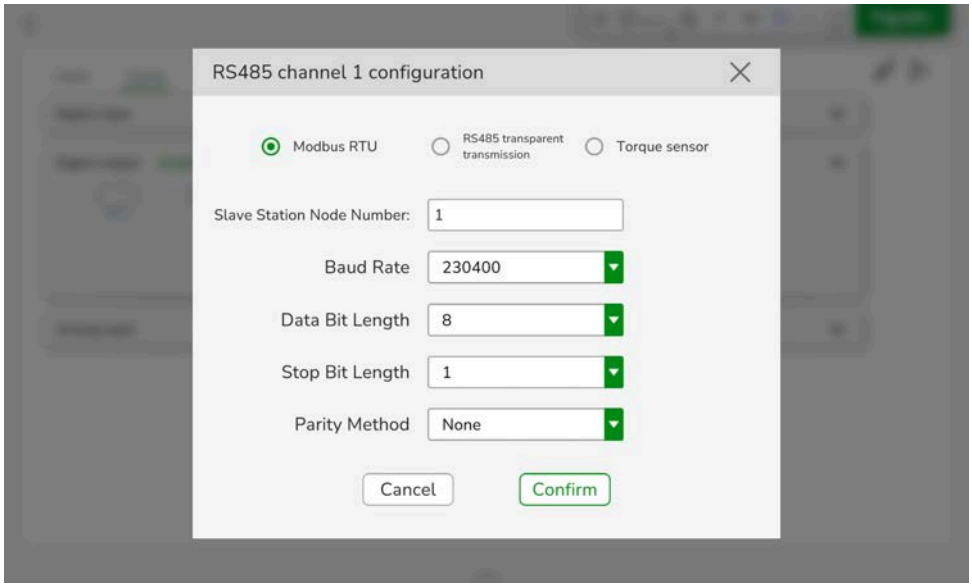
The two-way RS485 channel can be multiplexed as RS485 channel in the corresponding pin before configuration. Using RS485 channel 1 as an example, the TDO pin is multiplexed as RS485 channel 1, as presented in the following figure.



After setting, the **RS485 configuration** option is displayed.



Click **RS485 configuration** to display the RS485 channel configuration options. Here you can set the communication parameters.



RS485 Channel Configuration

When using the RS485 channel, the mode needs to be configured. There are three modes available:

- **Modbus RTU**

  To support external devices.
- **RS485 transparent transmission**

  Not supported.
- **Torque sensor**

  To connect the torque sensor of designated model.

RS485 Channel Communication Parameter Configuration

Set the serial communication parameters of the RS485 channel:

- **Baud rate** (maximum 2250000 supported)

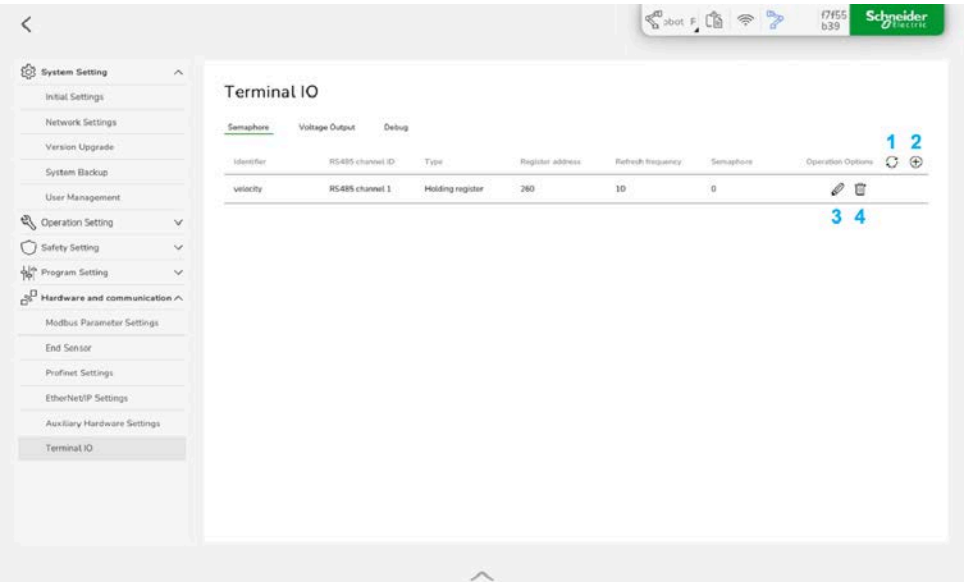
- **Data bit length** (8/9 supported)
- **Stop bit length** (1/2 supported)
- **Parity method** (odd parity/even parity/none)

When the channel mode is set as Modbus RTU, the Modbus **Slave station node number** needs to be additionally configured.

NOTE: A detected error in the configuration of the communication parameters results in the TIO not being able to communicate with the external equipment.

Semaphore

EcoStruxure Cobot Expert provides a semaphore function for user queries.



1 Refresh the semaphore values

2 Add a semaphore

3 Edit a semaphore

4 Delete a semaphore

To add a semaphore, click the **Add** icon.

Define the states to be queried in advance and capture the state value through subsequent refreshing and querying operations.

Semaphore Parameters:

- **Identifier**

  Unique identifier of the semaphore (Unicode and special characters are not supported). Used for subsequent refreshing, acquisition, and deletion operations.

- **RS485 channel ID**

  Used for designating the TIO RS485 channel of the semaphore source (RS485 channel 1 or RS485 channel 2)

- **Semaphore type**

  Data type of the semaphore. This parameter corresponds to the following Modbus function codes:

  - 01 means coil register
  - 02 means discrete input
  - 03 means input register
  - 04 means holding register

  Others are not supported.

- **Register address**

  Refers to the Modbus register address corresponding to the semaphore. This address is used for accessing the register designated by the Modbus RTU server in combination with the RS485 channel configuration and the semaphore type.

  NOTE: The semaphore must be defined in a situation where the related TIO pin has been multiplexed as RS485 channel and in the Modbus RTU mode. Changing the mode or the pin multiplexing will cause the semaphore configuration loss.

Semaphore Refreshing and Querying

Once a semaphore is defined, the monitoring or debugging can be performed through the debugging interface in the **Debug** tab (in **Settings** > **Hardware and Communication** > **Terminal IO**) or directly in the job program. The semaphore can be refreshed and queried through both methods.

The refresh operation can trigger the data interaction between the Lexium Cobot Controllers and the TIO devices during operation. Since the interaction between the Lexium Cobot Controllers and the TIO device is asynchronous with the refresh instruction, it is necessary to wait for a certain time (100 ms) to capture the value after refreshing. In addition, the refreshing frequency can be specified. If the frequency is 0, it is treated as a one-time refresh, and if the frequency is greater than 0, it is combined with the communication bandwidth to fulfill the refresh requirement as much as possible.

To refresh the semaphore value, click the **Refresh** icon in the **Semaphore** tab (in **Settings** > **Hardware and Communication** > **Terminal IO**).

Deleting a Semaphore

To delete a semaphore, perform the following steps:

Step	Action
1	In **Settings** > **Hardware and Communication** > **Terminal IO**, click on the **Semaphore** tab, select the **Delete** icon in the row of the semaphore to be deleted. **Result:** The confirmation prompt is displayed.
2	Click **Confirm**. **Result:** The semaphore is removed.

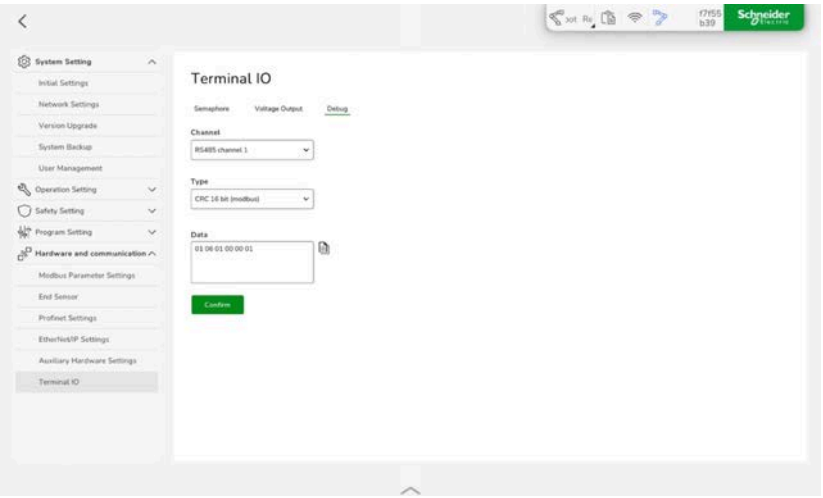
Immediate Instructions

The immediate instruction designates the immediate control command of the Lexium Cobot Controllers via the TIO external device, including the device position control, the speed control and the force control.

Sending an Immediate Instruction

To send an immediate instruction, perform the following steps:

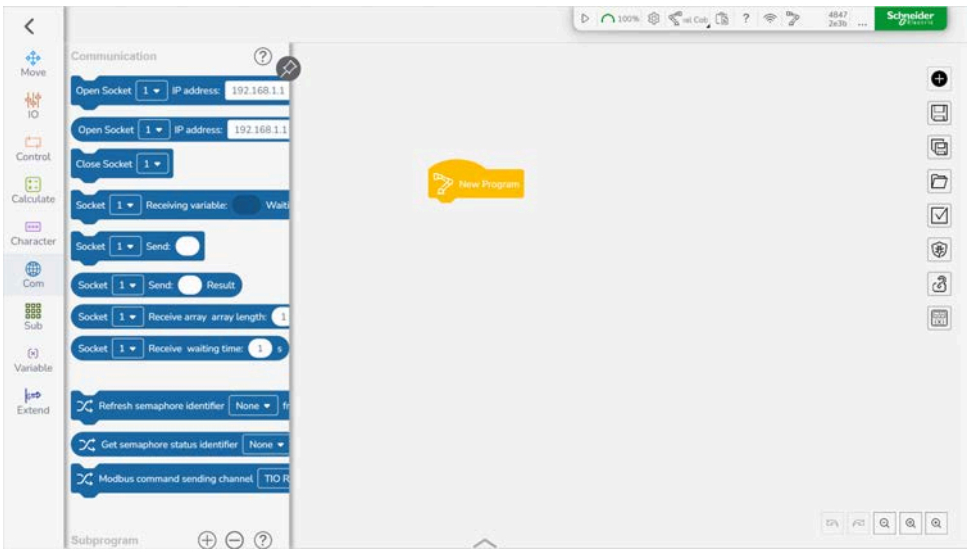
Step	Action
1	In **Settings** > **Hardware and Communication** > **Terminal IO**, click on the **Debug** tab.
2	Select the channel and the type.

Step	Action
3	Type in the hexadecimal data instruction in **Data**. **Example:** 01 06 01 00 00 01 
4	Click **Confirm**.

TIO Support in the Job Program

The job program supports refreshing and querying of the semaphores by appropriate instructions. The definition, modification and deletion of the semaphores can be added manually in the debugging interface. In addition, the instruction is given to send the immediate command to control the TIO device immediately.

The related commands are shown in the following graphic.

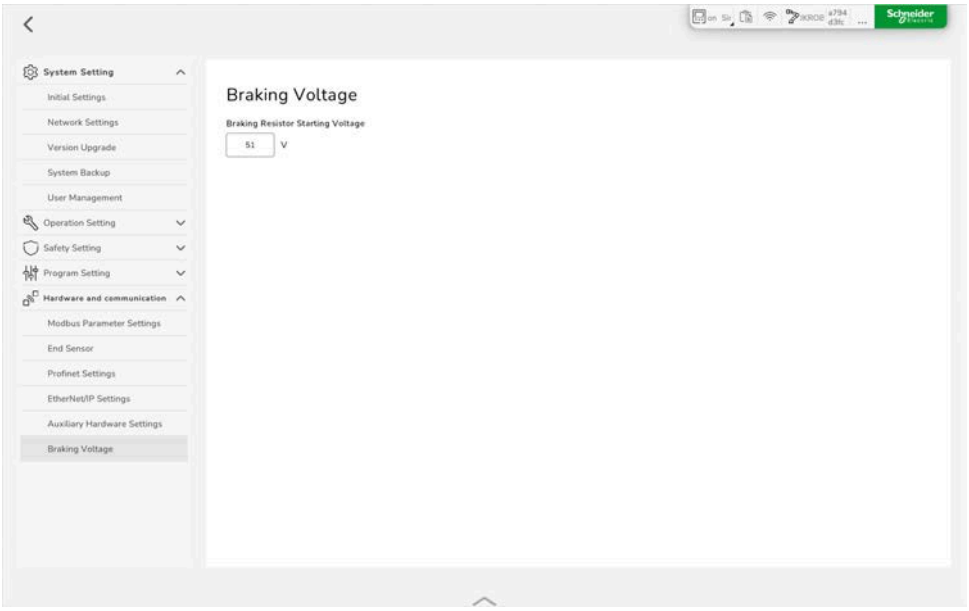


Braking Voltage

Overview

The Lexium Cobot Compact Controller provides a configurable braking resistor starting voltage setting to avoid overvoltage that may occur during deceleration and braking movement of the Lexium Cobot Arm.

To configure this setting, go to **Settings > Hardware and communication > Braking Voltage**.



Configuration Settings

Configuration of the braking resistor starting voltage according to the connected power supply:

Power supply	Input voltage	Braking resistor starting voltage
Modicon ABLU 48 V dc (commercial reference: ABLU3A48200)	48 V dc	51 V
48 V dc lithium battery	54.6 V dc	58 V

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that the braking resistor starting voltage is correctly configured according to the input voltage.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Setting the Braking Voltage

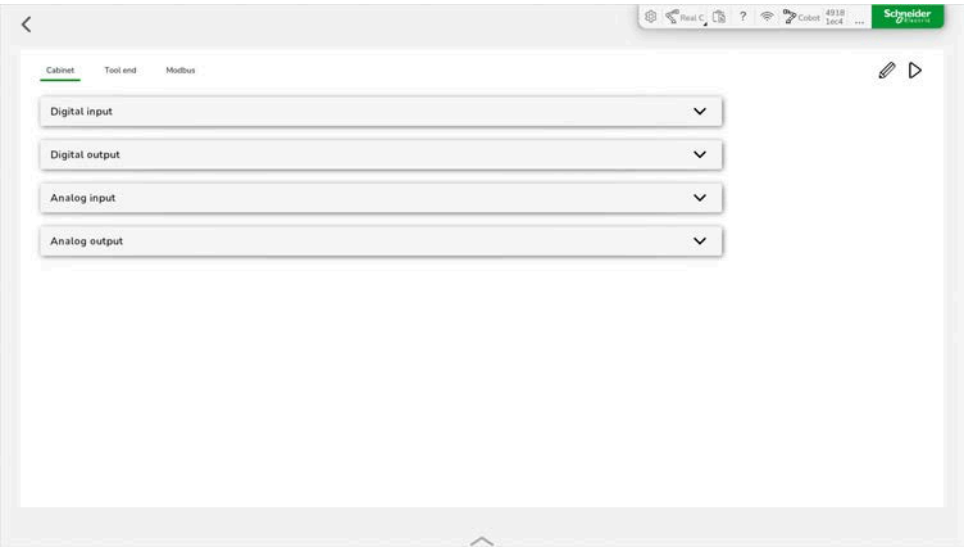
Step	Action
1	Disable and power off the Lexium Cobot Arm.
2	Click **Settings** > **Hardware and communication** > **Braking Voltage**.
3	In **Braking Resistor Starting Voltage**, set the value.

I/O Panel

What's in This Chapter

Overview

To view and set the electrical inputs and outputs of the Lexium Cobot Arm system, select **I/O Panel** in the feature bar.



NOTE: Disable the Lexium Cobot Arm for editing the inputs and outputs.

The **I/O Panel** consists by default of the following sections:

- **Cabinet**
- **Tool end**
- **Modbus**

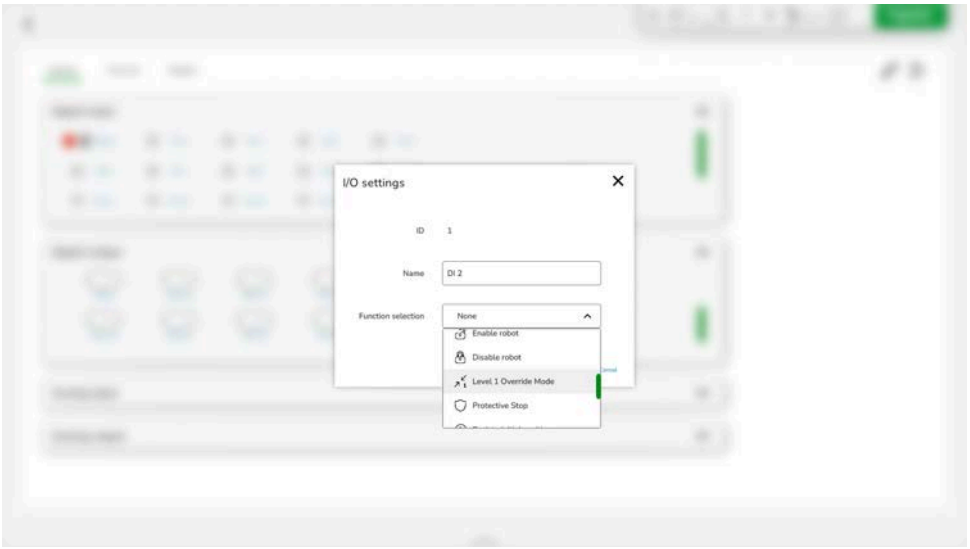
Optionally, the sections **Profinet** and **EtherNet/IP** can be activated by enabling them in the settings. For further information, refer to:

- Profinet Settings, page 136
- EtherNet/IP Settings, page 137

Function Settings

Supported Digital Input Functions

When you edit a digital input (DI), you can set the function of the input in the **I/O settings** dialog box.



In this dialog box, you can select the following functions from the **Function selection** dropdown list.

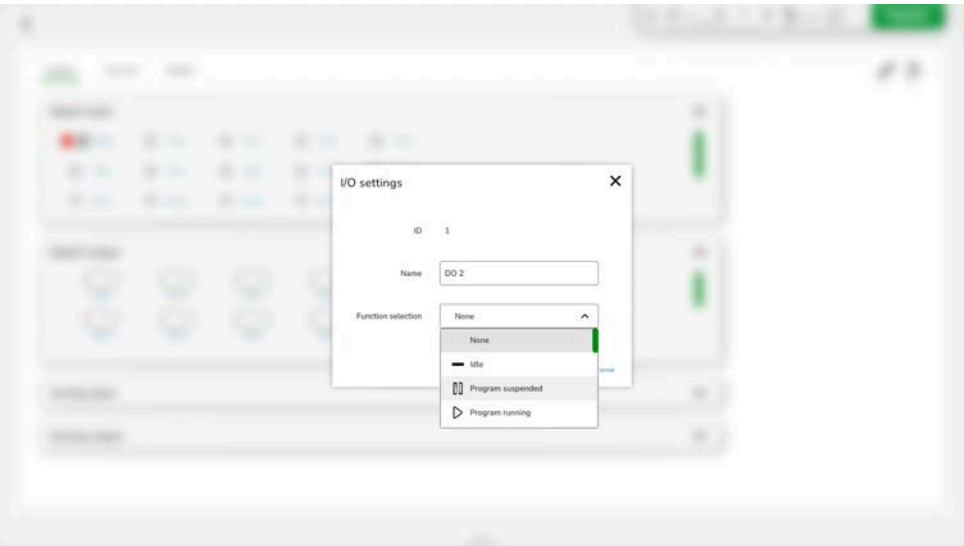
Function name	Triggering mode
None	None
Start program	Rising edge signal (FALSE>TRUE)
Pause program	Rising edge signal (FALSE>TRUE)
Resume program	Rising edge signal (FALSE>TRUE)
Stop program	Rising edge signal (FALSE>TRUE)
Turn on robot power	Rising edge signal (FALSE>TRUE)
Power off the robot	Rising edge signal (FALSE>TRUE)
Enable robot	Rising edge signal (FALSE>TRUE)
Disable robot	Rising edge signal (FALSE>TRUE)
Level 1 Override Mode (2)	FALSE signal
Protective Stop	FALSE signal
Back to initial position (=**Home**)	TRUE signal
Level 2 Override Mode (2)	FALSE signal
Clear fault (1)	Rising edge signal (FALSE>TRUE)
Free-drive mode On (hand guided)	Rising edge signal (FALSE>TRUE)
Free-drive mode Off (not hand guided)	Rising edge signal (FALSE>TRUE)

1 Only the collision message is cleared, the other messages are not cleared.

2 The **Level 2 Override Mode** parameter must be less than the **Level 1 Override Mode** parameter. Set the reduction ratio in **Settings** > **Safety Setting** > **Security Zone** > **Reduce Configuration**. This only affects the speed of the movement.

Supported Digital Output Functions

When you edit a digital output (DO), you can set the function of the output in the **I/O settings** dialog box.



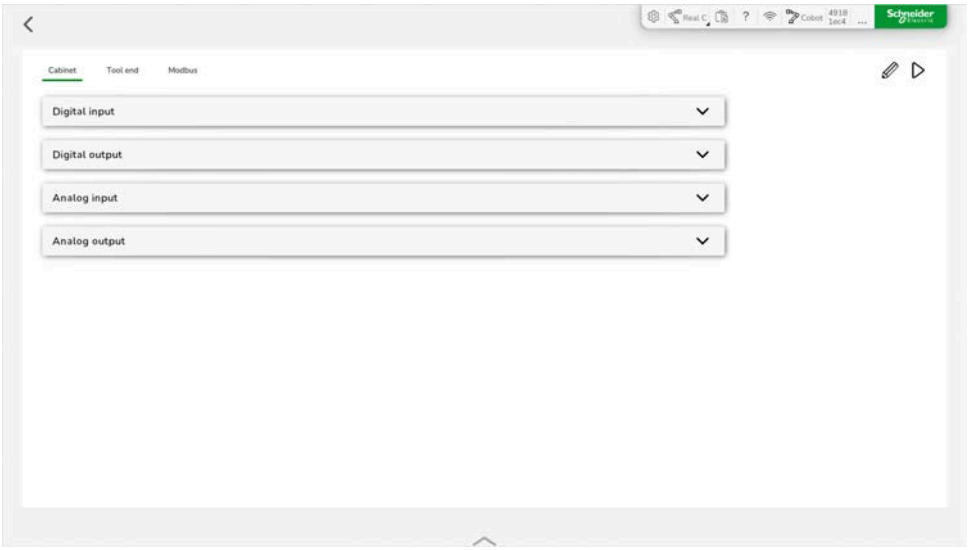
In this dialog box, you can select the following functions from the **Function selection** dropdown list.

State	Description
None	No function is assigned.
Idle	The program is stopped completely or there is no loaded program (no running or paused program).
Program suspended	The program is paused.
Program running	The program is being executed (not stopped nor paused).
Error	The Lexium Cobot detects an error. For example, position deviation.
Powered on	The Lexium Cobot Arm is powered on but not enabled.
Enabled	The Lexium Cobot Arm is powered on and enabled.
Moving	Triggered when the Lexium Cobot Arm is in motion (program operation, manual operation, secondary development control movement, and so on).
Static	Triggered when the Lexium Cobot Arm is not moving.
Started up	The Lexium Cobot Controller is powered on and started up. Independent of the state of the Lexium Cobot Arm.
Emergency Stop Status	When the system is in emergency stop state, the output signal is TRUE.
Level 1 Override Status	The Lexium Cobot is in **Level 1 Override Mode**. In this case, the output is TRUE.
Level 2 Override Status	The Lexium Cobot is in **Level 2 Override Mode**. In this case, the output is TRUE.
Safety Stop Status	Indicates whether the system is in a protective stop. When the protective stop is triggered, the output is TRUE.
Security Position	Triggered when the Lexium Cobot Arm is in the **Home Pose**. For further information, refer to Robot Pose, page 107.
Drag-and-drop Status	Indicates the status of the hand-guided mode. When the hand-guided mode is active, the output is TRUE.
Collision Status	Indicates a detected collision. When a collision is detected, the output is TRUE.

Cabinet Tab

Overview

To set the inputs and outputs of the Lexium Cobot Controllers, go to **I/O Panel > Cabinet**.



- The Lexium Cobot Cabinet Controller is equipped with 16 digital inputs, 16 digital outputs, and two analog interfaces which can be used as inputs or outputs.
- The Lexium Cobot Compact Controller is equipped with 5 digital interfaces which can be used as inputs or outputs.

When EcoStruxure Cobot Expert is connected to the Lexium Cobot Controller, the **I/O Panel** displays the physical signal in the Lexium Cobot Controller.

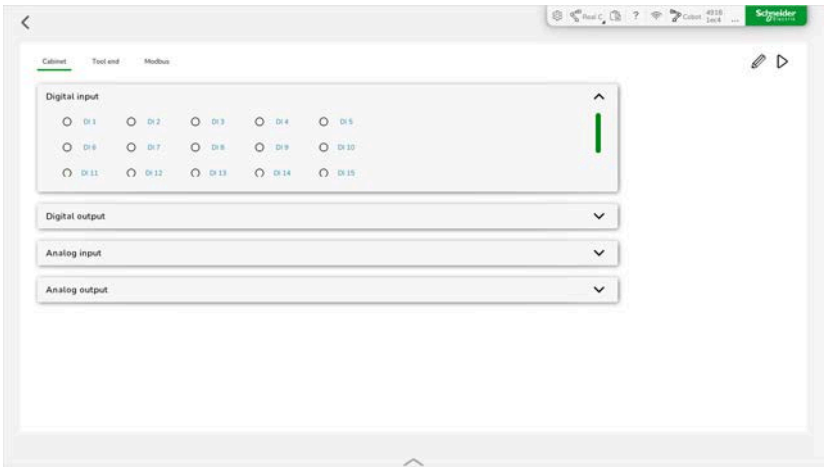
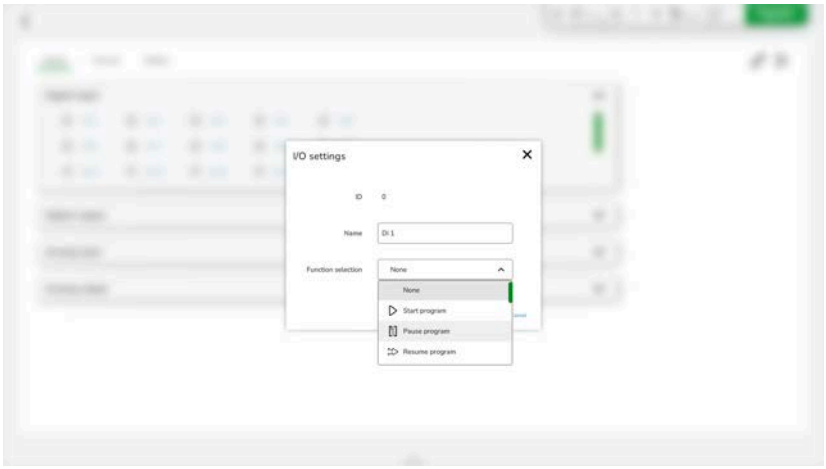
The **Cabinet** tab consists of four sections:

- **Digital input**
- **Digital output**
- **Analog input** (only available for the Lexium Cobot Cabinet Controller)
- **Analog output** (only available for the Lexium Cobot Cabinet Controller)

NOTE: Disable the Lexium Cobot Arm for editing the I/O.

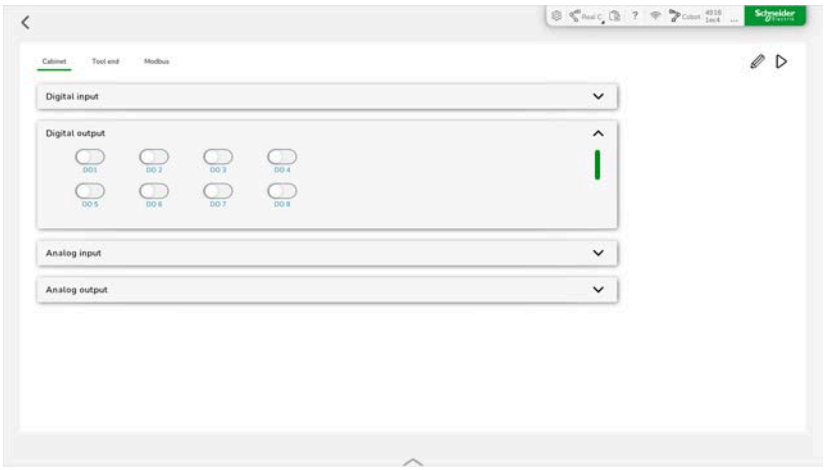
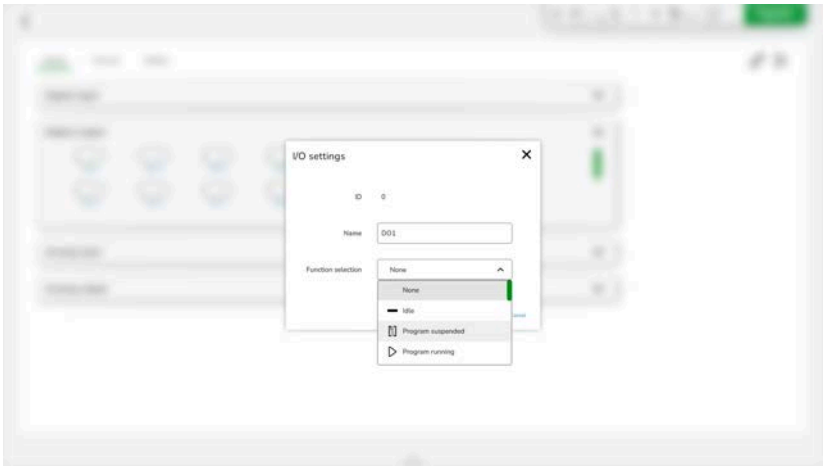
Setting a Digital Input Signal

To set a digital input signal (DI), perform the following steps:

Step	Action
1	In **I/O Panel** > **Cabinet**, click **Digital input** to expand the panel. 
2	Click the DI to be edited. **NOTE:** A digital interface can be used as either an input or an output. A crossed out signal means that it is configured for the other functionality (input or output). You can change this by changing the functionality in the panel where the signal is set as active. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DI from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DI is set and the selected function will be enabled when this DI signal is triggered.

Setting a Digital Output Signal

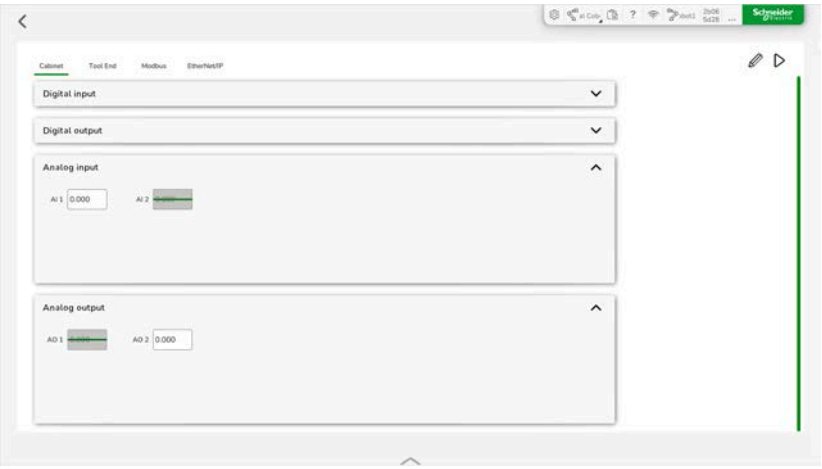
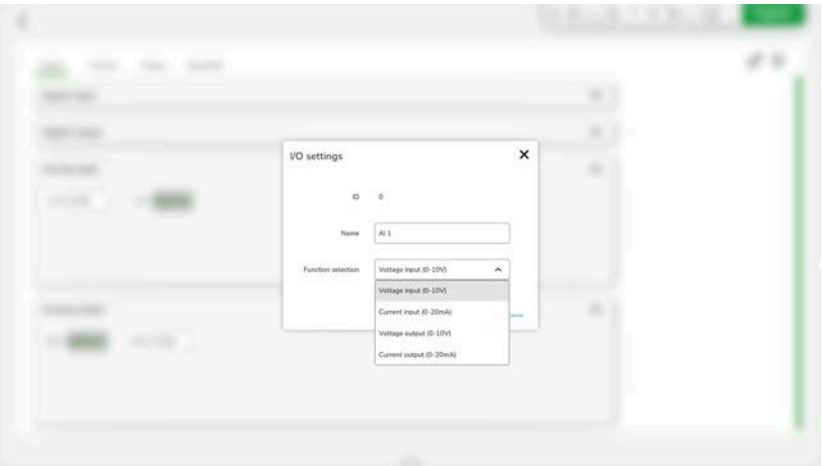
To set a digital output signal (DO), perform the following steps:

Step	Action
1	In **I/O Panel** > **Cabinet**, click **Digital output** to expand the panel. 
2	Click the DO name to be edited. **NOTE:** A digital interface can be used as either an input or an output. A crossed out signal means that it is configured for the other functionality (input or output). You can change this by changing the functionality in the panel where the signal is set as active. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DO from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DO is set.

Setting an Analog Input Signal

NOTE: Only available for the Lexium Cobot Cabinet Controller.

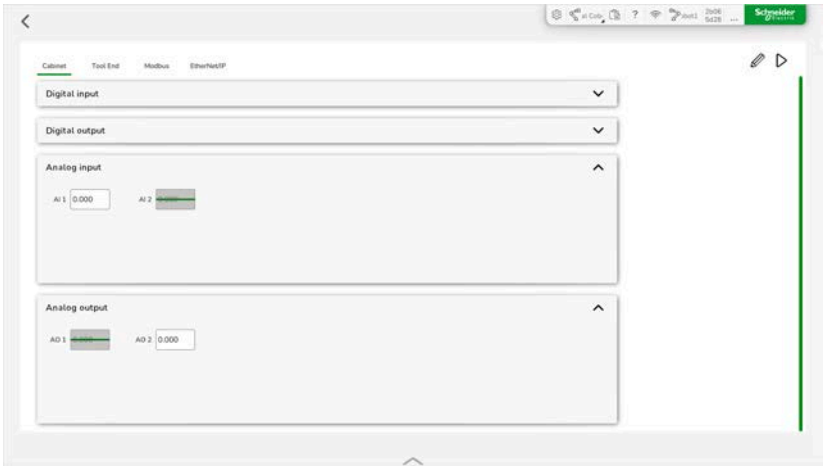
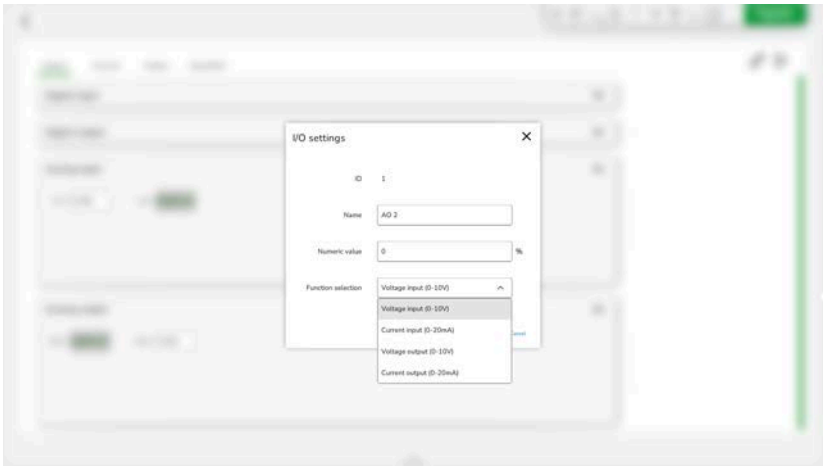
To set an analog input signal (AI), perform the following steps:

Step	Action
1	In **I/O Panel** > **Cabinet**, click **Analog input** to expand the panel. 
2	Click the AI to be edited. **Result:** The **I/O settings** dialog box is displayed. **NOTE:** An analog interface can be used either as an input or as an output. A crossed-out signal means that it is configured for the other functionality (input or output). You can change this by changing the functionality in the panel where the signal is set as active.
3	Optionally, edit the name.
4	Select the input function of the signal from **Function selection**.  The following options are available: • **Voltage input (0–10V)** • **Current input (0-20mA)** • **Voltage output (0-10V)** • **Current output (0-20mA)**
5	Click **Confirm**. **Result:** The AI is set.

Setting an Analog Output Signal

NOTE: Only available for the Lexium Cobot Cabinet Controller.

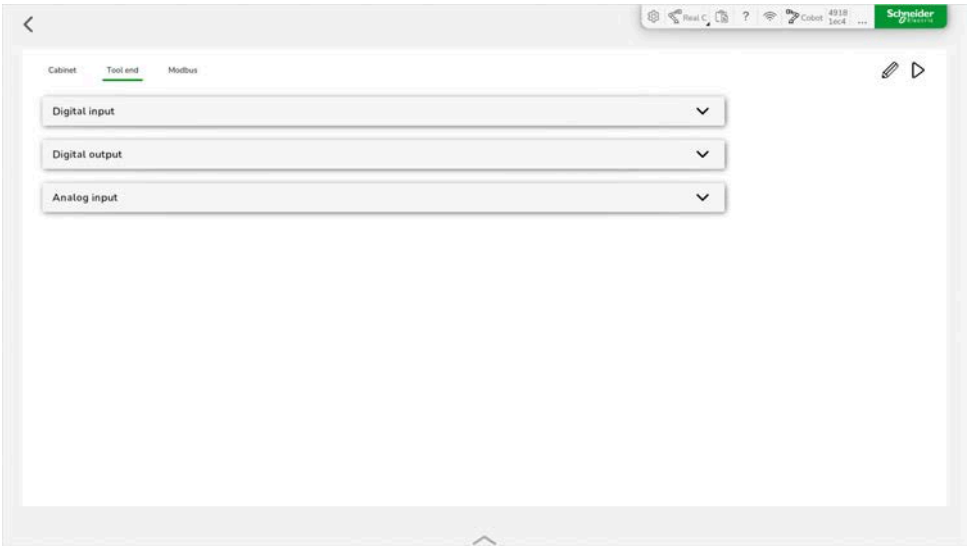
To set an analog output signal (AO), perform the following steps:

Step	Action
1	In **I/O Panel** > **Cabinet**, click **Analog output** to expand the panel. 
2	Click the signal to be edited. **Result:** The **I/O settings** dialog box is displayed. **NOTE:** An analog interface can be used either as an input or as an output. A crossed-out signal means that it is configured for the other functionality (input or output). You can change this by changing the functionality in the panel where the signal is set as active.
3	Optionally, edit the name.
4	Select the output function of the signal from **Function selection**.  The following options are available: • **Voltage input (0–10V)** • **Current input (0-20mA)** • **Voltage output (0-10V)** • **Current output (0-20mA)** **Result:** The field **Numeric value** is displayed.
5	In the field **Numeric value**, type in a value for the analog output. **NOTE:** The **Numeric value** represents the percentage of the maximum value. The range is 0...100. For example, when using the voltage output, a value of 50 represents 5 V dc.
6	Click **Confirm**. **Result:** The AO is set.

Tool End Tab

Overview

To set the inputs and outputs of the tool flange, go to **I/O Panel > Tool end**.



The **Tool end** tab consists of three sections:

- **Digital input** for the TDI
- **Digital output** for the TDO
- **Analog input** for the TAI

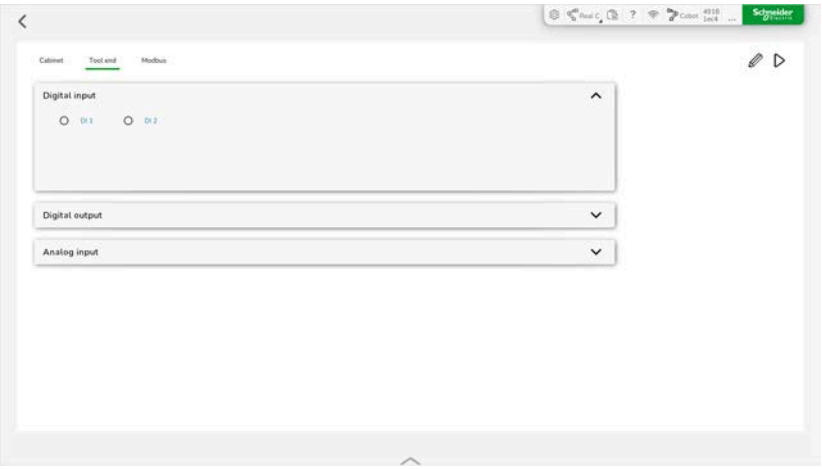
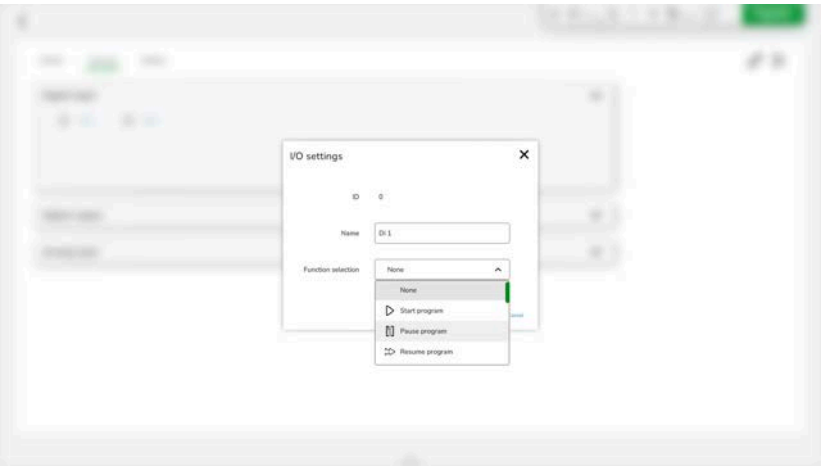
The TIO of the Lexium Cobot Arm is provided with two digital inputs, two digital outputs and two analog voltage inputs, with an input range of 0 ... 10 V dc.

NOTE: Disable the Lexium Cobot Arm for editing the I/O.

For the advanced configuration method, refer to Terminal I/O, page 140.

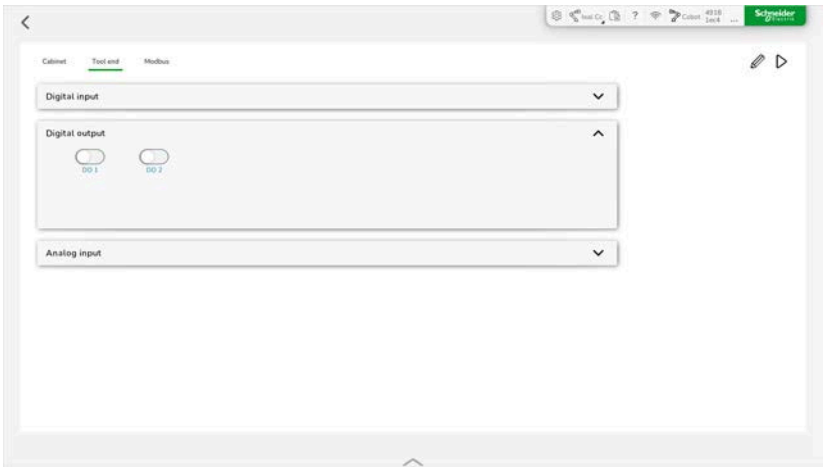
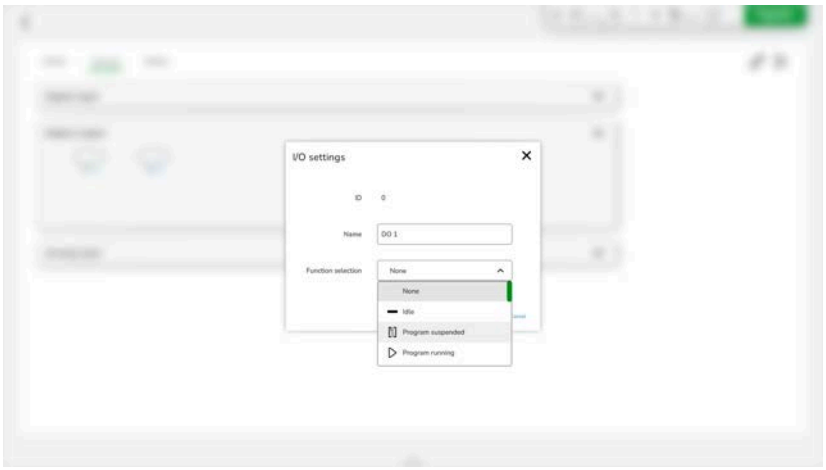
Setting a Tool End Digital Input Signal

To set a **Tool end** digital input signal (TDI), perform the following steps:

Step	Action
1	In **I/O Panel** > **Tool end**, click **Digital input** to expand the panel. 
2	Click the DI to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DI from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DI is set and the selected function will be enabled when this DI signal is triggered.

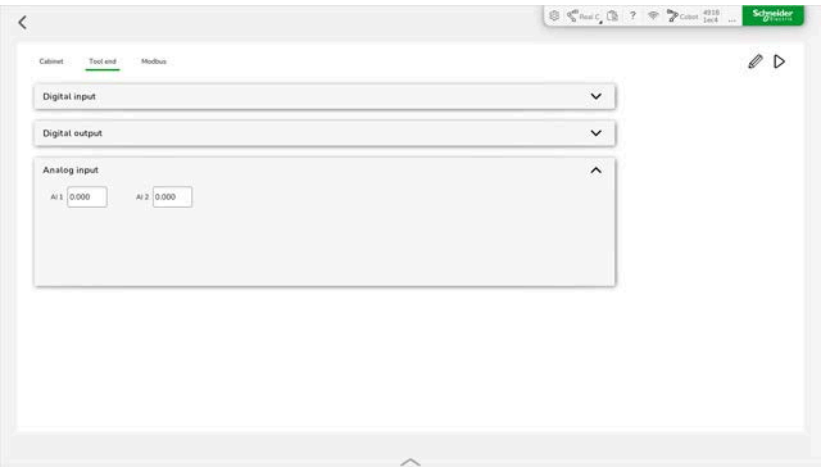
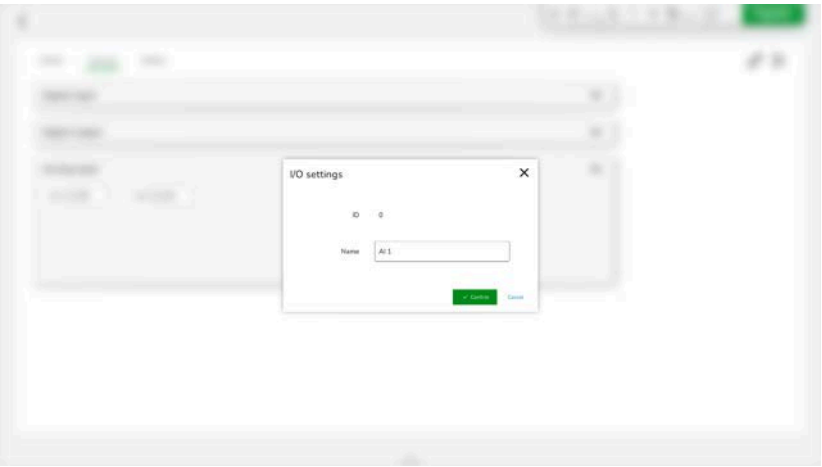
Setting a Tool End Digital Output Signal

To set a **Tool end** digital output signal (TDO), perform the following steps:

Step	Action
1	In **I/O Panel** > **Tool end**, click **Digital output** to expand the panel. 
2	Click the DO name to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DO from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DO is set.

Renaming a Tool End Analog Input Signal

To rename a **Tool end** analog input signal (TAI), perform the following steps:

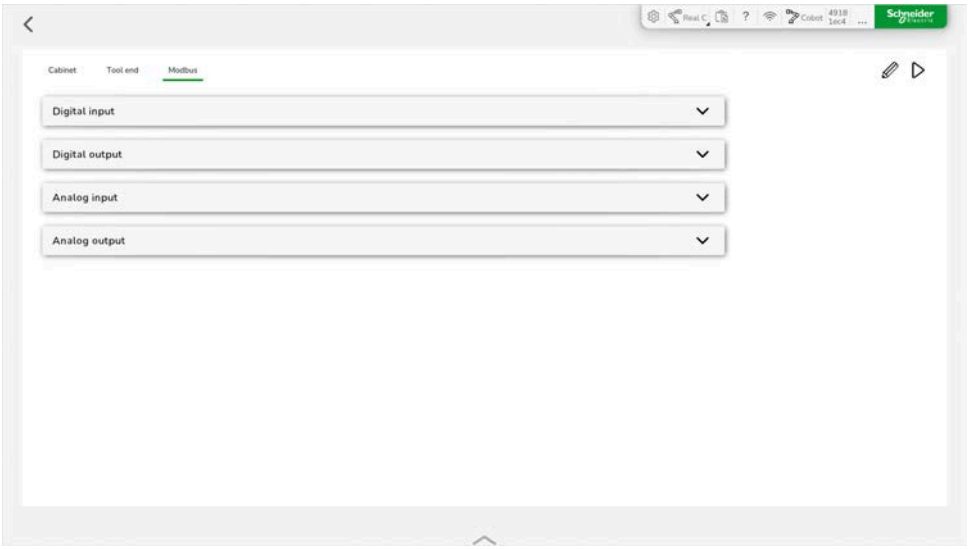
Step	Action
1	In **I/O Panel** > **Tool end**, click **Analog input** to expand the panel. 
2	Click the **Tool end** analog input to be edited. **Result:** The **I/O settings** dialog box is displayed. 
3	Edit the name.
4	Click **Confirm**. **Result:** The AI is renamed.

Modbus Tab

Overview

The Lexium Cobot Controllers support the Modbus communication protocol and can be used as a Modbus communication server for interaction with external devices.

To set the Modbus inputs and outputs of the Lexium Cobot Controller, go to **I/O Panel > Modbus**.



The **Modbus** tab consists of four sections:

- **Digital input**
- **Digital output**
- **Analog input**
- **Analog output**

I/O signals in the **Modbus** tab are the I/O data that is accessed by the Lexium Cobot and external devices via the Modbus communication protocol.

The Lexium Cobot Controllers support the following maximum number of inputs and outputs:

- 128 digital inputs and 128 digital outputs
- 16 integer analog inputs and 16 integer analog outputs
- 16 signed analog inputs and 16 signed analog outputs
- 32 analog inputs with floating point numbers and 32 analog outputs with floating point numbers

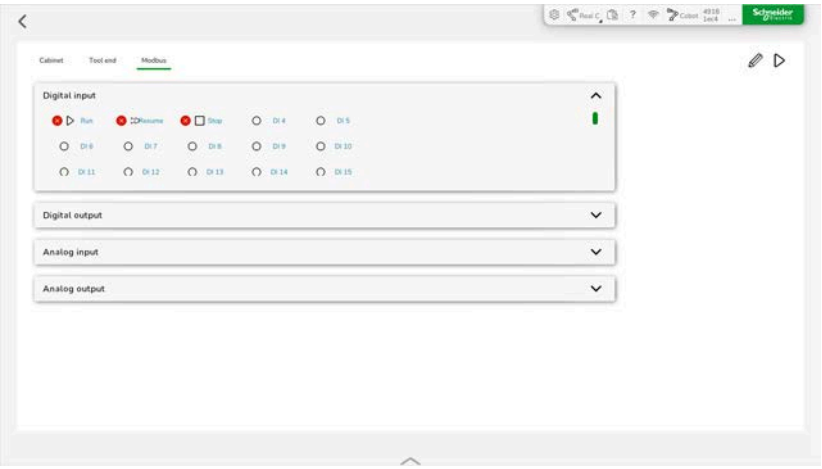
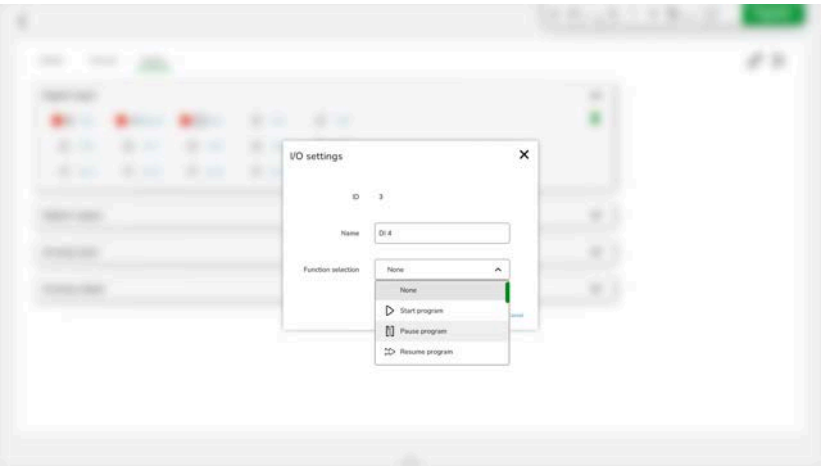
For the definition of the Modbus register addresses, refer to the Modbus Address Table, page 267.

NOTE: Disable the Lexium Cobot Arm for editing the I/O.

For the advanced configuration method, refer to Terminal I/O, page 140.

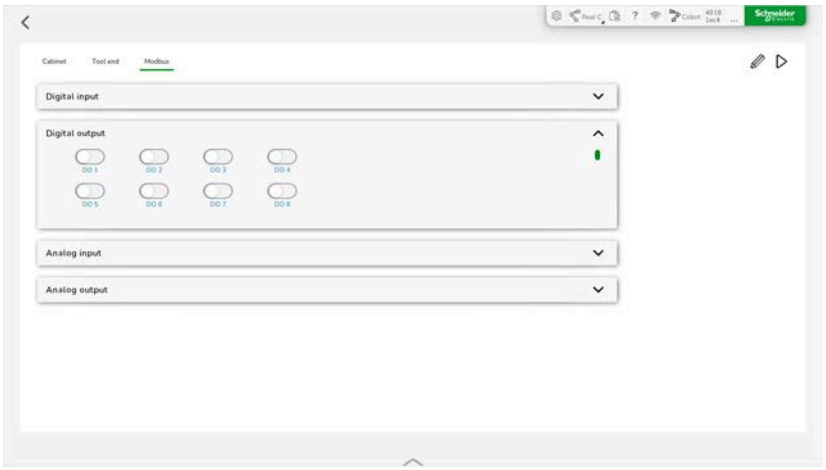
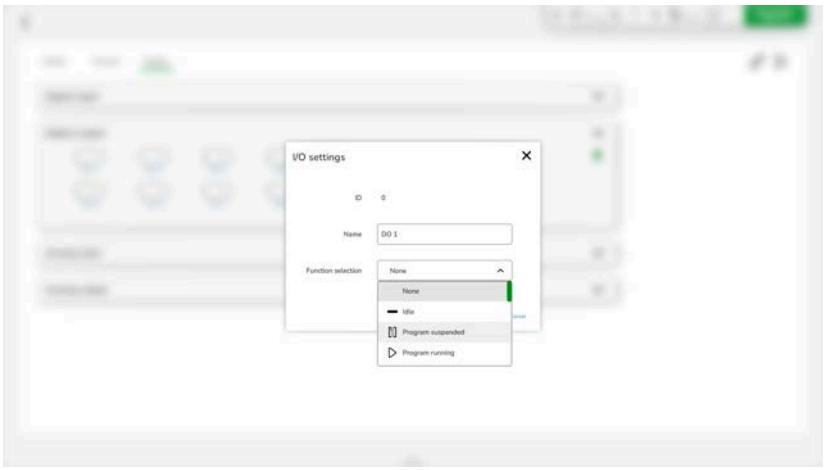
Setting a Modbus Digital Input Signal

To set a Modbus digital input signal (DI), perform the following steps:

Step	Action
1	In **I/O Panel** > **Modbus**, click **Digital input** to expand the panel. 
2	Click the DI to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DI from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DI is set and the selected function will be enabled when this DI signal is triggered.

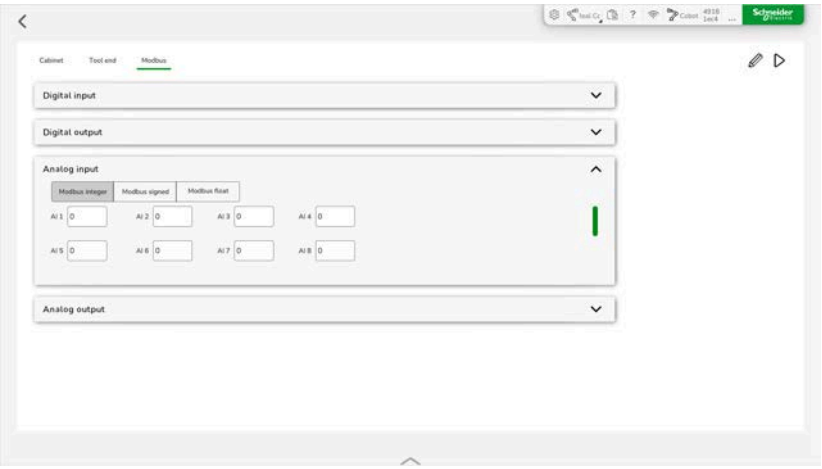
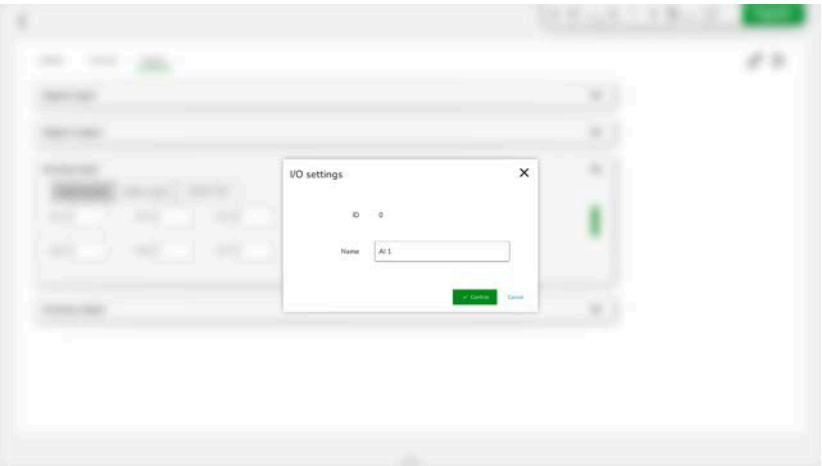
Setting a Modbus Digital Output Signal

To set a Modbus digital output signal (DO), perform the following steps:

Step	Action
1	In **I/O Panel** > **Modbus**, click **Digital output** to expand the panel. 
2	Click the DO name to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DO from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DO is set.

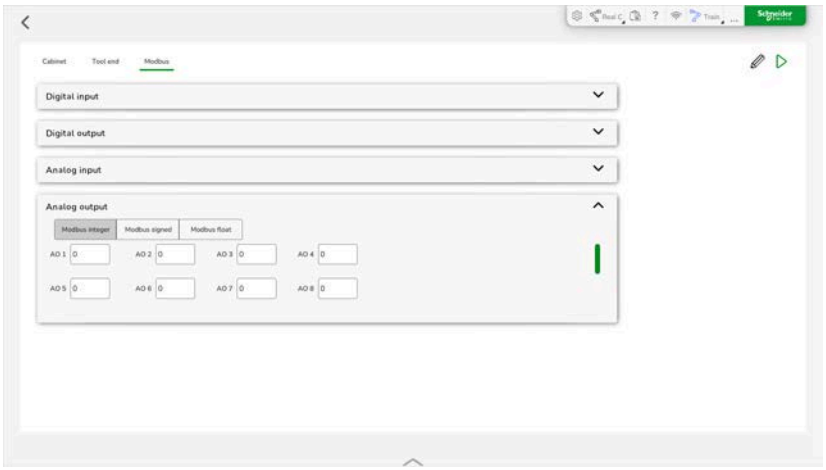
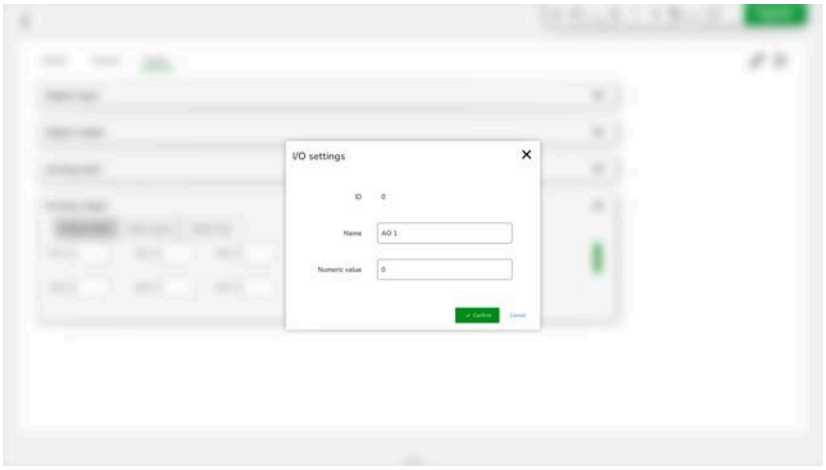
Renaming a Modbus Analog Input Signal

To rename a Modbus analog input signal (AI), perform the following steps:

Step	Action
1	In **I/O Panel** > **Modbus**, click **Analog input** to expand the panel. 
2	Select the tab of the applicable data type.
3	Click the AI to be edited. **Result:** The **I/O settings** dialog box is displayed. 
4	Edit the name.
5	Click **Confirm**. **Result:** The AI is renamed.

Setting a Modbus Analog Output Signal

To set a Modbus analog output signal (AO), perform the following steps:

Step	Action
1	In **I/O Panel** > **Modbus**, click **Analog output** to expand the panel. 
2	Select the tab of the applicable data type.
3	Click the signal to be edited. **Result:** The **I/O settings** dialog box is displayed. 
4	Optionally, edit the name.
5	Type in an initial **Numeric value**.
6	Click **Confirm**. **Result:** The AO is set.

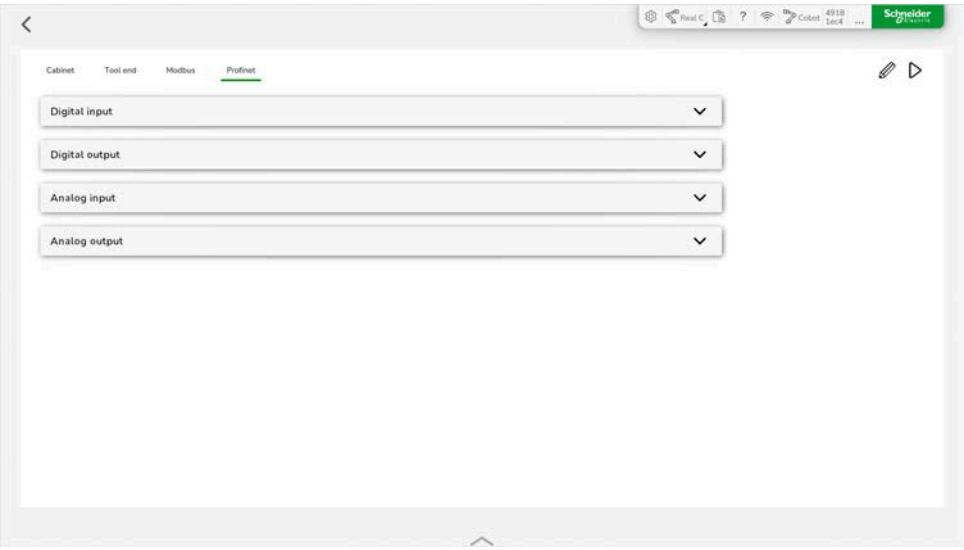
Profinet

Overview

The Lexium Cobot Controllers support the Profinet communication protocol and can be used as a Profinet I/O device server for communication with external devices.

To display the Profinet section, you have to enable Profinet in the settings. For further information, refer to Profinet Settings, page 136.

To set the inputs and outputs for Profinet, go to **I/O Panel > Profinet**.



The **Profinet** tab consists of four sections:

- **Digital input**
- **Digital output**
- **Analog input**
- **Analog output**

I/O signals in the **Profinet** tab are the I/O data that is accessed by the Lexium Cobot and external devices via the Profinet communication protocol.

The Lexium Cobot Controllers support the following maximum number of inputs and outputs:

- 64 digital inputs and 64 digital outputs
- 32 signed number analog inputs and 32 signed number analog outputs
- 32 floating-point number analog inputs and 32 floating-point number analog outputs

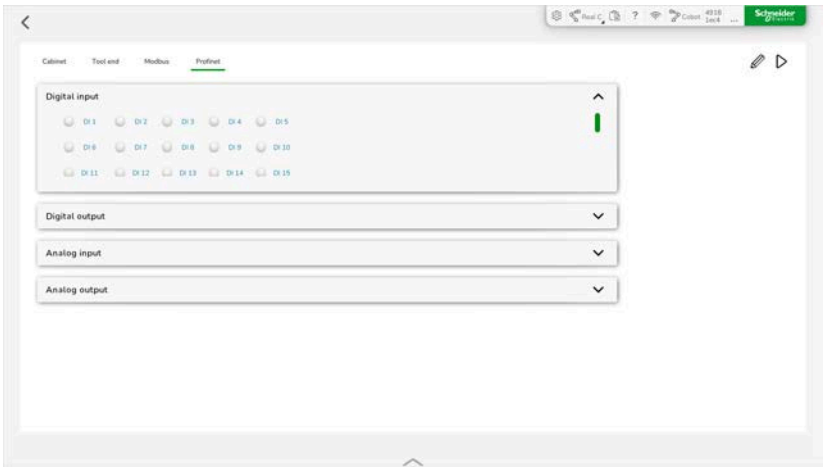
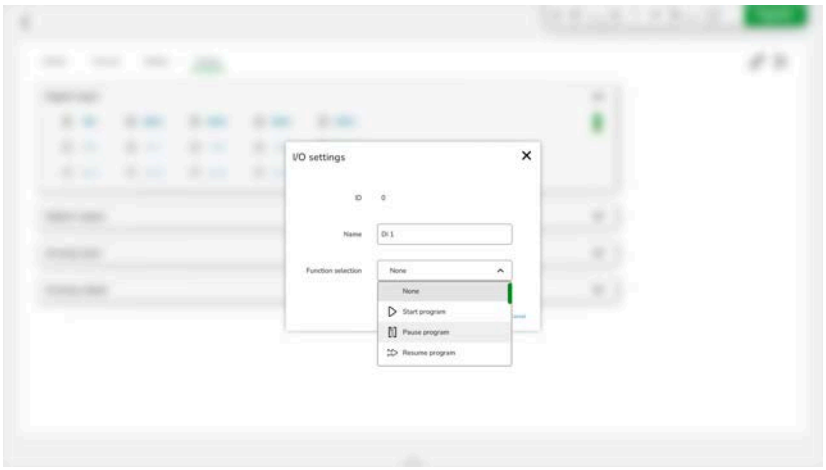
For the definition of the Profinet addresses, refer to the Profinet Address Table, page 273.

For GSDML-XML form device description files, contact your local Schneider Electric representative.

NOTE: Disable the Lexium Cobot Arm for editing the I/O.

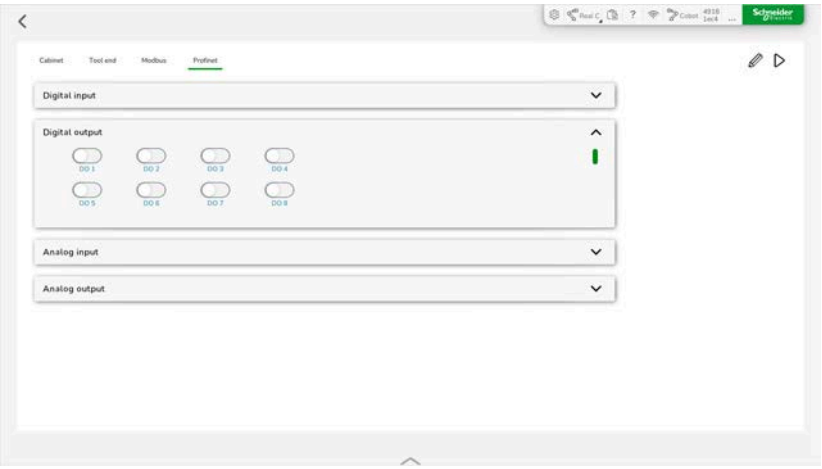
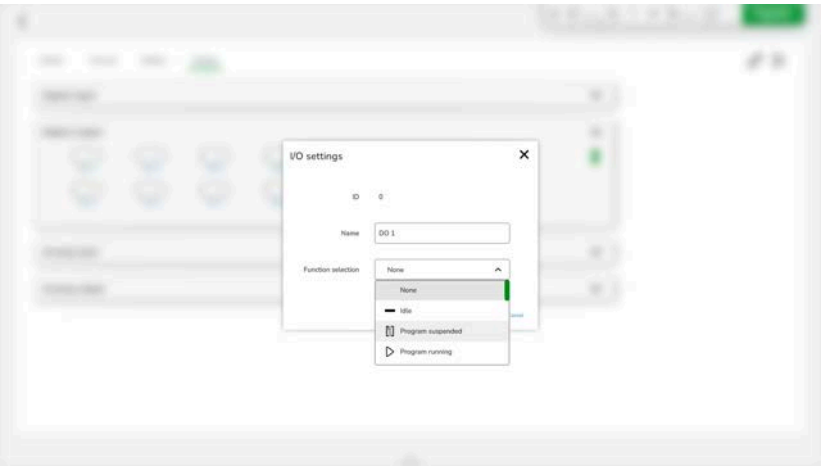
Setting a Profinet Digital Input Signal

To set a Profinet digital input signal (DI), perform the following steps:

Step	Action
1	In **I/O Panel** > **Profinet**, click **Digital input** to expand the panel. 
2	Click the DI to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DI from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DI is set and the selected function will be enabled when this DI signal is triggered.

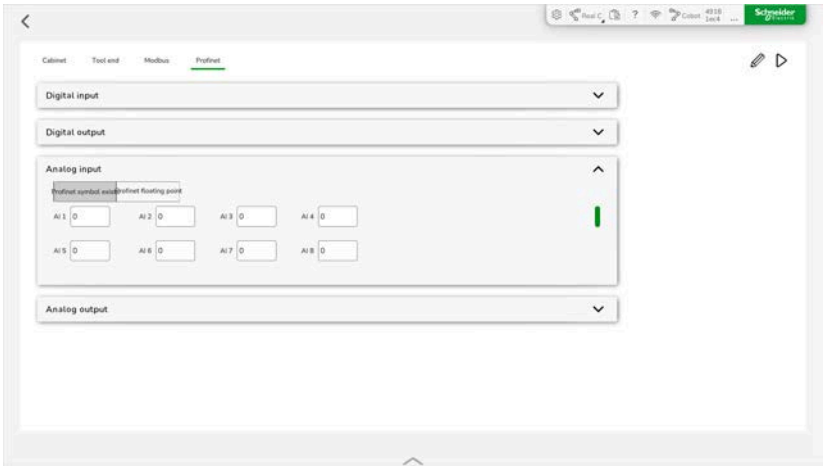
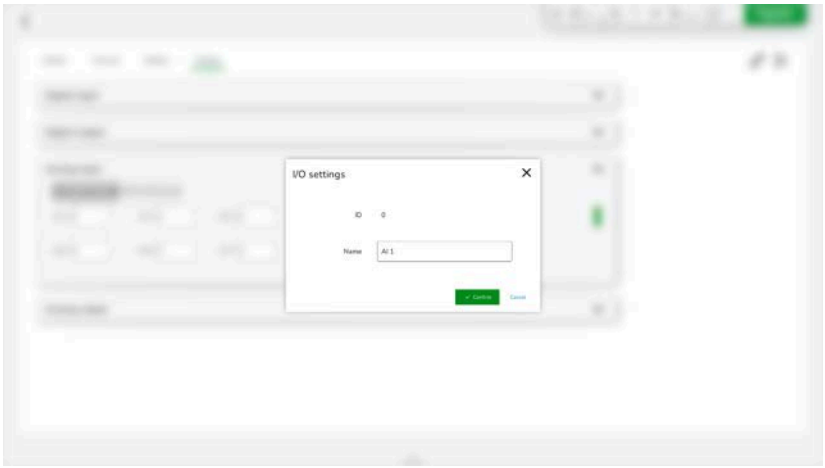
Setting a Profinet Digital Output Signal

To set a Profinet digital output signal (DO), perform the following steps:

Step	Action
1	In **I/O Panel** > **Profinet**, click **Digital output** to expand the panel. 
2	Click the DO name to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DO from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DO is set.

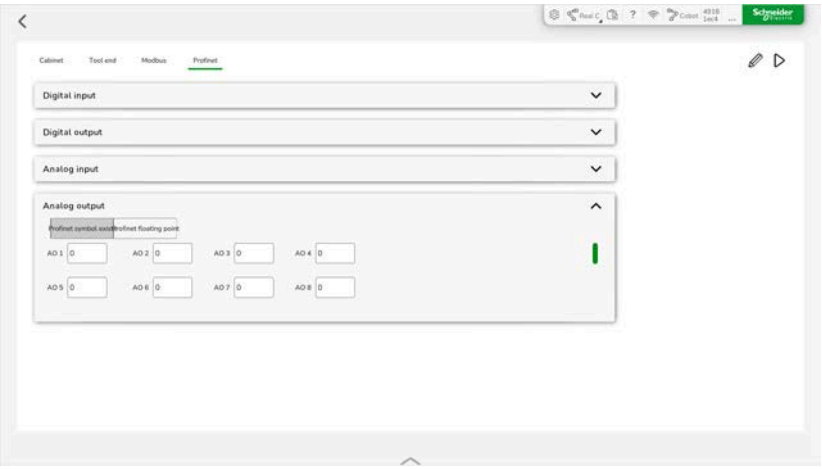
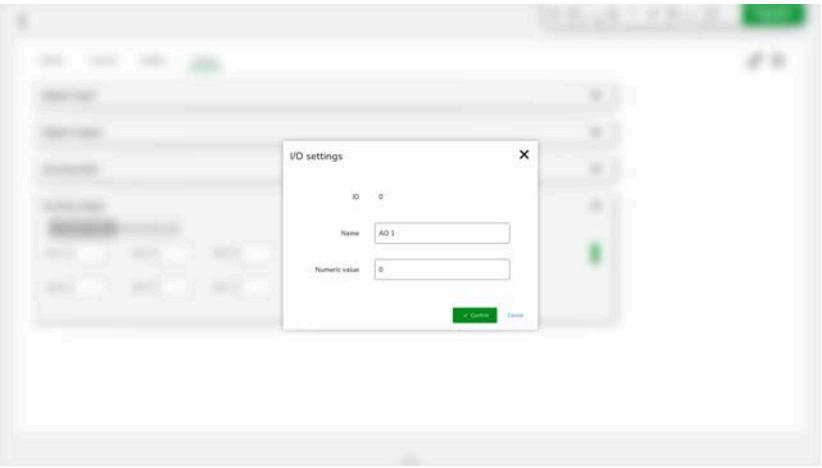
Renaming a Profinet Analog Input Signal

To rename a Profinet analog input signal (AI), perform the following steps:

Step	Action
1	In **I/O Panel** > **Profinet**, click **Analog input** to expand the panel.
2	Select the tab of the applicable data type. 
3	Click the AI to be edited. **Result:** The **I/O settings** dialog box is displayed. 
4	Edit the name.
5	Click **Confirm**. **Result:** The AI is renamed.

Setting a Profinet Analog Output Signal

To set a Profinet analog output signal (AO), perform the following steps:

Step	Action
1	In **I/O Panel** > **Profinet**, click **Analog output** to expand the panel. 
2	Select the tab of the applicable data type.
3	Click the signal to be edited. **Result:** The **I/O settings** dialog box is displayed. 
4	Optionally, edit the name.
5	Type in an initial **Numeric value**.
5	Click **Confirm**. **Result:** The AO is set.

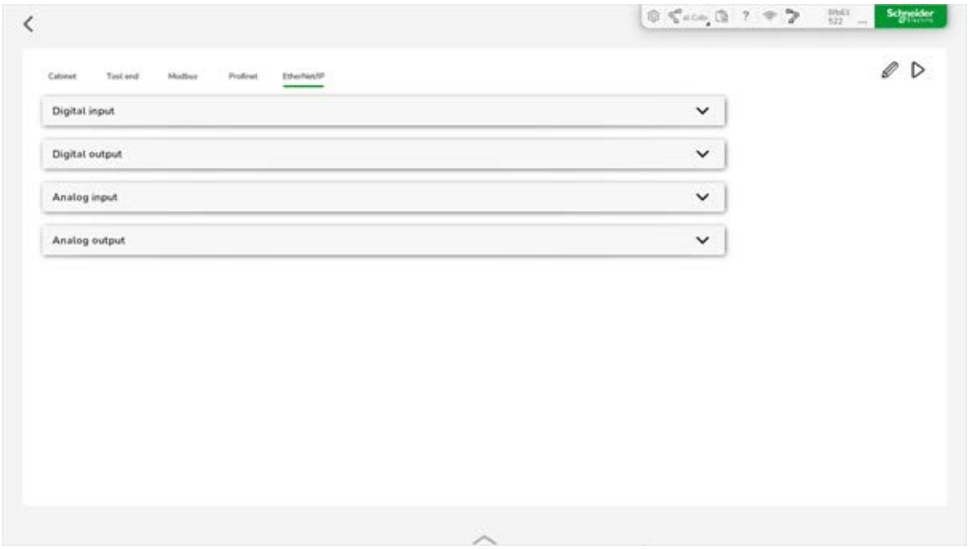
EtherNet/IP

Overview

The Lexium Cobot Controllers support the Ethernet/IP communication protocol and can be used as an Ethernet/IP communication adapter to interact with external devices.

To display the EtherNet/IP section, enable Ethernet/IP in the settings. For further information, refer to EtherNet/IP Settings, page 137.

To set the inputs and outputs for EtherNet/IP, go to **I/O Panel > EtherNet/IP**.



The **EtherNet/IP** tab consists of four sections:

- **Digital input**
- **Digital output**
- **Analog input**
- **Analog output**

I/O signals in the **EtherNet/IP** tab are the I/O data that is accessed by the Lexium Cobot and external devices via the EtherNet/IP communication protocol.

The Lexium Cobot Controllers support the following maximum number of inputs and outputs:

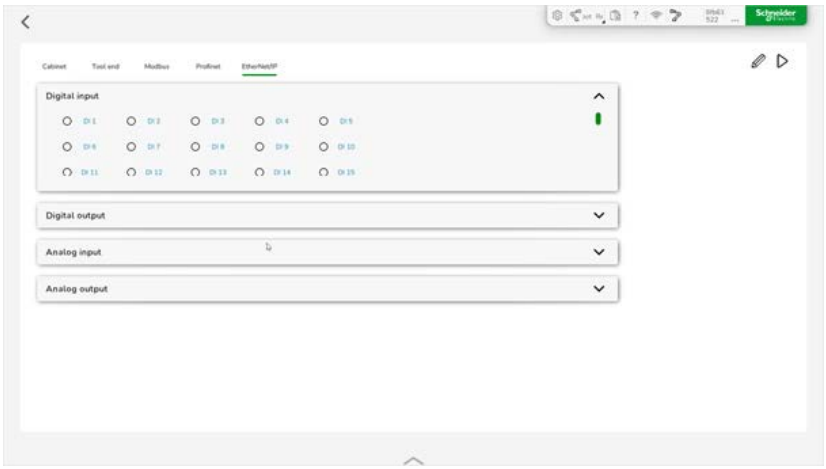
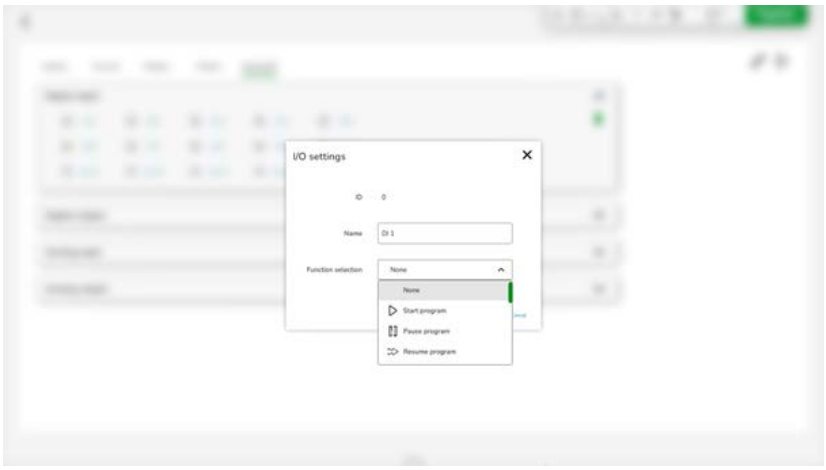
- 64 digital inputs and 64 digital outputs
- 24 signed number analog inputs and 24 signed number analog outputs
- 24 floating-point number analog inputs and 24 floating-point number analog outputs

For the definition of the EtherNet/IP register addresses, refer to the EtherNet/IP Address Table, page 277.

NOTE: Disable the Lexium Cobot Arm for editing the I/O.

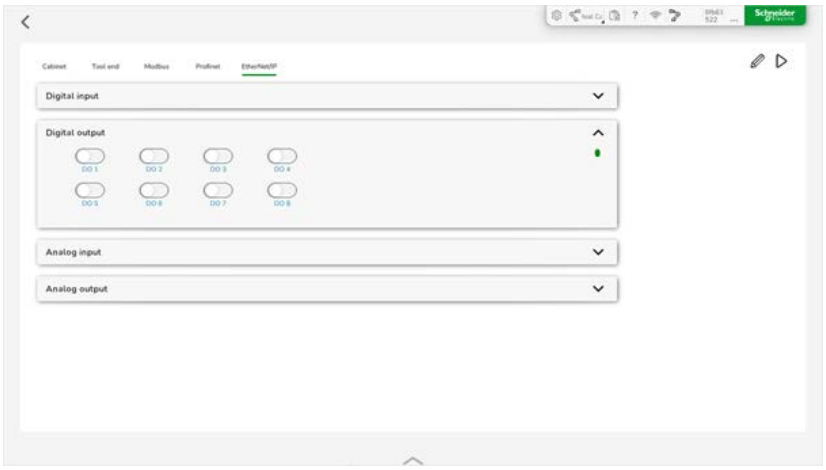
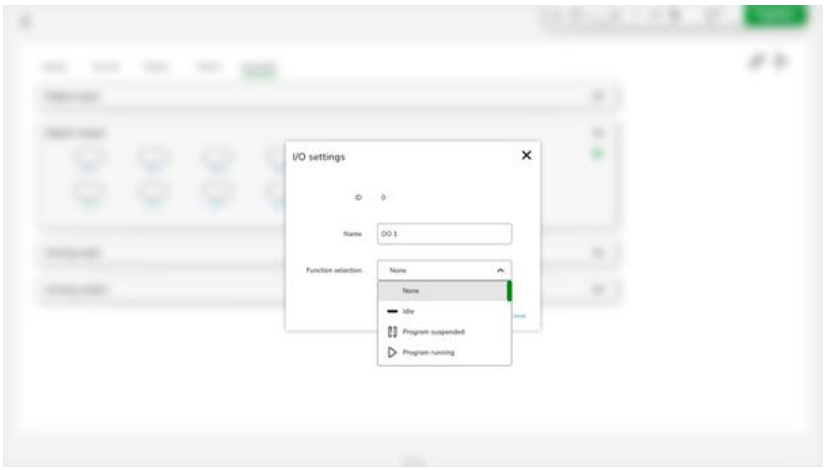
Setting an EtherNet/IP Digital Input Signal

To set an EtherNet/IP digital input signal (DI), perform the following steps:

Step	Action
1	In **I/O Panel** > **EtherNet/IP**, click **Digital input** to expand the panel. 
2	Click the DI to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DI from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DI is set and the selected function will be enabled when this DI signal is triggered.

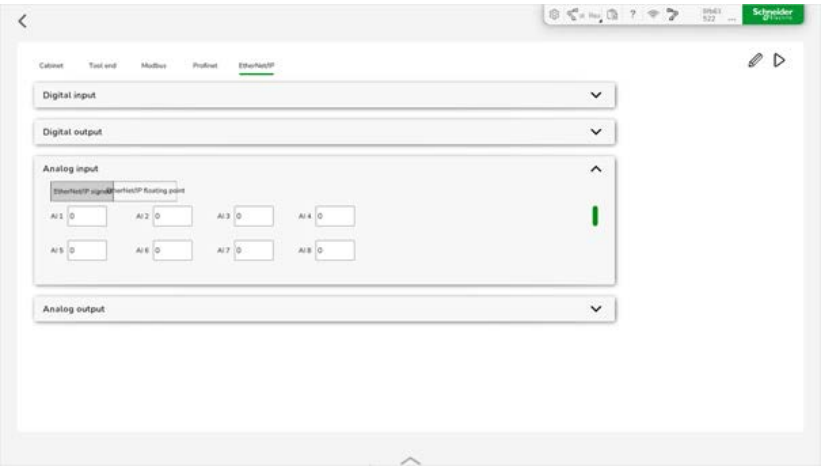
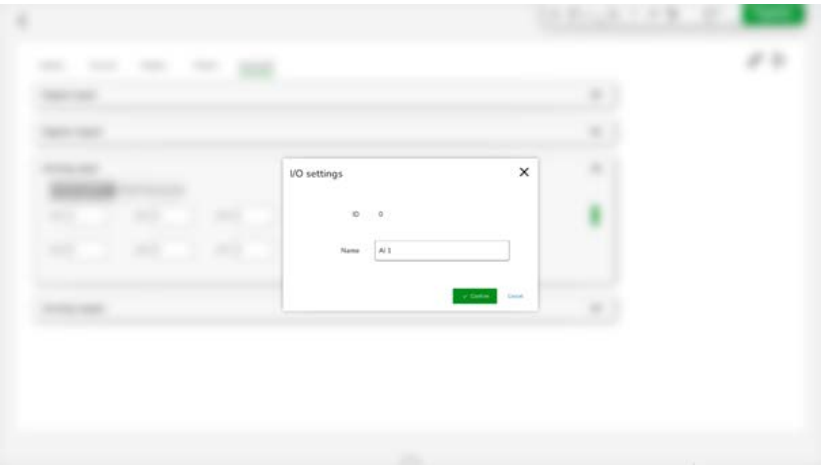
Setting an EtherNet/IP Digital Output Signal

To set an EtherNet/IP digital output signal (DO), perform the following steps:

Step	Action
1	In **I/O Panel** > **EtherNet/IP**, click **Digital output** to expand the panel. 
2	Click the DO name to be edited. **Result:** The **I/O settings** dialog box is displayed.
3	Optionally, edit the name.
4	Select the function of the DO from **Function selection**.  For further information about the different options, refer to Function Settings, page 148.
5	Click **Confirm**. **Result:** The DO is set.

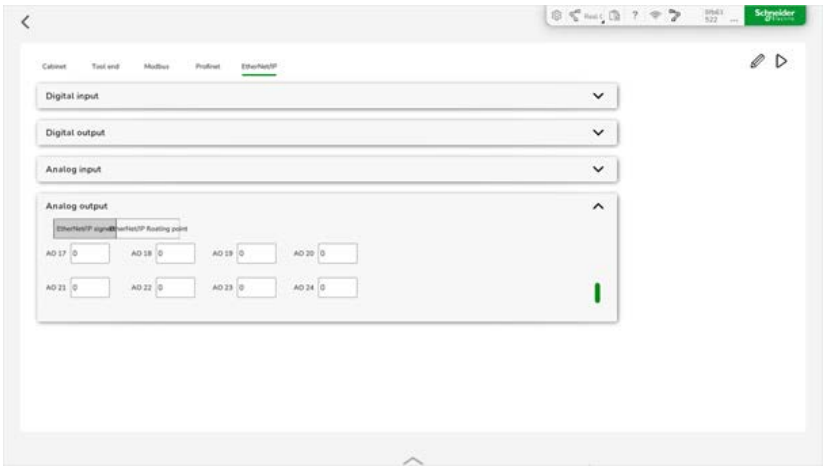
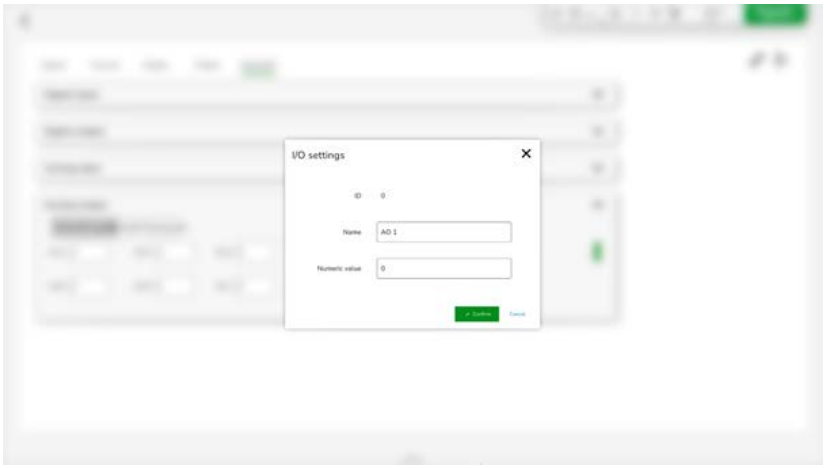
Renaming an EtherNet/IP Analog Input Signal

To rename an EtherNet/IP analog input signal (AI), perform the following steps:

Step	Action
1	In **I/O Panel** > **EtherNet/IP**, click **Analog input** to expand the panel.
2	Select the tab of the applicable data type. 
3	Click the AI to be edited. **Result:** The **I/O settings** dialog box is displayed. 
4	Edit the name.
5	Click **Confirm**. **Result:** The AI is renamed.

Setting an EtherNet/IP Analog Output Signal

To set an EtherNet/IP analog output signal (AO), perform the following steps:

Step	Action
1	In **I/O Panel** > **EtherNet/IP**, click **Analog output** to expand the panel. 
2	Select the tab of the applicable data type.
3	Click the signal to be edited. **Result:** The **I/O settings** dialog box is displayed. 
4	Optionally, edit the name.
5	Type in an initial **Numeric value**.
5	Click **Confirm**. **Result:** The AO is set.

Adding Extended I/O

Extended I/O

The **I/O Panel** has a dynamic I/O configuration function that allows the Lexium Cobot Controllers to act as a client of Modbus communication.

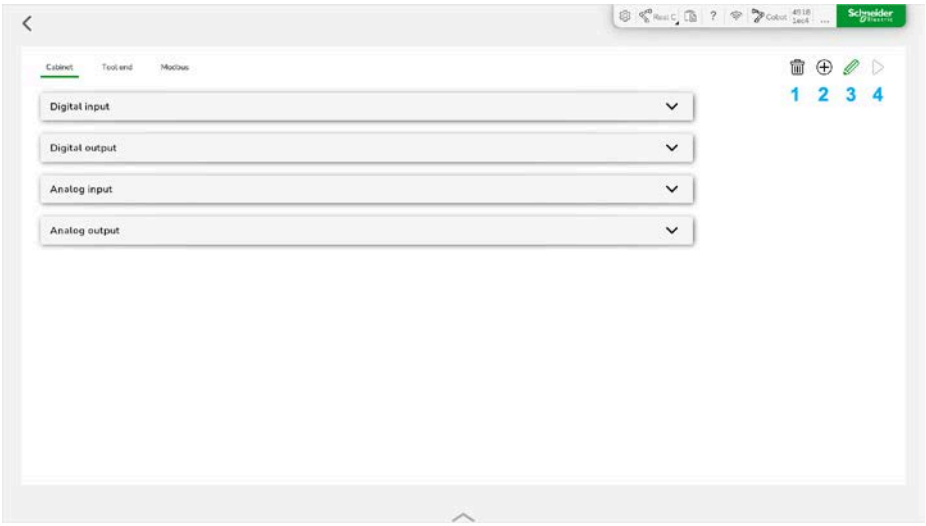
This function is located in **I/O Panel > Cabinet**, in the upper right corner of the and it has two modes:

- Operation mode

1 Edit for switching to the editing mode

2 Running indicates the active operating mode

- Edit mode



1 Delete for deleting a dynamic I/O tab

2 Add a new dynamic I/O tab

3 Edit an existing dynamic I/O tab

4 Run for switching to operating mode

When switched to the editing state:

Click the **Add** icon to configure the dynamic I/O.

The configuration consists of Modbus-TCP and Modbus-RTU. The Modbus setting is connected with the communication interface of the Lexium Cobot

Controller. The TCP/IP mode is connected through Ethernet, while the RTU mode is connected through the RS485 serial line interface.

NOTE:

- Disable and power off the Lexium Cobot Arm for configuring the extended I/O.
- Maximum number of extended I/O:

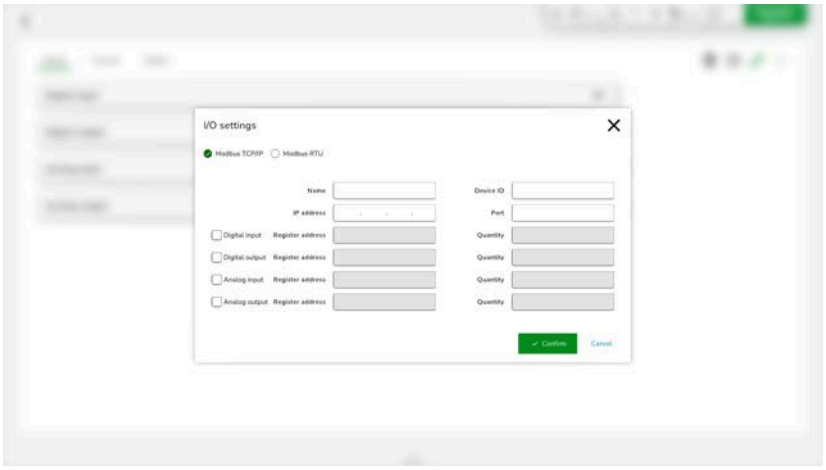
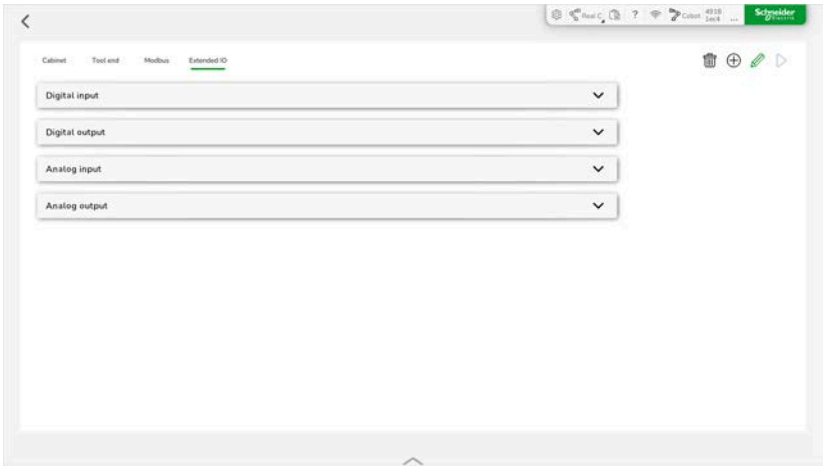
  32 for AIO

  64 for DIO

  Up to 8 modules are supported as an extension.

Adding an Extended I/O Configuration

To add an extended I/O configuration, perform the following steps:

Step	Action
1	In **I/O Panel** > **Cabinet**, click the **Edit** icon, then click the **Add** icon. **Result:** The I/O settings dialog box is displayed. 
2	Select **Modbus TCP/IP** or **Modbus RTU** and set the parameters according to your application.
3	Click **Confirm**. **Result:** The extended input and output configuration is added to the tabs. **Example:** 
4	Click the **Run** icon. **Result:** The connection of the extended I/O is established and you can configure the inputs and outputs.

Blockly Programming

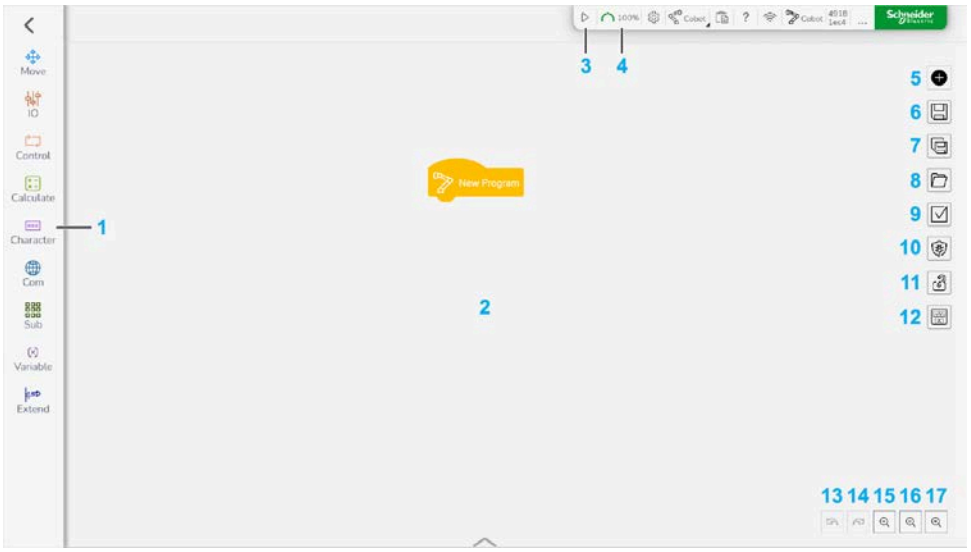
What's in This Chapter

Programming Control Interface

Overview

EcoStruxure Cobot Expert provides a visual programming interface which contains programming blocks and assisting dialog boxes to develop the program code for the Lexium Cobot.

To open the **Programming Control** interface, select **Programming Control** in the feature bar.



1 Instructions menu

2 Editing area

3 **Run** the program

4 **Speed** setting of the program

5 **New** programming file

6 **Save** the programming file

7 **Save as** the programming file

8 **Open** a programming file

9 **Advanced Operation**

10 **Debug** mode

11 **Lock** the program

12 **Variable Observation** in the editing area

13 **Undo** the last action

14 **Redo** the last undone action

15 **Zoom in** the editing area

16 **Reset view** of the editing area

17 **Zoom out** the editing area

The header of the job program consists of the yellow **New Program** instruction block, which is placed in by default.

The following instructions, connected to the head, form the body of the job program. The Lexium Cobot Arm executes the program step by step from top to bottom.

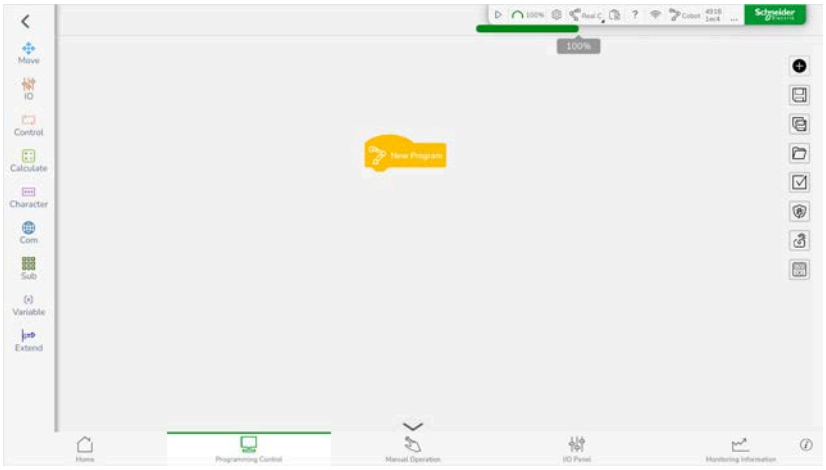
NOTE: To edit the name of the program, click the instruction block.

Running a Program

Step	Action
1	Power on and enable the Lexium Cobot Arm.
2	In the feature bar, select **Programming Control**. Then, click the **Run** icon from the top menu. **Result:** The Lexium Cobot runs the program.

NOTE: When the program is running, the **Run** icon is replaced by the **Pause** icon for pausing the program and the **Stop** icon for stopping the program.

Adjusting the Running Speed

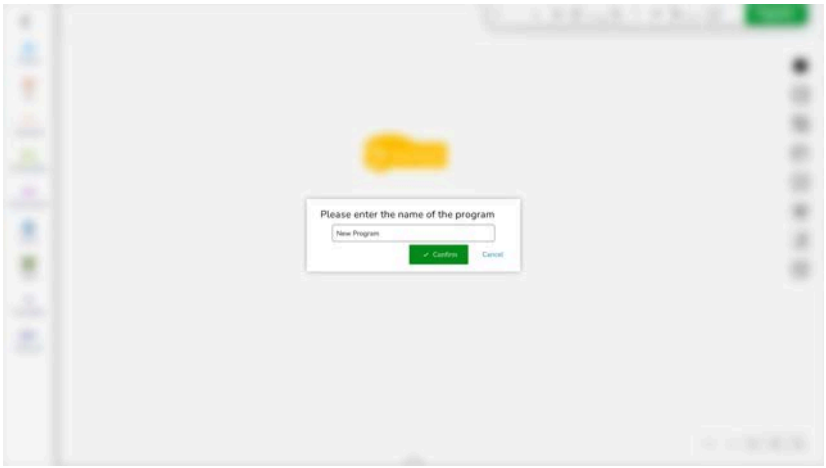
Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Speed** icon from the top menu. **Result:** The speed bar is displayed under the speed icon. 100% corresponds to the value that is defined in the program. 
2	There are two options to adjust the running speed: • Slide the speed bar to set the running speed of the Lexium Cobot Arm. • Click the percentage value under the speed bar and enter the running speed. 

Creating a New Program

In the feature bar, select **Programming Control**. Then, click the **New** icon on the right of the editing area.

Result: The new program is created.

Renaming a Program

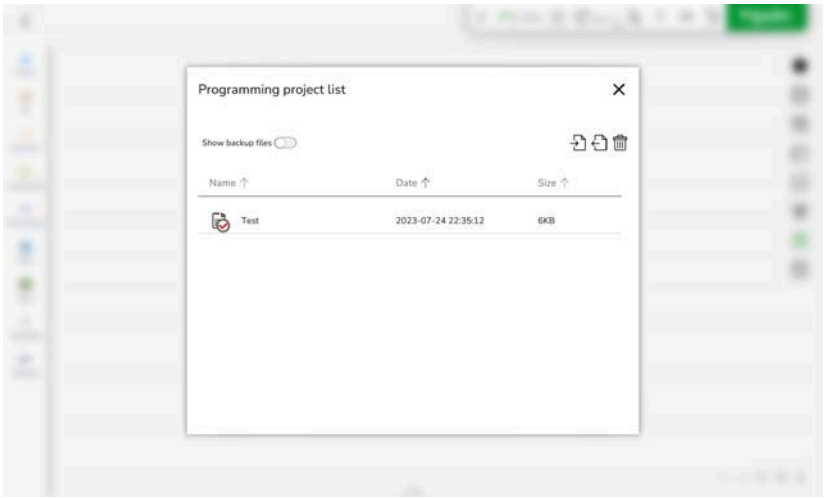
Step	Action
1	Click the **New Program** block. **Result:** The name setting dialog box is displayed. 
2	Type in the name of the program and click **Confirm**. **Result:** The program is renamed.

Saving a Program

You have two options to save the program:

- To save the program as the latest version, click the **Save** icon on the right of the editing area.
- To save the program under a new file name, click the **Save as** icon on the right of the editing area.

Displaying the Programs

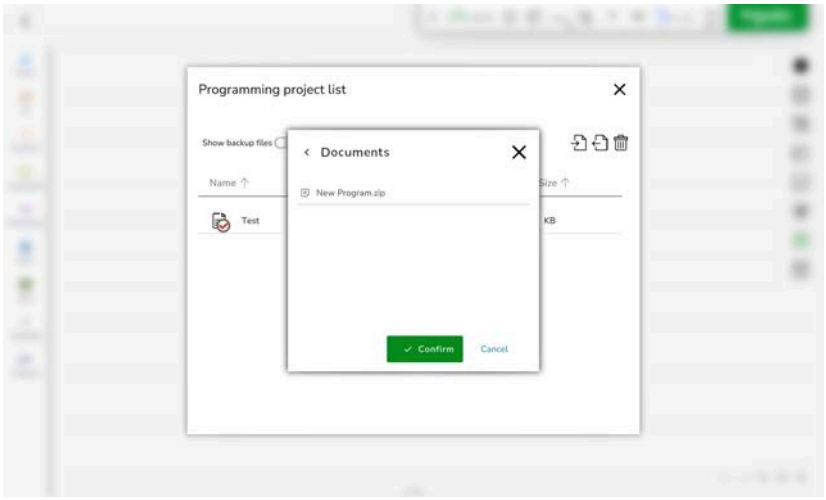
Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Open** icon on the right of the editing area. **Result:** The programs saved in the Lexium Cobot Controller are displayed. 
2	To sort the program list differently, click **Name**, **Date**, or **Size**.

Open a Saved Program

Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Open** icon on the right of the editing area. **Result:** The programs saved in the Lexium Cobot Controller are displayed.
2	Click the name of the program to be opened. **Result:** The program opens.

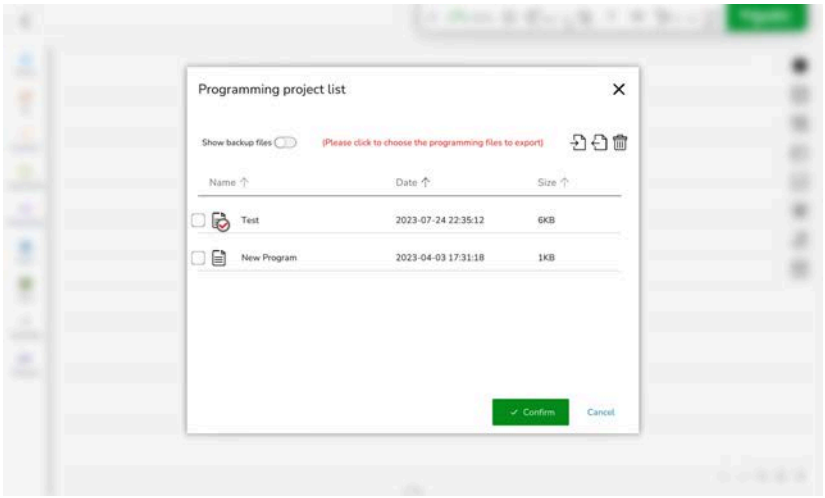
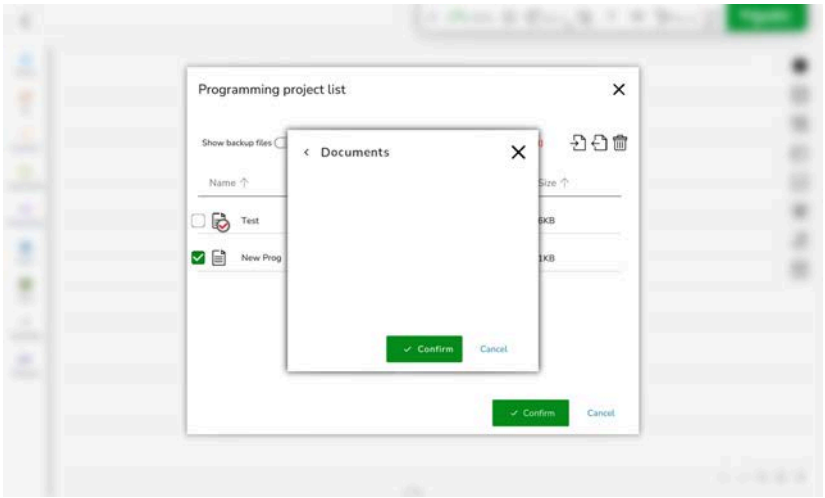
Importing a Program

To import an exported program to the connected Lexium Cobot Controller, perform the following steps:

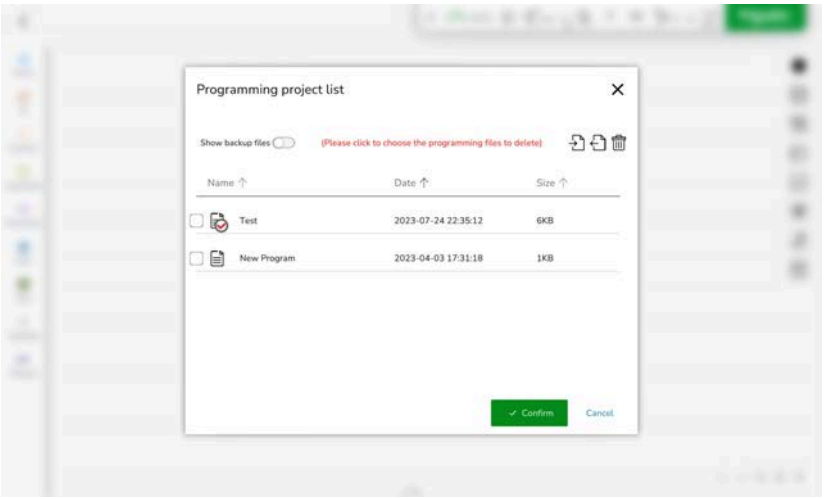
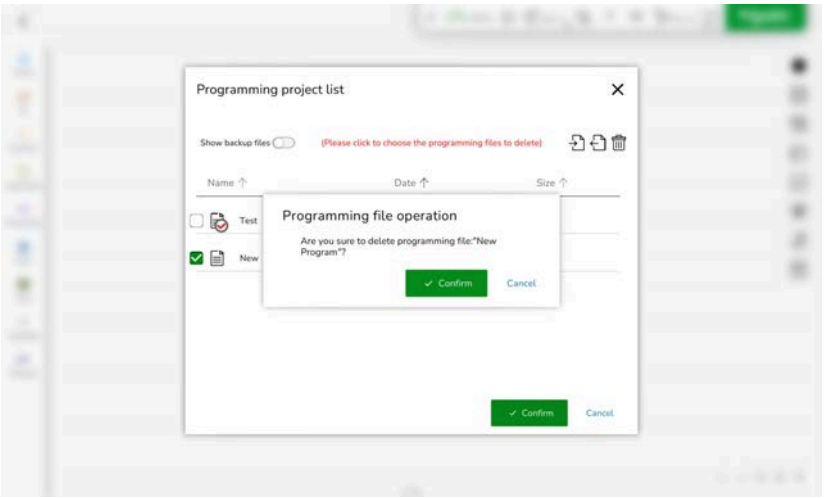
Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Open** icon on the right of the editing area. **Result:** The programs saved in the Lexium Cobot Controller are displayed.
2	In the **Programming project list** dialog box, click the **Import** icon:  **Result:** The file manager dialog box is displayed. 
3	Select the program to be imported.
4	Click **Confirm**. **Result:** The program is imported to the Lexium Cobot Controller and is added to the **Programming project list**.

Exporting a Program

To export the program saved in the Lexium Cobot Controller, perform the following steps:

Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Open** icon on the right of the editing area. **Result:** The programs saved in the Lexium Cobot Controller are displayed.
2	In the **Programming project list** dialog box, click the **Export** icon:  **Result:** Checkboxes for the listed programs are displayed. 
3	Select the program to be exported. **NOTE:** Up to five programs can be selected at the same time.
4	Click **Confirm**. **Result:** The file manager dialog box is displayed. 
5	Select the path to export the program.
6	Click **Confirm**. **Result:** The program is exported.

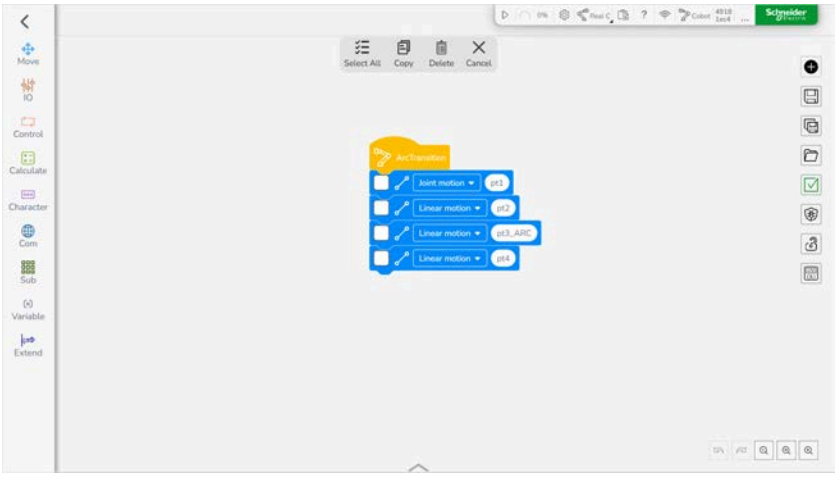
Deleting a Program

Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Open** icon on the right of the editing area. **Result:** The programs saved in the Lexium Cobot Controller are displayed.
2	In the **Programming project list** dialog box, click the **Delete** icon:  **Result:** Checkboxes for the listed programs are displayed. 
3	Select the program to be deleted. **NOTE:** Up to five programs can be selected at the same time.
4	Click **Confirm**. **Result:** The following confirmation prompt is displayed. 
5	Click **Confirm**. **Result:** The program is deleted.

Display the Advanced Operation Menu

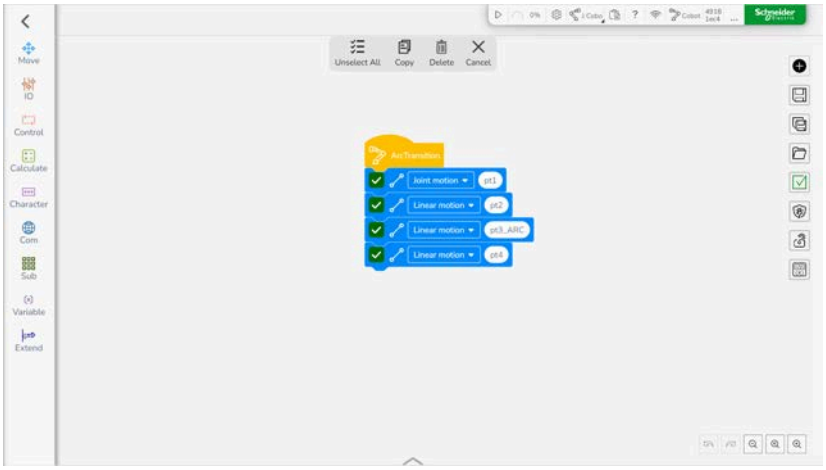
In the feature bar, select **Programming Control**. Then, click the **Advanced Operation** icon on the right of the editing area.

Result: The **Advanced Operation** menu and the check boxes of the instruction blocks are displayed.



Select All Instructions

To select all instructions in the editing area, perform the following steps:

Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Advanced Operation** icon on the right of the editing area. **Result:** The **Advanced Operation** menu and the check boxes of the instruction blocks are displayed.
2	Click the **Select All** icon. **Result:** All instructions are selected. 

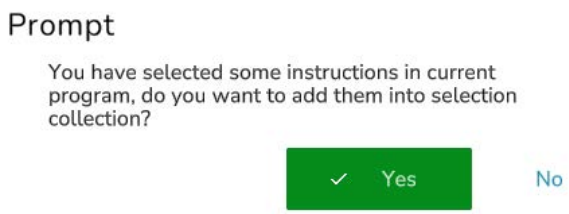
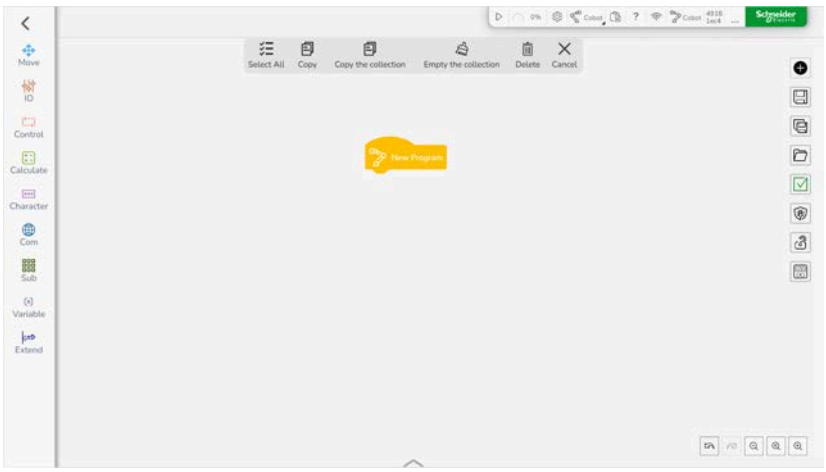
Copy Instructions

To copy selected instructions from the sequence and paste them into the editing area, perform the following steps:

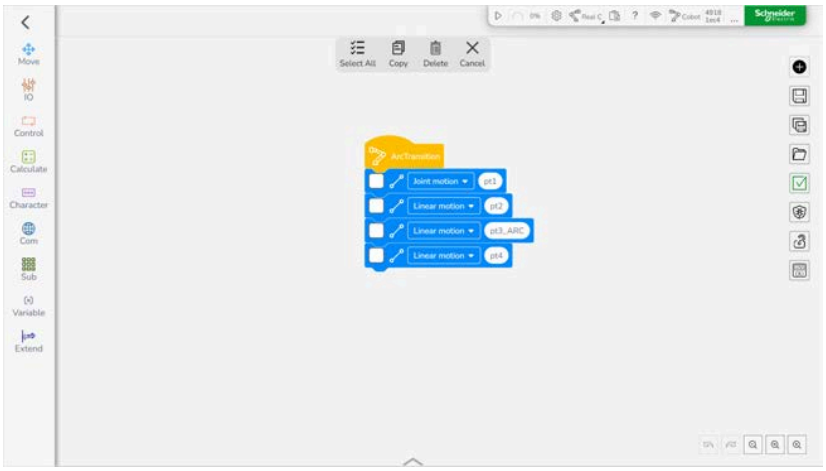
Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Advanced Operation** icon on the right of the editing area. **Result:** The **Advanced Operation** menu and the check boxes of the instruction blocks are displayed.
2	Select the instructions to be copied.
3	Click **Copy**. **Result:** The selected instructions are inserted into the editing area.

Copy Instructions Across Programs

To copy selected instructions and insert them into the editing area of another program, perform the following steps:

Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Advanced Operation** icon on the right of the editing area. **Result:** The **Advanced Operation** menu and the check boxes of the instruction blocks are displayed.
2	Select the instructions to be copied.
3	Open the program or subprogram into which you want to insert the instructions. **Result:** The following confirmation prompt is displayed. 
4	Click **Yes**. **Result:** The subprogram or program is opened and the **Advanced Operation** menu is displayed. 
5	Click **Copy the collection** to finish the cross-program copying instruction. **Result:** The selected instructions are inserted into the editing area of the program.

Delete Instructions

Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Advanced Operation** icon on the right of the editing area. **Result:** The subprogram or program is opened and the **Advanced Operation** menu is displayed. 
2	Select the instructions to be deleted.
3	Click **Delete**. **Result:** The selected instructions are deleted.

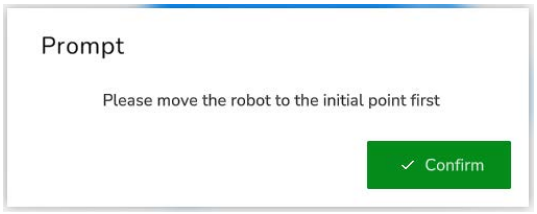
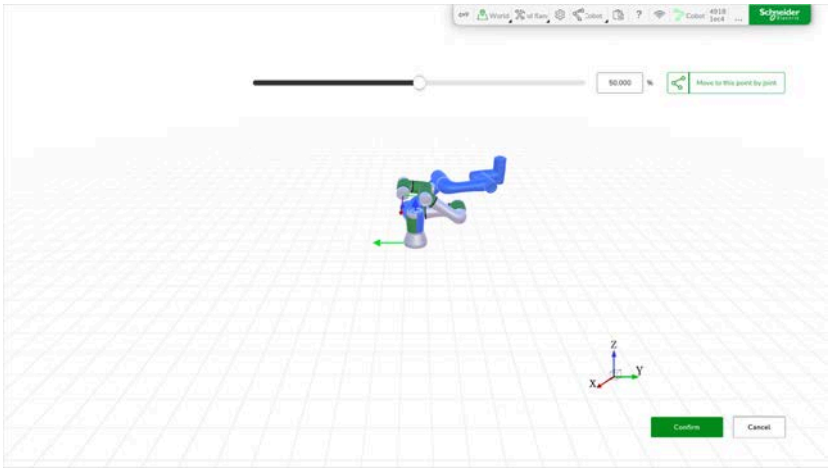
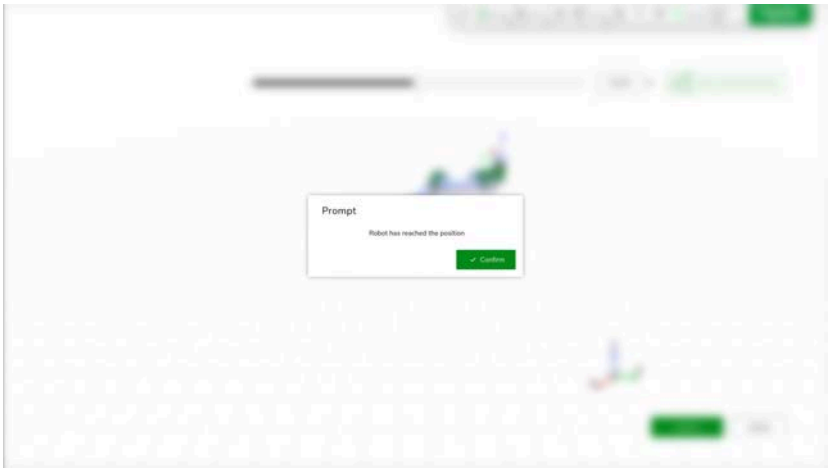
Hide the Advanced Operation Menu

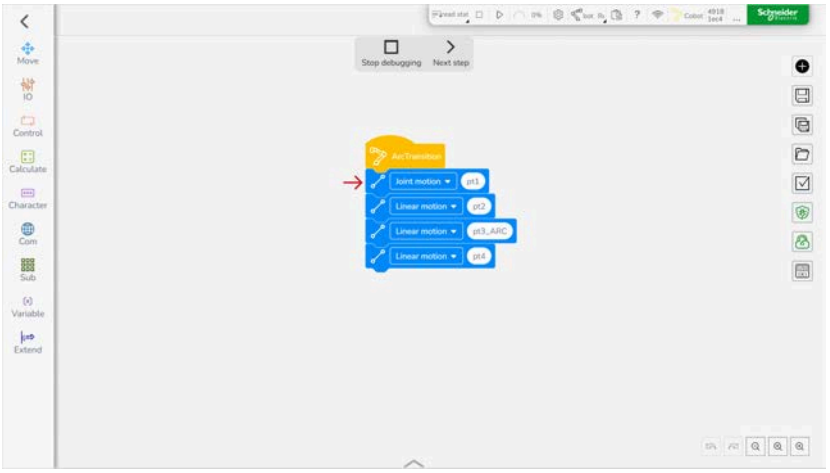
Click **Cancel** or **Advanced Operation** icon.

Result: The **Advanced Operation** menu is closed.

Single-Step Debugging of the Program

To execute the program instruction in single step, perform the following steps:

Step	Action
1	Power on and enable the Lexium Cobot Arm.
2	In the feature bar, select **Programming Control**. Then, click the **Debug** mode icon on the right of the editing area. Then click **Single-Step Debugging**. **Result:** The following prompt is displayed: 
3	Click **Confirm**. **Result:** The **Manual Operation** interface is displayed. 
4	Click and hold **Move to this point by joint** until the prompt **Robot has reached the position** is displayed. 
5	Click **Confirm** in the prompt and then **Confirm** in the **Manual Operation** interface.

Step	Action
6	Click **Single-Step Debugging**. **Result:** The Lexium Cobot starts to run the program. The red arrow on the left side of the instruction indicates the next instruction to be executed. During the single-step debugging operation, the illuminated ring of the Lexium Cobot Arm flashes yellow at the hardware and at the representation in the **Home** screen of the software. 
7	Click **Next step**. **Result:** The Lexium Cobot executes the next instruction.
8	Repeat Step 7 for the other instructions, then click **Stop Debugging** to stop running the program.

Locking the Program

In the feature bar, select **Programming Control**. Then, click the **Lock** icon on the right of the editing area so that it becomes green.

Result: The program is locked and cannot be edited.

Unlocking the Program

In the feature bar, select **Programming Control**. Then, click the **Lock** icon on the right of the editing area so that it becomes grey.

Result: The program is unlocked and can be edited.

Variable Observation

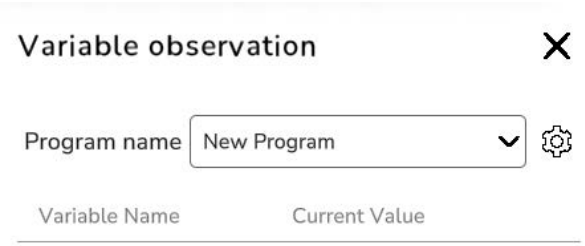
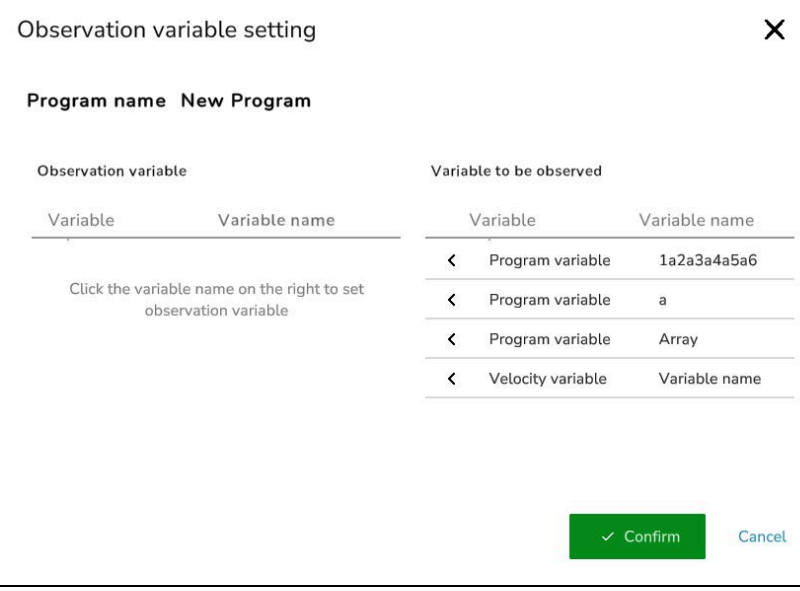
You can monitor the values of the variables in real time in the running program.

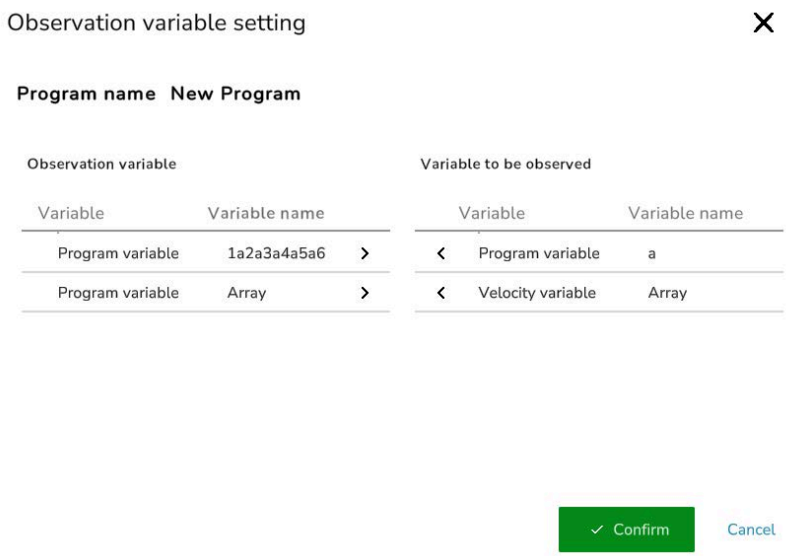
Four types of variables are available:

- System variables
- Program variables
- Speed variables
- Position variables

Monitoring a Variable

To monitor a variable, perform the following steps:

Step	Action
1	In the feature bar, select **Programming Control**. Then, click the **Variable Observation** icon on the right of the editing area:  **Result:** The **Variable observation** dialog box is displayed. 
2	Click the **Settings** button. **Result:** The **Observation variable setting** is displayed. 

Step	Action
3	In the **Variable to be observed** table, click on the variables you want to monitor. **Result:** The selected variables are displayed in the **Observation variable** table. 
4	Click **Confirm**.

Types of Instructions

How to Use Instructions

For building up the program code, EcoStruxure Cobot Expert provides colored and differently shaped program blocks called instructions. The instructions represent commands, functions, logical operations, variables, and data containers. To realize the programming, drag and drop the appropriate instructions from the instruction menu to the appropriate position in the editing area and edit the parameters of the instruction.

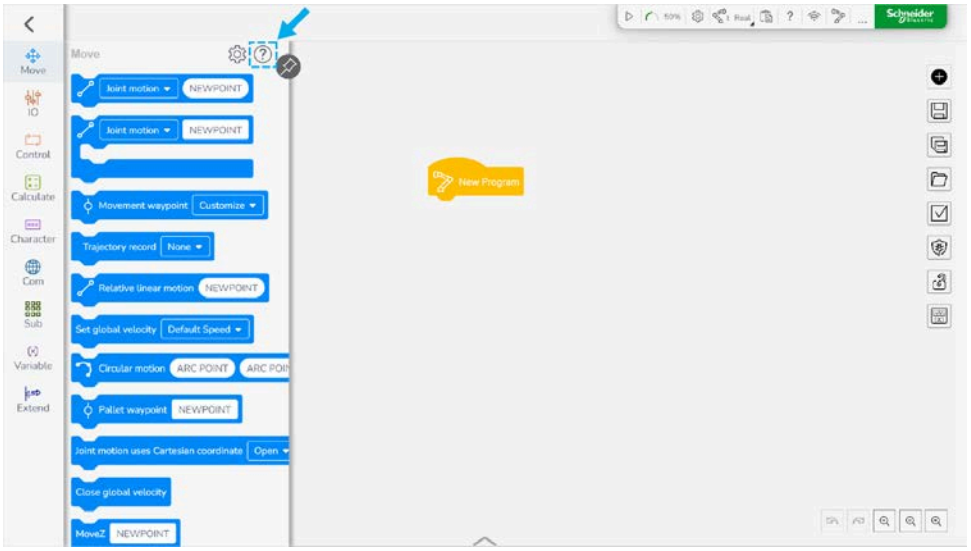
The different colors indicate the category of an instruction. Each category has its own color.

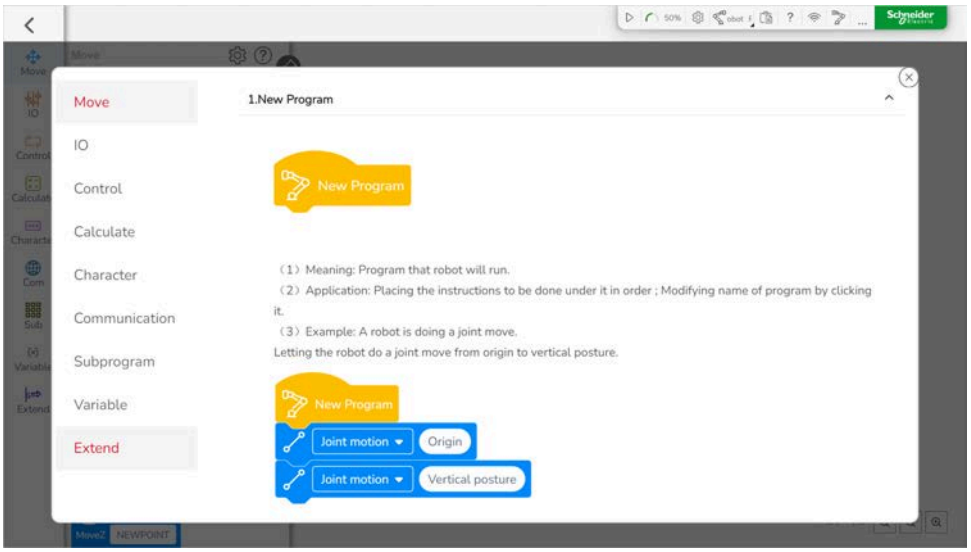
The different shapes indicate the type of an instruction and help to understand how to position the block and how the instruction works during programming.

There are three types of instructions:

- Action instructions
- Judgement instructions
- Data instructions

The following sections briefly describe these types. For detailed information on the particular instructions, refer to the help in the instructions menu of the **Programming Control** interface.

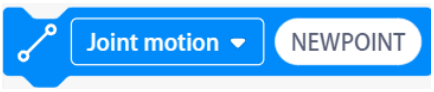


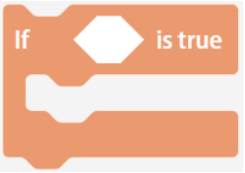


Action Instructions

Instructions that have a shape with noses and notches are action instructions.

For example, *joint motion [x]* and *if [x] is true*.





Action instructions in direct series can form a complete program.

Judgement Instructions

Instructions that have pointed sides are judgement instructions.

For example, *Digital output [x] [x] to be [x]* and *[x] less than [x]*.

The judgement instruction is placed into the judgement field of other instructions, such as *if [x] is true*, *if [x] is true, otherwise* and *wait until [x]* to state a judgement condition.

Data Instructions

Instructions with rounded sides are data instructions.

For example, *Get Analog output [x] [x]* and *Get [x]*.

The data instruction is used to capture or store data. It is placed in the data fields of instructions similar to data instruction shape.

Script Editing via Subprogram

What's in This Chapter

The Lexium Cobot programming script is a specific programming language (DSL) for controlling Lexium Cobot systems. The scripts can be implemented using the specified grammar in the programming script.

Grammar of Lexium Cobot Programming Script

Identifiers

In the Lexium Cobot programming script, identifiers are case-insensitive and their naming must meet the following rules:

- Use only English letters, numbers and underlines
- Do not use a number as the first character
- Do not use reserved keywords as identifiers

  NOTE: A maximum of 255 characters are supported, but it is a good practice not to exceed 30 characters.

Example:

```
# Correct
_var1 = 1
var2 = 1
VAR3 = 1
_2KDDinKAEld74Zl8WzKP = 1

# Incorrect
4VAR = 1
if = 1
```

Reserved Keywords

The following list presents reserved keywords in the Lexium Cobot programming script.

NOTE: Do not use these reserved words as constants, variables, or any other identifier names.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Script Comments

The Lexium Cobot programming script supports single-line comments. Single-line comments must start with # and can be placed at the end of statements or expressions.

Example:

```
# the first comment
str = "Hello, World!" # the second comment
```

Statements

The Lexium Cobot programming script only supports single-line statement. It neither supports splitting a statement across multiple lines nor incorporating multiple statements within a single line.

Data Types

The Lexium Cobot programming script supports three types of data:

- Scalar
- String
- Array

Furthermore, you can define system variables on basis of the data type array.

Scalar

The Lexium Cobot programming script does technically not distinguish between Boolean type, integer, and float.

For Boolean type, false corresponds to 0 and true corresponds to 1.

Example:

```
var = 1
#or
var = 1.0
#or
var = (expr1 > expr2)
```

String

For definition of strings, use English double quotes (" ") and the escape characters presented in the following table.

Escape character	Description
\\	Backslash
\'	Single quote
\"	Double quote
\n	Newline
\t	Horizontal tab
\r	Enter

NOTE: The supported escape characters by the Lexium Cobot programming script should be observed when defining strings, otherwise parse errors may occur.

Example:

```
string1 = "Hello World"  #Result: Hello World
string2 = "Hello \"World\""  #Result: Hello "World"
```

Array

Overview

An array is a container that holds a number of data of the same data type. Only scalar type data is supported, not string arrays or nested arrays.

Array Definition

Syntax:

```
arr = [...] # define an array
arr = []# define an empty array
```

Sub-Interval Access of Array

Access to a subsequence of a specific interval within an interval of a specific array is supported and returned in the form of an array. For accessing a sub-interval of an array, use the following syntax:

```
array[startIndx : endIndex : step]
```

If `step` is `0`, the program reports errors and the program execution is terminated. In other cases, although no errors are reported (exceeding the array limit), if `startIndex`, `endIndex` and `step` do not comply with the logical conditions, a value is returned that complies with the conditions within the effective range of the defined array. Otherwise, an empty array is returned if it does not exist.

A special syntax for accessing a subinterval with steps of 1 is provided by default, without using the step parameter:

```
array[startIndex, endIndex]
```

Example:

```
a = [1,2,3,4,5,6,7,8,9,0]
b = a[0:5]#Result: [1, 2, 3, 4, 5]
b = a[-5:10:1] #Result: [6, 7, 8, 9, 0]
```

Array and Pose Representation

In the Lexium Cobot programming script, the 6-element array is used to represent the Lexium Cobot joint position or spatial pose. The length unit is mm and the angle unit is °.

Example:

```
endPosJ = [90,90,90,90,90,90] # joint space position array
endPosL = [663.5,8.159996,6.950005,90,0,0] # Cartesian
spatial position array
```

System Variable

The variable defined in the operating program is released at the end of the program execution. For variables whose required value can be kept for a long time, the Lexium Cobot programming script provides a system variable mechanism.

The system variable can be used directly in the program and the variable value can be retained after the connection is terminated or when the value is modified in the program.

NOTE:

- System variables support only the data type scalar
- System variables do not support negative index and interval access
- A program can store up to 100 system variables

The syntax for accessing system variables is as follows:

```
sysvar[id], id∈[5500, 5599]
```

Example:

```
sysvar[5500] = 100
a = sysvar[5500]
```

Expressions

Arithmetic Operations

Arithmetic operators are used for the four arithmetic operations and are grouped according to the precedence of the operator. The precedence of (*, /, %, **) is higher than that of (+, -). Operators with higher priority combine more closely than operators with lower priority, and the operators in the following table comply with left-associativity. The operation is performed from the center to the right when the operator priority is the same.

Operator	Function	Usage
*	Multiplication	a = (b * c)
/	Division	a = (b / c)
%	Complementation	a = (b % c)
**	Exponentiation	a = (b ** c)
+	Addition	a = (b + c)
-	Subtraction	a = (b - c)

Logical and Relational Operators

The logical operator is applicable to a data type that can be arbitrarily converted to a Boolean value. The relational operator is applicable to the arithmetic operators. The type of the returned value of the logical operator and the relational operator are of the Boolean type.

Associativity	Operator	Function	Usage
Right	!	Logical negation	! expr
Left	&&	logic and	expr && expr
Left	||	Logic or	expr || exprr
Left	<	Less than	expr < expr
Left	>	More than	expr > expr
Left	==	Equal to	expr == expr
Left	!=	Not equal to	expr != expr
Left	<=	Less than or equal to	expr <= expr
Left	>=	More than or equal to	expr >= expr

Bitwise Operators

The bitwise operator is applicable to operation objects of type integer, where the operation object is considered as a collection of binary digits. It supports only the operation XOR.

Operator	Function	Usage
^	XOR	expr1 ^ expr2

Statements

Under normal conditions, statements are implemented sequentially. Usually, it is not sufficient to execute the statements in sequence. Therefore, the Lexium Cobot programming script provides a set of control flow statements that support more complex implementation control.

Simple Statement

In the Lexium Cobot programming script, statements need to be placed in a line separately and most statements do not need terminators.

Simple statement includes expression statement, function call statement, and so on.

Conditional Statement

Overview

The if statement evaluates whether a certain condition is true or not and, based on the result of the evaluation, proceeds with executing another statement. The if statement includes two modes, namely one type of statement with an else branch and the other type of statement without an else branch.

if...end Statement

The if...end statement has the following syntax:

```
if(condition):
statement
end
```

Example:

```
condition = get_digital_output(0,1)
if(condition):
endPosJ = [0,0,0,0,0,0 ]
movj(endPosJ,0,60,200,0)
end
```

if...else...end Statement

The if...else...end statement has the following syntax:

```
if(condition):
statement
else:
statement
end
```

if...elif...else...end Statement

The if...elif...else...end statement has the following syntax:

```
if(condition1):
statement
elif(condition2):
statement
else:
```

```
statement
end

condition1 = get_digital_output(0,1)
condition2 = get_digital_output(1,1)

if(condition1):
endPosJ = [0,0,0,0,0,0]
movj(endPosJ,0,60,200,0)
elif (condition2):
endPosJ = [1,2,3,4,5,6]
movj(endPosJ,0,10,50,0)
else:
endPosL = [663.5,8.159996,6.950,90,0 ,0]
movl(endPosL,0,250,250,0)
end
```

Loop Statement

In the while structure the statement is executed as long as the result of the condition evaluation is true (generally it is a statement block). The condition cannot be empty. If the first evaluation of the condition is false, the statement is not executed at all.

The while-loop statement has the following syntax:

```
while(condition):
statement
end
```

Example:

```
while(i <= 4):
endPosJ =[0,0,0,0,0,0 ]
endPosL =[663.5,8.159996,6.950005,90,0 ,0]
movl(endPosL,0,250,2 50,0)
i = (i+1)
end
```

Jump Statement

Overview

Jump statements interrupt the execution of the while statement. The Lexium Cobot programming script provides two types of jump statements, namely the break statement and the continue statement.

Break Statement

The break statement terminates the execution of a while statement and continues the execution from the first statement after these statements. In nested loops, break exits only from the loop in which it occurs.

Example:

```
while(condition1):
statement
...
if(condition2):
break
end
...
statement
end
```

Continue Statement

The continue statement ends the present iteration of the loop and immediately starts the next iteration. The continue statement occurs only within the while loop. Continue exits only from the loop in which it occurs. It interrupts the present iteration, but continues the execution of the present loop. In the while statement, this means that the value of the condition is continuously verified.

Example:

```
while(condition1):
statement
...
if(condition2):
continue
end
...
statement end
```

Motion-Related Commands

movl()

movl(target, motionType, speed, acceleration, arcTransition, abortion)

Functional Description

This function performs a linear movement.

If the parameter *motionType* is set to 0, the motion is absolute and is in user frame.

If the parameter *motionType* is set to 1, the motion is relative and is in user frame.

If the parameter *motionType* is set to 2, the motion is relative and is in tool frame.

Parameters

Input	Data type	Description
target	Array with six elements	Describes the Cartesian target [X, Y, Z, RX, RY, RZ] either as absolute or relative pose
motionType	Scalar	Defines the motion as absolute or relative • 0: absolute motion in user frame. Target position is *var_pos*. • 1: relative motion in user frame. Target position is present pose + *target* in user frame • 2: relative motion in tool frame. Target position is present pose + *target* in tool frame
speed	Scalar	TCP Speed in mm/s
acceleration	Scalar	Acceleration in mm/s^2
arcTransition	Scalar	Arc transition parameter • 0: the robot stops at the target point • 0<: the robot does not stop at the target and smoothly changes to the next motion segment
abortion	Array with three elements	Abortion condition. It is an optional parameter. Array includes three parameters: • Input source: scalar 0: Controller DI 1: Tool DI 2: Modbus DI • Input index (address): scalar • Input triggering state: scalar **NOTE:** The numeration of DI addresses starts with 0, for example, DI1 has index 0, DI2 has the index 1 and so on.

Example

```
abslinear = [150, 400, 700, -90, 0, 0]      #Defining the target
rellinear_uf = [0, 0, -200, 0, 0, 0]        #Defining the target
stop = [0,0,1]                              #Defining the abortion condition: If DI1 of the controller is set to 1, the corresponding motion segment will be aborted
```

movl(abslinear,0,500, 500, 0)	#Result: absolute joint motion to the joint position *absolute* with maximum speed 60°/s, acceleration 200°/s², with arc transition and without abortion condition.
movl(rellinear_uf,1,250,250,1, stop)	#Result: relative joint motion according to *relative* with maximum speed 120°/s, acceleration 240°/s², without arc transition and with *stop* as an abortion condition.

movj()

movj(target, motionType, speed, acceleration, arcTransition, abortion)

Functional Description

This function performs a joint movement.

If the parameter *motionType* is set to 0, the motion is absolute.

If the parameter *motionType* is set to 1, the motion is relative.

Parameters

Input	Data type	Description
target	Array with six elements	Describes the joint positions [J1, J2, J3, J4, J5, J6] either as absolute or relative position
motionType	Scalar	Default value: 0 Defines the motion as absolute or relative • 0: absolute motion. Target position is *var_pos*. • 1: relative motion. Target position is present pose + *Target*.
speed	Scalar	Default value: 60 Joint speed in °/s **NOTE:** This is a maximum speed for each joint. The actual speed is adjusted during the motion.
acceleration	Scalar	Default value: 0 Joint acceleration in °/s
arcTransition	Scalar	Default value: 0 Arc transition parameter • 0: the robot stops at the target point • 0<: the robot does not stop at the target and smoothly changes to the next motion segment
abortion	Array with 3 elements	Abortion condition. It is an optional parameter. Not chosen by default. Array includes three parameters: • Input source: scalar 0: Controller DI 1: Tool DI 2: Modbus DI • Input index (address): scalar • Input triggering state: scalar **NOTE:** The numeration of DI addresses starts with 0, for example, DI1 has index 0, DI2 has the index 1 and so on.

Example

absolute = [0,30,60,30,0,0]	#Defining the target for absolute movement
relative = [180, 60, 0,0,0,0]	#Defining the target for relative movement
stop = [0,0,1]	#Defining the abortion condition: If DI1 of the controller is set to 1, the corresponding motion segment will be aborted
movj(absolute,0,60,200,10)	#Result: absolute joint motion to the joint position *absolute* with maximum speed 60°/s, acceleration 200°/s², with *ArcTransition* and without abortion condition
movj(relative, 1, 120, 240, 0, stop)	#Result: relative joint motion according to *relative* with maximum speed 120°/s, acceleration 240°/s², without *ArcTransition* and with *stop* as an abortion condition

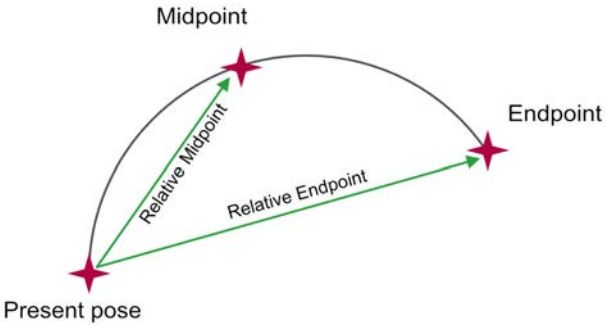
movc()

movc(midpoint, endpoint, motionType, speed, acceleration, arcTransition, counter, abortion)

Functional Description

This function performs a circular movement. The circle is defined by three points:

- Present pose
- Midpoint
- Endpoint



Parameters

Input	Data type	Description
midpoint	Array with six elements	Describes the middle point [X, Y, Z, RX, RY, RZ] either as absolute or relative pose
endpoint	Array with six elements	Describes the end point [X, Y, Z, RX, RY, RZ] either as absolute or relative pose (relative to the first point)
motionType	Scalar	Value range: 0,1 Defines the motion as absolute or relative • 0: absolute motion. Target position is *var_pos*. • 1: relative motion. Target position for middle point is present pose + *Midpoint*, the target position for end point is present pose + *Endpoint*.
speed	Scalar	Joint speed in °/s
acceleration	Scalar	Joint acceleration in °/s²
arcTransition	Scalar	Arc transition parameter • 0: the robot stops at the target point. • 0<: the robot does not stop at the target and smoothly changes to the next motion segment.

Input	Data type	Description
counter	Scalar	Defines number of circles to be performed: • 0: motion only up to the endpoint • >0: defines number of circles
abortion	Array with 3 elements	Abortion condition. It is an optional parameter. Array includes 3 parameters: • Input source: scalar 0: Controller DI 1: Tool DI 2: Modbus DI • Input index (address): scalar • Input triggering state: scalar **NOTE:** The numeration of DI addresses starts with 0, for example, DI1 has index 0, DI2 has the index 1 and so on.

Example

```
stop = [0,0,1]                                  #Defining the abortion condition: If DI1 of the controller is set to 1, the corresponding motion segment will be aborted

midpointCart = [-400, 100, 550, -90, 0, 90 ]

endpointCart = [-400, 50, 600, -90, 0, 90]      #Defining midpoint and endpoint for absolute motion

midpointRel = [0, 50, 50, 0, 0, 0]

endpointRel = [0, 0, 100, 0, 0, 0]              #Defining midpoint and endpoint for relative motion

movc(midpointCart, endpointCart, 0, 120, 250, 0, 1.5, stop)   #Result: circle motion with Cartesian definition of the points with maximum speed 120 mm/s, acceleration 240 mm/s², without ArcTransition, 1.5 circles performed and with stop as an abortion condition

movc(midpointRel, endpointRel, 1, 60, 120, 0, 0)              #Result: circle motion with relative definition of the points with maximum speed 60 mm/s, acceleration 120 mm/s², without ArcTransition, motion performed only up to endpoint and without an abortion condition
```

get_atl_joint_pose()

… = get_atl_joint_pose()

Functional Description

This function obtains the current joint position of the Lexium Cobot Arm. The returned value is an array with six elements representing the joint positions [J1, J2, J3, J4, J5, J6].

Example

```
jointPose = get_atl_joint_pose()    #Result: joint position will be saved in variable JointPose.
```

get_atl_TCP_pose()

… = get_atl_TCP_pose()

Functional Description

This function obtains the current TCP pose of the Lexium Cobot Arm. The returned value is an array with six elements representing the Cartesian pose [X, Y, Z, RX, RY, RZ].

Example

```
TCPPose = get_atl_TCP_pose()    #Result: TCP position will be saved in variable TCPPose.
```

get_atl_flange_pose()

… = get_atl_flange_pose()

Functional Description

This function obtains the end flange position of the Lexium Cobot Arm. The returned value is an array with six elements representing the Cartesian pose [X, Y, Z, RX, RY, RZ].

Example

```
endFlangePose = get_atl_flange_pose()    #Result: end flange position will be saved in variable EndFlangePose.
```

enable_speed_override()

enable_speed_override(motionType, speed, acceleration)

Functional Description

This function sets the speed and acceleration limit. The limit can be removed by function *disable_speed_override()*.

Parameters

Input	Data type	Description
motionType	Scalar	Defines the motion type which is limited • 0: Cartesian motion. The speed and acceleration units are mm/s and mm/s². If Cartesian is chosen as motion type, the function will have no impact on joint motion • 1: joint motion. The speed and acceleration units are °/s and °/s². If joint motion is chosen as motion type, the function will have no impact on any Cartesian motion (linear, circle)
speed	Scalar	Defines the speed limit.
acceleration	Scalar	Defines the acceleration limit.

Example

enable_speed_override(0, 150, 300)	#Result: the linear motion speed is limited to 150 mm/s, acceleration to 300 mm/s^2.

disable_speed_override()

disable_speed_override(motionType)

Functional Description

This function removes the speed and acceleration limitsset by the *enable_speed_override()* function.

Parameters

Input	Data type	Description
motionType	Scalar	Defines the motion type for which the limit is removed. • 0: Cartesian motion • 1: joint motion **NOTE:** Only the limit for the selected motion type is removed.

Example

enable_speed_override(0, 150, 300)	#Result: the linear motion speed is limited to 150 mm/s, acceleration to 300 mm/s^2
disable_speed_override(0)	#Result: previously defined speed and acceleration limit for linear motion is removed.

I/O Control

set_digital_output()

set_digital_output(source, index, state, immediate)

Functional Description

This function sets the digital output signal.

Parameters

Input	Data type	Description
source	Scalar	Defines the source of the DO: • 0: Cabinet IO • 1: Tool IO • 2: Extended IO • 3: reserved • 4: Modbus IO • 5: Profinet IO • 6: EtherNet/IP IO
index	Scalar	Index of the controlled digital output. **NOTE:** The numeration of DO addresses starts with 0, for example, DO1 has index 0, DO2 has the index 1 and so on.
state	Scalar	State of the digital output: • 0: Off • 1: On
immediate	Scalar	Defines whether the command is executed immediately or before the next motion command. • 0: non-immediate (postponed) command • 1: immediate command

Example

set_digital_output(0,1,1,0)	#Result: DO2 of the controller is set to On before next motion.
set_digital_output(1,0,0,1)	#Result: DO1 of the Tool IOs is set to Off immediately

set_analog_output()

set_analog_output(source, index, state, immediate)

Functional Description

This command is used to control analog output signal. The output can be used either as voltage or current output. The output must be configured in the EcoStruxure Cobot Expert. For detailed information, refer to Setting an Analog Output Signal, page 154.

Parameters

Input	Data type	Description
source	Scalar	Defines the source of the AO: • 0: Cabinet IO • 1: Tool IO • 2: Extended IO • 3: reserved • 4: Modbus IO • 5: Profinet IO • 6: EtherNet/IP IO
index	Scalar	Index of the controlled analog output. **NOTE:** The numeration of AO addresses starts with 0, for example, AO1 has index 0, AO2 has the index 1 and so on.
numeric value	Scalar	Defines the output value. The value represents the percentage of the maximum range (10 V or 20 mA).
immediate	Scalar	Defines whether the command is executed immediately or before the next motion command. • 0: non-immediate (postponed) command • 1: immediate command

Example

set_analog_output(0,0,50,0)	#Result: AO1 of the cabinet is set to 50 % of the range (5 V or 10 mA depending on the configuration before the next motion command).
set_analog_output(1,0,0,1)	#Result: AO2 of the Tool end is set to 100 % of the range (10 V or 20 mA depending on the configuration) immediately.

get_digital_output()

… = get_digital_output(source, index)

Functional Description

This function gets the state of a digital output. The range of the returned value is [0,1]:

- 0: Off
- 1: On

Parameters

Input	Data type	Description
source	Scalar	Defines the source of the DO: • 0: Cabinet IO • 1: Tool IO • 2: Extended IO • 3: reserved • 4: Modbus IO • 5: Profinet IO • 6: EtherNet/IP IO
index	Scalar	Index of the controlled digital output. **NOTE:** The numeration of DO addresses starts with 0, for example, DO1 has index 0, DO2 has the index 1, and so on.

Example

```
DOstate = get_digital_output (0,0)   #Result: State of DO1 of the controller is saved in variable DOstate.
```

get_analog_output()

… = get_analog_output(source, index)

Functional Description

This function gets the value of an analog output. The value represents the percentage of the maximum range (10 V dc or 20 mA). The output can be used either as voltage or current output. The output must be configured in EcoStruxure Cobot Expert. For detailed information, refer to Setting an Analog Output Signal, page 154.

Parameters

Input	Data type	Description
source	Scalar	Defines the source of the AO: • 0: Cabinet IO • 1: Tool IO • 2: Extended IO • 3: reserved • 4: Modbus IO • 5: Profinet IO • 6: EtherNet/IP IO
index	Scalar	Index of the controlled analog output. **NOTE:** The numeration of AO addresses starts with 0, for example, AO1 has index 0, AO2 has the index 1, and so on.

Example

```
AOstate = get_analog_output (0,0)   #Result: value of AO1 of the controller is saved in variable AOstate.
```

get_digital_input()

… = get_digital_input(source, index)

Functional Description

This function gets the state of a digital input. The range of the returned value is [0,1]:

- 0: Off
- 1: On

Parameters

Input	Data type	Description
source	Scalar	Defines the source of the DI: • 0: Cabinet IO • 1: Tool IO • 2: Extended IO • 3: reserved • 4: Modbus IO • 5: Profinet IO • 6: EtherNet/IP IO
index	Scalar	Index of the controlled digital input. **NOTE:** The numeration of DI addresses starts with 0, for example, DI1 has index 0, DI2 has the index 1, and so on.

Example

```
DIstate = get_digital_input(0,0)    #Result: state of DI1 of the controller is saved in variable DIstate.
```

get_analog_input()

… = get_analog_input(source, index)

Functional Description

This function gets the value of an analog input. The value represents the percentage of the maximum range (10 V dc or 20 mA). The input can be used either as voltage or current input. The input must be configured in EcoStruxure Cobot Expert. For detailed information, refer to Setting an Analog Input Signal, page 153.

Parameters

Input	Data type	Description
source	Scalar	Defines the source of the AI: • 0: Cabinet IO • 1: Tool IO • 2: Extended IO • 3: reserved • 4: Modbus IO • 5: Profinet IO • 6: EtherNet/IP IO
index	Scalar	Index of the controlled analog input. **NOTE:** The numeration of AI addresses starts with 0, for example, AI1 has index 0, AI2 has the index 1, and so on.

Example

```
AIstate = get_analog_output (0,0)    #Result: value of AI1 of the controller is saved in variable AIstate.
```

wait_input()

wait_input(source, index, expectedValue, time)

Functional Description

This function monitors for an expected value of a digital input signal. Feedback of this function can be acquired with *get_timeout* function. If the expected value is detected, the next command is executed.

Parameters

Input	Data type	Description
source	Scalar	Defines the source of the AI: • 0: Cabinet IO • 1: Tool IO • 2: Extended IO • 3: reserved • 4: Modbus IO • 5: Profinet IO • 6: EtherNet/IP IO
index	Scalar	Index of the controlled input. **NOTE:** The numeration of AI addresses starts with 0, for example, AI1 has index 0, AI2 has the index 1 and so on.
expectedValue	Scalar	Defines which value is expected. If the expected value is detected, the next command is executed and function *get_timeout()* returns value 1.
time	scalar	The maximum waiting time in [s]. If the set limit is expired without input signal, timeout mark is set and function *get_timeout()* returns value 1. The next command is executed. If 0 is set as value, there is no time limit.

Example

wait_input(0,0,1,0)	#Result: monitoring for value 1 on DI1 of the controller without time limit. No further command execution until DI1 has value 1.

get_timeout()

… = get_timeout()

Functional Description

This function obtains the feedback of the *wait_input()* command. The returned value can be either:

- 0: no timeout, the signal was detected within expected time
- 1: timeout, no signal detected within expected time

Example

wait_input(0,0,1,10)	#Result: monitoring for value 1 on DI1 of the controller for 10 seconds.
timeOut = get_timeout	#Result: the feedback value is saved in variable *tmeOut*. 1 if no signal in *wait_input* within 10 second, 0 if signal detected within 10 seconds

Parameter Setting

set_payload()

set_payload(mass, centerOfMass)

Functional Description

This function sets the payload settings of the Lexium Cobot Arm.

Parameters

Input	Data type	Description
mass	Scalar	Defines the mass of the load in kg
centerOfMass	Array with 3 elements	Describes the offset of the center of mass in the End Flange Coordinate System [X, Y, Z] in mm

Example

```
CoM = [10,20,30]          #Defining centerOfMass parameter in the variable CoM.
set_payload(3.5, CoM)     #Result: payload settings changed to 3.5kg with an offset defined in CoM.
```

get_payload()

… = get_payload()

Functional Description

This function obtains the payload settings of the Lexium Cobot Arm. The returned value is an array with four elements:

- [0]: load mass in kg
- [1]-[3]: the offset of the center of mass in the End Flange Coordinate System [X, Y, Z] in mm

Example

```
payload = get_payload()   #Result: payload settings are saved in variable payload.
```

get_collision_level()

… = get_collision_level()

Functional Description

This function obtains the collision level settings of the Lexium Cobot Arm. The returned value is a scalar with value range [0…5]:

- 0: unlimited mode
- 1-5: collision levels

Example

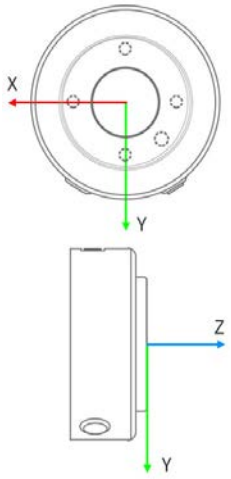
```
colLevel = get_collision_level()    #Result: collision level is saved in variable colLevel.
```

set_tool()

set_tool(offset)

Functional Description

This function sets the offset of the tool end relative to the end flange.



Parameters

Input	Data type	Description
offset	Array with six elements	Defines the offset [X, Y, Z, RX, RY, RZ] relative to the end flange.

Example

```
tool1= [10,20,30,40,50,60]    #Defining offset parameters in the variable tool1.
set_tool(tool1)               #Result: offset of the tool end is set according to parameters defined in variable tool1.
```

set_tool_id()

set_tool_id(id)

Functional Description

This function sets one of the pre-defined TCP settings.

Parameters

Input	Data type	Description
id	Scalar	Selects the pre-defined TCP settings from the TCP list in EcoStruxure Cobot Expert.

Example

```
set_tool_id(1)          #Result: offset of the tool end is set according to the first pre-defined TCP setting.
```

get_tool_offsets()

… = get_tool_offsets()

Functional Description

This function obtains the offset of the tool end relative to the end flange. The returned value is an array with six scalar elements [X, Y, Z, RX, RY, RZ].

Example

```
offset = get_tool_offsets()          #Result: offset of the tool end is saved in variable offset.
```

get_tool_offsets_of()

… = get_tool_offsets_of(id)

Functional Description

This function obtains the offset values of the designated tool end relative to the end flange. The returned value is an array with six scalar elements [X, Y, Z, RX, RY, RZ].

Parameters

Input	Data type	Description
id	Scalar	Represents the designated pre-defined TCP setting.

Example

```
offset = get_tool_offsets_of(2)          #Result: offset values of second pre-defined TCP setting are saved in variable offset.
```

set_user_frame()

set_user_frame(userFrame)

Functional Description

This function sets the user frame relative to the world frame.

Parameters

Input	Data type	Description
userFrame	Array with six elements	Defines the offset of user frame [X, Y, Z, RX, RY, RZ] relative to the world frame.

Example

```
userFrame1= [10,20,30,40,50,60]    #Defining offset parameters in the variable userFrame1.
set_user_frame(userFrame1)         #Result: offset of the user frame is set according to parameters
                                   defined in variable userFrame1.
```

set_user_frame_id()

set_user_frame_id(id)

Functional Description

This function sets one of the pre-defined user frames.

Parameters

Input	Data type	Description
id	Scalar	Chooses the pre-defined user frame from the list in EcoStruxure Cobot Expert.

Example

```
set_user_frame_id(1)               #Result: the first pre-defined user frame is set
```

get_user_frame()

… = get_user_frame()

Functional Description

This function obtains the offset of the frame. The returned value is an array with six scalar elements [X, Y, Z, RX, RY, RZ].

Example

```
userFrame = get_user_frame()       #Result: offset of the frame is saved in variable userFrame.
```

get_user_frame_of()

… = get_user_frame_of(id)

Functional Description

This function obtains the offset values of a pre-defined user frame. The returned value is an array with six scalar elements [X, Y, Z, RX, RY, RZ].

Parameters

Input	Data type	Description
id	Scalar	Represents the designated pre-defined user frame.

Example

```
userFrame1 = get_user_frame_of(2)    #Result: offset values of second pre-defined user frame are saved in variable userFrame1.
```

Pose Calculation

pose_add()

… = pose_add(pos1, pos2)

Functional Description

This function calculates the addition of two poses. The resulting pose is calculated as follows:

- res.P = pos1.P + pos2.P
- res.R = pos2.R * pos1.R

The returned value is an array with six scalar elements [X, Y, Z, RX, RY, RZ].

Parameters

Input	Data type	Description
pos1, *pos2*	Array with six elements	Defines the two poses [X, Y, Z, RX, RY, RZ] to be added.

pose_sub()

… = pose_sub(pos1, pos2)

Functional Description

This function calculates the subtraction of two poses. The resulting pose is calculated as follows:

- res.P = pos1.P - pos2.P
- res.R = inv(pos2.R) * pos1.R

The returned value is an array with six scalar elements [X, Y, Z, RX, RY, RZ].

Parameters

Input	Data type	Description
pos1, *pos2*	Array with six elements	Defines the two poses [X, Y, Z, RX, RY, RZ] to be subtracted.

pose_dist()

… = pose_dist(pos1, pos2)

Functional Description

This function calculates the distance between two poses. Only the position coordinate is considered. The returned value is a scalar.

Parameters

Input	Data type	Description
pos1, *pos2*	Array with six elements	Defines the two poses [X, Y, Z, RX, RY, RZ].

kine_inverse()

… = kine_inverse(posJ, posC)

Functional Description

This function calculates the inverse kinematic solution. It calculates the joint position [J1, J2, J3, J4, J5, J6] corresponding to Cartesian position posC [X, Y, Z, RX, RY, RZ].

The returned value is an array with six scalar elements [J1, J2, J3, J4, J5, J6].

Parameters

Input	Data type	Description
posJ	Array with six elements	Defines the joint position [J1, J2, J3, J4, J5, J6] near the calculated values to confirm the selection of calculated position.
posC	Array with six elements	Defines the Cartesian pose [X, Y, Z, RX, RY, RZ].

kine_forward()

… = kine_forward(posJ)

Functional Description

This function calculates the forward kinematic solution. It calculates the Cartesian position [X, Y, Z, RX, RY, RZ] corresponding to joint position [J1, J2, J3, J4, J5, J6].

The returned value is an array with six scalar elements [X, Y, Z, RX, RY, RZ].

Parameters

Input	Data type	Description
posJ	Array with six elements	Defines the joint position [J1, J2, J3, J4, J5, J6].

Auxiliary Function Library

Mathematical Calculations Library

Function	Description
res = atan2(y,x)	Arc-tangent function, which will return the arc-tangent of value y/x, in degree.
res = abs(arg)	Find the absolute value of expression
res = acos(arg)	Arc-cosine function, in degree
res = asin(arg)	Arc-sine function, in degree
res = cos(arg)	Cosine function
res = sin(arg)	Sine function
res = tan(arg)	Tangent function
res = floor(arg)	Round down to an integer
res = ceil(arg)	Round up to an integer
res = round(arg)	Round off
res = sqrt(arg)	Take the square root
res = rad2deg(arg)	Radians to degrees
res = deg2rad(arg)	Degrees to radians

String Operations

string_concat()

… = string_concat(str1, str2)

Functional Description

This function concatenates two strings.

The returned value is a new string.

Parameters

Input	Data type	Description
str1	String	Defines the first string to be concatenated.
str2	String	Defines the second string to be concatenated.

Example

```
str1 = "hello,"                 #defines the first string str1.
str2 = "world"                  #defines the first string str2.
str3 = string_concat(str1, str2)  #the result of concatenation is saved in variable str3.
```

get_string_from_array()

… = get_string_from_array(arr, sep, str)

Functional Description

This function converts an array into a string.

The returned value is a scalar representing the length of the string.

Parameters

Input	Data type	Description
arr	Array	Defines the array.
sep	String	Defines the separator for the string representation.
str	String	Defines the variable where the result is saved.

Example

```
arr = [1,2,3,4,5]               #defines the array arr.
sep = "; "                      #defines the separator "; "
```

str = ""	#defines the string *str* where the result is saved.
strLen = get_string_from_array (arr, sep, str)	#Result: string “1; 2; 3; 4; 5” is saved in variable *str*. The returned value 13 is saved in variable *strLen*.

get_array_from_string()

… = get_array_from_string(str, sep, arr)

Functional Description

This function converts a string into an array.

The returned value is a scalar representing the number of elements in the array.

Parameters

Input	Data type	Description
str	String	Defines the variable where the result is saved.
sep	String	Defines the separator for the string representation.
arr	Array	Defines the array.

Example

str = "1,2,3,4,5"	#defines the string *str*.
arr = [1,0,0,0,0]	#defines the array *arr* where the result is saved.
sep = ","	#defines the separator “,”
str = ""	#defines the string *str* where the result is saved.
resNum = get_array_from_string (str, sep, arr)	#Result: array [1,2,3,4,5] is saved in variable *arr*. The returned value 5 is saved in variable *resNum*.

get_length()

… = get_length(str_arr)

Functional Description

This function obtains the length of a string or a number of elements in an array.

The returned value is a scalar.

Parameters

Input	Data type	Description
str_arr	String or array	Defines the string or array.

Example

```
str_arr = "1,2,3,4,5"          #defines the string str_arr .
length = get_length (str_arr)  #Result: length of the variable str_arr is saved in variable length.
```

strcmp()

… = strcmp(str1, str2)

Functional Description

This function compares two strings.

The returned value is a scalar that represents a Boolean value:

- 0 represents TRUE – strings are equal
- Else represents FALSE – strings are different

Parameters

Input	Data type	Description
str1, *str2*	String	Define the strings to be compared.

Example

```
str1 = "1,2,3,4,5"          #defines the string str1.
str2 = "1,2,3,4,5"          #defines the string str2.
cmpRes = strcmp (str1, str2)  #Result: comparison result is saved in variable cmpRes.
```

Program Control and Debugging

log_message()

log_message(level, message)

Functional Description

This function adds new log information that can be displayed in the log.

Parameters

Input	Data type	Description
level	Scalar	Log message type: • 1 – **Info** • 2 – **Warning** • 3 – **Error**
message	String or scalar	Log message text

Example

```
log_message (1, “Hello”)        #Log information of type Info with the text "Hello" is displayed in the log.
```

get_system_clock()

… = get_system_clock()

Functional Description

This function obtains clock information from the system. The time is reset when the controller is rebooted.

The returned value is a scalar representing ms passed after last restart.

sleep()

sleep(time)

Functional Description

This function delays for a period of time.

Parameters

Input	Data type	Description
time	Scalar	Defines the delay time in [s]

pause()

pause()

Functional Description

This function pauses the program.

exit()

exit()

Functional Description

This function stops the program.

Network Communication

socket_open()

… = socket_open(ip, port, tlsEnabled)

Functional Description

This function opens the specified IP and port number, stores the created SOCKET handle in a variable and returns.

NOTE: A TCP server is required to implement the TLS encryption.

Parameters

Input	Data type	Description
ip	String	Represents the TCP server address in string format, for example, “192.168.1.10”.
port	Scalar	Represents the TCP server port number.
tlsEnabled	Scalar	Enable the TLS option for the socket, 0 for disabled and 1 for enabled.

socket_close()

… = socket_close(ip, port, tlsEnabled)

Functional Description

This function opens the specified IP and port number, stores the created SOCKET handle in a variable and returns.

NOTE: A TCP server is required to implement the TLS encryption.

Parameters

Input	Data type	Description
ip	String	Represents the TCP server address in string format, for example, “192.168.1.10”.
port	Scalar	Represents the TCP server port number.
tlsEnabled	Scalar	Enable the TLS option for the socket, 0 for disabled and 1 for enabled.

socket_get_var()

… = socket_get_var(sockid, type, argname)

Functional Description

This function requests the setting of server parameters. Returned value type depends on parameter type.

The function sends the string *get <argname>* through the socket and the data form *<argname><value>* is expected to be received.

There are 2 seconds timeout and it returns 0 after timeout.

When an array is expected, the sending form from server is: *<arrName><[num1, num2, ..., numN]>*

When a string is expected, sending form from server is: *<strName><"stringValue">*

Parameters

Input	Data type	Description
sockid	Scalar	Represents the socket ID and must be created first.
type	Scalar	Represents the parameter type: • 0: integer • 1: floating point number • 2: string
argname	String	Represents the variable name to be obtained as string, for example, "argname".

socket_read_real()

… = socket_read_real(sockid, num)

Functional Description

This function obtains an array of real values from the server to be stored in a returned variable of type scalar.

There is a two-second timeout. If exceeded, it returns 0.

Function sending format is *get#real#num#* and expected receiving data format is *< [num1, num2, ..., numN]>*

Parameters

Input	Data type	Description
sockid	Scalar	Represents the socket ID and must be created first.
num	Scalar	Represents the number of values expected to be received.

socket_read_string()

… = socket_read_string(sockid, prefix, suffix)

Functional Description

This function obtains a string from the server and stores it in the returned variable of type string.

There is a two-second timeout. If exceeded, it returns 0.

Function is sent in the form of *get#string#prefix#suffix#* and expected data receiving form is *"prefixSTRINGsuffix"*.

Parameters

Input	Data type	Description
sockid	Scalar	Represents the socket ID and must be created first.
prefix	Scalar	Prefix requirements for the string expected to be received.
suffix	Scalar	Suffix requirements for the string expected to be received.

socket_send()

… = socket_send(sockid, var)

Functional Description

This function sends a variable value in string format through the specified socket. The data is sent in the following form:

- Number: 123.4
- Number array: 11, 22, 33
- String: “string”

The returned value is a scalar representing the send result:

- 1: successful
- 2: unsuccessful

Parameters

Input	Data type	Description
sockid	Scalar	Represents the socket ID and must be created first.
var	Scalar / array / string	Represents the variable to be sent.

socket_recv()

… = socket_recv(sockid, timeout)

Functional Description

This function receives data from the server with a defined timeout.

NOTE: The function only receives data and does not send a request to the server.

If data is not received in time, it returns an empty string.

If data is successfully received, the function returns the received string.

Parameters

Input	Data type	Description
sockid	Scalar	Represents the socket ID and must be created first.
timeout	Scalar	Represents the receive timeout setting in [s].

TCP/IP Protocol Communication

What's in This Chapter

TCP/IP Communication Control Commands

This chapter describes the TCP/IP communication control commands used in and released with the LexiumCobotCommunication library for EcoStruxure Machine Expert.

The string commands are in JSON format.

NOTE:

- Implement and connect the TCP/IP client to the Lexium Cobot Controller on port 10001.
- Use the Ethernet connector CN13 of the Lexium Cobot Controller or CN26 of the Lexium Cobot Compact Controller to establish the connection.
- Configure the network settings using EcoStruxure Cobot Expert. For further information, refer to Network Settings, page 68.
- You can send a new command only after the receive message is received by the TCP/IP client.
- The control source in EcoStruxure Cobot Expert must be delegated to **Remote Control**. For further information, refer to Delegating the Control of the Lexium Cobot, page 50.

Get Control Source (get_control_source)

Task

This command retrieves the source controlling the Lexium Cobot.

Send Message

```
{"cmdName":"get_control_source"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*get_control_source*	Name of the command to send.

Receive Message

```
{"errorCode":"0","errorMsg":"","control_source":
<INT>,"cmdName":"get_control_source"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
control_source	INT	*1*: Stick *2*: App *3*: Remote	Source controlling the Lexium Cobot system.
cmdName	STRING	*get_control_source*	Name of the command sent.

Power On the Lexium Cobot Arm (power_on)

Task

This command energizes the Lexium Cobot Arm.

Send Message

```
{"cmdName":"power_on"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*power_on*	Name of the command to send.

Receive Message

```
{"errorCode":"0","errorMsg":"","power
status":"True","cmdName":"power_on"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
power status	STRING	*True*	Power status of the Lexium Cobot Arm.
cmdName	STRING	*power_on*	Name of the command sent.

Power Off the Lexium Cobot Arm (power_off)

Task

This command de-energizes the Lexium Cobot Arm.

Send Message

```
{"cmdName":"power_off"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*power_off*	Name of the command to send.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"power_off"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*power_off*	Name of the command sent.

Enable the Lexium Cobot Arm (enable_robot)

Task

This command enables the Lexium Cobot Arm.

Send Message

```
{"cmdName":"enable_robot"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*enable_robot*	Name of the command to send.

Receive Message

```
{"errorCode":"0","errorMsg":"","enabled
status":"True","cmdName":"enable_robot"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
enabled status	STRING	*True*	Enabled status of the Lexium Cobot Arm.
cmdName	STRING	*enable_robot*	Name of the command sent.

Disable the Lexium Cobot Arm (disable_robot)

Task

This command disables the Lexium Cobot Arm.

Send Message

```
{"cmdName":"disable_robot"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*disable_robot*	Name of the command to send.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"disable_robot"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*disable_robot*	Name of the command sent.

Joint Movement with Joint Position (moveJ)

Task

This command triggers a joint movement of the Lexium Cobot to user-defined target joint positions.

Send Message

```
{
"cmdName":"moveJ",
"relFlag":<INT>,
"jointPosition":[<ARRAY [0..5] OF REAL>],
"speed":<REAL>,
"acc":<REAL>,
"arc_transition":<REAL>,
"command_id":<INT>
}
```

Parameter	Data type	Value	Description
cmdName	STRING	moveJ	Name of the command to send.
relFlag	INT	*0*	Set to *0*. jointPosition is interpreted as absolute joint positions.
jointPosition	ARRAY [0..5] OF REAL	User defined value	Target angle value in degree of each joint. [j1, j2, j3, j4, j5, j6]
speed	REAL	User defined value	Value represents the maximum speed of the joints in °/s. Value must be greater than *0.0*.
acc	REAL	User defined value	Value represents the maximum acceleration of the joints in °/s². Value must be greater than *0.0*.
arc_transition	REAL	User defined value	When the value is not equal to *0.0*, the connection between two motion segments is blended if possible. Value must be greater than or equal to *0.0*.
command_id	INT	User defined value	Identifier of the motion command. Value must be greater than or equal to *0*.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"moveJ"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*moveJ*	Name of the command sent.

Joint Movement with Cartesian Position (moveTCP)

Task

This command triggers a joint movement of the Lexium Cobot to a user-defined target position.

This command does not move in a straight line from the present position to the target position. This command first performs inverse kinematics to the defined target point of Cartesian position and then performs the moveJ command to move to the specified position.

Send Message

```
{
"cmdName":"moveTCP",
"cartPosition":[<ARRAY [0..5] OF REAL>],
"speed":<REAL>,
"acc":<REAL>,
"arc_transition":<REAL>,
"command_id":<INT>
}
```

Parameter	Data type	Value	Description
cmdName	STRING	*moveTCP*	Name of the command to send.
relFlag	INT	*0*	Set to *0*. *cartPosition* is interpreted as absolute Cartesian pose of the TCP.
cartPosition	ARRAY [0..5] OF REAL	User defined value	Target position of the robot TCP. Target position value in mm for x, y and z. Target orientation value in degree for rx, ry and rz. [x, y, z, rx, ry, rz]
speed	REAL	User defined value	Value represents the maximum speed of the joints in °/s. Value must be greater than *0.0*.
acc	REAL	User defined value	Value represents the maximum acceleration of the joints in °/s². Value must be greater than *0.0*.
arc_transition	REAL	User defined value	When the value is not equal to *0.0*, the connection between two motion segments is blended if possible. Value must be greater than or equal to *0.0*.
command_id	INT	User defined value	Identifier of the motion command. Value must be greater than or equal to *0*.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"moveTCP"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*moveTCP*	Name of the command sent.

Linear Movement (moveL)

Task

This command triggers a linear movement of the Lexium Cobot to a user-defined target position.

Send Message

```
{
"cmdName":"moveL",
"relFlag":<INT>,
"cartPosition":[<ARRAY [0..5] OF REAL>],
"speed":<REAL>,
"acc":<REAL>,
"arc_transition":<REAL>,
"command_id":<INT>
}
```

Parameter	Data type	Value	Description
cmdName	STRING	*moveL*	Name of the command to send.
relFlag	INT	*0*	Set to *0*. *cartPosition* is interpreted as absolute Cartesian pose of the TCP.
cartPosition	ARRAY [0..5] OF REAL	User defined value	Target position of the robot TCP. Target position value in mm for x, y and z. Target orientation value in degree for rx, ry and rz. [x, y, z, rx, ry, rz]
speed	REAL	User defined value	Value represents the maximum linear speed in mm/s. Value must be greater than *0.0*.
acc	REAL	User defined value	Value represents the maximum linear acceleration in mm/s². Value must be greater than *0.0*.
arc_transition	REAL	User defined value	When the value is not equal to *0.0*, the connection between two motion segments is blended if possible. Value must be greater than or equal to *0.0*.
command_id	INT	User defined value	Identifier of the motion command. Value must be greater than or equal to *0*.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"moveL"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*moveL*	Name of the command sent.

Circular Movement (moveC)

Task

This command triggers a circular movement of the Lexium Cobot to a user-defined target position using a user-defined circular pose to define the trajectory.

Send Message

```
{
"cmdName":"moveC",
"relFlag":<INT>,
"circPosition":[<ARRAY [0..5] OF REAL>],
"cartPosition":[<ARRAY [0..5] OF REAL>],
"speed":<REAL>,
"acc":<REAL>,
"arc_transition":<REAL>,
"command_id":<INT>
}
```

Parameter	Data type	Value	Description
cmdName	STRING	*moveC*	Name of the command to send.
relFlag	INT	*0*	Set to *0*. *cartPosition* is interpreted as absolute Cartesian pose of the TCP.
circPosition	ARRAY [0..5] OF REAL	User defined value	Circular pose of the robot TCP. Circular position value in mm for x, y and z. Circular orientation value in degree for rx, ry and rz. [x, y, z, rx, ry, rz]
cartPosition	ARRAY [0..5] OF REAL	User defined value	Target position of the robot TCP. Target position value in mm for x, y and z. Target orientation value in degree for rx, ry and rz. [x, y, z, rx, ry, rz]
speed	REAL	User defined value	Value represents the maximum circular speed in mm/s. Value must be greater than *0.0*.
acc	REAL	User defined value	Value represents the maximum circular acceleration in mm/s². Value must be greater than *0.0*.
arc_transition	REAL	User defined value	When the value is not equal to *0.0*, the connection between two motion segments is blended if possible. Value must be greater than or equal to *0.0*.
command_id	INT	User defined value	Identifier of the motion command. Value must be greater than or equal to *0*.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"moveC"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*moveC*	Name of the command sent.

Jog Command (jog) - Continuous Jog Mode

Task

This command starts a continuous jog movement of a single component of the Lexium Cobot, both in joint space and in Cartesian space.

The positive and negative direction of the movement is reflected by the positive or negative value of the parameter *speed*.

A jog movement can be stopped using the command Stop Robot Movement (*stop_program*).

Send Message

```
{"cmdName":"jog","jog_mode":1,"coord_map":<INT>,"jnum":
<INT>,"speed":<REAL>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*jog*	Name of the command to send.
jog_mode	INT	*1*	Continuous jogging.
coord_map	INT	*0*: move in world coordinate frame *1*: move in joint space *2*: move in tool coordinate frame	Refers to the coordinate frame in which the Lexium Cobot Arm moves.
jnum	INT	*0...5*	Joint space: axis number 1 to 6 correspond to *0* to *5* respectively. Cartesian space: x, y, z, rx, ry, rz correspond to *0* to *5* respectively.
speed	REAL	User defined value.	Joint space: Value represents the maximum speed of the joints in °/s. Cartesian space: Value represents the maximum speed along the Cartesian direction in mm/s or the orientation in °/s. Value must be greater than or less than *0.0*.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"jog"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*jog*	Name of the command sent.

Jog Command (jog) - Relative Jog Mode

Task

This command starts a relative jog movement of a single component of the Lexium Cobot, in joint as well as in Cartesian space.

The positive and negative direction of the movement is reflected by the positive or negative value of the parameter *poscmd*.

A jog movement can be stopped using the command Stop Robot Movement (*stop_program*).

Send Message

```
{"cmdName":"jog","jog_mode":2,"coord_map":<INT>,"jnum":
<INT>,"speed":<REAL>,"poscmd":<REAL>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*jog*	Name of the command to send.
jog_mode	INT	*2*	Relative jogging.
coord_map	INT	*0*: move in world coordinate frame *1*: move in joint space *2*: move in tool coordinate frame	Refers to the coordinate frame in which the Lexium Cobot Arm moves.
jnum	INT	*0...5*	Joint space: axis number 1 to 6 correspond to *0* to *5* respectively. Cartesian space: x, y, z, rx, ry, rz correspond to *0* to *5* respectively.
speed	REAL	User defined value.	Joint space: Value represents the maximum speed of the joints in °/s. Cartesian space: Value represents the maximum speed along the Cartesian direction in mm/s or the orientation in °/s. Value must be greater than *0.0*.
poscmd	REAL	User defined value.	Refers to the position value. Unit of single joint move is degrees. Unit of Cartesian space single axis move is mm.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"jog"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*jog*	Name of the command sent.

Jog Command (jog) - Absolute Jog Mode

Task

This command starts an absolute jog movement of a single component of the Lexium Cobot, in joint as well as in Cartesian space.

A jog movement can be stopped using the command Stop Robot Movement (*stop_program*).

Send Message

```
{"cmdName":"jog","jog_mode":3,"coord_map":<INT>,"jnum":
<INT>,"speed":<REAL>,"poscmd":<REAL>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*jog*	Name of the command to send.
jog_mode	INT	*3*	Absolute jogging.
coord_map	INT	*0*: move in world coordinate frame *1*: move in joint space *2*: move in tool coordinate frame	Refers to the coordinate frame in which the Lexium Cobot Arm moves.
jnum	INT	*0...5*	Joint space: axis number 1 to 6 correspond to *0* to *5* respectively. Cartesian space: x, y, z, rx, ry, rz correspond to *0* to *5* respectively.
speed	REAL	User defined value.	Joint space: Value represents the maximum speed of the joints in °/s. Cartesian space: Value represents the maximum speed along the Cartesian direction in mm/s or the orientation in °/s. Value must be greater than *0.0*.
poscmd	REAL	User defined value.	Refers to the position value. Unit of single joint move is degrees. Unit of Cartesian space single axis move is mm.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"jog"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*jog*	Name of the command sent.

Clear Error (clear_error)

Task

This command clears a detected collision error or advisory of the Lexium Cobot.

Send Message

```
{"cmdName":"clear_error"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*clear_error*	Name of the command to send.

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"clear_error"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*clear_error*	Name of the command sent.

Load Program (load_program)

Task

This command loads the program specified by its name on the Lexium Cobot Controller.

Send Message

```
{"cmdName":"load_program","programName":"<STRING(24)>"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*load_program*	Name of the command to send.
programName	STRING(24)	User defined value.	Specifies the name of the program to be loaded. Example: Test.jks

NOTE: Ensure that the file extension *.jks is added to the name of the program.

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"load_program"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*load_program*	Name of the command sent.

Start Program (start_program)

Task

This command starts the loaded program on the Lexium Cobot Controller.

Send Message

```
{"cmdName":"start_program"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*start_program*	Name of the command to send.

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"start_program"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*start_program*	Name of the command sent.

Pause Program (pause_program)

Task

This command pauses the program running on the Lexium Cobot Controller.

Send Message

```
{"cmdName":"pause_program"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*pause_program*	Name of the command to send.

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"pause_program"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*pause_program*	Name of the command sent.

Resume Program (resume_program)

Task

This command resumes the paused program on the Lexium Cobot Controller.

Send Message

```
{"cmdName":"resume_program"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*resume_program*	Name of the command to send.

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"resume_program"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*resume_program*	Name of the command sent.

Get Name of Loaded Program (get_loaded_program)

Task

This command retrieves the name of the loaded program from the Lexium Cobot Controller.

Send Message

```
{"cmdName":"get_loaded_program"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*get_loaded_program*	Name of the command to send.

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"get_loaded_
program","programName":"<STRING(24)>"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*get_loaded_program*	Name of the command sent.
programName	STRING(24)	-	Name of the loaded program. Example: Test.jks

Stop Robot Movement (stop_program)

Task

This command stops and aborts the execution of the program running on the Lexium Cobot Controller or stops robot movements executed by move or jog commands.

Send Message

```
{"cmdName":"stop_program"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*stop_program*	Name of the command to send.

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"stop_program"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*stop_program*	Name of the command sent.

Get User Coordinate Frames (get_coordsys_offsets)

Task

This command retrieves the offset of user defined coordinate frames of the Lexium Cobot.

Send Message

```
{"cmdName":"get_coordsys_offsets"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*get_coordsys_offsets*	Name of the command to send.

Receive Message

```
{
"errorCode":"0",
"errorMsg":"",
"cmdName":"get_coordsys_offsets",
"coordsys_offsets":[
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
]
}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*get_coordsys_offsets*	Name of the command sent.
coordsys_offsets	ARRAY [0..10] OF ARRAY [0..5] OF REAL	-	Offset of user defined coordinate frames of the Lexium Cobot. [[x, y, z, rx, ry, rz], …] The array index 0 refers to the World coordinate frame.

Set User Coordinate Frame Offset (set_coordsys_offset)

Task

This command sets a user defined coordinate frame of the Lexium Cobot.

Send Message

```
{"cmdName":"set_coordsys_offset","coordsys_offset":[< ARRAY
[0..5] OF REAL>],"id":<INT>,"name":"<STRING(12)>"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_coordsys_offset*	Name of the command to send.
coordsys_offset	ARRAY [0..5] OF REAL	User defined value.	Offset of user defined coordinate frame to be set of the Lexium Cobot. [x, y, z, rx, ry, rz]
id	INT	User defined value.	Identifier of the user defined coordinate frame. Valid values: *1..10*
name	STRING(12)	User defined value.	Name of the user defined coordinate frame.

Receive Message

```
{"errorCode": "0", "errorMsg": "", "cmdName": "set_coordsys_
offset"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_coordsys_offset*	Name of the command sent.

Select User Coordinate Frame (set_coordsys_id)

Task

This command selects a user defined coordinate frame of the Lexium Cobot.

Send Message

```
{"cmdName":"set_coordsys_id", "coordsys_id":<INT>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_coordsys_id*	Name of the command to send.
coordsys_id	INT	User defined value.	Identifier of the user defined coordinate frame to be selected. Valid values: *0...10* The value *0* refers to the World coordinate frame.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"set_coordsys_id"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_coordsys_id*	Name of the command sent.

Get Tool Coordinate Frames (get_tool_offsets)

Task

This command retrieves the offset of user defined tool coordinate frames of the Lexium Cobot.

Send Message

```
{"cmdName":"get_tool_offsets"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*get_tool_offsets*	Name of the command to send.

Receive Message

```
{
"errorCode":"0",
"errorMsg":"",
"cmdName":"get_tool_offsets",
"tool_offsets":[
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
[<ARRAY [0..5] OF REAL>],
]
}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*get_tool_offsets*	Name of the command sent.
tool_offsets	ARRAY [0..10] OF ARRAY [0..5] OF REAL	-	User defined tool coordinate frames of the Lexium Cobot. The array index 0 refers to the center of the end flange.

Set Tool Coordinate Frame (set_tool_offset)

Task

This command sets a user defined tool coordinate frame of the Lexium Cobot.

Send Message

```
{"cmdName":"set_tool_offset","tool_offset":[< ARRAY [0..5]
OF REAL>],"id":<INT>,"name":"<STRING(12)>"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_tool_offset*	Name of the command to send.
tool_offset	ARRAY [0..5] OF REAL	User defined value.	Offset of user defined tool coordinate frame to be set of the Lexium Cobot. [x, y, z, rx, ry, rz]
id	INT	User defined value.	Identifier of the user defined tool coordinate frame. Valid values: *1..10*
name	STRING(12)	User defined value.	Name of the user defined tool coordinate frame.

Receive Message

```
{"errorCode": "0", "errorMsg": "", "cmdName": "set_tool_
offset"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_tool_offset*	Name of the command sent.

Select Tool Coordinate Frame (set_tool_id)

Task

This command selects a user defined tool coordinate frame of the Lexium Cobot.

Send Message

```
{"cmdName":"set_tool_id", "tool_id":<INT>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_coordsys_id*	Name of the command to send.
set_tool_id	INT	User defined value.	Identifier of the user defined tool coordinate frame to be selected. Valid values: *0...10* The value *0* refers to the center of the end flange.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"set_tool_id"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_tool_id*	Name of the command sent.

Get System Variables (get_system_variables)

Task

This command retrieves the user defined system variables of the Lexium Cobot.

Send Message

```
{"cmdName":"get_system_variables"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*get_system_ variables*	Name of the command to send.

Receive Message

```
{
"errorCode": "0",
"errorMsg": "",
"cmdName":"get_system_variables",
"system_variables":[{"alias":"<STRING(20)>","id":
<INT>,"value":<REAL>}] "
}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*get_system_ variables*	Name of the command sent.
system_ variables	ARRAY [0..X] OF SYSTEM-VARIABLE	-	User defined system variables of the Lexium Cobot.

System Variable

Parameter	Data type	Value	Description
alias	STRING(20)	-	Refers to the name of the system variable.
id	INT	-	Refers to the identifier of the system variable.
value	REAL	-	Refers to the value of the system variable. The variable range is [-999,999,999,999.999 … 999,999,999,999.999]

Set System Variable (set_system_variable)

Task

This command modifies a user defined system variable of the Lexium Cobot.

Send Message

```
{
"cmdName":"set_system_variable",
"system_variable_id":<INT>,
"system_variable_name":"<STRING(20)>","system_variable_
value":<REAL>
}
```

Parameter	Data type	Value	Description
cmdName	STRING	*get_system_variables*	Name of the command to send.
system_variable_id	INT	User defined value.	Identifier of the system variable to be modified. Range: [5500..5599]
system_variable_name	STRING(20)	User defined value.	Defines the name of the system variable.
system_variable_value	REAL	User defined value.	Defines the value of the system variable. The variable range is [-999,999,999,999.999 … 999,999,999,999.999]

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"set_system_
variable"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_system_variable*	Name of the command sent.

Exceptions

If	Then
The parameter *system_variable_id* is out of [5500, 5599]	The receive message is: *{"errorCode":"10","errorMsg":"param id_new should between 5500~5599"}*
The parameter *system_variable_value* is not in the valid range of +-999,999,999,999.999	The receive message is: *{"errorCode":"10","errorMsg":"system_variable_value must be in the range +-999,999,999,999.999"}*
The parameter *system_variable_id* does not exist	The receive message is: *{"errorCode":"13","errorMsg":"system_variable_id not exist"}*

Set Analog Output (set_analog_output)

Task

This command sets the analog output of the Lexium Cobot Cabinet Controller specified by its type and index.

Send Message

```
{"cmdName":"set_analog_output","type":<INT>,"index":
<INT>,"value":<REAL>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_analog_output*	Name of the command to send.
type	INT	*0*: Cabinet	Defines the type of the analog output of the Lexium Cobot Cabinet Controller.
index	INT	User defined value.	Defines the index of the analog output of the Lexium Cobot Cabinet Controller.
value	REAL	User defined value.	Defines the value of the analog output of the Lexium Cobot Cabinet Controller. Unit: [%] Value range: *0* ≤ value ≤ *100*

NOTE: The index 0 refers to analog output AO1 and index 1 refers to analog output AO2 of the Lexium Cobot Cabinet Controller.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"set_analog_
output"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_analog_output*	Name of the command sent.

Set Digital Output (set_digital_output)

Task

This command sets the digital output of the Lexium Cobot system specified by its type and index.

Send Message

```
{"cmdName":"set_digital_output","type":<INT>,"index":
<INT>,"value":<INT>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_digital_output*	Name of the command to send.
type	INT	*0*: Cabinet *1*: Tool End	Defines the type of the digital output of the Lexium Cobot system.
index	INT	User defined value.	Defines the index of the digital output of the Lexium Cobot system.
value	INT	*0* or *1*	Defines the state of the digital output of the Lexium Cobot system.

NOTE: For the Lexium Cobot system, the index 0 refers to digital output DO1, the index 1 refers to digital output DO2, and so on.

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"set_digital_
output"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_digital_output*	Name of the command sent.

Set Speed Override (set_speed_rate)

Task

This command sets the speed override for executing a program on the Lexium Cobot Controller.

Send Message

```
{"cmdName":"set_speed_rate","rate_value":<REAL>}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_speed_rate*	Name of the command to send.
rate_value	REAL	User defined value.	Specifies the value of the speed override. Value range: *0.0 ≤ rate_value ≤ 1.0*

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":" set_speed_rate"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_speed_rate*	Name of the command sent.

Get Payload (get_payload)

Task

This command retrieves the parameters of the payload which is mounted to the Lexium Cobot Arm.

Send Message

```
{"cmdName":"get_payload"}
```

Parameter	Data type	Value	Description
cmdName	STRING	*get_payload*	Name of the command to send.

Receive Message

```
{"errorCode":"0","errorMsg": "","cmdName":"get_
payload","centroid": [<ARRAY [0..2] OF REAL>],"mass":<REAL>}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*get_payload*	Name of the command sent.
centroid	ARRAY [0..2] OF REAL	-	Offset of the center of mass of the payload mounted on Lexium Cobot Arm related to the End Flange Coordinate System. Unit: [mm]
mass	REAL	-	Mass of the payload mounted on the Lexium Cobot Arm. Unit: [kg]

Set Payload (set_payload)

Task

This command sets the parameters of the payload which is mounted to the Lexium Cobot Arm. The parameters to be set consist of the mass of the payload and its center of mass.

Send Message

```
{"cmdName":"set_payload","mass":<REAL>,"centroid":[< ARRAY
[0..2] OF REAL>]}
```

Parameter	Data type	Value	Description
cmdName	STRING	*set_payload*	Name of the command to send.
mass	REAL	User defined value.	Mass of the payload mounted on the Lexium Cobot Arm. Unit: [kg]
centroid	ARRAY [0..2] OF REAL	User defined value.	Center of mass of the payload mounted on the Lexium Cobot Arm. Unit: [mm]

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"set_payload"}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*set_payload*	Name of the command sent.

Calculate Forward Kinematics (kine_forward)

Task

This command calculates the user-defined positions of the robot joints, transferred at jointPosition to the corresponding Cartesian pose.

Send Message

```
{"cmdName":"kine_forward","jointPosition":[< ARRAY [0..5] OF
REAL>]}
```

Parameter	Data type	Value	Description
cmdName	STRING	*kine_forward*	Name of the command to send.
jointPosition	ARRAY [0..5] OF REAL	User defined value.	Joint positions to be transferred to corresponding Cartesian pose. [j1, j2, j3, j4, j5, j6] Unit: [°]

Receive Message

```
{"errorCode":"0","errorMsg":"","cmdName":"kine_
forward","cartPosition":[< ARRAY [0..5] OF REAL>]}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*kine_forward*	Name of the command sent.
cartPosition	ARRAY [0..5] OF REAL	-	Cartesian pose referring to the joint positions provided. [x, y, z, rx, ry, rz] Unit: [mm]

Calculate Inverse Kinematics (kine_inverse)

Task

This command calculates the user-defined Cartesian pose to the corresponding positions of the robot joints. The corresponding positions of the robot joints are determined by the provided joint positions on jointPosition.

Send Message

```
{"cmdName":"kine_inverse","jointPosition":[< ARRAY [0..5] OF
REAL>],"cartPosition":[< ARRAY [0..5] OF REAL>]}
```

Parameter	Data type	Value	Description
cmdName	STRING	*kine_inverse*	Name of the command to send.
jointPosition	ARRAY [0..5] OF REAL	User defined value.	Joint positions as a reference. [j1, j2, j3, j4, j5, j6] Unit: [°]
cartPosition	ARRAY [0..5] OF REAL	User defined value.	Cartesian pose to be transferred to corresponding joint positions. [x, y, z, rx, ry, rz] Unit: [mm]

Receive Message

```
{"errorCode": "0","errorMsg":"","cmdName":"kine_
inverse","jointPosition":[< ARRAY [0..5] OF REAL>]}
```

Parameter	Data type	Value	Description
errorCode	STRING	*0*	Detected error code.
errorMsg	STRING	-	Detected error message.
cmdName	STRING	*kine_inverse*	Name of the command sent.
jointPosition	ARRAY [0..5] OF REAL	-	Joint positions referring to the Cartesian pose provided. [j1, j2, j3, j4, j5, j6] Unit: [°]

Robot Feedback Data

This chapter describes the TCP/IP robot feedback data used in and released with LexiumCobotCommunication library for EcoStruxure Machine Expert.

NOTE:

- Implement and connect a TCP/IP client to the Lexium Cobot Controller on port 10000 to retrieve the robot feedback data.
- Use the Ethernet connector CN13 of the Lexium Cobot Cabinet Controller or CN26 of the Lexium Cobot Compact Controller to establish the connection.
- Configure the network settings using EcoStruxure Cobot Expert. For further information, refer to Network Settings, page 68.

The data is transferred in ASCII code every 20 ms from the Lexium Cobot Controller (TCP/IP server).

The robot feedback data is in JSON format.

Element	Description
`{`	Start sign of the protocol.
`"len": <INT>,`	Length of the protocol in bytes.
`"error_code": <STRING>,`	Error code provided by the Lexium Cobot Controller. **NOTE:** Only reported once if an error is detected.
`"error_msg": <STRING>,`	Error message provided by the Lexium Cobot Controller. **NOTE:** Only reported once if an error is detected.
`"protective_stop": <INT>,`	*0*: The Lexium Cobot is not in protective stop. *1*: The Lexium Cobot is in protective stop.
`"joint_position": [<ARRAY [0..5] OF REAL>],`	Positions of the Lexium Cobot joints.
`"cartesian_position": [<ARRAY [0..5] OF REAL>],`	Cartesian pose of the Lexium Cobot TCP.
`"din": [<ARRAY [0..15] OF INT>],`	Digital inputs of the Lexium Cobot Controller.
`"dout": [<ARRAY [0..15] OF INT>],`	Digital outputs of the Lexium Cobot Controller.
`"ain": [<ARRAY [0..1] OF REAL>],`	Analog inputs of the Lexium Cobot Cabinet Controller. Unit: [%]
`"aout": [<ARRAY [0..1] OF REAL>],`	Analog outputs of the Lexium Cobot Cabinet Controller. Unit: [%]
`"tio_din": [<ARRAY [0..1] OF INT>],`	Digital inputs of the Lexium Cobot Tool End.
`"tio_dout": [<ARRAY [0..1] OF INT>],`	Digital outputs of the Lexium Cobot Tool End.
`"tio_ain": [<ARRAY [0..1] OF REAL>],`	Analog inputs of the Lexium Cobot Tool End. Unit: [%]
`"task_mode": <INT>,`	Mode in which the Lexium Cobot system is running. *1*: The Lexium Cobot system is in manual mode. *2*: The Lexium Cobot system is in automatic mode. *4*: The Lexium Cobot system is in hand-guide (drag) mode.

Element	Description
`"program_state": <INT>,`	State of the executed program. *0*: The program state of the Lexium Cobot system is idle. No program is loaded. *1*: The program state of the Lexium Cobot system is reading. A program is loading. *2*: The program state of the Lexium Cobot system is paused. The loaded program is paused. *3*: The program state of the Lexium Cobot system is waiting. The loaded program is fully loaded and executing.
`"enabled": <BOOL>,`	*false*: The Lexium Cobot Arm is disabled. *true*: The Lexium Cobot Arm is enabled.
`"speed_rate": <REAL>,`	Configured speed override for executing a program on the Lexium Cobot Controller. Value range: *0.0 ≤ speed_rate ≤ 1.0*
`"tool_id": <INT>,`	Identifier of the active user defined tool coordinate frame of the Lexium Cobot.
`"coordsys_id": <INT>,`	Identifier of the active user defined coordinate frame of the Lexium Cobot.
`"on_soft_limit": <INT>,`	*0*: The Lexium Cobot is not in soft limit. *1*: The Lexium Cobot is in soft limit. One or some of the Lexium Cobot joints have reached their soft limit, which can be configured in **Safety Setting** in EcoStruxure Cobot Expert.
`"emergency_stop": <INT>,`	*0*: The Lexium Cobot is not in emergency stop. *1*: The Lexium Cobot is in emergency stop.
`"powered_on": <INT>,`	*0*: The Lexium Cobot Arm is powered off. *1*: The Lexium Cobot Arm is powered on.
`"in_position": <BOOL>,`	*false*: • The Lexium Cobot is not in position. *true*: • The Lexium Cobot is in standstill. • The Lexium Cobot has reached the jog target. • The Lexium Cobot has reached the target position of a move command (TCP/IP) • The Lexium Cobot is executing a program and has reached the target position of a move command (Blockly).
`"collision_stop": <INT>,`	*0*: The Lexium Cobot is not in collision stop. *1*: The Lexium Cobot is in collision stop.
`"tio_button": [<ARRAY [0..2] OF INT>],`	*0*: Button not pressed *1*: Button pressed *[0]* : **FREE** button *[1]* : **POINT** button *[2]* : **play/pause** button
`"command_id": <INT>,`	Identifier of the active move command in case the move command is specified in this library. Identifier of the program command in case of a running program.
`"safety_checksum": <STRING(10)>`	Lexium Cobot safety checksum.
`}`	End sign of the protocol.

Appendices

What’s in This Part

Additional Information on the Lexium Cobot

What's in This Chapter

Data Types of Lexium Cobot Parameters

Get the Joint Position

Data type: array

Length: 6

Meaning: six elements in the array represent the angle values (unit: degree) from the first joint to the sixth joint in sequence.

Get the Tool and Center Position

Data type: array

Length: 6

Meaning: six elements in the array represent the spatial positions of the origin of the existing tool coordinate system in the user coordinate system, corresponding to X, Y, Z (unit: mm), RX, RY and RZ (unit: degree) from the 0 element to the 5th element in sequence.

Get the Flange Center Pose

Data type: array

Length: 6

Meaning: it represents the spatial positions of the Lexium Cobot Arm end flange center in the user coordinate system, corresponding to X, Y, Z (unit: mm), RX, RY and RZ (unit: degree) from the 0 element to the 5th element in sequence.

Capture the End Payload

Data type: array

Length: 4

Meaning: it represents the Lexium Cobot Arm stored end payload information, corresponding to the load mass (unit: kg) and X, Y and Z distances (unit: mm) of the centroid of payload relative to the flange center from the 0 element to the 3rd element in sequence.

Capture the End Force

Data type: array

Length: 6

Meaning: capture the net torque value (unit: N.m) after the sensor end load is compensated by the end torque sensor.

Capture the Sensitivity

Data type: number

Meaning: the configured collision sensitivity value.

System Time

Data type: number

Meaning: capture the system time in ms since the last start of the Lexium Cobot Controller.

Modbus Address Table

ID	Type	Name	Data type	Function code	Register type
8	Common digital input	DO0	BOOL	02	Discrete input is readable but not writable
9		DO1			
10		DO2			
…		…			
133		DO125			
134		DO126			
135		DO127			
40	Common digital output	DI0	BOOL	01/05/15	Coil state
41		DI1			
42		DI2			
…		…			
165		DI125			
166		DI126			
167		DI127			
96	Analog input	AO00	UINT16	04	Input register is readable but not writable
97		AO01			
98		AO02			
99		AO03			
…		…			
109		AO13			
110		AO14			
111		AO15			
112		AO16	INT16		
113		AO17			
114		AO18			
…		…			
125		AO29			
126		AO30			
127		AO31			
128		AO32	FLOAT32 (Big-Endian)		
129					
130		AO33			
131					
132		AO34			
133					
…		…			
…					
186		AO61			
187					
188		AO62			
189					
190		AO63			
191					

ID	Type	Name	Data type	Function code	Register type
100	Analog output	AI00	UINT16	03/06	Holding register is readable and writable
101		AI01			
102		AI02			
103		AI03			
104		AI04			
…		…			
111		AI11			
112		AI12			
113		AI13			
114		AI14			
115		AI15			
116		AI16	INT16		
117		AI17			
118		AI18			
119		AI19			
120		AI20			
…		…			
127		AI27			
128		AI28			
129		AI29			
130		AI30			
131		AI31			
132		AI32	FLOAT32 (Big-Endian)		
133					
134		AI33			
135					
136		AI34			
137					
138		AI35			
139					
140		AI36			
141					
…		…			
…					
186		AI59			
187					
188		AI60			
189					
190		AI61			
191					
192		AI62			
193					
194		AI63			
195					

ID	Type	Name	Data type	Function code	Description	Unit	Register type
300	Lexium Cobot data	Servo version No.	INT32	04	–	–	Input register is readable but not writable
302		Lexium Cobot serial No.					
304		Joint 1 voltage	INT32		Voltage of each joint	V	
306		Joint 2 voltage					
308		Joint 3 voltage					
310		Joint 4 voltage					
312		Joint 5 voltage					
314		Joint 6 voltage					
316		Joint 1 temperature			Temperature of each joint	°C	
318		Joint 2 temperature					
320		Joint 3 temperature					
322		Joint 4 temperature					
324		Joint 5 temperature					
326		Joint 6 temperature					
328		Joint 1 servo error code	INT32		Servo serial No. of each joint	–	
330		Joint 2 servo error code					
332		Joint 3 servo error code					
334		Joint 4 servo error code					
336		Joint 5 servo error code					
338		Joint 6 servo error code					
340		Joint 1 error state	UINT16		Servo error state • 0 means no error detected • 1 means error detected		
341		Joint 2 error state					
342		Joint 3 error state					
343		Joint 4 error state					
344		Joint 5 error state					
345		Joint 6 error state					

ID	Type	Name	Data type	Function code	Description	Unit	Register type
346	Lexium Cobot data	Joint 1 enabling state	UINT16	04	Servo enabling state • 0 means disabling • 1 means enabling	–	Input register is readable but not writable
347		Joint 2 enabling state					
348		Joint 3 enabling state					
349		Joint 4 enabling state					
350		Joint 5 enabling state					
351		Joint 6 enabling state					
352		Joint 1 collision state			Servo collision detection state • 0 means no collision detected • 1 means collision detected		
353		Joint 2 collision state					
354		Joint 3 collision state					
355		Joint 4 collision state					
356		Joint 5 collision state					
357		Joint 6 collision state					
358		Joint 1 current	Float32		Current of each joint	A	
360		Joint 2 current					
362		Joint 3 current					
364		Joint 4 current					
366		Joint 5 current					
368		Joint 6 current					
370		Sensor force x			Force/Torque of each joint	N	
372		Sensor force y					
374		Sensor force z					
376		Sensor torque rx				Nm	
378		Sensor torque ry					
380		Sensor torque rz					
382		Joint 1 position			Position of each joint	°	
384		Joint 2 position					
386		Joint 3 position					
388		Joint 4 position					
390		Joint 5 position					
392		Joint 6 position					

ID	Type	Name	Data type	Function code	Description	Unit	Register type
394	Lexium Cobot data	Joint 1 speed	Float32	04	Speed of each joint	°/s	Input register is readable but not writable
396		Joint 2 speed					
398		Joint 3 speed					
400		Joint 4 speed					
402		Joint 5 speed					
404		Joint 6 speed					
406		TCP position X			TCP	mm	
408		TCP position Y					
410		TCP position Z					
412		TCP position RX				°	
414		TCP position RY					
416		TCP position RZ					
418		TCP speed X			TCP speed	mm/s	
420		TCP speed Y					
422		TCP speed Z					
424		TCP speed RX				°/s	
426		TCP speed RY					
428		TCP speed RZ					
430		TCP_OFFSET_X			Tool coordinate system	mm	
432		TCP_OFFSET_Y					
434		TCP_OFFSET_Z					
436		TCP_OFFSET_RX				°	
438		TCP_OFFSET_RY					
440		TCP_OFFSET_RZ					
442		BASE_OFFSET_X			User coordinate system	mm	
444		BASE_OFFSET_Y					
446		BASE_OFFSET_Z					
448		BASE_OFFSET_RX				°	
450		BASE_OFFSET_RY					
452		BASE_OFFSET_RZ					

ID	Type	Name	Data type	Function code	Description	Unit	Register type
454	Lexium Cobot data	PROTECTIVE_STOP	UINT16	04	Lexium Cobot collision detected: 1 No Lexium Cobot collision detected: 0	–	Input register is readable but not writable
455		EMERGENCY_STOP			Emergency stop		
456		POWER_ON			Power-on		
457		ROBOT_ENABLE			Upper enabling		
458		ON_SOFT_LIMIT			Software limit		
459		INPOS			Reach the target position		
460		Motion mode			• Servo position mode: 4 • Admittance control mode: 2 • Hand-guided mode: 1 • Other mode (Jog and other operation): 0		
461		Reduction mode level			Reduction mode level: • First-level reduction: 1 • Second-level reduction: 2 • Protective stop: 3		
462		Speed magnification	FLOAT32		Speed setting of program		
464		MOTION_ERRCODE	INT32		Error code		
466		CAB_TEMPERATURE	FLOAT32		Lexium Cobot Controller temperature		
468		CAB_ AVERAGEPOWER			Lexium Cobot Controller power		
470		CAB_ AVERAGECURRENT			Lexium Cobot Controller current		
472		UHI_PULES	FLOAT32		Conveyor belt pulse		
474		UHI_SPEED			Conveyor belt movement speed		
476		UHI_DIR	UINT16		Conveyor belt movement direction		
477		UHI_ORIGIN_PULES	INT32		Original pulse of conveyor belt		
479		Reserved	UINT16		–		

Profinet Address Table

Transmission type Lexium Cobot > External Controller (R->P)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
0	Robot serial number (int32)									Robot state, safety–related settings 1_R->P_Robot_Safety 32+4 Bytes
32	Servo version number (int32)									
64	CAB_AVERAGECURRENT (float) [A]									
96	CAB_AVERAGEPOWER (float) [W]									
128	CAB_TEMPERATURE (float) [℃]									
160	Power state	Enable state	Reten-tion	Retention						
192	MOTION_ERRCODE (int32)									
224	Move mode (uint8)			Reduc-tion mode level	Emer-gency Stop	Protec-tive stop	Soft limit state	Reten-tion	Reten-tion	
256	Reserved (int) 4 bytes									
288	Joint 1 voltage (float) [V]									Joint parameters 2_R->P_Joints 172+48 Bytes
320	Joint 2 voltage (float) [V]									
352	Joint 3 voltage (float) [V]									
384	Joint 4 voltage (float) [V]									
416	Joint 5 voltage (float) [V]									
448	Joint 6 voltage (float) [V]									
480	Joint 1 current (float) [A]									
512	Joint 2 current (float) [A]									
544	Joint 3 current (float) [A]									
576	Joint 4 current (float) [A]									
608	Joint 5 current (float) [A]									
640	Joint 6 current (float) [A]									
672	Joint 1 position (float)[°]									
704	Joint 2 position (float) [°]									
736	Joint 3 position (float) [°]									
768	Joint 4 position (float) [°]									
800	Joint 5 position (float) [°]									
832	Joint 6 position (float) [°]									
864	Joint 1 speed (float) [°/s]									
896	Joint 2 speed (float) [°/s]									
928	Joint 3 speed (float) [°/s]									
960	Joint 4 speed (float) [°/s]									
992	Joint 5 speed (float) [°/s]									
1024	Joint 6 speed (float) [°/s]									
1088	Joint 1 temperature (float) [°C]									
1120	Joint 2 temperature (float) [°C]									
1152	Joint 3 temperature (float) [°C]									

Transmission type Lexium Cobot > External Controller (R->P)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
1184	Joint 4 temperature (float) [°C]									Joint parameters 2_R->P_Joints 172+48 Bytes
1216	Joint 5 temperature (float) [°C]									
1248	Joint 6 temperature (float) [°C]									
1280	Joint 1 torque (float) [Nm]									
1312	Joint 2 torque (float) [Nm]									
1344	Joint 3 torque (float) [Nm]									
1376	Joint 4 torque (float) [Nm]									
1408	Joint 5 torque (float) [Nm]									
1440	Joint 6 torque (float) [Nm]									
1472	Joint 1 servo error code (int32)									
1504	Joint 2 servo error code (int32)									
1536	Joint 3 servo error code (int32)									
1568	Joint 4 servo error code (int32)									
1600	Joint 5 servo error code (int32)									
1632	Joint 6 servo error code (int32)									
1664	Joint error state (uint8)			Joint enable state	Joint collision state (uint8)				Reten-tion	
1696 ~	Reserved (float) 48 Bytes									
2048	TCP position X (float) [mm]									TCP and BASE parameters 3_R->P_TCP_BASE 96+48 Bytes
2080	TCP position Y (float) [mm]									
2112	TCP position Z (float) [mm]									
2144	TCP position RX (float) [mm]									
2176	TCP position RY (float) [mm]									
2208	TCP position RZ (float) [mm]									
2240	TCP speed X (float) [mm/s]									
2272	TCP speed Y (float) [mm/s]									
2304	TCP speed Z (float) [mm/s]									
2336	TCP speed RX (float) [mm/s]									
2368	TCP speed RY (float) [mm/s]									
2432	TCP_OFFSET_X (float) [mm]									
2464	TCP_OFFSET_Y (float) [mm]									
2496	TCP_OFFSET_Z (float) [mm]									
2528	TCP_OFFSET_RX (float) [mm]									
2560	TCP_OFFSET_RY (float) [mm]									
2592	TCP_OFFSET_RZ (float) [mm]									
2400	TCP speed RZ (float) [mm/s]									
2624	BASE_OFFSET_X (float) [mm]									
2656	BASE_OFFSET_Y (float) [mm]									
2688	BASE_OFFSET_Z (float) [mm]									

Transmission type Lexium Cobot > External Controller (R->P)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
2720	BASE_OFFSET_RX (float) [mm]									3_R->P_TCP_BASE 96+48 Bytes
2752	BASE_OFFSET_RY (float) [mm]									
2784	BASE_OFFSET_RZ (float) [mm]									
2816 ~	Reserved 48 Bytes									
3200	Boolean register 0-31									Boolean output register DO 0~63 4_R->P_DO 8+4 Bytes
3232	Boolean register 32-63									
3264	Reserved (4 Bytes)									
3296	Integer register 0									Integer output register AO 0~31 5_R->P_AO_INT 128 Bytes
3328	Integer register 1									
3360	Integer register 2									
3392	Integer register 3									
3424	Integer register 4									
3456	Integer register 5									
3488	Integer register 6									
3520	Integer register 7									
3552	Integer register 8									
3584	Integer register 9									
3616	Integer register 10									
3648	Integer register 11									
3680	Integer register 12									
3712	Integer register 13									
3744	Integer register 14									
3776	Integer register 15									
3808	Integer register 16									
3840	Integer register 17									
3872	Integer register 18									
3904	Integer register 19									
3936	Integer register 20									
3968	Integer register 21									
4000	Integer register 22									
4032	Integer register 23									
4064	Integer register 24									
4096	Integer register 25									
4128	Integer register 26									
4160	Integer register 27									
4192	Integer register 28									
4224	Integer register 29									
4256	Integer register 30									
4288	Integer register 31									

Transmission type Lexium Cobot > External Controller (R->P)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
4320	Floating point number register 0									Floating point number output register AO 0~31 6_R->P_AO_FLOAT 128 Bytes
4352	Floating point number register 1									
4384	Floating point number register 2									
4416	Floating point number register 3									
4448	Floating point number register 4									
4480	Floating point number register 5									
4512	Floating point number register 6									
4544	Floating point number register 7									
4576	Floating point number register 8									
4608	Floating point number register 9									
4640	Floating point number register 10									
4672	Floating point number register 11									
4704	Floating point number register 12									
4736	Floating point number register 13									
4768	Floating point number register 14									
4800	Floating point number register 15									
4832	Floating point number register 16									
4864	Floating point number register 17									
4896	Floating point number register 18									
4928	Floating point number register 19									
4960	Floating point number register 20									
4992	Floating point number register 21									
5024	Floating point number register 22									
5056	Floating point number register 23									
5088	Floating point number register 24									
5120	Floating point number register 25									
5152	Floating point number register 26									
5184	Floating point number register 27									
5216	Floating point number register 28									
5248	Floating point number register 29									
5280	Floating point number register 30									
5312	Floating point number register 31									

EtherNet/IP I/O Address Table

Transmission type Lexium Cobot > External Controller (R->P)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
0	Robot serial number (int32)									Robot state, safety–related settings 1_R->P_Robot_Safety 20 Bytes
32	Servo version number (int32)									
64	Power state	Enable state	Reten-tion	Retention						
96	MOTION_ERRCODE (int32)									
128	Move mode (uint8)			Reduc-tion mode level	Emer-gency Stop	Protec-tive stop	Soft limit state	Reten-tion	Reten-tion	
160	Joint 1 current (float) [A]									Joint parameters 2_R->P_Joints 124 Bytes + 20 Bytes
192	Joint 2 current (float) [A]									
224	Joint 3 current (float) [A]									
256	Joint 4 current (float) [A]									
288	Joint 5 current (float) [A]									
320	Joint 6 current (float) [A]									
352	Joint 1 position (float)[°]									
384	Joint 2 position (float) [°]									
416	Joint 3 position (float) [°]									
448	Joint 4 position (float) [°]									
480	Joint 5 position (float) [°]									
512	Joint 6 position (float) [°]									
544	Joint 1 speed (float) [°/s]									
576	Joint 2 speed (float) [°/s]									
608	Joint 3 speed (float) [°/s]									
640	Joint 4 speed (float) [°/s]									
672	Joint 5 speed (float) [°/s]									
704	Joint 6 speed (float) [°/s]									
736	Joint 1 torque (float) [Nm]									
768	Joint 2 torque (float) [Nm]									
800	Joint 3 torque (float) [Nm]									
832	Joint 4 torque (float) [Nm]									
864	Joint 5 torque (float) [Nm]									
896	Joint 6 torque (float) [Nm]									
928	Joint 1 servo error code (int32)									
960	Joint 2 servo error code (int32)									
992	Joint 3 servo error code (int32)									
1024	Joint 4 servo error code (int32)									
1056	Joint 5 servo error code (int32)									
1088	Joint 6 servo error code (int32)									
1120	Joint error state (uint8)			Joint enable	Joint collision state (uint8)				Reten-tion	

Transmission type Lexium Cobot > External Controller (R->P)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
1152~-1280	reserved 20 Bytes									Joint parameters 2_R->P_Joints 124 Bytes + 20 Bytes
1312	Sensor force x (float) [N]									TCP parameters 3_R->P_TCP 76 Bytes + 48 Bytes
1344	Sensor force y (float) [N]									
1376	Sensor force z (float) [N]									
1408	Sensor torque rx (float) [Nm]									
1440	Sensor torque ry (float) [Nm]									
1472	Sensor torque rz (float) [Nm]									
1504	TCP position X (float) [mm]									
1536	TCP position Y (float) [mm]									
1568	TCP position Z (float) [mm]									
1600	TCP position RX (float) [mm]									
1632	TCP position RY (float) [mm]									
1664	TCP position RZ (float) [mm]									
1696	TCP_OFFSET_X (float)[mm]									
1728	TCP_OFFSET_Y (float)[mm]									
1760	TCP_OFFSET_Z (float)[mm]									
1792	TCP_OFFSET_RX (float)[mm]									
1824	TCP_OFFSET_RY (float)[mm]									
1856	TCP_OFFSET_RZ (float)[mm]									
1888	TCP linear speed V (float) [mm/s]									
1920-~2272	reserved 48 Bytes									
2304	Boolean register 0-31									Boolean output register DO 0~63 4_R->P_DO 8+4 Bytes
2336	Boolean register 32-63									
2368	Reserved (4 Bytes)									
2400	Integer register 0									Integer output register AO 0~23 5_R->P_AO_INT 96 Bytes
2432	Integer register 1									
2464	Integer register 2									
2496	Integer register 3									
2528	Integer register 4									
2560	Integer register 5									
2592	Integer register 6									
2624	Integer register 7									
2656	Integer register 8									
2688	Integer register 9									
2720	Integer register 10									
2752	Integer register 11									
2784	Integer register 12									

Transmission type Lexium Cobot > External Controller (R->P)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
2816	Integer register 13									Integer output register AO 0~23 5_R->P_AO_INT 96 Bytes
2848	Integer register 14									
2880	Integer register 15									
2912	Integer register 16									
2944	Integer register 17									
2976	Integer register 18									
3008	Integer register 19									
3040	Integer register 20									
3072	Integer register 21									
3104	Integer register 22									
3136	Integer register 23									
3168	Floating point number register 0									Floating point number output register AO 0~23 6_R->P_AO_FL OAT 96 Bytes
3200	Floating point number register 1									
3232	Floating point number register 2									
3264	Floating point number register 3									
3296	Floating point number register 4									
3328	Floating point number register 5									
3360	Floating point number register 6									
3392	Floating point number register 7									
3424	Floating point number register 8									
3456	Floating point number register 9									
3488	Floating point number register 10									
3520	Floating point number register 11									
3552	Floating point number register 12									
3584	Floating point number register 13									
3616	Floating point number register 14									
3648	Floating point number register 15									
3680	Floating point number register 16									
3712	Floating point number register 17									
3744	Floating point number register 18									
3776	Floating point number register 19									
3808	Floating point number register 20									
3840	Floating point number register 21									
3872	Floating point number register 22									
3904	Floating point number register 23									

Transmission type External Controller > Lexium Cobot (P->R)										
Bit	0	1	2~7	8~15	16	17	18	19~23	24~31	**Unit Modules**
0	Boolean register 0-31									Boolean input register DI 0~63 1_P->R_DI 8+4 Bytes
32	Boolean register 32-63									
64	Reserved (4 Bytes)									
96	Integer register 0									Integer input register AI 0~23 2_P->R_AI_INT 96 Bytes
128	Integer register 1									
160	Integer register 2									
800	Integer register 22									
832	Integer register 23									
864	Floating point number register 0									Floating point number input register AI 0~23 3_P>R_AI_ FLOAT 96 Bytes
928	Floating point number register 1									
1568	Floating point number register 22									
1600	Floating point number register 23									

Further Information About the Manufacturer

What's in This Chapter

Contact Addresses

Manufacturer

Schneider Electric Automation GmbH

(Subsidiary of Schneider Electric Industries SAS FR-92500 Rueil-Malmaison)

Schneiderplatz 1

97828 Marktheidenfeld, Germany

Phone: +49 9391 603 5000

www.se.com

Other Contacts

See the homepage for additional contact addresses:

Contact Center | Schneider Electric Global (se.com)

Product Training Courses

Product Training Courses

Schneider Electric offers a number of product training courses.

The Schneider Electric training instructors will help you take advantage of the extensive possibilities offered by the system.

See the website (www.se.com) for further information and the seminar schedule.

Glossary

F

Factory Pose: The factory pose describes the folding positions of the Lexium Cobot Arm in delivery condition.

Flange Center Point (FCP): Center of the outer contact surface of the tool flange of the Lexium Cobot Arm.

H

Hand-Guided Mode: In hand-guided mode, also called drag mode or free-drive mode, the Lexium Cobot Arm is moved manually by hand.

Home Pose: The initial pose of the Lexium Cobot Arm. You can define the home pose in the software and reach the home pose through the Home button on the Control Stick.

M

Manual Operation: Moving the Lexium Cobot Arm manually using the Control Stick or the control elements in the manual operation interface of the software.

Modbus RTU: Modbus Remote Terminal Unit. Serial line connection by using Modbus communication.

O

Open Pose: Zero position pose for joint zeroing.

P

Pose: The Lexium Cobot pose includes the cartesian position X, Y, Z and the orientation RX, RY and RZ.

R

Roll-Pitch-Yaw (RPY): Special Euler angles (attitude angles) used to describe the orientation of the robot in three-dimensional space. Roll-Pitch-Yaw angles are used to express the orientation of the spherical wrist in robots. The orientation of the end-effector can be obtained by a combination of the roll-pitch-yaw angles.

S

Safety Control Board (SCB): Part of the Lexium Cobot Controller that is specifically intended for the safety-related functions of the Lexium Cobot. The Safety Control Board has its own firmware version, which is related to the firmware of the Lexium Cobot Controller.

T

Tool Center Point (TCP): The Tool Center Point (TCP) is the part of the end effector mounted on the tool flange of the Lexium Cobot Arm that comes into contact with the workpiece. The TCP is used for the positioning of the robot in the Cartesian space and must be defined so that the Lexium Cobot Arm can move to the same position from different angles

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

EIO0000004780.05