WEISS ROBOTICS

# GRIPLINK PLUGIN FOR UNIVERSAL ROBOTS

Version 3.0.0

April 2026

# Table of Contents

# 1 Introduction

GRIPLINK technology enables automation components from WEISS ROBOTICS to be integrated into robot systems via a standardized communication interface. The GRIPLINK Plugin for Universal Robots serves as the controller-side interface and enables the connection and operation of compatible WEISS ROBOTICS devices on Universal Robots systems.

The GRIPLINK Plugin supports both devices that use the GRIPLINK protocol natively via TCP/IP and compatible Modbus RTU devices that are connected via the UR controller's Tool I/O interface.

The term "GRIPLINK controller" is used in this manual as a generic term for all products that integrate GRIPLINK technology. These include both standalone GRIPLINK controllers (e.g., GRIPLINK-ET4) and devices with embedded GRIPLINK functionality, such as WPG series or INTRAPAL series servo grippers from WEISS ROBOTICS. All of these products implement the same GRIPLINK protocol and can therefore be controlled in the same way via the plugin.

This manual describes the functions of the GRIPLINK Plugin. For information on installation, commissioning, and operation of the respective device, please refer to the corresponding operating manual. Further information and documents can be found at **https://weiss-robotics.com/**

## 1.1 Notation and Symbols

For clarity, the following symbols are used in this manual:

Function- or safety-related note. Failure to observe these instructions may endanger the safety of personnel and the system, damage the device, or impair its function.

Additional information to aid in understanding the described topic.

Reference to further information.

## 1.2 Intended Use

The GRIPLINK Plugin software is intended for communication between compatible devices from WEISS ROBOTICS and a robot controller from Universal Robots. It enables the integration and control of supported devices via the designated communication interfaces Ethernet and Tool I/O. The requirements of the applicable guidelines as well as the installation and operating instructions in this manual must be observed and adhered to. Any other use or use beyond the intended scope is considered improper. The manufacturer is not liable for any resulting damage.

## 1.3 System Requirements

For Ethernet operation, this plugin is compatible with GRIPLINK protocol version 3 and higher. For Tool I/O operation, the device- and firmware-dependent requirements listed below apply.

***The following Universal Robots controllers are supported:***

- **Ethernet:** UR e-Series with software version 5.5 (or higher) and UR CB3.1 with software version 3.11 (or higher).
- **Tool I/O (Modbus RTU):** UR e-Series with software version 5.4 (or higher).

| Interfaces | Supported devices / firmware | License |
|---|---|---|
| Ethernet | • GRIPLINK-ET4 from FW version 5.2.0<br>• WPG series from FW version 2.4.0<br>• INTRAPAL series from FW version 5.2.0 | - |
| Tool I/O (Modbus RTU) | • GRIPKIT EASY from firmware version 2.1.1<br>• GRIPKIT LS from firmware version 1.1.0<br>• INTRAPAL series from firmware version 5.3.0 | • OPT-GKEASY-MB for GRIPKIT EASY<br>• OPT-IPL-MB for INTRAPAL series |

The IP address of the GRIPLINK controller must be on the same subnet as that of the robot controller. The GRIPLINK controller manual describes the exact procedure for changing the IP address.

The Tool I/O interface is supported by Universal Robots only starting with the e-Series. Robot controllers of the CB-Series, including the CB3.1, do not support the Tool I/O interface. On these systems, the GRIPLINK Plugin can therefore only be used in Ethernet mode.

**Important Note:** Updating the URCap may affect existing systems, configurations, and robot programs. This may require adjustments to the application. Therefore, carefully review the system requirements and changes from previous versions before each update.

It is recommended that you perform a complete functional test of the application after migration.

## 1.4 License Terms

The GRIPLINK Plugin is protected by copyright. The applicable license terms are included with the software package. By installing the software, you accept these license terms.

## 1.5 Demo Programs

The demo programs included in the software package demonstrate the plugin's functionality. They are intended solely for testing purposes!

***ExampleBasicGripCycle***

Simple gripping cycle with GRIPLINK GRIP and GRIPLINK RELEASE. After gripping, a message is displayed via the HOLDING/NO PART branch. Depending on the result, either GRIPLINK RELEASE and GRIPLINK DISABLE or just GRIPLINK DISABLE is executed.

***ExampleFlexGripCycle***

Demonstrates flexible gripping with GRIPLINK FLEXGRIP and GRIPLINK FLEXRELEASE. After the FLEXGRIP command, the device state is read via GRIPLINK GET DEV STATE. Depending on the device state, either FLEXRELEASE and DISABLE are executed, or the program is terminated with a stop (e.g., in case of an error or unexpected condition).

***ExampleAsyncGrip***

Shows how a gripping operation is started asynchronously using `GL_GRIP(...,...,do_block = 0, ...)`. The gripping state is then checked via the program node GRIPLINK GET DEV STATE with the "Wait for state transition (WSTR)" option enabled. Depending on the result, the program branches to GRIPLINK RELEASE and DISABLE, DISABLE only, or a halt if an error condition exists.

***ExampleMultiGrip***

Example of parallel gripping with three grippers on ports 0, 1, and 2. First, the grippers are initialized using GRIPLINK ENABLE and GRIPLINK HOME. Next, a quasi-parallel GRIPLINK GRIP is started with `GL_GRIP(...,...,do_block = 0, ...)`. Afterward, the states are checked via GRIPLINK GET DEV STATE with the "Wait for state transition (WSTR)" option enabled. Depending on the result, a distinction is made between:

- At least one gripper is in S_NO_PART
- All grippers are in S_NO_PART.
- All grippers are in S_HOLDING.

# 2 Installation

## 2.1 Software Installation

Make sure you are using the latest version of the GRIPLINK Plugin. The latest version can be downloaded at **www.griplink.de**.

1. Download the plugin file “griplink_plugin_universalrobots_3.0.0.zip”.
2. Extract the previously downloaded ZIP archive of the GRIPLINK Plugin to the root directory of a USB flash drive and insert it into the USB port of the teach pendant.
3. Open the settings and navigate to the “System/URCaps” menu.

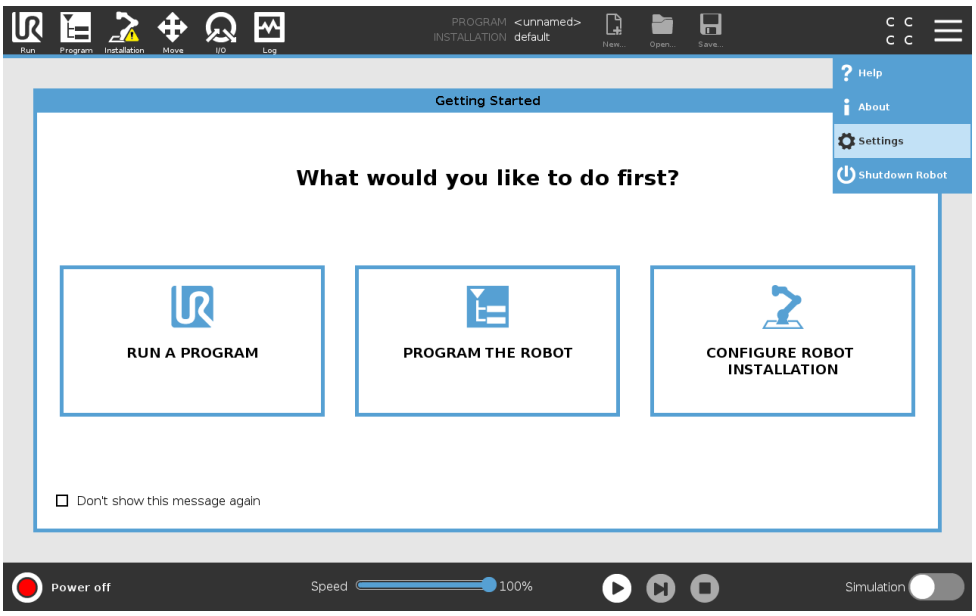



Figure 1: PolyScope 5 - Getting Started.

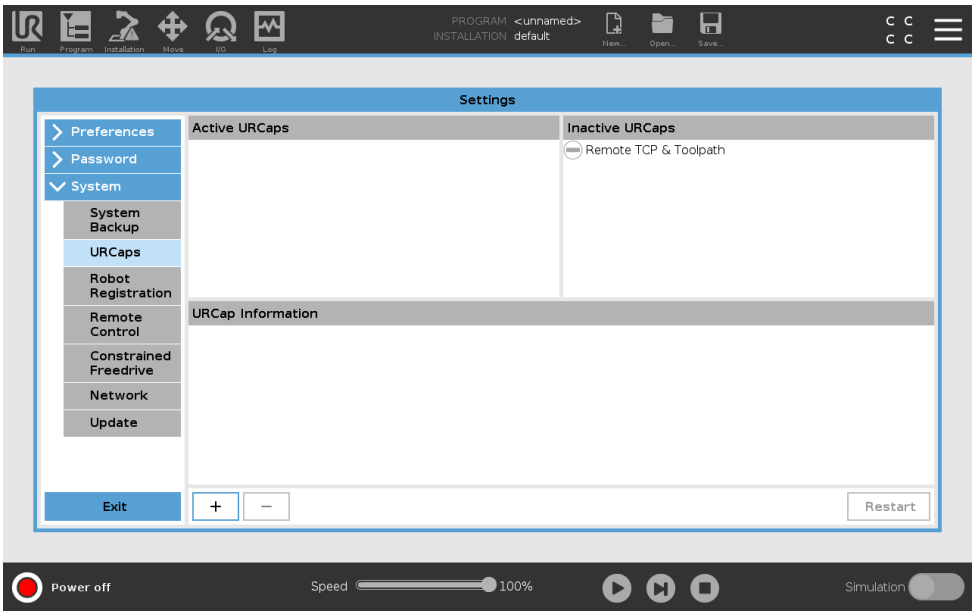



Figure 2: PolyScope 5 - Settings/System/URCaps.

4. Press the "+" button and select the previously extracted .urcap file.

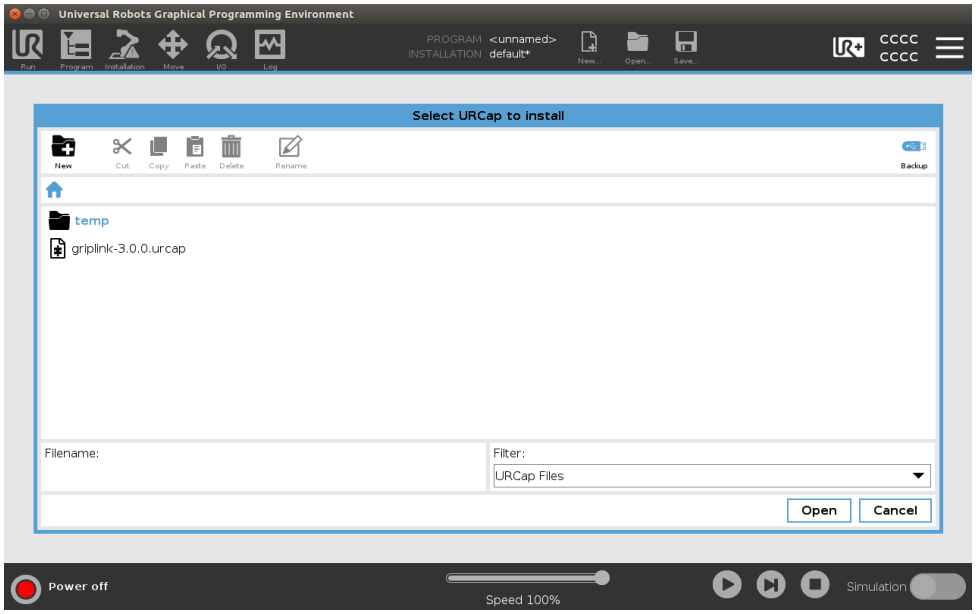



Figure 3: PolyScope 5 - Select URCap to install.

5. Restart the robot by pressing the "Restart" button.

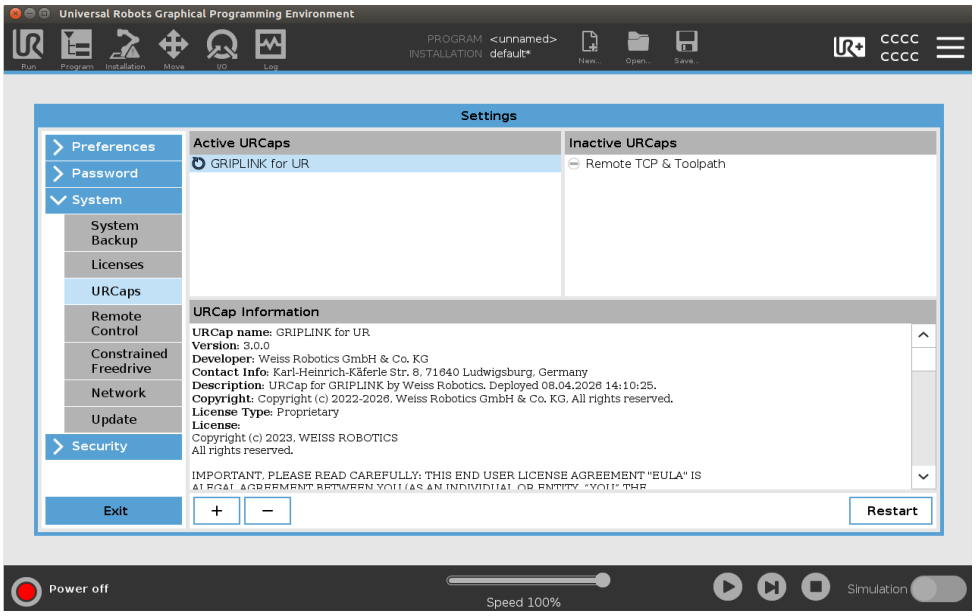



Figure 4: PolyScope 5 - Active URCaps: GRIPLINK for UR.

### 2.1.1 Verifying the Installation

After you have completed the installation process, the "GRIPLINK" entry will appear in the "Installation" main menu, and various GRIPLINK program nodes will appear in the "Program" main menu under the "URCaps" menu item.

## 2.2 Uninstalling the Software

To remove the GRIPLINK Plugin from your robot, follow the instructions in the robot controller manual.

## 2.3 Behavior in Case of an Error

If an error occurs within the GRIPLINK Plugin or during communication with a connected device, execution of the running robot program is stopped and an error message is displayed. As a result, any ongoing robot movements may also be stopped. This prevents the robot from continuing its motion while a communication or device error is present.

An error can occur, for example, if the connection to the GRIPLINK controller is interrupted, invalid parameters are passed, or a connected device reports an error condition. The same applies if the addressed device is already in the FAULT state or switches to this state during the execution of a command.

Additional diagnostic information can be found in the state display in the installation node and—if logging is enabled—in the log files in the ~/.griplink directory on the robot controller. An overview of typical error causes and recommended actions can be found in **Chapter 8 (Troubleshooting)**.

# 3 Hardware Setup

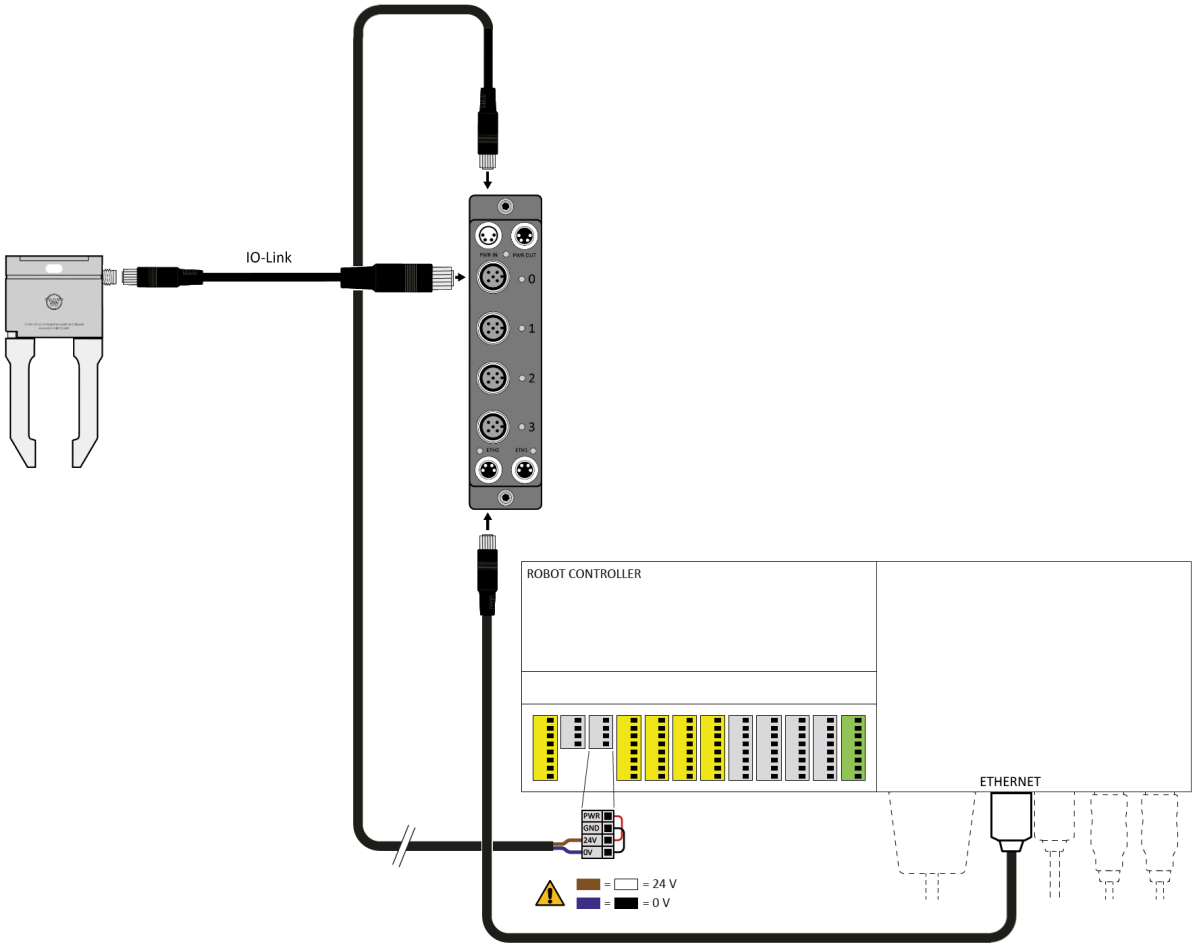



Figure 5: IO-Link gripper connected to the robot controller (Ethernet) via the GRIPLINK controller.

If you are unsure whether the power supply via the robot controller's connection is sufficiently rated to operate all devices connected to the GRIPLINK, use an external power supply!

The power supply must be sufficiently rated for the connected devices!

## 3.1 Mounting on the Robot Flange

Mechanical mounting on the robot flange is described in the operating instructions for the respective device.

Installation, removal, and wiring work may only be carried out when the system is de-energized. Before connecting, disconnecting, or replacing a Tool I/O device, ensure that the Tool I/O interface is at **0 V**.

## 3.2 Power Supply

When using multiple devices, ensure that the power supply is sufficiently rated. This is the only way to guarantee proper and trouble-free operation of all connected devices.

The permissible current consumption of the robot's Tool I/O interface must not be exceeded by connected devices. Exceeding this limit can lead to malfunctions or damage to the robot system as well as to the connected devices.

Please refer to the information in the robot's operating manual regarding the permissible current consumption limits, as well as the current consumption specifications in the operating manual of the respective device.

## 3.3 Tool I/O Topology

In Tool I/O mode, communication takes place via the robot's Tool I/O interface. The port mapping described in this section applies to the Tool I/O integration of the GRIPLINK Plugin and must be distinguished from the generic port definition of the GRIPLINK Unified Command Set. The logical port mapping is based on the respective Modbus station address. Each station address maps to exactly one master with a main port and up to three associated subports.

For gripper modules with a finger interface, the fingers are assigned to the corresponding subports of the respective main port. For example, if a device is connected to main port 0, finger 0 and finger 1 are mapped to ports 1 and 2. If a device is connected to main port 4, the corresponding assignment is to ports 5 and 6.

| Main Port | Station Address (Modbus RTU) | Assigned Subports |
|---|---|---|
| 0 | 1 | 1, 2, 3 |
| 4 | 2 | 5, 6, 7 |
| 8 | 3 | 9, 10, 11 |
| 12 | 4 | 13, 14, 15 |

When using multiple devices, ensure that the wiring is correct and that the communication interface is configured appropriately for the installation.

The Tool I/O interface must only be operated with the original Tool I/O cable for the respective device. Other cables, applications, or wiring may impair the data connection and cause damage.

# 4 Plugin Functionality

## 4.1 Plugin Structure

The GRIPLINK Plugin consists of the installation node for basic configuration, the GRIPLINK toolbar for commissioning and diagnostics, as well as the program nodes and URScript functions for use in the robot program.

The graphical program nodes and the URScript functions access the same communication interface. This allows supported devices to be addressed both via the PolyScope user interface and directly from script code. The toolbar is intended exclusively for manual operation, diagnostics, and commissioning; changes to the actual process flow must be made in the robot program.

## 4.2 Operation via Program Nodes

After installation, the plugin provides various program nodes that can be inserted into a robot program just like other robot commands.

## 4.3 Operation via URScript

The plugin includes a set of basic functions in URScript that can also be called independently from the robot program using the corresponding script program node. These basic functions are automatically integrated into the robot program when the plugin is installed. Section 7 describes the available functions in more detail.

# 5 Preparing the Robot

The basic settings for the GRIPLINK Plugin must be configured during installation.

In Ethernet mode, the GRIPLINK controller is configured via its web interface, which can be accessed at the set IP address (default: 192.168.1.40).

In Tool I/O mode, device-specific parameterization of GRIPKIT EASY can be performed as needed using the "GRIPKIT Configurator" application.

## 5.1 Setup

### 5.1.1 Ethernet Interface

Enter the IP address of the GRIPLINK controller connected to the robot in the "GRIPLINK Controller IP Address" field. The default IP address is 192.168.1.40 and is located in the 192.168.1.0/24 subnet. This IP address can be adapted to the respective plant network. The robot controller and the GRIPLINK controller must be on the same subnet.

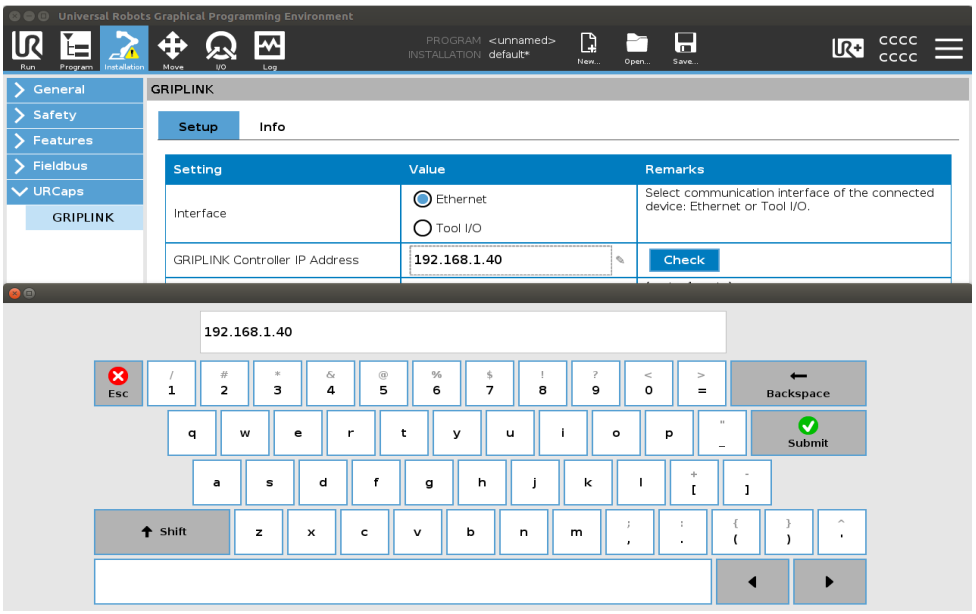



Figure 6: PolyScope 5 - Installation/URCaps/GRIPLINK/Setup (Change IP Address).

Use the "Check" button to verify whether the configured GRIPLINK controller is connected.

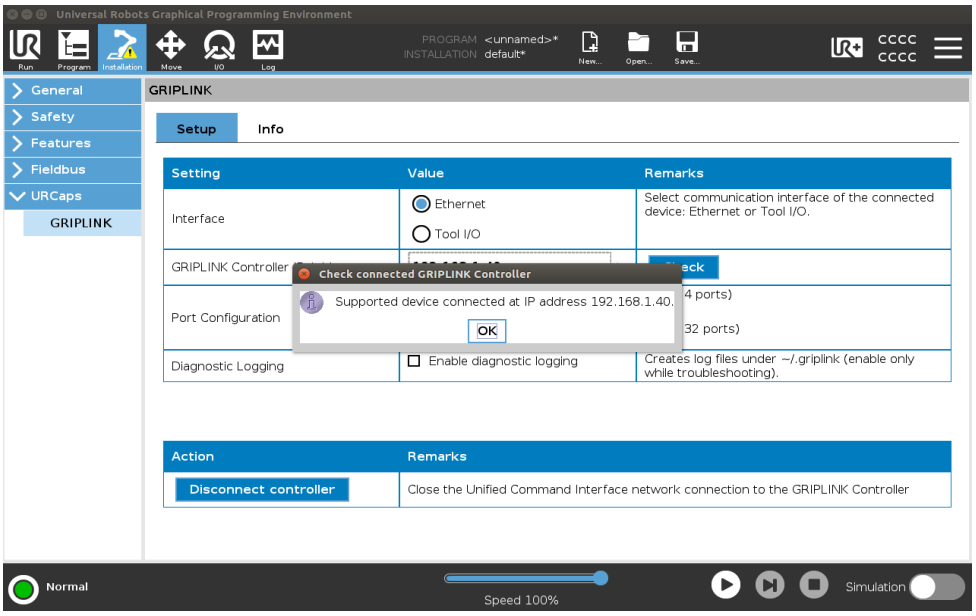



Figure 7: Installation/URCaps/GRIPLINK/Setup: Check connected GRIPLINK controller.

If a connection between the robot controller and the GRIPLINK controller is active, no gripping commands can be executed via the action buttons of the program nodes. To disconnect the robot controller from the GRIPLINK controller, use the "Disconnect" button.

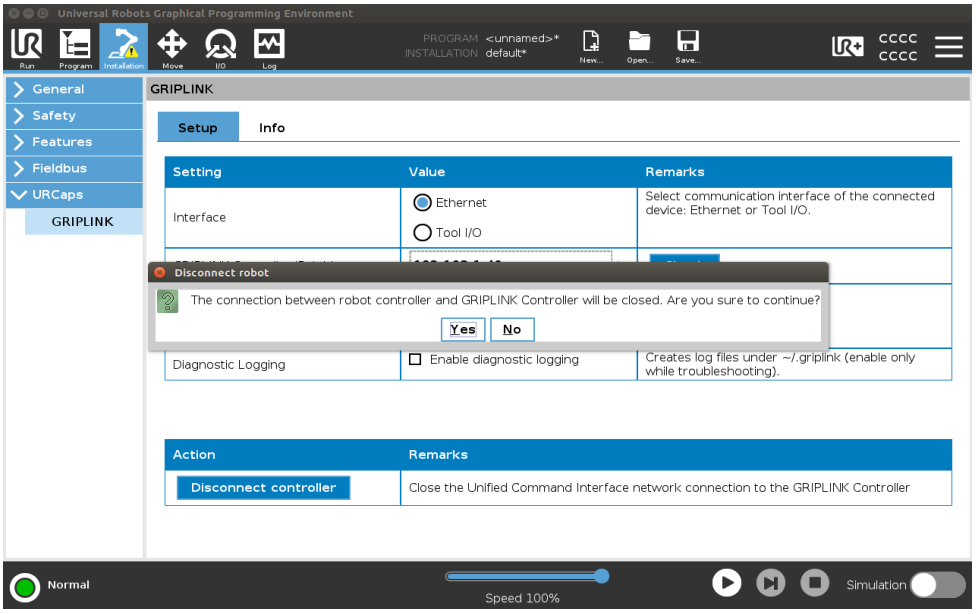



Figure 8: PolyScope 5 - Installation/URCaps/GRIPLINK/Setup: Disconnect (confirmation dialog).

Do not disconnect the connection while a robot program is running! Risk of damage and injury from falling parts!

### 5.1.2 Tool I/O Interface

To use the Tool I/O interface, the I/O Interface must be correctly configured in the Installation tab of PolyScope. To do this, select the option “GRIPLINK for UR” from the “Controlled by” dropdown menu.

For Tool I/O operation, the following communication parameters are automatically set by the plugin: Baud Rate 115200, Parity None, Stop Bits One, RX Idle Chars 1.5, and TX Idle Chars 3.5.

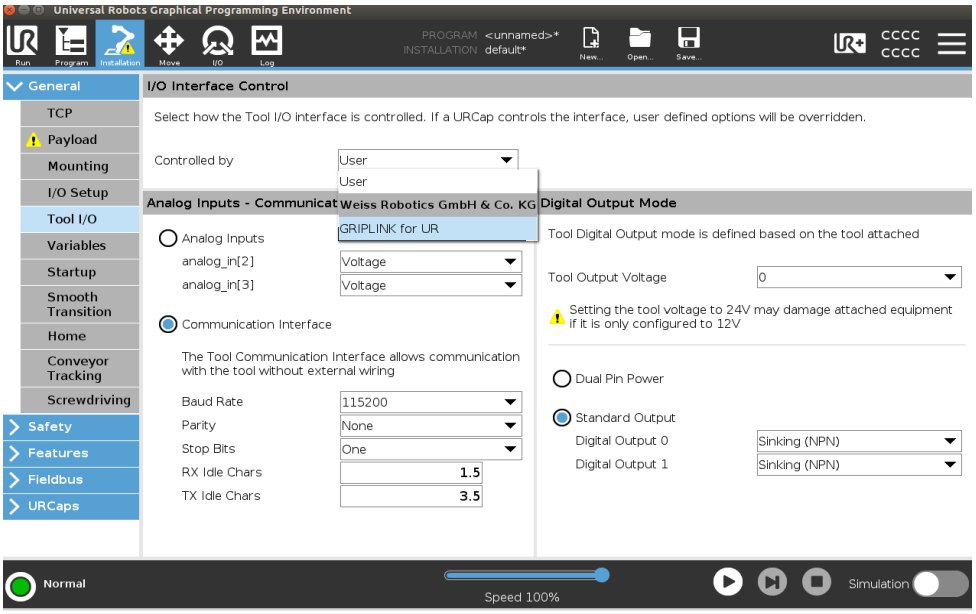



Figure 9: PolyScope 5 – Configuration of the Tool I/O interface with “GRIPLINK for UR” selected as the control method.

If the interface is not configured correctly, the connected devices cannot be controlled via the URCap.

During operation, only the GRIPLINK Plugin is permitted to access the Tool I/O interface. The simultaneous use of other URCaps or functions that also control the Tool I/O interface can lead to conflicts, communication errors, or unreliable operation.

Two interactive indicators are displayed in the installation node: the Tool I/O access and the state display.

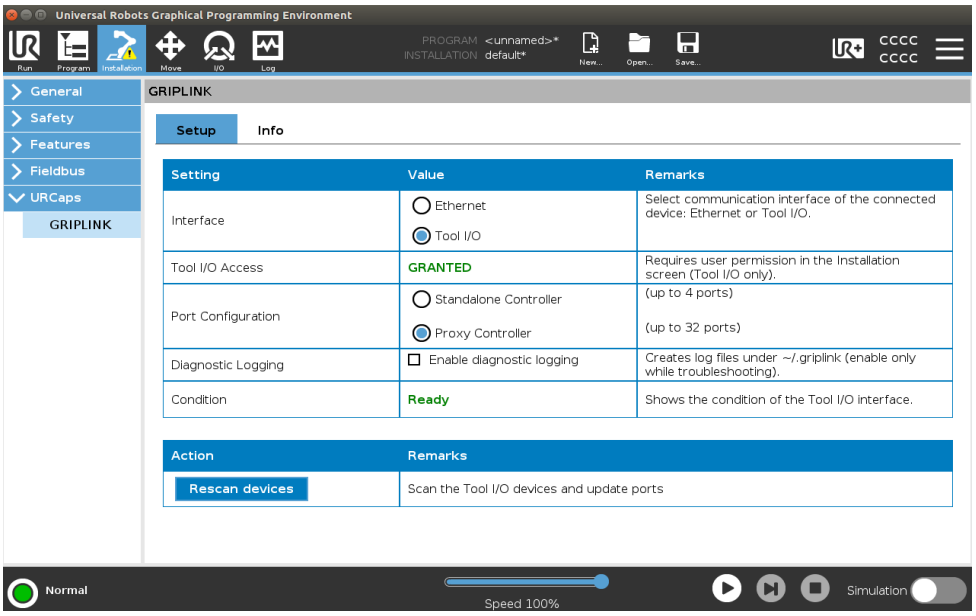



Figure 10: Tool I/O interface with granted access and status display.

The status display provides information about the current operating status of the Tool I/O interface and assists with diagnostics during commissioning and operation.

| Tool I/O Access | Meaning |
|---|---|
| NOT_GRANTED | The user has not selected “GRIPLINK for UR” in the robot’s Tool I/O settings. The URCap therefore has no access to the Tool I/O interface. |
| GRANTED | "GRIPLINK for UR" is selected. The URCap is permitted to access the Tool I/O interface. |

| Status Indicator | Meaning |
|---|---|
| Tool I/O Interface Error | Error in the Tool I/O interface. Check access, wiring, and the connected device. |
| Waiting for Tool I/O Access | Access to Tool I/O has not yet been granted. Select "GRIPLINK for UR" in the Tool I/O settings. |
| Scanning devices | The device search is currently in progress. Once the search is complete, the detected devices, including port assignments, will be displayed. |
| Waiting for device | No device has been found yet. Connect a compatible device and then perform a rescan. |
| Ready | At least one compatible device has been found; the Tool I/O interface is ready for operation. |

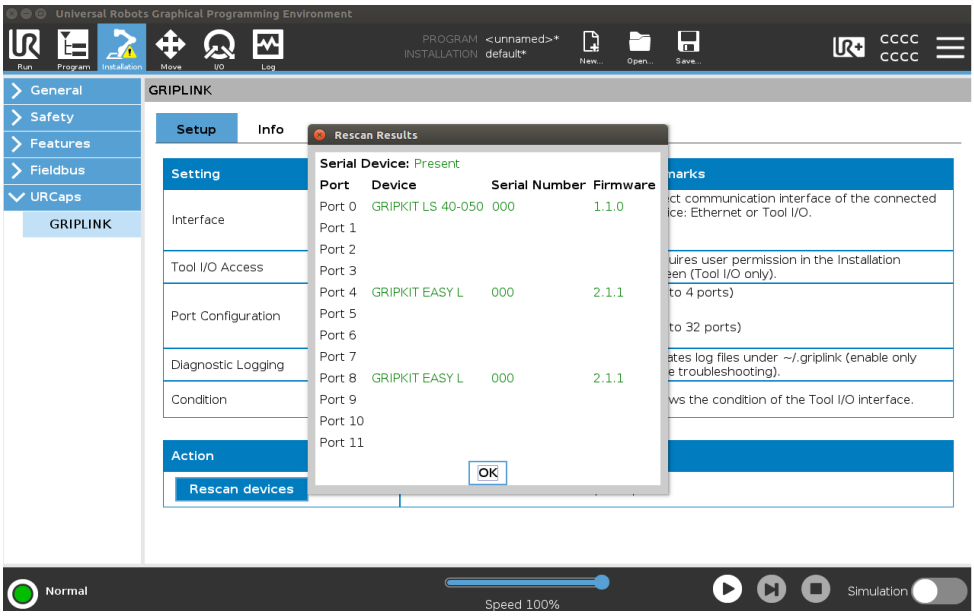



Figure 11: Rescan result in the Tool I/O interface showing detected devices, port assignments, serial numbers, and firmware versions.

The **"Rescan devices"** action performs a new device search on the Tool I/O interface. The currently detected devices, including port assignment, serial number, and firmware version, are displayed and updated in the URCap. A rescan is particularly useful after initial commissioning, after a tool change, or if a connected device has not yet been detected.

#### 5.1.2.1 Commissioning the Tool I/O Interface

***The following sequence is recommended for commissioning the Tool I/O interface:***

1. Ensure that the Tool I/O interface is at 0 V before connecting the device.
2. Mechanically mount a compatible Tool I/O device and connect the original Tool I/O cable.
3. In the robot's Tool I/O settings, under **"Controlled by,"** select the **"GRIPLINK for UR"** option.
4. In the installation node of the GRIPLINK Plugin, select the **"Tool I/O"** communication interface.
5. Save the installation after selecting the "Tool I/O" communication interface in the GRIPLINK Plugin.
6. Check whether the access status in the GRIPLINK Plugin installation node shows **"GRANTED"**.
7. If no device is detected yet, run **"Rescan devices."**
8. Before the first production run, verify that the device has been detected correctly. If necessary, also use the **"GRIPLINK Check Device"** program node.

#### 5.1.2.2 Tool Change at the Tool I/O interface

***The following sequence is recommended for a safe tool change at the Tool I/O interface:***

1. Terminate the running robot program and bring the robot to a safe stop.

2. In the robot's Tool I/O settings, first select **"Controlled by: User."**
3. Ensure that the voltage at the Tool I/O interface is **0 V** before disconnecting any cables or tools.
4. Disconnect the Tool I/O cable and remove the previous tool while it is de-energized.
5. Mechanically install the new tool and reconnect the Tool I/O cable.
6. Then, in the Tool I/O settings, select **"Controlled by: GRIPLINK for UR"** again.
7. In the GRIPLINK Plugin, select the **"Tool I/O"** interface and then execute **"Rescan devices."**
8. Before starting production, verify that the newly installed tool has been correctly recognized and that the Tool I/O interface is operational.

### 5.1.3 Port Configuration

Below the IP address, the operating mode of the GRIPLINK controller is selected using two radio buttons.

***Single Controller***

The robot controller communicates with a single GRIPLINK controller. Ports 0 through 3 can be selected in the program nodes; they correspond to the four IO-Link ports of this GRIPLINK controller.

***Proxy Controller***

The robot controller communicates with a GRIPLINK controller that acts as a proxy master for a network of multiple GRIPLINK controllers.
From the URCap's perspective, this makes a GRIPLINK controller with up to 32 ports available, which can be addressed in the program nodes via port numbers 0 through 31.
The proxy configuration (roles, licenses, etc.) is performed via the GRIPLINK controller's web interface.

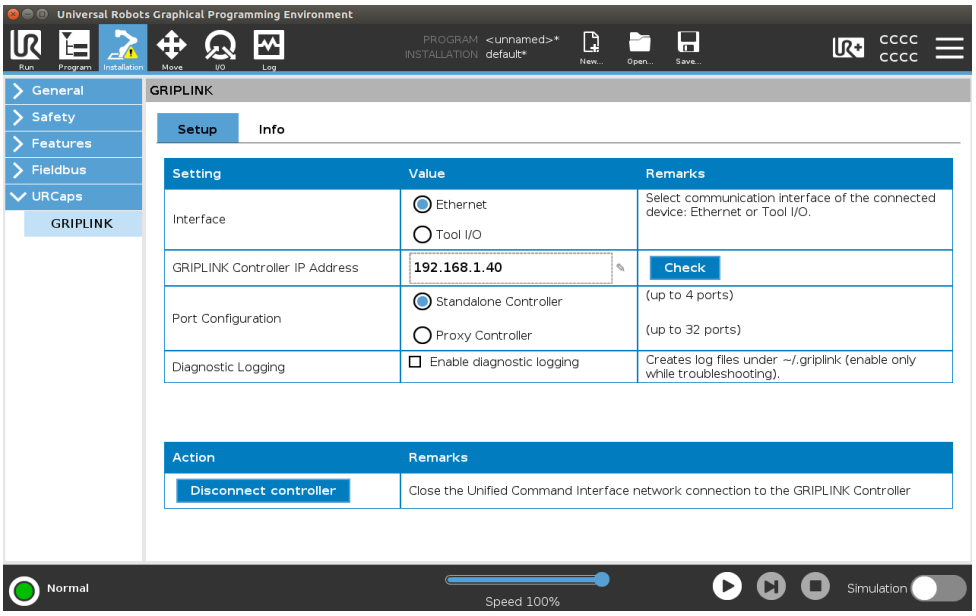



Figure 12: Installation/ URCaps/ GRIPLINK/Setup: Select Controller Mode (Single vs. Proxy).

### 5.1.4 Logging Configuration

The "Enable Logging" checkbox turns logging for the GRIPLINK Plugin on or off. By default, logging is disabled.

***Logging enabled***

Relevant information, warnings, and error messages from the plugin are logged in log files in the `~/.griplink` directory on the robot controller.

***Logging disabled***

No additional GRIPLINK-specific log entries are generated.

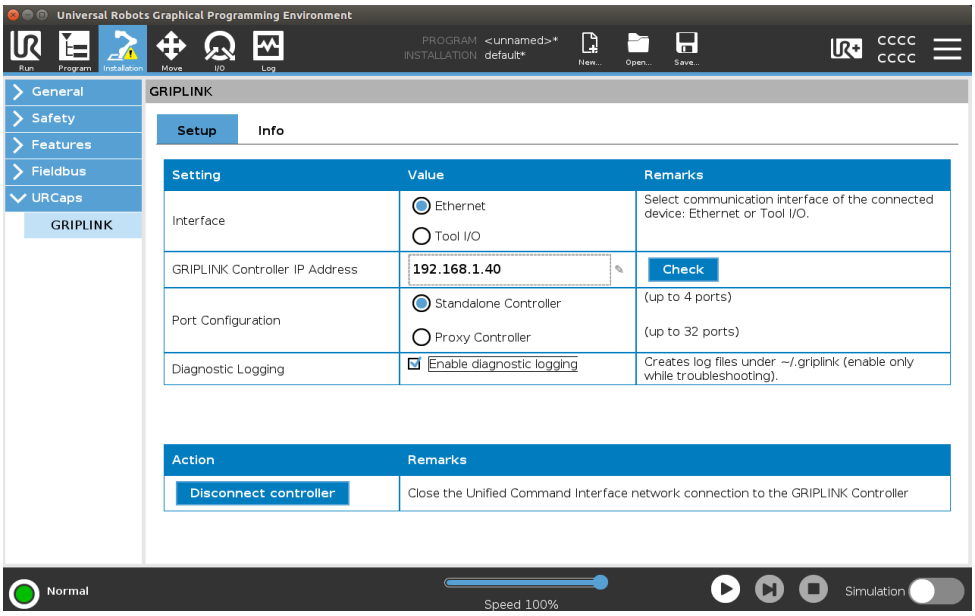



Figure 13: PolyScope 5 - Installation/URCaps/GRIPLINK/Setup: Proxy Controller selected & Logging enabled.

- After checking the checkbox, logging begins as soon as the GRIPLINK installation page is opened.
- The log files are used for diagnostics and troubleshooting related to the GRIPLINK Plugin.
- For regular operation, it is recommended that you disable logging to limit the volume of logs and reduce storage requirements.

# 6 GRIPLINK Toolbar

The GRIPLINK toolbar provides a user-friendly interface for the connected devices supported and recognized by the current system. It can be displayed within the PolyScope 5 view and supports the following tasks in particular:

- Commissioning
- Diagnostics
- Gripping and releasing during setup and testing
- Viewing presets
- Monitoring important device data during programming

The toolbar displays the data and provides the supported functions for the currently selected port.

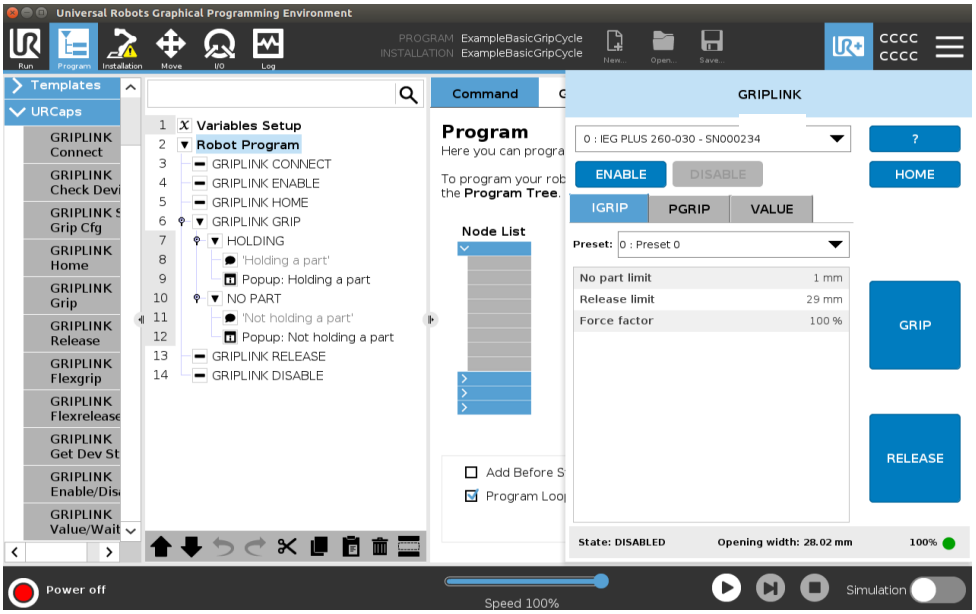



Figure 14: PolyScope 5 - GRIPLINK Toolbar.

## 6.1 Toolbar Layout

The upper section of the toolbar contains the device selection combo box as well as the buttons for controlling the selected device.

***Device selection (combo box)***

Dropdown menu for selecting the desired device, e.g.

- 0: IEG PLUS 260-030 – SN000000

The port number, device type, serial number, or an optional tag configured on the GRIPLINK controller are displayed.

***ENABLE / DISABLE (Button)***

Enables or disables the selected device. The current state is displayed in the state bar, e.g.

- State: DISABLED
- State: RELEASED
- State: HOLDING
- State: INVALID

***HOME (Button)***

The HOME button performs a homing procedure for the selected device.

The current device state, telemetry data, and the device's status indicator for monitoring are displayed cyclically in the lower section of the toolbar.

***State***

State displays the current device state (e.g., RELEASED, HOLDING, FAULT).

***Telemetry Data***

The telemetry data contains device-specific measured values, e.g., the opening width (in mm) for gripper modules.

***Status display***

The status indicator visualizes the current device status. Green indicates an operational state, while red indicates a fault state.

## 6.2 IGRIP Tab - Working with Grip Presets

The **IGRIP tab** displays the configured grip presets for the selected device.

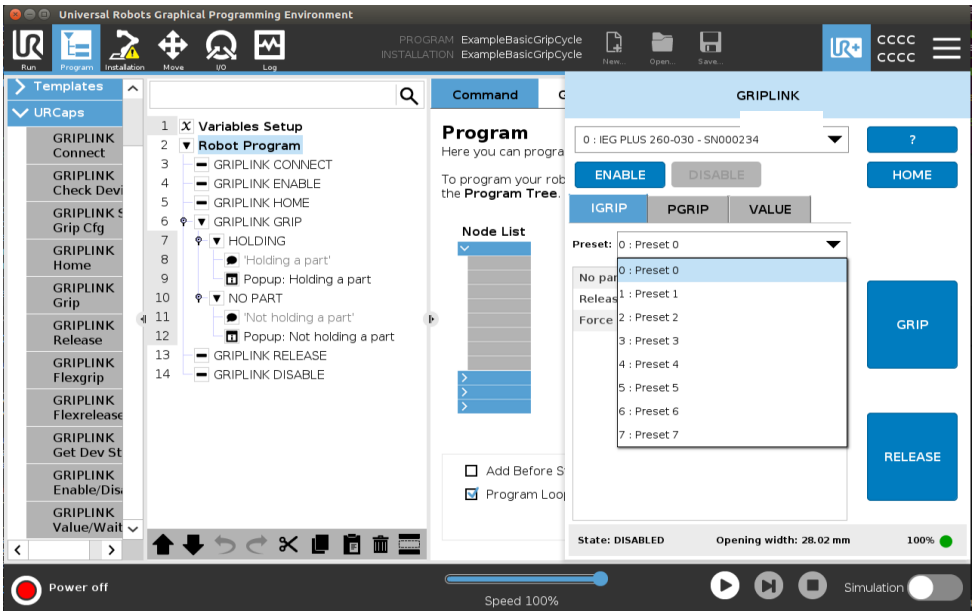



Figure 15: GRIPLINK Toolbar (IGRIP tab): Preset selection open.

***Preset selection (combo box)***

Dropdown menu with the available grip presets 0…7. Each row displays the index and the tag, e.g.
0: Workpiece_1.

***Preset Parameter Display (Table)***

Depending on the device, the following are displayed, among other things:

- No part limit (mm)
- Release limit (mm)
- Force factor (%)

***GRIP (button)***

The GRIP button performs a gripping operation for the selected preset.

***RELEASE (Button)***

The RELEASE button performs a release operation for the selected preset.

***Typical use***

- Testing a newly configured preset.
- Viewing presets and their configured parameters in the toolbar.
- Quick teach-in: Select the preset on the GRIPLINK controller, then execute several grip cycles via the toolbar and observe the behavior.
- Checking whether a component is reliably detected with the set limits.

## 6.3 PGRIP Tab - Flexgrip/Flexrelease

The **PGRIP tab** enables parameterized, flexible gripping and releasing (analogous to the **"GRIPLINK Flexgrip"** and **"GRIPLINK Flexrelease"** program nodes**)**.

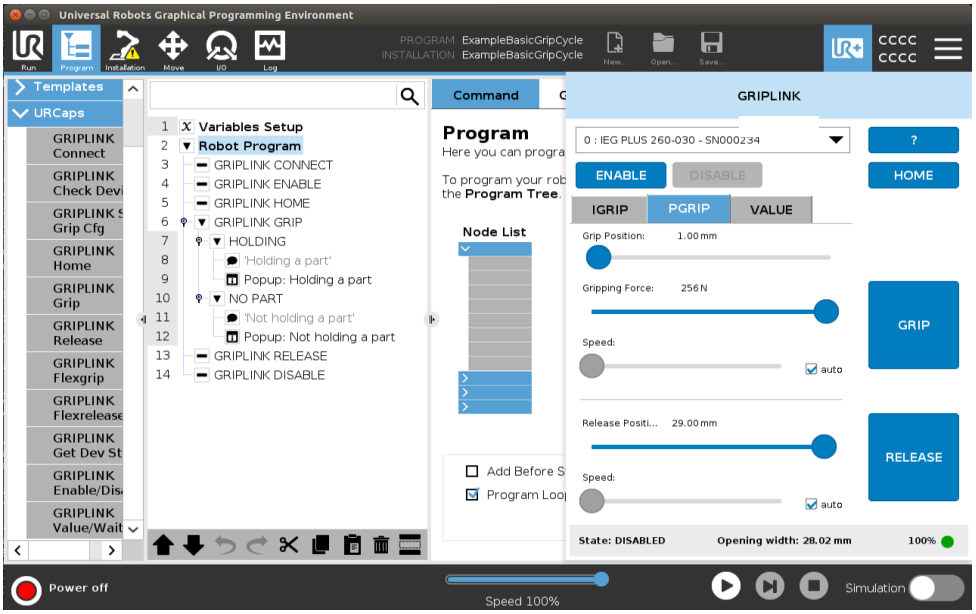



Figure 16: GRIPLINK Toolbar (PGRIP tab): Position/Force/Speed.

***Available controls***

- Grip Position: Target position for gripping in mm (slider).
- Gripping Force:  Gripping force in N (slider).
- Speed: gripping speed; in Ethernet mode in mm/s and in Tool I/O mode as a percentage (%) of the maximum device speed (Optional via slider).
- Release Position: Target position for release in mm (slider).
- GRIP/RELEASE: Start the corresponding movement using the set parameters.

***Applications***

- Determining a suitable gripping position and gripping force for a new component.
- Testing how gently or quickly a part can be gripped or released.
- Checking tolerances (e.g., different workpiece widths) before creating a fixed preset.

## 6.4 VALUE Tab - Monitor Device Data

The **VALUE tab** displays the current values of the selected device. The available indices depend on the device type. The values are updated cyclically.

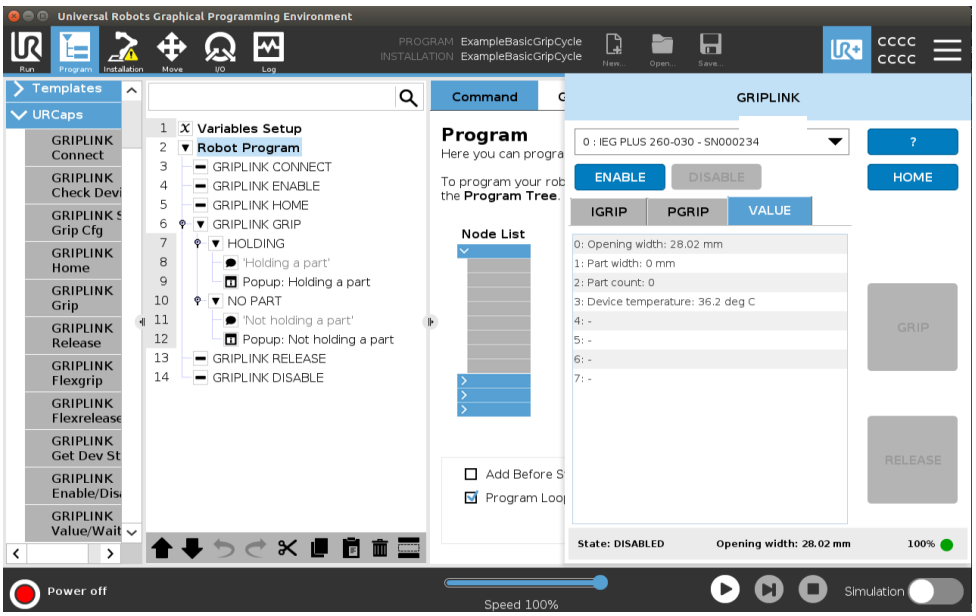



Figure 17: GRIPLINK Toolbar (VALUE tab) - Display telemetry data/status values.

***Example for the IEG PLUS series***

- Index 0: Opening width (mm)
- Index 1: Part width (mm)
- Index 2: Part count
- Index 3: Device temperature (°C)
- Index 4–7: Not used in the IEG PLUS series

***Typical application***

- Checking whether a component has been correctly detected (part width / part count).
- Monitoring the opening width during a program run.
- Checking the device temperature during long cycles.
- Assistance in selecting appropriate limit values (No Part Limit, Release Limit, etc.).

## 6.5 Notes on Using the toolbar

While a robot program is running, the toolbar actions must not be used. Devices cannot be controlled via the toolbar while a program is active.

If buttons are pressed multiple times in quick succession, they are temporarily disabled and then automatically reactivated. This prevents overloading of communication and robot control.

The toolbar interface depends on the current device state and device properties. Controls are automatically enabled or disabled depending on the state (e.g., `DISABLED`, `FAULT`, `HOLDING`).

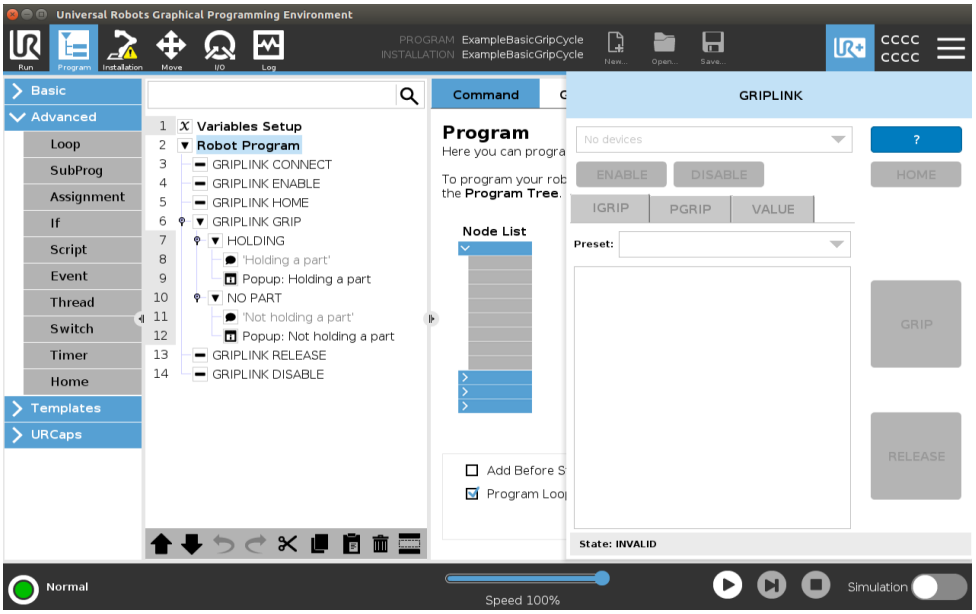



Figure 18: GRIPLINK Toolbar – No devices connected and controls disabled.

The toolbar does not modify existing program nodes. It is intended solely for manual operation, diagnostics, and commissioning. Changes intended to have a permanent effect on the sequence must be made in the robot program.

# 7 Program Nodes

## 7.1 Basic Program Flow

Every robot program that controls devices such as grippers and sensors via GRIPLINK should follow the design guidelines below. This ensures robust initialization of the devices and a reproducible sequence during operation.

### 7.1.1 Initialization Section (e.g., BeforeStart)

In the initialization section, the connection to the GRIPLINK controller is established and all connected devices are prepared.

***Establishing a connection to the GRIPLINK controller***

The GRIPLINK CONNECT program node is inserted at the beginning of the robot program. It establishes the network connection to the GRIPLINK controller configured in the installation.

***Initializing devices***

For each device connected to the GRIPLINK controller, an initialization section is executed once, for example in the form of a subprogram.

### 7.1.2 Typical sequence per device

***GRIPLINK Check Device***

Checks the corresponding port to see if the expected device is connected (by comparing the Vendor ID and Product ID).

***GRIPLINK Set Grip Config, GRIPLINK Home, and GRIPLINK Enable:***

- Pre-configuration of grip presets (no-part limit, release limit, force factor, etc.).
- Homing of the device.
- Activation of the device for subsequent operation.


Figure 19: Program nodes in the initialization section (BeforeStart).

### 7.1.3 Main program section

In the main part of the robot program, the devices are used in a cycle:

***Gripping and releasing components***

- using predefined grip presets via GRIPLINK Grip and GRIPLINK Release.
- or using parameterized gripping operations via GRIPLINK Flexgrip/Flexrelease.

***Read, check, and monitor device state and values***

- Check device states with **GRIPLINK Get Dev State** (see Appendix A for all device states).
- Read and monitor device values using **GRIPLINK VALUE/WAIT** (e.g., opening width, part width, part count).

Following this recommended procedure supports robust initialization and reproducible operation.

The following sections describe the available commands of the plugin. Each command can be used both as URScript code and, where available, as a graphical program node.

## 7.2 Establish Connection - GRIPLINK Connect

This command establishes the connection between the GRIPLINK controller and the robot controller. The IP address set on the plugin's installation page is displayed.

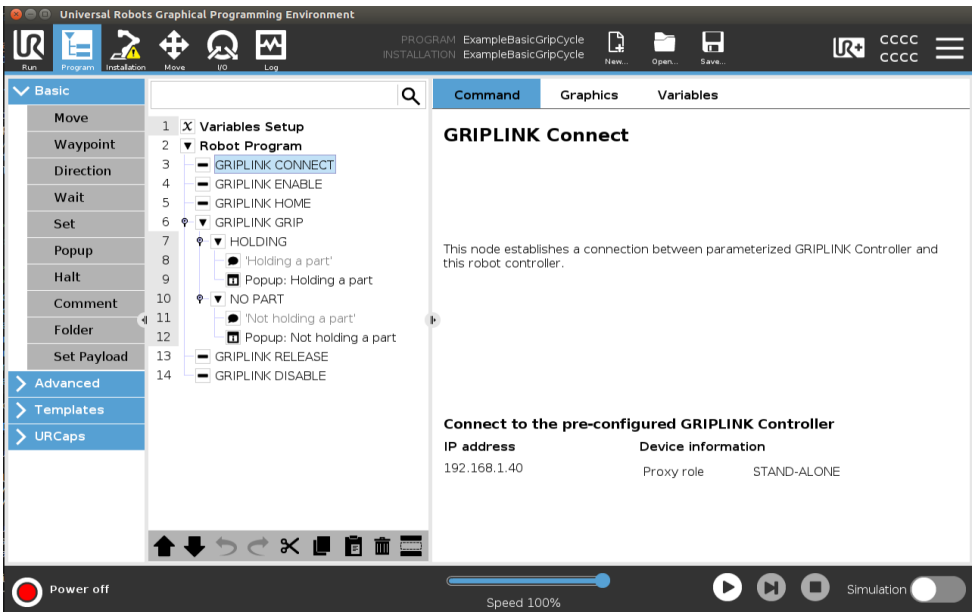



Figure 20: GRIPLINK CONNECT - Connection to the GRIPLINK controller.

If the GRIPLINK controller is accessible at the IP address set during installation, additional information such as the proxy role is displayed.
If not, the connection must be checked.

The Tool I/O interface is addressed via the IP address 127.0.0.1. Therefore, the IP address 127.0.0.1 must be used when calling the URScript function `GL_CONNECT()`. The GRIPLINK socket is identical for both interfaces and is named "sock_griplink".

***Command call with URScript code***

```
GL_CONNECT(
      <GRIPLINK_IP>,
      <SOCKET_NAME>
)
```

| Parameters | Type | Meaning |
|---|---|---|
| <GRIPLINK_IP> | String | IP address of the GRIPLINK controller<br>E.g., "192.168.1.40" |

| | | |
|---|---|---|
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

Example for the Ethernet interface:

```
GL_CONNECT("192.168.1.40","sock_griplink")
```

Example for the Tool I/O interface:

```
GL_CONNECT("127.0.0.1","sock_griplink")
```

## 7.3 Checking a Connected Device - GRIPLINK Check Device

Before using a device in the program, it is recommended to verify that the expected device is actually connected. The "GRIPLINK Check Device" program node checks the selected port to see if the Vendor and Product IDs (VID/PID) of the connected device match the expected values. The robot program is stopped immediately if they do not match.

For more information on Vendor and Product IDs, refer to the operating instructions for the respective device.

The port, vendor ID, and product ID can be configured in the settings. In the example, a CRG 30-050 from WEISS ROBOTICS (VID 815, PID 40) is expected at port 0.

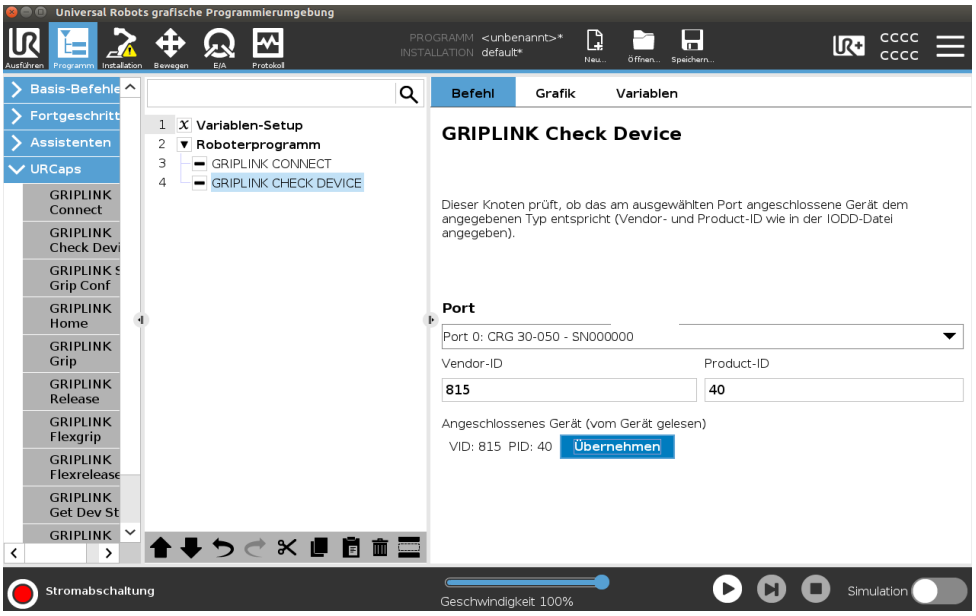



Figure 21: GRIPLINK CHECK DEVICE - Device check at the port based on Vendor ID and Product ID.

***Command call with URScript code***

```
GL_DEVASSERT(
      <PORT>,
      <VENDOR_ID>,
      <PRODUCT_ID>,
      <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |

| | | |
|---|---|---|
| <VENDOR_ID> | Integer | Vendor ID of the expected device |
| <PRODUCT_ID> | Integer | Product ID of the expected device |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Ensure that a WEISS ROBOTICS IEG 55-020 (VID 815, PID 20) is
connected
# to port 0
GL_DEVASSERT(0,815,20,"sock_griplink")
```

## 7.4 Home Device - GRIPLINK Home

If necessary, a device must be homed at the beginning. The "GRIPLINK Home" program node is used for this purpose.

***Port (combo box)***

Select the port to which the device is connected. In online mode, the drop-down list displays the connected devices with

- device type,
- serial number, and
- if applicable, a freely selectable tag that can be configured on the GRIPLINK controller.

This allows the desired gripper or sensor to be uniquely identified and selected directly from the dropdown list.

***Actions Home / Enable / Disable (Button)***

The buttons at the bottom allow the following actions to be performed directly for the selected port:

- Home: Start the device’s homing sequence
- Enable: Activate the device
- Disable: Deactivate the device

These functions are particularly helpful during commissioning and testing of the application, as they allow devices to be controlled selectively without having to run the entire robot program.

Further details can be found in the operating instructions for the respective device.

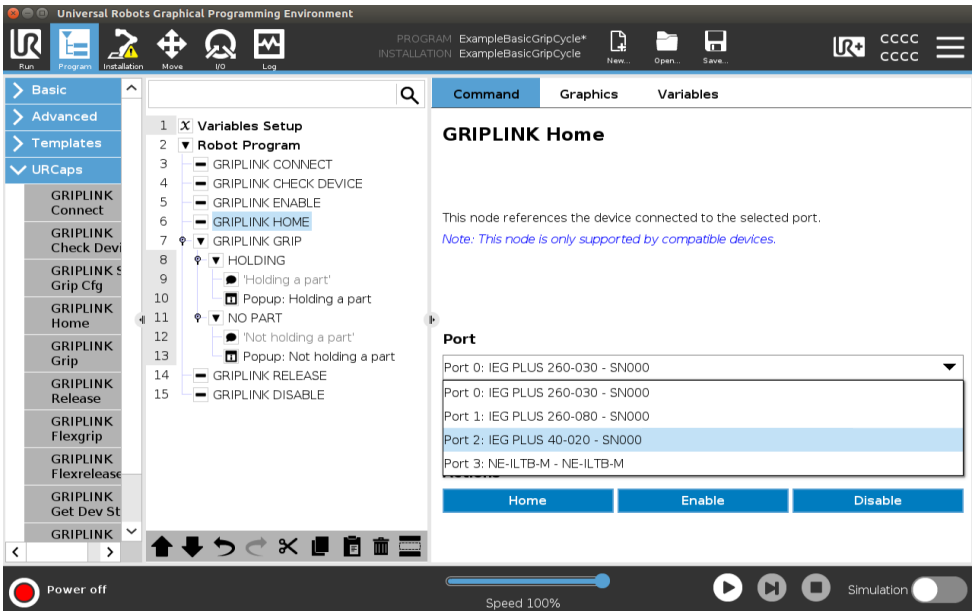



Figure 22: GRIPLINK Home - Device homing (port selection).

A port must be selected!

***Command call with URScript code***

```
GL_HOME(
      <PORT>,
      <SOCKET_NAME>
)
```

| Parameters | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br><br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
      # Home device connected to port 0
      GL_HOME(0,"sock_griplink")
```

## 7.5 Enable and disable device - GRIPLINK Enable/Disable

Devices can be enabled and disabled during operation. The "GRIPLINK ENABLE/DISABLE" program node combines both functions into a single program node. The following settings are available.

***Port (combo box)***

Select exactly one port whose device is to be enabled or disabled. A port must be selected for the command to be executed.

***Action Selection (combo box)***

Select the action to be performed:

- **Enable**: Enables the device on the selected port
- **Disable**: Disables the device on the selected port

The selected action is stored in the program. The program node adjusts its title according to the user's selection, depending on whether the node is executed as **Enable** or **Disable**.

***Wait for state transitions (Checkbox)***

This option determines whether the program checks, after executing the command, whether the device state on the selected port has actually changed.

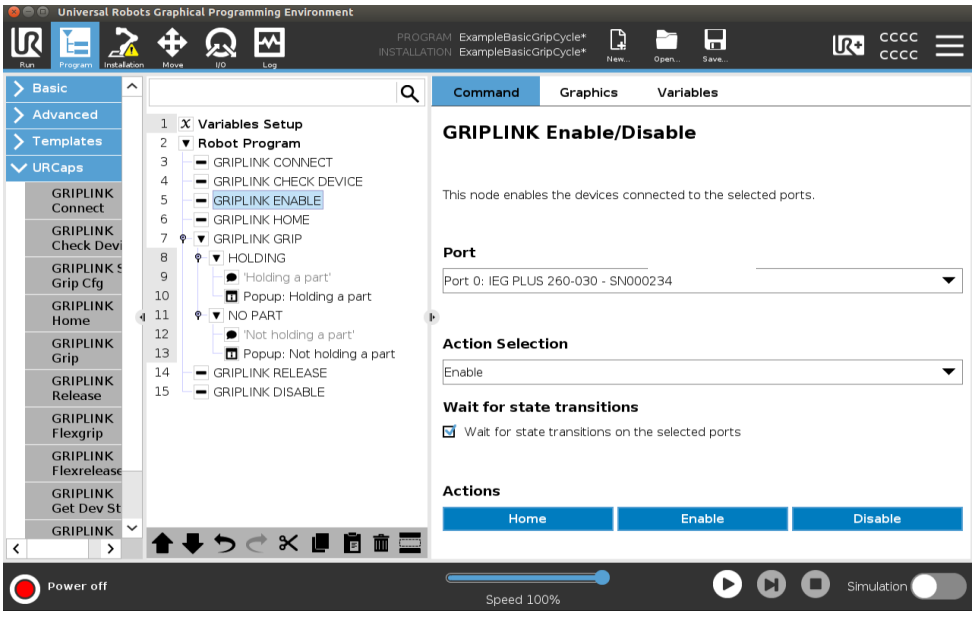



Figure 23: GRIPLINK Enable/Disable - Enable action (with Wait for state transitions).

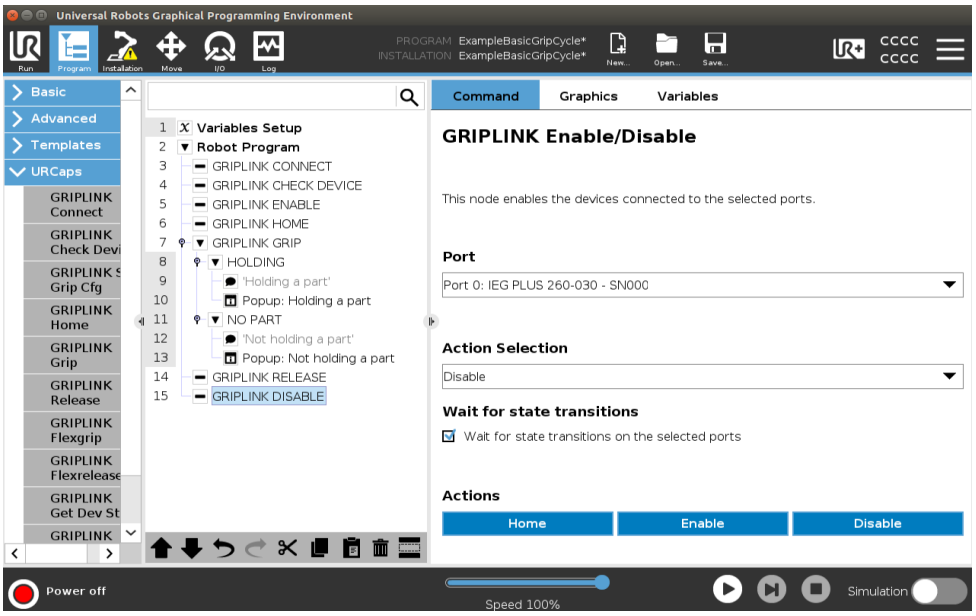



Figure 24: GRIPLINK Enable/Disable - Disable action (with Wait for state transitions).

Some devices must first be enabled using the Enable command before they can be used! Further details can be found in the operating instructions for the respective device driver.

Ensure that the devices are in a valid state when the "Wait for state transitions" checkbox is selected!

If a device is already in the DISABLED state when the "GRIPLINK Disable" program node is called with the "Wait for state transitions" checkbox selected, this may result in a timeout error.

***Command call with URScript code***

```
GL_ENABLE(
     <PORT>,
     <WSTR_ENABLED>,
     <SOCKET_NAME>
)

GL_DISABLE(
     <PORT>,
     <WSTR_ENABLED>,
     <SOCKET_NAME>
)
```

| Parameters | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |

| | | |
|---|---|---|
| <WSTR_ENABLED> | Boolean | "True" to wait for the device's state transition<br>"False" to not wait for the device's state transition |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br><br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Disable device connected to port 0, wait until disabled
GL_DISABLE(0,True,"sock_griplink")

# Enable device connected to port 0 again, wait until enabled
GL_ENABLE(0,True,"sock_griplink")
```

## 7.6 Gripping and Releasing - GRIPLINK Grip/Release

Every gripper module supports the basic commands “Grip” and “Release.” Depending on the gripper module, up to eight freely configurable grip presets can be executed.

For more information on gripping operations and their parameterization, refer to the operating instructions for the respective gripper module.

***In the "GRIPLINK Grip" and "GRIPLINK Release" program nodes,***

- the port of the desired gripper and
- the index of the grip preset

can be selected. For the Release node, the blocking behavior (blocking / non-blocking) can also be set.

***Blocking behavior***

- **GRIPLINK Grip:** The "GRIPLINK Grip" graphical program node is set to blocking by default. The robot program remains at this program node until the gripping operation is complete.
- **GRIPLINK Release:** For the "GRIPLINK Release" node, you can use the checkbox to select whether the command is executed in blocking or non-blocking mode.

In **blocking mode**, the command is sent to the GRIPLINK controller and is only acknowledged once the gripping or release operation is complete.
In **non-blocking mode**, the gripping or release operation is started and the node is completed immediately; the robot program continues while the gripper is still gripping or releasing.

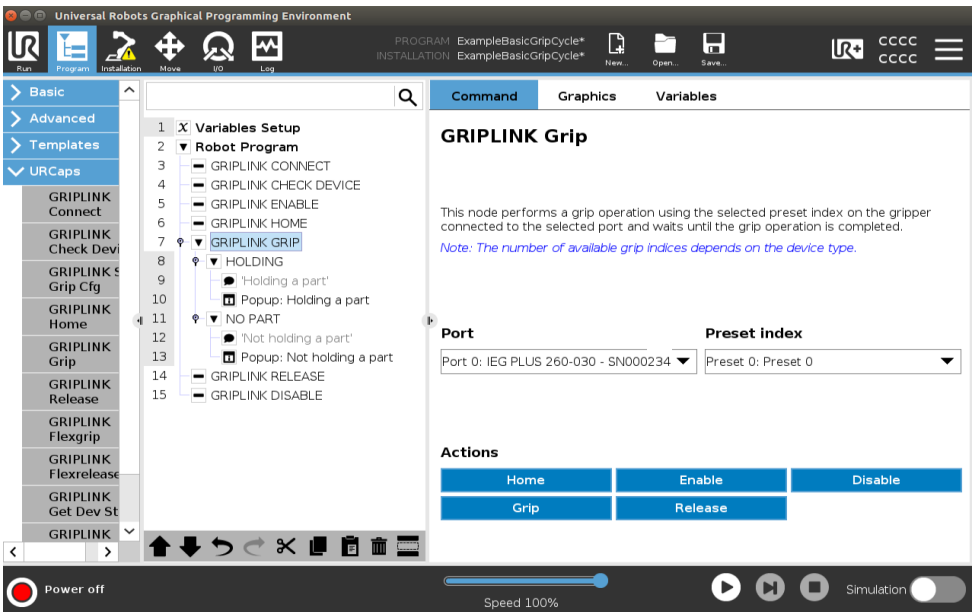



Figure 25: GRIPLINK GRIP - Execute gripping operation at the selected port with preset index (blocking).

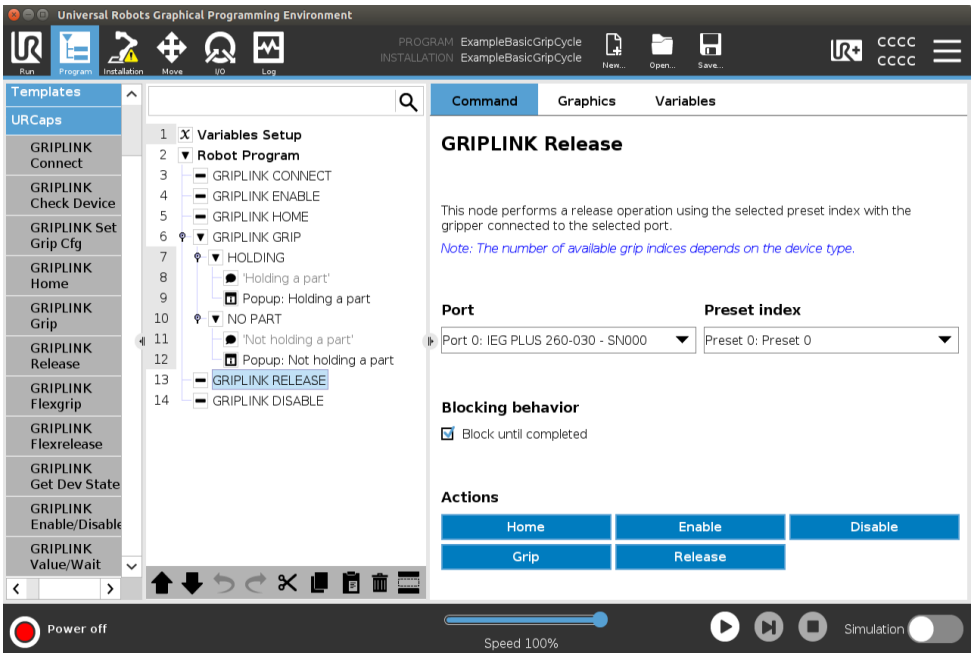



Figure 26: GRIPLINK RELEASE - Release operation at the selected port with preset index (optionally blocking)

The graphical “GRIPLINK Grip” program node is always executed in blocking mode. A non-blocking execution of the grip command is only available via the URScript function `GL_GRIP(..., <DO_BLOCK> = 0)`.

Ensure that the system’s motion and safety concept is compatible with the selected blocking behavior. In many applications, a blocking implementation is advisable so that the robot only continues moving once the gripping or release operation has been reliably completed.

If a device is already in the **"HOLDING"** or **"NO_PART"** state, the GRIPLINK protocol does not allow the command to be executed again using the "GRIPLINK Grip" program node or `GL_GRIP()` in URScript code.

For GRIPKIT EASY gripper modules with a firmware version below 3.0.0, only blocking execution is supported. Non-blocking calls are not permitted for these devices.

If the "GRIPLINK Grip" program node is nevertheless called in one of these states (S_HOLDING or S_NO_PART), this results in the following message:
"Service unavailable within current state".

***Command call with URScript code***

```
GL_GRIP(
     <PORT>,
     <PRESET_INDEX>,
```

```
     <DO_BLOCK>,
     <SOCKET_NAME>
)
GL_RELEASE(
     <PORT>,
     <PRESET_INDEX>,
     <DO_BLOCK>,
     <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <PRESET_INDEX> | Integer | Preset index (0...7) |
| <DO_BLOCK> | Integer | 0: non-blocking execution<br>1: blocking execution |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

Example 1 - Blocking execution with GRIPLINK grip/release:

```
# Grip with gripper connected to port 0 and preset index 3
# Execute the command with blocking
GL_GRIP(0,3,1,"sock_griplink")

# ...
# Move part to place position
# ...

# Release the part again
GL_RELEASE(0,3,1,"sock_griplink")
```

Example 2 - Non-blocking gripping, WSTR, and Get Dev State:

```
# Grip with the gripper connected to port 0 and preset index 3
# Execute the command without blocking
GL_GRIP(0,3,0,"sock_griplink")

# ...
# Wait for state transition on port 0
GL_WSTR(0,"sock_griplink")

# Get device state to evaluate
# See Appendix A for the device states
gripper0_state = GL_DEVSTATE(0,"sock_griplink")

if (gripper0_state == S_HOLDING):
    # Release part again (blocking)
    GL_RELEASE(0,3,1,"sock_griplink")
    GL_DISABLE(0,True,"sock_griplink")
else:
    # Process other states here (e.g., S_NO_PART, S_FAULT,…)
    GL_DISABLE(0,True,"sock_griplink")
end
```

### 7.6.1 Evaluation of the gripping state

The "GRIPLINK Grip" program node detects after gripping whether the gripper has gripped a component or not.
This allows workflows to be implemented easily based on the gripper state, e.g., alternative workflows in the event that no workpiece is present (NO_PART) and the regular process sequence when the gripper successfully holds the workpiece (HOLDING).

***Depending on the result, the child nodes are executed as follows:***

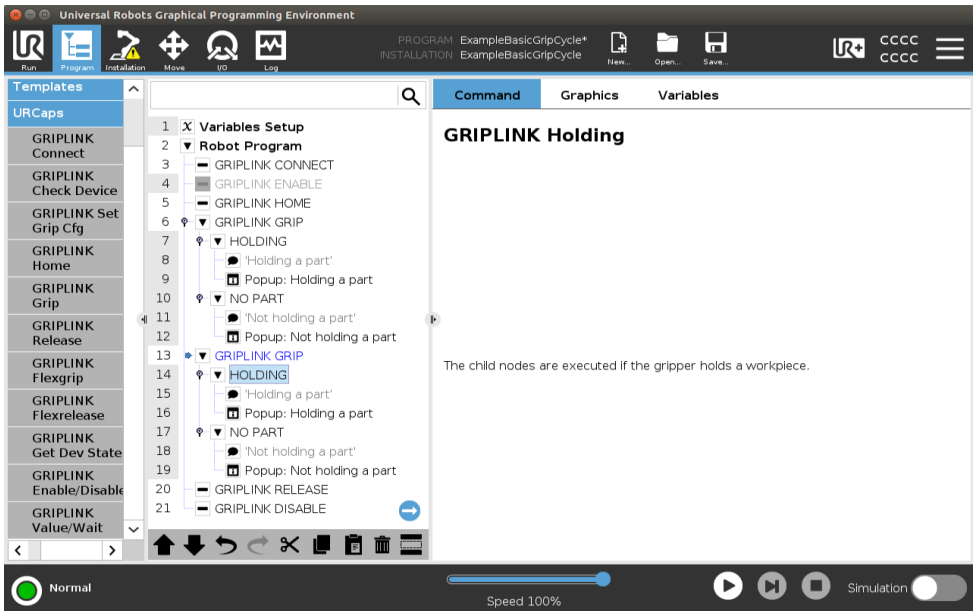



Figure 27: Child node “HOLDING” – Executing the subordinate actions when the device is holding a workpiece.

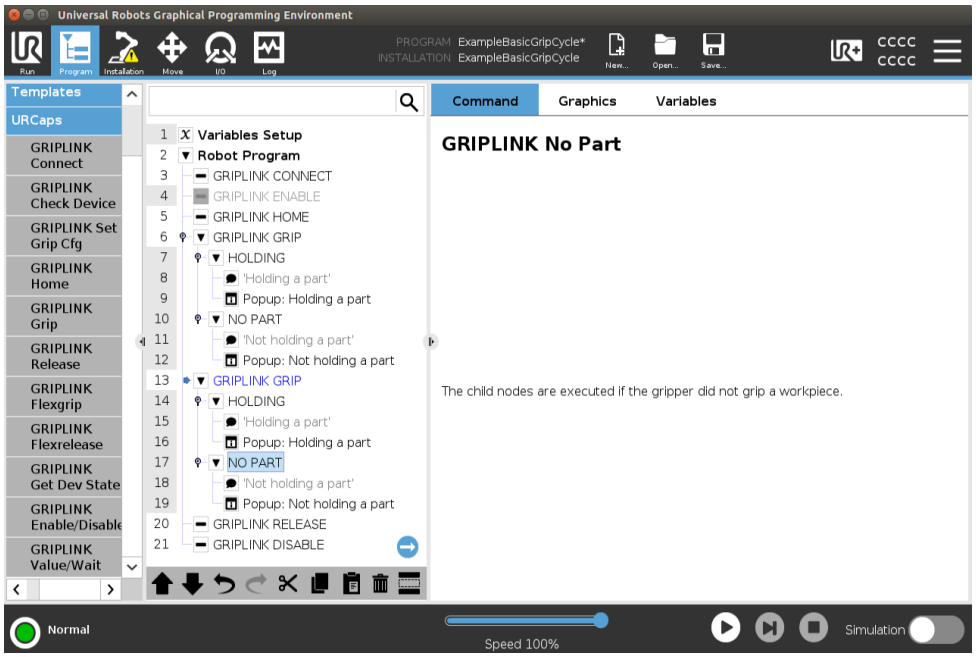



Figure 28: Child node “NO PART” – Execute the subordinate actions if no workpiece has been gripped.

## 7.7 Flexible Gripping, Releasing, and Pre-positioning - GRIPLINK Flexgrip/Flexrelease

The gripper modules of the WPG series, INTRAPAL series, IEG PLUS series, CRG series, and STERIGRIP series support flexible gripping and releasing. Here, position, gripping force, and motion parameters can be specified independently of presets. This enables fast pre-positioning and gentle gripping, which can reduce cycle times and protect the gripped parts.

For more information on the motion parameters, please refer to the operating instructions for the respective gripper module.

***In the "GRIPLINK Flexgrip" program node, you can:***

- Select the port of the desired gripper
- Specify the target position in mm
- Set the desired gripping force in N
- Specify the speed in mm/s for Ethernet mode or in % for Tool I/O mode
- Specify acceleration in Ethernet mode in mm/s² or in Tool I/O mode in %

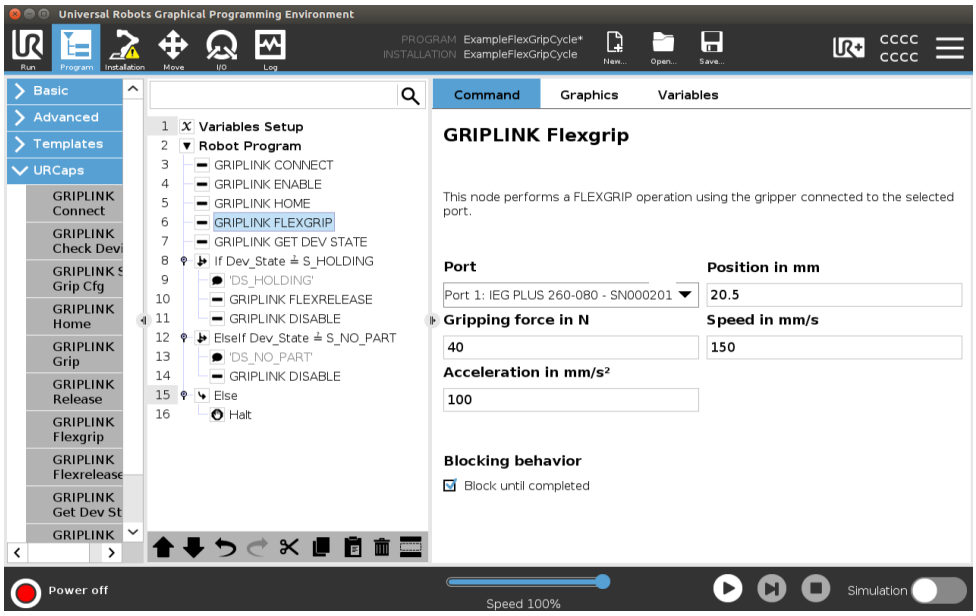



Figure 29: GRIPLINK FLEXGRIP - Flexible gripping with parameters (position, gripping force, speed, acceleration; optionally blocking).

***In the "GRIPLINK Flexrelease" program node, you can:***

- Select the port of the desired gripper
- Specify the target position in mm
- Specify the speed in Ethernet mode in mm/s or in Tool I/O mode in %
- Specify acceleration in Ethernet mode in mm/s² or in Tool I/O mode in %

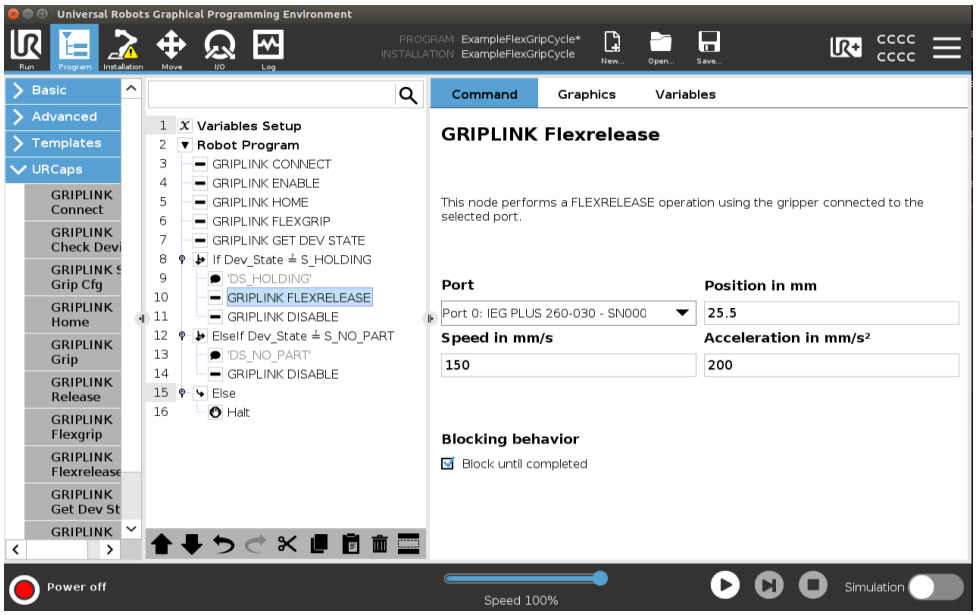



Figure 30: GRIPLINK FLEXRELEASE - Flexible release with parameters (position, speed, acceleration; optionally blocking).

***The "Blocking behavior" checkbox is also available in both nodes:***

***Blocking (checkbox checked ☑: do_block = 1)***

- The command is sent to the GRIPLINK controller and is not acknowledged until the movement is complete.
- From the robot program's perspective, the node remains blocking until the operation is complete.
- With **FLEXRELEASE**, an additional internal check is performed to determine whether the target position has been reached (via GL_WAITVAL).

***Non-blocking (checkbox unchecked ☐: do_block = 0)***

- The movement is started and confirmed immediately.
- The robot program continues to run while the gripping or release movement is executed in the background.
- A non-blocking execution can be used, for example, to perform parallel gripping processes with multiple grippers.

Clear the “Block until completed (Blocking behavior)” checkbox only if the system’s safety and motion concept allows non-blocking execution.

For GRIPKIT EASY gripper modules with a firmware version below 3.0.0, only blocking execution is supported. Non-blocking calls are not permitted for these devices.

If a device is already in the **"HOLDING"** or **"NO_PART"** state, the GRIPLINK protocol does not permit re-executing the command using the "GRIPLINK Flexgrip" program node or `GL_FLEXGRIP()` in URScript code.

If the "GRIPLINK Flexgrip" program node is nevertheless called in one of these states (S_HOLDING or S_NO_PART), this results in a message:
“Service unavailable within current state”.

***Command call with URScript code***

```
GL_FLEXGRIP(
      <PORT>,
      <POSITION>,
      <GRIPPING_FORCE>,
      <SPEED>,
      <ACCELERATION>,
      <DO_BLOCK>,
      <SOCKET_NAME>
)
GL_FLEXRELEASE(
      <PORT>,
      <POSITION>,
      <SPEED>,
      <ACCELERATION>,
      <DO_BLOCK>,
      <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <POSITION> | Double | Target position in mm |
| <GRIPPING_FORCE> | Double | Gripping force in N (only for GL_FLEXGRIP) |
| <SPEED> | Double | Ethernet interface: Speed in mm/s<br>Tool I/O interface: Speed in % |
| <ACCELERATION> | Double | Ethernet interface: Acceleration in mm/s²<br>Tool I/O interface: Acceleration in % |
| <DO_BLOCK> | Integer | 0: non-blocking execution<br>1: blocking execution |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Prepositioning with WPG at port 0 to 50.0 mm at optimal speed
# and acceleration
# with blocking until flexrelease ends
GL_FLEXRELEASE(0,50,0,0,1,"sock_griplink")

# Grip part with No Part Limit 45.0 mm with 100 N and optimal speed
# and acceleration
# with blocking until flexgrip ends
GL_FLEXGRIP(0,45,100,0,0,1,"sock_griplink")
```

## 7.8 State Query - GRIPLINK Get Dev State

The “GRIPLINK Get Dev State” program node is used to query the current device state. The retrieved device state is written as an integer to a user-selected result variable.

The following parameters can be set in the node:

***Port (combo box)***

Selection of the port where the device whose state is to be queried is located.

***Result Variable (combo box)***

Selection of a numeric variable into which the device state is written. This allows states to be defined centrally and reused. All variables created in the **Installation → General → Variables** tab are available for selection here.

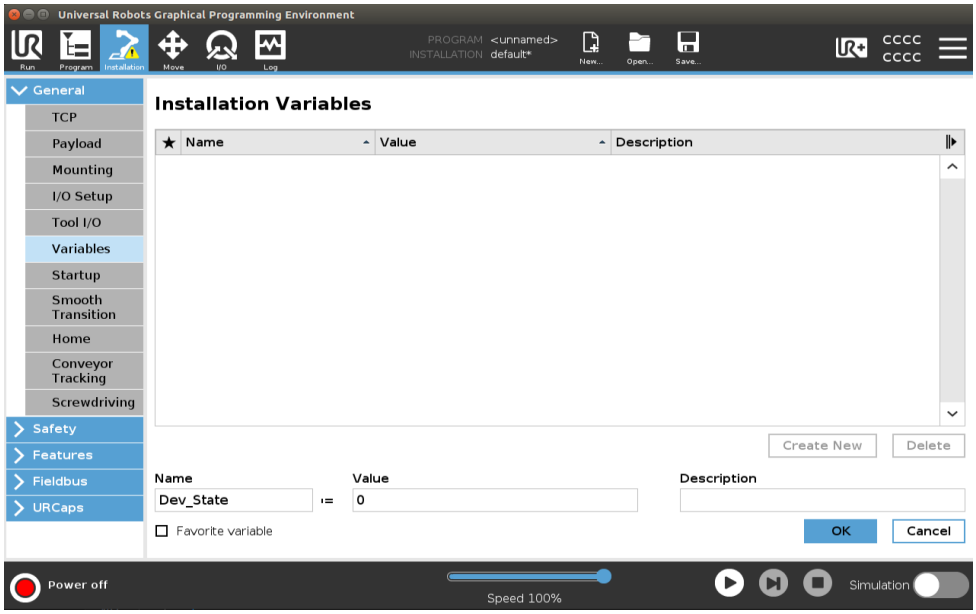



Figure 31: PolyScope 5 - Installation/General/Variables: Create an installation variable.

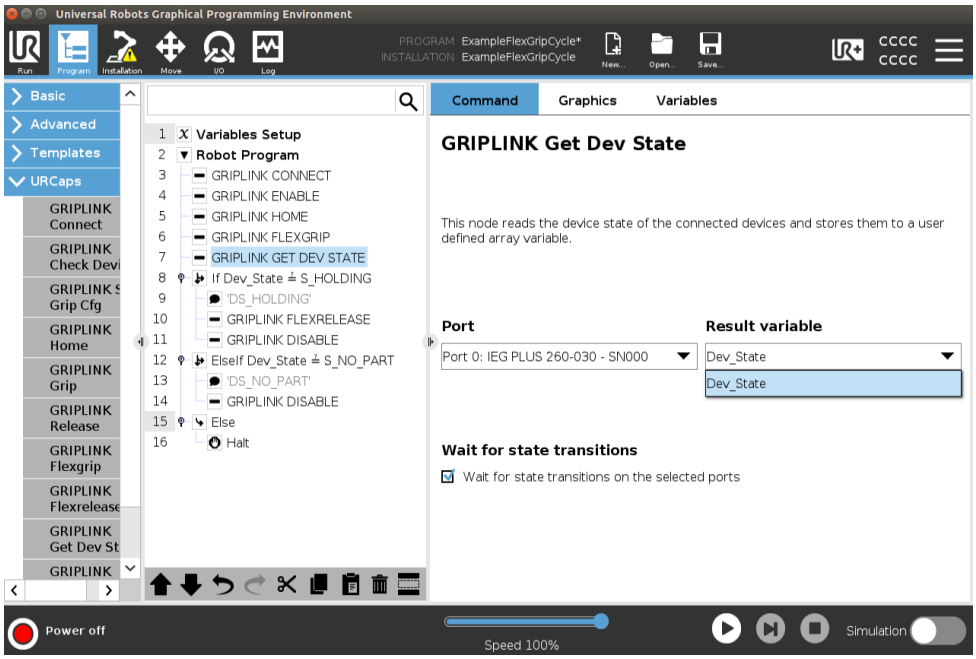



Figure 32: GRIPLINK Get Dev State - Read the device state on the selected port and save it to the variable `Dev_State` (optionally with “Wait for state transitions”).

The node has the checkbox “Wait for state transitions on the selected port”:

***Checkbox unchecked*** ☐

- The current device state is read immediately using `GL_DEVSTATE(...)`.
- The result is immediately written to the result variable.
- Typical use: simple periodic state polling in a running program.

***Checkbox checked*** ☑

- The node first waits internally using `GL_WSTR(...)` for a state transition of the device on the selected port.
- Only after a state change is detected is the current device state read using `GL_DEVSTATE(...)` and written to the result variable.
- Typical use: following a previous command (e.g., Grip, Release, Enable, Disable), when it is necessary to evaluate which state was actually reached as soon as the device has visibly changed.

The possible device states are listed in Appendix A.

If no state change occurs over a longer period of time, a timeout may occur if the "Wait for state transitions on the selected port" option is enabled.

The read state can then be processed using if-else constructs. In the following example, the state of a gripper at Port 0 is evaluated. Depending on the state, the system either moves to the placement position or executes an alternative sequence.

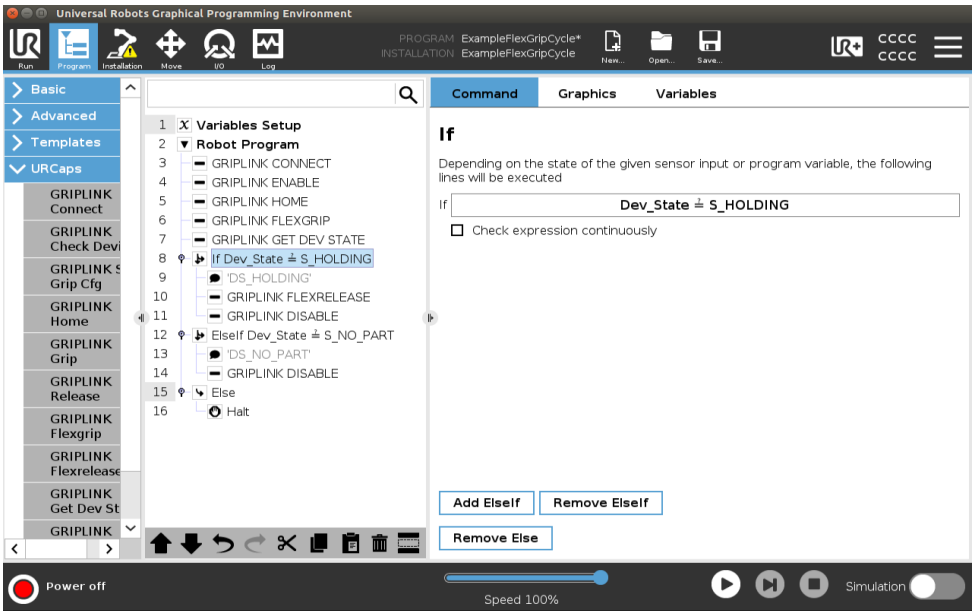



Figure 33: If condition – Evaluation of Dev_State (e.g., `Dev_State == S_HOLDING`) for flow branching.

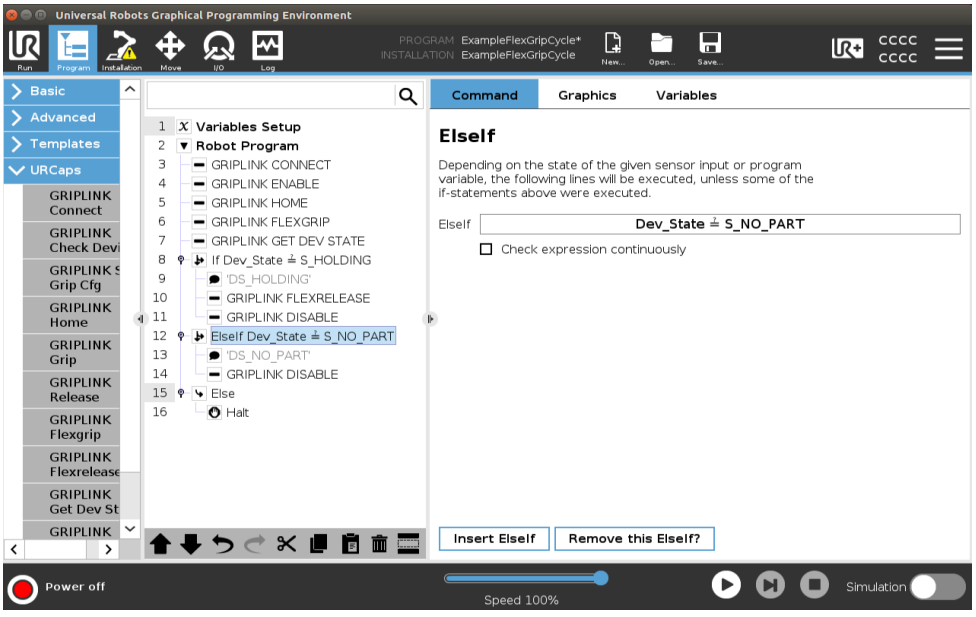



Figure 34: ElseIf condition—alternative evaluation of Dev_State (e.g., `Dev_State == S_NO_PART`).

***Command call with URScript code***

```
<RETURN_VARIABLE> = GL_DEVSTATE(
     <PORT>,
     <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0…31) |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

| Type Return value | Meaning |
|---|---|
| Integer | Device state (see Appendix A) |

Predefined constants can be used for the device states.

Example 1 - Query state directly (without WSTR):

```
# Gripper connected to port 0 and preset index 3
# Execute the command WITH blocking
GL_GRIP(0,3,1,"sock_griplink")


# Read the current device state immediately
gripper0_state = GL_DEVSTATE(0,"sock_griplink")

# When the state is HOLDING, move to the target position
if (gripper0_state == S_HOLDING):
     # Part was successfully gripped
     # Move to the placement position
     # ...
else:
     # Handle other states here (e.g., S_NO_PART, S_FAULT, …)
end
```

Example 2 - Evaluate state after state transition (using WSTR):

```
# Gripper connected to port 0 and preset index 3
# Execute the command WITHOUT blocking
GL_GRIP(0,3,0,"sock_griplink")

# Wait for a state transition on port 0
GL_WSTR(0,"sock_griplink")


# Read the current device state after the transition
gripper0_state = GL_DEVSTATE(0, "sock_griplink")

# When state is HOLDING, move to the target position
if (gripper0_state == S_HOLDING):
     # Part was successfully gripped
     # Move to place position
     # ...
else:
     # Handle other states here (e.g., S_NO_PART, S_FAULT, …)
end
```

## 7.9 Read Device Values and Wait - GRIPLINK Value/Wait

The “GRIPLINK Value/Wait” program node is used to read a device value and, optionally, wait until it reaches a specified target value.
The behavior of the program node is controlled via the checkbox **“Wait** for **device value to reach the given value within the given window**”:

- Checkbox unchecked ☐: Read device value
- Checkbox checked ☑: Wait for device value within the window, then read device value

Devices such as sensors provide data in the form of indexed values. These values can be read; depending on the device, a varying number of indices are available.

### 7.9.1 Read device value

The following parameters can be set in the program node:

***Port (combo box):***

Select the port to which the desired gripper or sensor is connected.

***Value index (combo box):***

Select which device value should be read. The available indices depend on the device type.
Example (IEG PLUS gripper):

- Index 0: Opening width (mm)
- Index 1: Part width (mm)
- Index 2: Part count
- Index 3: Device temperature (°C)

***Result variable:***

Name of the numeric variable to which the read device value is written. Any numeric variable defined under Installation → General → Variables can be selected.

For gripper modules, the value with index 0 corresponds to the opening width in millimeters.

***Behavior***

- The current device value is read once (via `GL_VALUE(...)`).
- The result is written to the specified result variable.
- The program continues without a wait.

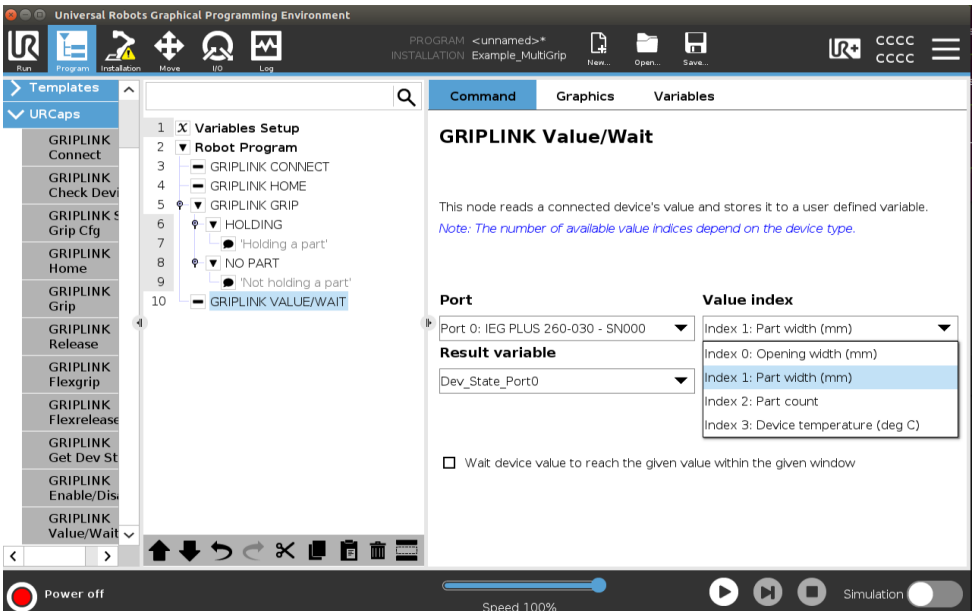
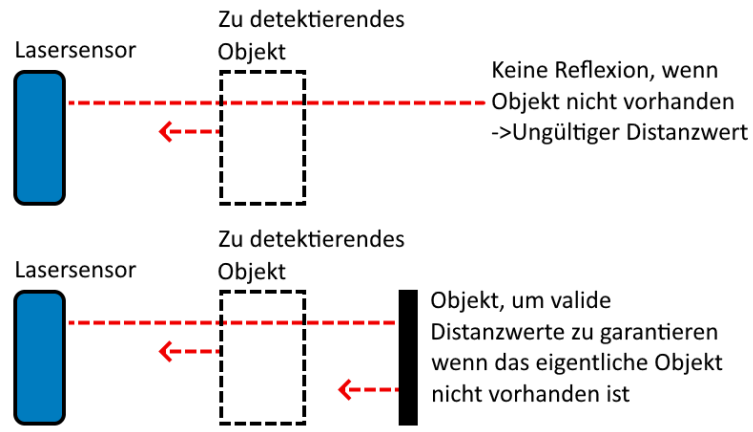



Figure 35: GRIPLINK VALUE/WAIT - Read device value (Value Index) and write it to a variable (e.g., `Dev_State_Port0`).

A maximum of eight values can be read (index 0…7); however, depending on the device, there may be fewer.

**Therefore, ensure that a valid value index is always selected. Further information can be found in the manual for the respective device driver.**
The value 2147483647 is used in GRIPLINK, for example, for sensors whose measured value lies outside the measurement range.

**To prevent errors in program execution, ensure that valid values (< 2147483647) are always read from the respective device!**
**This can be ensured, for example, by having a distance sensor always detect an object and thus always return a valid distance instead of a zero value:**




***Command call with URScript code***

```
<RETURN_VARIABLE> = GL_VALUE(
     <PORT>,
     <VALUE_INDEX>,
     <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <VALUE_INDEX> | Integer | Value index (0...7) |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

| Type Return value | Meaning |
|---|---|
| Integer | Value of the read device value |

Example:

```
# Read value 0 (position) of a gripper connected to port 0
gripper0_position = GL_VALUE(0,0,"sock_griplink")
```

### 7.9.2 Waiting for and reading device values

After setting a device value, it may be necessary to wait until this value has actually been reached. For actuators, this could be a target position, for example.

In **wait mode** (checkbox checked ☑), the node ensures that

- a specific **target value** is reached and
- it must be reached within a tolerance **window** and
- within a specified time in ms (**timeout)**.

If the target range is not reached within the specified time, a **timeout error** is generated. In wait mode, the following additional settings are available:

***Value threshold***

Target value to which the device value should converge (e.g., 20 mm workpiece width).

***Window size***

Size of the tolerance window around the target value. Example:

- Window size = 1.5 mm
- Valid range: 20 mm ± 0.75 mm

***Timeout in ms***

Maximum wait time in milliseconds. If the target range is not reached within this time, the node generates a timeout error.

***Process for waiting and reading the device value***

1. The node uses `GL_WAITVAL(...)` to wait until the device value enters the window defined around the threshold value.
2. Afterward, the current device value is read using `GL_VALUE(...)` and written to the result variable.

***Typical applications***

- Waiting for a sensor distance to ensure that an object is actually present.
- Monitoring whether a value remains within a defined tolerance (e.g., `opening width, part width`).

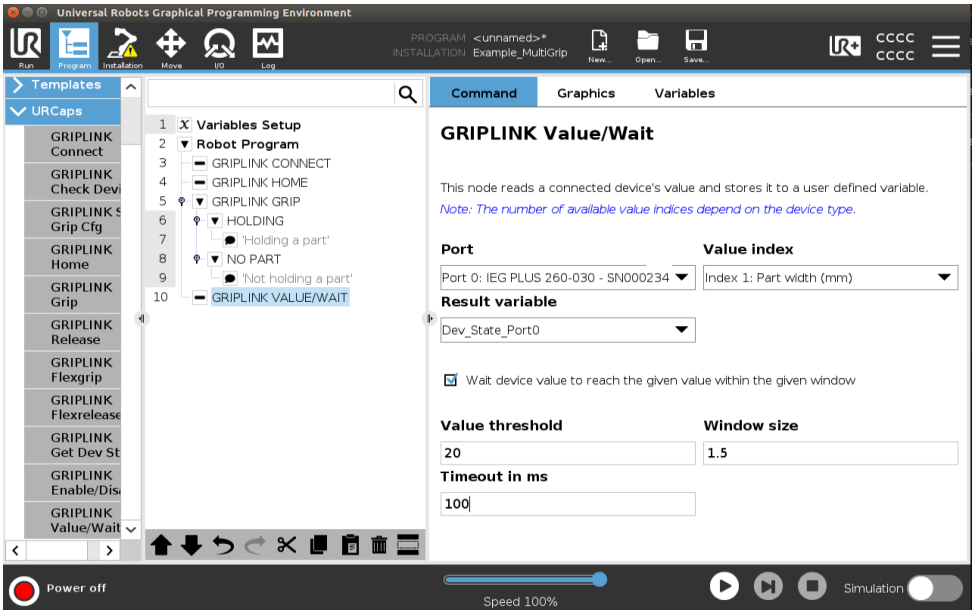



Figure 36: GRIPLINK VALUE/WAIT - Wait for a device value, then read the value and store it in a variable.

This function is not available on all devices.

The window size should be selected appropriately in advance depending on the rate of change of the device value to enable reliable and accurate detection!

***Command call with URScript code***

```
GL_WAITVAL(
      <PORT>,
      <VALUE_INDEX>,
      <VALUE_THRESHOLD>,
      <WINDOW_SIZE>,
      <TIMEOUT_MS>,
      <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <VALUE_INDEX> | Integer | Value index (0...7)<br>Depending on the device, fewer than 8 values may be available! |
| <VALUE_THRESHOLD> | Double | Threshold value to be reached |
| <WINDOW_SIZE> | Double | Window size that defines the range of values centered around the threshold value within which the WAITVAL command completes without error |
| <TIMEOUT_MS> | Integer | Maximum duration in ms for which the system waits for the target value to be reached |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br><br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Wait for the sensor connected to port 0 to reach a value of 4000 +/-
500
# within at most 15 seconds
# Value index 0
# Value threshold 4000
# Window size 500
# 15 s => 15,000 ms
GL_WAITVAL(0,0,4000,500,15000,"sock_griplink")

# Read value 0 afterwards
gripper0_position = GL_VALUE(0,0,"sock_griplink")
```

## 7.10 Configuring a Grip Preset - GRIPLINK Set Grip Config

Depending on the gripper module, up to eight freely configurable grip presets can be executed. The "GRIPLINK Set Grip Config" program node allows you to adjust the grip parameters during operation, for example, to initialize a gripper module at the start of a program or to react flexibly to changes in the sequence.

For more information on the grip parameters, particularly the limitations, refer to the operating instructions for the respective gripper modules.

In this program node, you can select the port of the desired gripper and the index of the grip preset. Using the "Value Interpretation" drop-down menu, you can customize how values are displayed based on the meaning of each parameter. For certain devices, irrelevant parameters are hidden.

The following parameters can be set in the program node:

***Port (combo box)***

Selection of the port for the desired gripper.

***Preset Index (combo box)***

Selection of the preset to be configured (0…7).

***Grip Preset Tag (text input)***

Free text describing the preset (e.g., `workpiece1`). The tag facilitates the assignment of the grip preset to a workpiece.

***Value Interpretation (combo box)***

Specifies how the individual parameter fields are interpreted and labeled.

***WEISS ROBOTICS IO-Link Gripper***

Parameters are displayed as No Part Limit (mm), Release Limit (mm), and Force Factor (%).

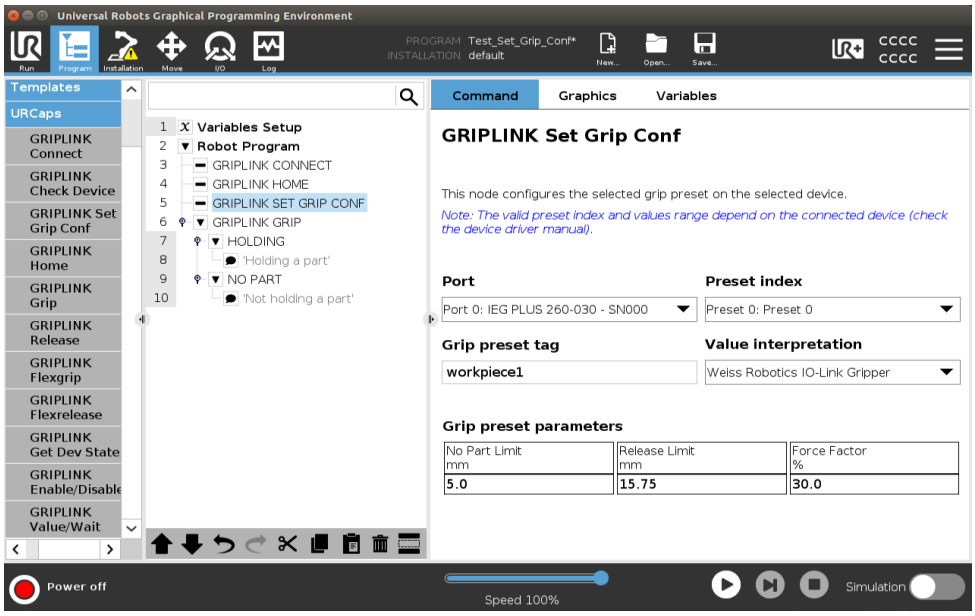



Figure 37: GRIPLINK SET GRIP CONF - Value Interpretation: WEISS ROBOTICS IO-Link Gripper.

***WEISS ROBOTICS WPG/INTRAPAL Gripper***

Parameters are displayed as No Part Limit (mm), Release Limit (mm), Force Factor (%), Grip Velocity (mm/s), Grip Acceleration (mm/s²), Release Velocity (mm/s), and Release Acceleration (mm/s²).

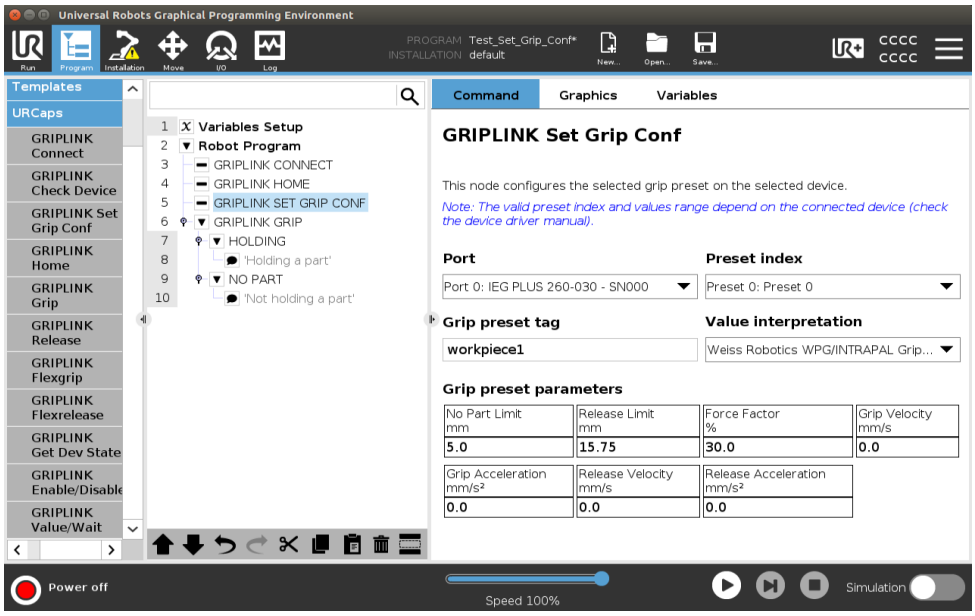



Figure 38: GRIPLINK SET GRIP CONF - Value Interpretation: WEISS ROBOTICS WPG/INTRAPAL Gripper.

***Generic Vacuum Picker***

The parameters are displayed as Part Present Setpoint (kPa), Energy Saving Setpoint (kPa), and Blow-off Duration (ms).

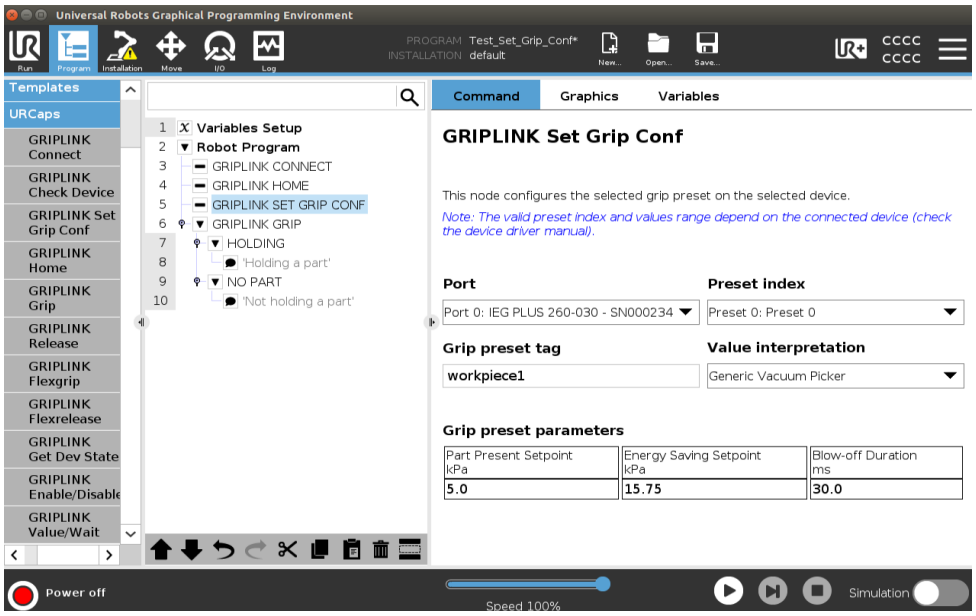



Figure 39: GRIPLINK SET GRIP CONF - Value Interpretation: Generic Vacuum Picker.

***Generic***

All up to eight available parameters are displayed neutrally as Value-0 to Value-7.

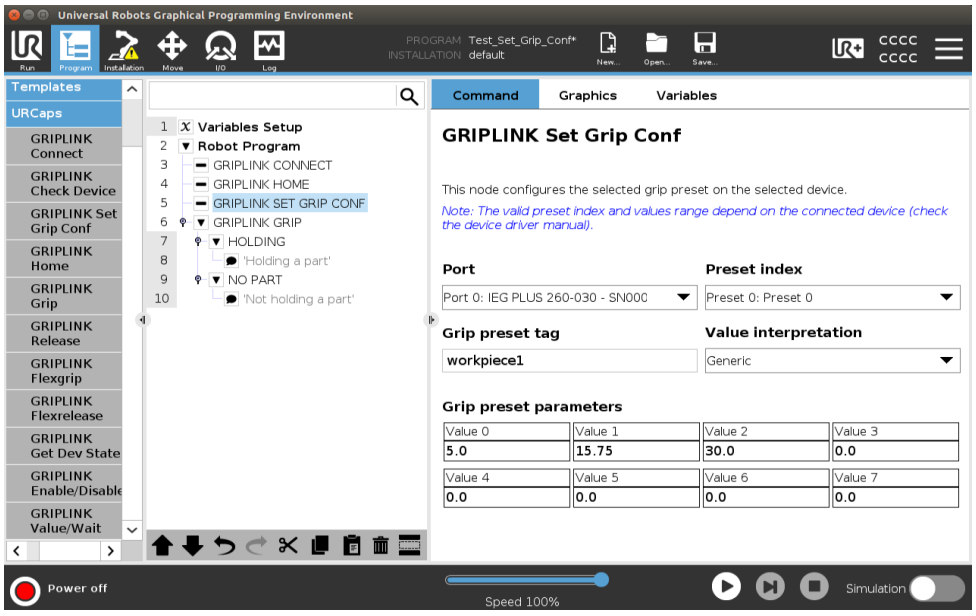



Figure 40: GRIPLINK SET GRIP CONF - Value Interpretation: Generic.

For gripper modules from WEISS ROBOTICS, select the **"Weiss Robotics"** parameter template to ensure that the entered values are validated and displayed correctly.

Note that, depending on the gripper module, not all eight configurable grip presets may be available.

For other devices, the generic view can be used. Here, all eight available parameters are visible, even if not all of them are used by the device.

***Command call with URScript code***

```
GL_SETGRIPCFG(
      <PORT>,
      <PRESET_INDEX>,
      <PRESET_TAG>,
      <PRESET_PARAMS>,
      <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31). |
| <PRESET_INDEX> | Integer | Preset index (0...7). |
| <PRESET_TAG> | String | Preset tag - Freely selectable tag for the preset (e.g., "workpiece1"). |
| <PRESET_PARAMS> | Double List | List of numerical parameter values for this preset. The list contains at least one and at most 8 elements. |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink".<br><br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Prepare preset parameters for params 0, 1, and 2
preset_params = [5,15.75,30]

# Set grip preset 3 with the tag "workpiece1" for the device connected
to port 0
GL_SETGRIPCFG(0,3,"workpiece1",preset_params,"sock_griplink")
```

## 7.11 Set Device Value - GRIPLINK Set Value

Some devices support setting device values, e.g., for positions.

This function is not available on all devices.

Device values are set exclusively via the URScript function `GL_SETVAL(...)`, e.g., in a script node.

Setting device values can cause actuators to move.

***Command call with URScript code***

```
GL_SETVAL(
      <PORT>,
      <VALUE_INDEX>,
      <VALUE>,
      <SOCKET_NAME>
)
```

| Parameter | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <VALUE_INDEX> | Integer | Value index (0...7)<br>Depending on the device, fewer than 8 values may be available! |
| <VALUE> | Double | Value to be set |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Set device value 2 of the device connected to port 3 to value 12
GL_SETVAL(3,2,12,"sock_griplink")
```

## 7.12 Control of the LED light ring - GRIPLINK LED

The CRG series gripper modules from WEISS ROBOTICS feature an LED light ring for indicating various operating states. This can be controlled using a standalone instruction. All CRG gripper modules have eight configurable presets that can be activated via the instruction.

For more information on LED visualization and its configuration, please refer to the operating instructions for the CRG gripper modules.

The LED function is only available on compatible devices!

The LED ring is controlled exclusively via the URScript function `GL_LED(...)`, e.g., in a script node.

***Command call with URScript code***

```
GL_LED(
      <PORT>,
      <PRESET_INDEX>,
      <SOCKET_NAME>
)
```

| Parameters | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <PRESET_INDEX> | Integer | Preset index (0...7) |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Set LED pattern preset 4 of CRG 200-085 connected to port 0
GL_LED(0,4,"sock_griplink")
```

## 7.13 Mechanical Clamping Control - GRIPLINK Clamp

The grippers of the CRG series from WEISS ROBOTICS are equipped with the PERMAGRIP feature. This makes it possible to switch off motor control during prolonged gripping while still holding the workpiece securely.

For more information on mechanical clamping, refer to the operating manual for the respective gripper module.

Clamping is controlled exclusively via the URScript function `GL_CLAMP(...)`.

This function is not available on all devices.

***Command call with URScript code***

```
GL_CLAMP(
     <PORT>,
     <CLAMP_STATE>,
     <SOCKET_NAME>
)
```

| Parameters | Type | Meaning |
|---|---|---|
| <PORT> | Integer | Port index (0...31) |
| <CLAMP_STATE> | Boolean | "True" to enable clamping<br>"False" to disable clamping |
| <SOCKET_NAME> | String | Name of the socket to be used, e.g., "sock_griplink"<br><br>The socket name is used in all subsequent nodes in the robot program! |

Example:

```
# Enable clamping of CRG 200-085 at port 0
GL_CLAMP(0,True,"sock_griplink")

# Grip with CRG 200-085 at port 0 using preset 0 without blocking.
GL_GRIP(0,0,0,"sock_griplink")

# Hold part for a longer time
#...
```

# 8 Troubleshooting

The GRIPLINK Plugin outputs error messages during operation. Typical messages, possible causes, and recommended actions are described below. The information provided here supports commissioning, diagnostics, and service.

## 8.1 No access to the Tool I/O interface

***Possible cause:***

In the robot's Tool I/O menu, the **"GRIPLINK for UR"** option was not selected under **"Controlled by."**

***Recommended action:***

Check the robot's settings under **"Installation / Tool I/O"** and select **"GRIPLINK for UR"** there. Then check in the GRIPLINK Plugin's installation node whether Tool I/O access displays the status "GRANTED**."**

## 8.2 Device is not recognized in Tool I/O mode

***Possible cause:***

The connected device is not compatible, not wired correctly, not connected with the original Tool I/O cable, or the Tool I/O interface has not yet been rescanned.

***Recommended action****:*

Check the wiring, power supply, and device compatibility. Ensure that the original Tool I/O cable is being used. Then, in the installation node of the GRIPLINK Plugin, perform the **"Rescan devices"** action and check whether the device appears in the results display.

## 8.3 Timeout when the "Wait for state transitions" option is enabled

***Possible cause:***

The expected state transition did not occur because the device was already in the target state, the preceding command did not trigger a state change, or the device was not in a valid initial state.

***Recommended action:***

Check the program flow and the device state before the command. Enable the wait function only if a state transition is actually expected. If necessary, first check the current device state using **GRIPLINK Get Dev State**.

## 8.4 Non-blocking execution on GRIPKIT EASY with firmware versions below 3.0.0

***Possible cause:***

For GRIPKIT EASY devices with a firmware version below **3.0.0,** non-blocking execution of gripping and release commands is not permitted.

***Recommended action:***

Use only blocking execution for these devices. Check the firmware version in use and adapt existing programs accordingly.

# Appendix A Device State

The following table lists the possible state values for connected devices. The constants listed in the right-hand column can be used in the robot program (see Section 7.8).

| Device state | Value | Meaning | Name of the URScript constant |
|---|---|---|---|
| NOT CONNECTED | 0 | Gripper module not connected | S_NOT_CONNECTED |
| NOT INITIALIZED | 1 | Gripper module not initialized | S_NOT_INITIALIZED |
| DISABLED | 2 | Drive inactive<br>Fingers can be moved manually.<br>Device is disabled | S_DISABLED |
| RELEASED | 3 | Workpiece released | S_RELEASED |
| NO PART | 4 | No workpiece found | S_NO_PART |
| HOLDING | 5 | Workpiece is held | S_HOLDING |
| ENABLED | 6 | Drive active<br>Finger position is held.<br>Device is enabled. | S_ENABLED |
| FAULT | 7 | Fault condition | S_FAULT |

Do not assign values other than those listed in the table to the URScript constants, as this will impair the function of the URCap and may result in malfunction!

# Appendix B Status codes

If an error occurs during the execution of a GRIPLINK command, a corresponding status code is output. This describes the cause of the error and can be used for diagnosis in the robot program or during commissioning. The following table provides an overview of the possible status codes and their meanings.

| Status code | Identifier | Description |
|---|---|---|
| 0 | E_SUCCESS | No error. Command successfully executed. |
| 1 | E_OVERRUN | Data overrun |
| 2 | E_RANGE_ERROR | Value out of range |
| 3 | E_NOT_AVAILABLE | Function or data not available |
| 4 | E_NOT_INITIALIZED | Device not initialized |
| 5 | E_TIMEOUT | Timeout |
| 6 | E_INSUFFICIENT_RESOURCES | Not enough memory available |
| 7 | E_CHECKSUM_ERROR | Checksum error |
| 8 | E_ACCESS_DENIED | Access denied |
| 9 | E_INVALID_HANDLE | Invalid handle |
| 10 | E_INVALID_PARAMETER | Invalid parameter |
| 11 | E_INDEX_OUT_OF_BOUNDS | Index out of bounds |
| 12 | E_IO_ERROR | Generic I/O error |
| 13 | E_READ_ERROR | Read error |
| 14 | E_WRITE_ERROR | Write error |
| 15 | E_NOT_FOUND | Resource not found |
| 16 | E_NOT_OPEN | File or device not open |
| 17 | E_EXISTS | Resource already exists |
| 18 | E_NO_COMM | Connection error |
| 19 | E_STATE_CONFLICT | Invalid state |
| 20 | E_NOT_SUPPORTED | Command or function not supported |
| 21 | E_INCONSISTENT_DATA | Data inconsistent |

| | | |
|---|---|---|
| 22 | E_CMD_SYNTAX | Syntax error |
| 23 | E_CMD_UNKNOWN | Unknown command |
| 24 | E_CMD_ABORTED | Command aborted |
| 25 | E_CMD_FAILED | Command failed |
| 26 | E_AXIS_BLOCKED | Axis is blocked |
| 27 | E_PENDING | Action pending / Not yet finished |

# Appendix C Tool I/O-specific status codes

Status codes starting with 100 are Tool I/O-specific error codes of the URCap and support diagnostics and error handling in the Tool I/O interface.

| Status code | Meaning |
|---|---|
| 100 | Access to the Tool I/O interface was not granted. |
| 101 | Previously granted access to the Tool I/O interface has been revoked. |
| 102 | A device scan is already in progress. Please wait for it to complete. |
| 103 | The Tool I/O interface is currently being initialized. |
| 104 | The Tool I/O interface is currently being terminated. |

# Appendix D Changes from Previous Versions

| Date | Version | Type | Description | Author |
|---|---|---|---|---|
| 04/06/2026 | 3.0.0 | Added | Support for GRIPKIT devices and devices from the INTRAPAL series via Tool I/O (Modbus RTU). | H. E. |
| 04/06/2026 | 3.0.0 | (BREAKING) | Ethernet-based communication (TCP/IP) and Tool I/O-based communication (Modbus RTU) have been combined into a single Plugin.<br>The previous FlexGrip Plugin is being discontinued and replaced by the GRIPLINK Plugin. | H. E. |
| 04/06/2026 | 3.0.0 | Added | Turkish localization (TR). | H. E. |
| 12/16/2025 | 2.2.0 | Added | Demo programs for PolyScope | H. E. |
| 12/16/2025 | 2.2.0 | Added | Port configuration and logging configuration. | H. E. |
| 12/16/2025 | 2.2.0 | Added | GRIPLINK Toolbar (IGRIP/PGRIP/VALUE). | H. E. |
| 12/16/2025 | 2.2.0 | Changed | Revision of program nodes:<br>• Port combo box with online information.<br>• Common Enable/Disable node<br>• DO_BLOCK for GRIP/RELEASE and FLEXGRIP/FLEXRELEASE<br>• Floating-point support for FLEXGRIP/FLEXRELEASE and VALUE/WAIT<br>• Variable selection for VALUE/WAIT and Get Dev State, as well as merging WAITVAL and VALUE into a single node. | H. E. |
| 12/16/2025 | 2.2.0 | Removed | MULTIGRIP, MULTIRELEASE, and Get Position program nodes. | H. E. |