WEISS ROBOTICS

# GRIPLINK-PLUGIN FOR STÄUBLI

Version 3.0.0

July 2023

# Content

# 1 Introduction

With GRIPLINK technology, IO-Link compatible automation components can be connected to robot systems from leading manufacturers via a simple network connection. The GRIPLINK plug-in for STÄUBLI is the control-side link and enables an easy integration of GRIPLINK technology from WEISS ROBOTICS into robot systems from the manufacturer STÄUBLI.

These instructions describe the functions of the GRIPLINK plug-in. For information on mounting, commissioning and operation of the GRIPLINK controller, refer to the operating instructions of the respective module. These can be found online at **www.griplink.de/manuals**

## 1.1 Notation and symbols

For a better overview, the following symbols are used in these instructions:

Function or safety-relevant note. Non-observance may endanger the safety of personnel and plant, damage the device or impair the function of the device.

Additional information for a better understanding of the described facts.

Reference to further information.

## 1.2 Intended use

The "GRIPLINK Plugin" software is intended for communication between the GRIPLINK Controller from WEISS ROBOTICS and a robot controller. The requirements of the applicable directives and the installation and operating instructions in these instructions must be observed and complied with. Any other use or use beyond the scope of this manual is considered improper use. The manufacturer is not liable for any damage resulting from this.

## 1.3 System requirements

One of the following STÄUBLI robot controllers is required for operation:

- CS9 (s8.10.6)

The following robot options are required to run the software:

- Stäubli SRS

Contact your STÄUBLI dealer to obtain the robot options.

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

## 1.4 License terms

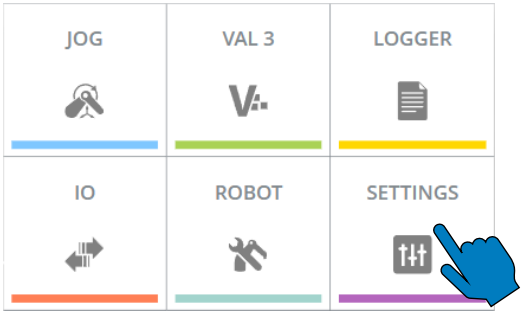
The GRIPLINK plugin is protected by copyright. The respective valid license terms are included in the software package. With the installation you accept these license terms.

# 2 Installation

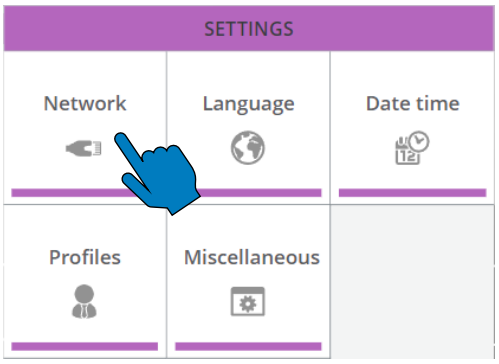
## 2.1 Preparing the Robot

Turn on the robot controller and configure the IP address (e.g. 192.168.1.14). Ensure, that robot controller and GRIPLINK Controller operate in the same network.
Perform the following steps:

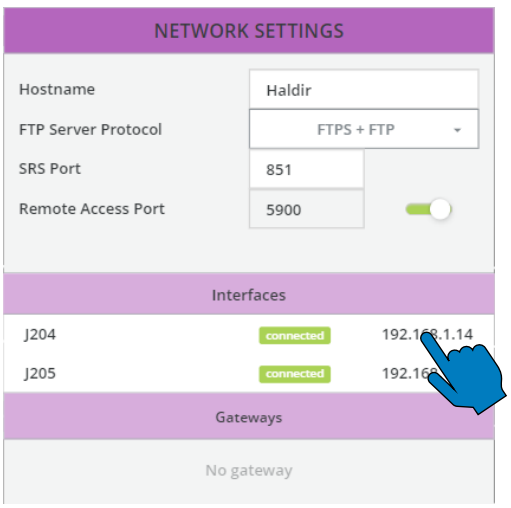
1. Press 
2. Navigate to “SETTINGS“

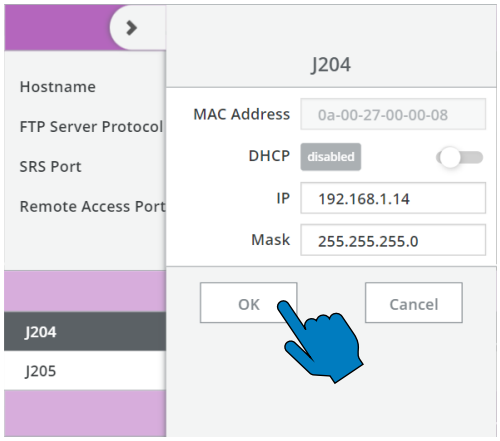



3. Navigate to “network“




4. Set the IP address for the interface in use to a valid address



o operate the GRIPLINK controller, the GRIPLINK plugin is required on the robot controller. To install the GRIPLINK plugin, follow points 1 to 4.

Make sure that you are using the latest version of the GRIPLINK plugin. The current version can be downloaded from **www.griplink.de/plugins.**

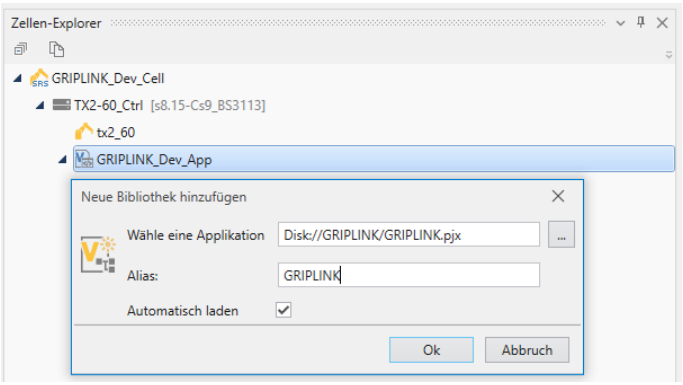
1. Unpack the previously downloaded ZIP archive with the GRIPLINK plugin into the SRS directory of the cell you are using: Staubli > SRS > CellName > ControllerName > usr > usrapp.
2. Open the STAUBLI Robotics Suite:
    - Open the cell
    - Go to the Cell Explorer
    - Select the Val3 application that should use the GRIPLINK plugin, then "right click" > Add > New Library > ... > GRIPLINK > GRIPLINK.pjx > Open > Specify Alias > Load Automatically > Ok

In this manual GRIPLINK is used as alias for all examples.
Commands are called via "call Alias:Command()".

## 2.2 Installation of the software via STAUBLI Robotics Suite (SRS)

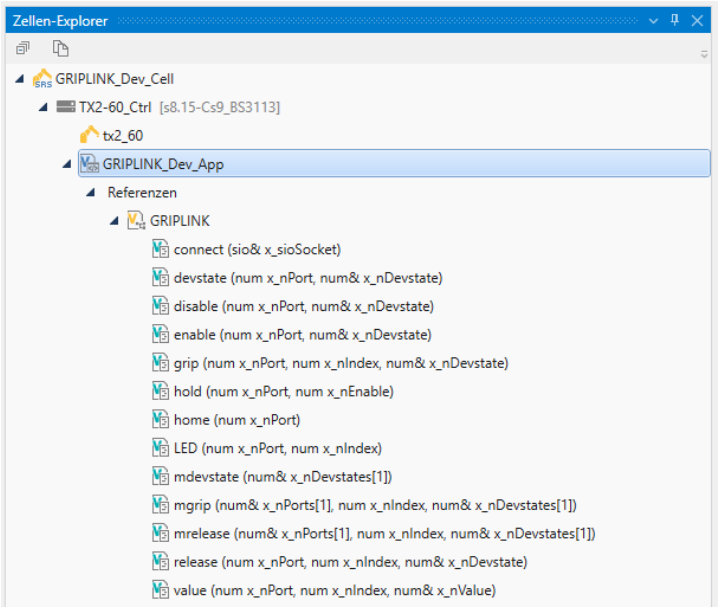
Ensure, you are using the latest version of the GRIPLINK plugin. The latest version can be downloaded here: **www.griplink.de/software**

1. Download the plugin file “griplink_staubli_<Version>.zip“.
2. Unzip the downloaded zip archive to the directory “usrapp“:
   „…/<Cell name>/<Controller name>/usr/usrapp“
3. Open the cell in SRS and navigate to the cell explorer.
4. Right click on the VAL3 program, the plugin should be used with. Add the plugin program “GRIPLINK.pjx“ as a library.




Enter a unique name into the “Alias” field. This alias is required to use the plugin functions in the subprograms of your robot application.

5. The program “GRIPLINK” is now displayed in the cell explorer



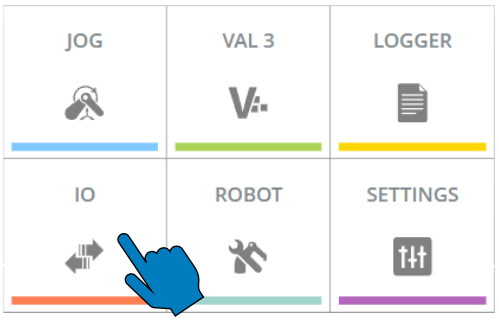
## 2.3 Setting up the socket connection for the GRIPLINK Controller

Communication over the network interface is done using socket variables. Those have to be configured in advance.

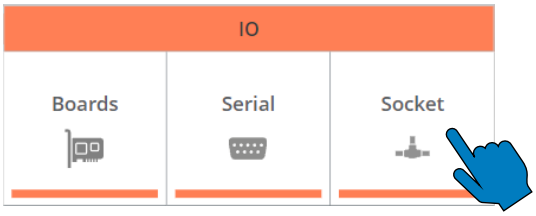
### 2.3.1 Teach Pendant

Perform the following steps:

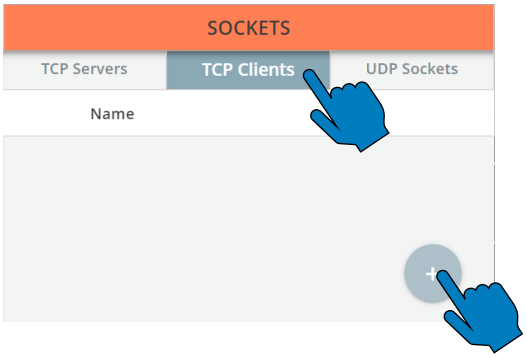
6. Press 
7. Navigate to “IO“

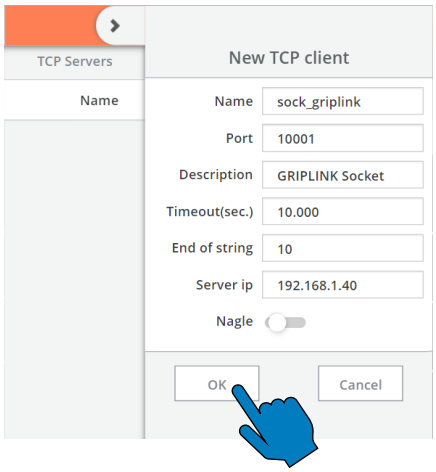



8. Navigate to “Socket“




9. Select “TCP Clients” and open the input mask by clicking  . Enter the name of the socket connection and the values for the remaining parameters. Confirm by clicking “OK”.



Enter the following values:

| Field | Value | Unit |
|---|---|---|
| Server ip | IP address of the GRIPLINK Controller | - |
| Port | 10001 | - |
| Timeout | 10 | Seconds |
| End of string | 10 | - |

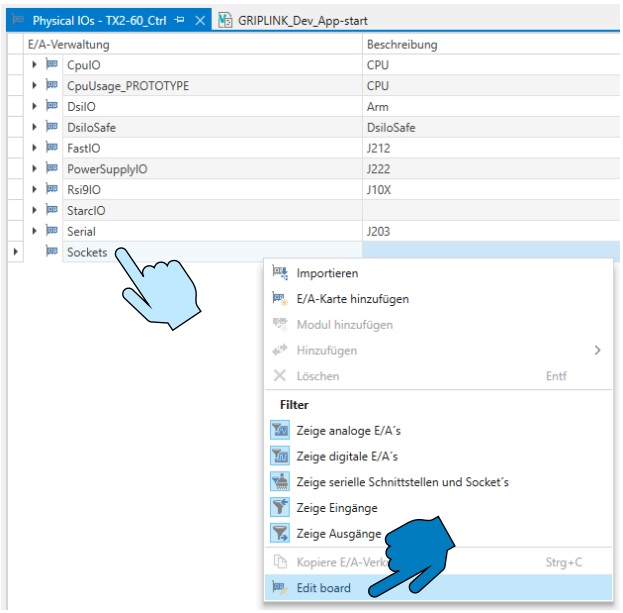
Ensure, that the correct name is entered. It must match the name used in the robot program.

Any other value than the values above lead to malfunctioning!

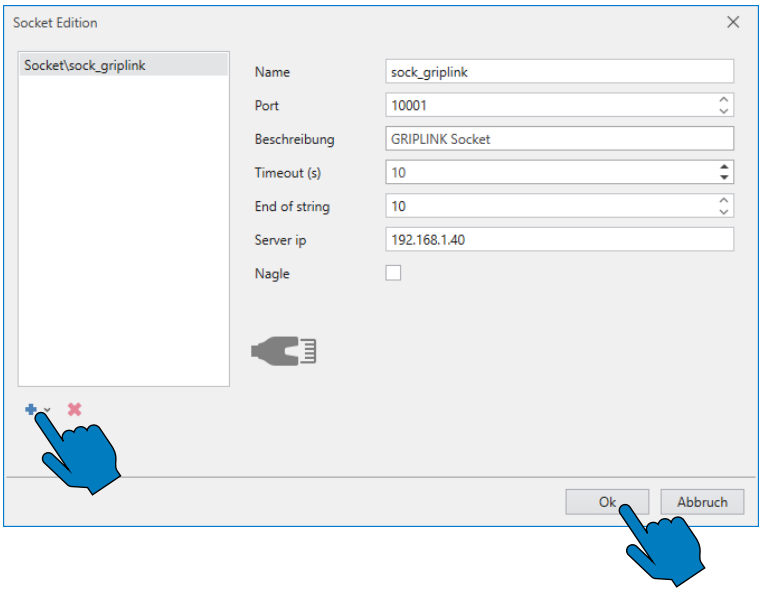
### 2.3.2 STAUBLI Robotics Suite (SRS)

Perform the following steps:

1. Open the I/O-Manager of the robot controller in SRS
2. Open the input mask for a new socket connection by right clicking on “Sockets”




3. Add a new socket by clicking on . Enter the name of the socket connection and all remaining parameters of the interface. Confirm by clicking “OK”.



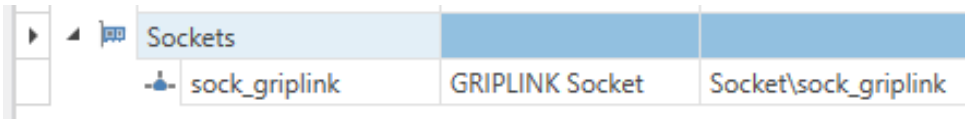

Enter the following values:

| Field | Value | Unit |
|---|---|---|
| Server ip | IP address of the GRIPLINK Controller | - |
| Port | 10001 | - |
| Timeout | 10 | Seconds |
| End of string | 10 | - |

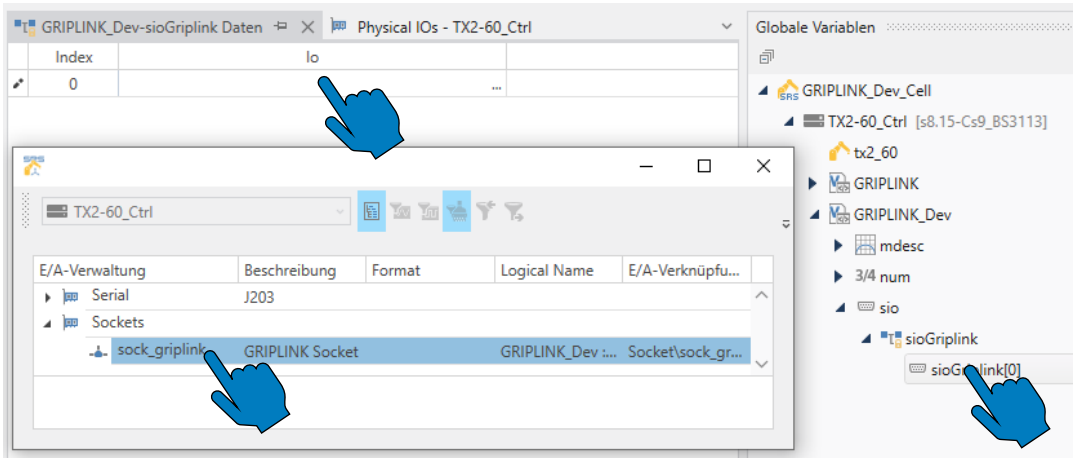
Ensure, that the correct name is entered. It must match the name used in the robot program.

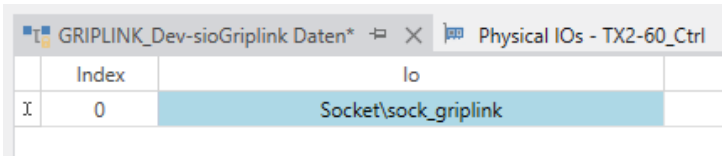
Any other value than the values above lead to malfunctioning!

4. The socket connection is now displayed in the table.




5. The socket connection can now be assigned to a socket variable of the robot program.




When loading VAL3 applications from STAUBLI Robotics Suite to a robot controller it might happen, that the SIO variable has to be linked once more (refer to section 4.1.3).
Create a SIO socket interface (refer to section 2.3.1).

Press and open the Application Manager.
Click on “VAL3 Application” → “Drive” → “Program”
→ “Global Variables” → “sio” → <variable name>
Click on “Link (F2)” → “Edit (F6)” → Socket → „TCP Clients“ → „Socket Name“ → „OK (F8)“ → „OK (F8)“ → „Save (F8)“

## 2.4 Behavior in case of error

If an error occurs within the GRIPLINK plug-in or during communication with the GRIPLINK controller, an error message is always written to the logger. As a rule, the running movements of the robot are also aborted. The same applies if the addressed device is in the FAULT state or changes there due to a command.

The most common error cases and possible solutions are listed in section 4.1.

# 3 Command Reference

The GRIPLINK plugin provides the user with a collection of gripping module-specific functions. Both single and multiple commands are available. The commands are called with the alias of the library reference defined by the user.

***Return values:***

Any return values of a command are written to variables that are passed as "Pass by Reference" when the command is called

***Multi commands:***

With the multi-commands (prefix M), several devices can be addressed simultaneously. These commands are particularly suitable for handling large or bendable workpieces with several gripping modules.

***The basic program flow with the GRIPLINK plugin is always as follows:***

1. Establish connection with GRIPLINK Controller using griplink:connect()
2. For servo gripping modules without absolute encoder: Reference gripping module with griplink:home()
3. Activate device with griplink:enable()
4. Grip/release with griplink:grip()/griplink:mgrip() or griplink:release()/griplink:mrelease()
5. Before terminating the program, close connection to GRIPLINK Controller using griplink:bye()

The basic program flow must be followed for the plugin and connected devices to function normally.

The available commands of the GRIPLINK plug-in are described below. For any example, the library prefix “griplink” is used (refer to section 2.2).

## 3.1 Establish connection – CONNECT

This command establishes the connection between GRIPLINK controller and the robot controller. The socket IO variable, with which the further commands are executed, is passed as a parameter. This must have been linked to a socket accordingly before (refer to section 2.3).

If the connection was established successfully, the plugin automatically checks if the connected GRIPLINK controller supports the version of the used GRIPLINK protocol.

***Syntax***

```
connect(sio& x_sioSocket)
```

***Parameters***

| | |
|---|---|
| sio& x_sioSocket | Reference to socket variable of the used TCP/IP socket |

***Example***

Establish connection between robot and the GRIPLINK via socket variable `sioGriplink`:

```
call GRIPLINK:connect(sioGriplink)
```

If GRIPLINK commands are executed before a CONNECT, a log message is written and the program execution is stopped.

The CONNECT command should be called only once at the beginning of the robot program.

## 3.2 Close connection – BYE

This command closes the connection between robot controller and GRIPLINK controller.

Disconnecting an existing connection is not necessary in normal program operation and should only be used with caution!

***Syntax***

```
bye()
```

***Example***

Close connection to the GRIPLINK Controller:

```
call GRIPLINK:bye()
```

## 3.3 Check connected Device – DEVASSERT

This command checks whether the expected device is connected to the selected port. If the device connected to the port does not have the expected vendor and product ID, the robot program will be stopped immediately.

***Syntax***

```
devassert(num x_nPort, num x_nVID, num x_nPID)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nVID | Vendor ID as specified by the IODD of the expected device |
| num x_nPID | Product ID as specified by the IODD of the expected device |

***Example***

Ensure that the IEG 55-020 device from Weiss Robotics (VID 815, PID 20) is connected to port 3:

```
call GRIPLINK:devassert(3,815,20)
```

The vendor and product ID can be found in the device description or IODD provided by the manufacturer.

## 3.4 Activate device – ENABLE

This command activates the device connected to the selected port. IO-Link gripping modules then change to the RELEASED state and execute the release action of the last selected grip preset.

Using the function argument `x_bWstrEnabled` can be used to activate waiting for a state transition (WSTR) of the respective device. The command execution then waits until the state of the device at the port with the passed index has changed to ENABLED or FAULT. If there is no change of state, the command returns with the TIMEOUT error code.

If WSTR is enabled, the device state is written to the variable `x_nDevstate` passed by reference after command execution.

***Syntax***

```
enable(num x_nPort, num& x_nDevstate, bool x_bWstrEnabled)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num& x_nDevstate | Reference to return variable of the device state after successful command execution and with activated WSTR. |
| bool x_bWstrEnabled | Set to „true“, to activate WSTR<br>Set to „false“, to deactivate WSTR |

***Example***

Activate the device at port 0 and store the new device state into the variable `l_nDevstate`:

```
call GRIPLINK:enable(0, l_nDevstate, true)
```

WSTR should be used only if the device is in DISABLED state before, otherwise the command execution timeout will occur.

Only if WSTR was activated the new device state is written into the return variable!

## 3.5 Deactivate device – DISABLE

This command activates the device connected to the selected port. IO-Link gripping modules then change to the RELEASED state and execute the release action of the last selected grip preset.

The function argument `x_bWstrEnabled` can be used to enable waiting for a state transition (WSTR) of the respective device. The command execution then waits until the state of the device at the port with the passed index has changed to DISABLED or FAULT. If there is no change of state, the command returns with the error code TIMEOUT.

If WSTR is enabled, the device state is written to the variable `x_nDevstate` passed by reference after command execution.

***Syntax***

```
disable(num x_nPort, num& x_nDevstate, bool x_bWstrEnabled)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num& x_nDevstate | Reference to return variable of the device state after successful command execution and with activated WSTR. |
| bool x_bWstrEnabled | Set to „true“, to activate WSTR<br>Set to „false“, to deactivate WSTR |

***Example***

Deactivate the device at port 0 and store the new device state into the variable `l_nDevstate`:

```
call GRIPLINK:disable(0, l_nDevstate, true)
```

WSTR should be used only if the device is not in DISABLED state before, otherwise the command execution timeout will occur.

Only if WSTR was activated the new device state is written into the return variable!

## 3.6 Reference Gripping Module – HOME

This command references the gripper connected to the selected port. The command is blocking and waits until the referencing process has been completed or a timeout has occurred.

The device state is queried after command execution and written to the variable `x_nDevstate` passed by reference.

***Syntax***

```
home(num x_nPort, num& x_nDevstate)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num& x_nDevstate | Reference to return variable of the device state after successful command execution and with activated WSTR. |

***Example***

Reference the gripper at port 0 and store the new device state in the variable `l_nDevstate`:

```
call GRIPLINK:home(0, l_nDevstate)
```

The behavior may vary depending on the connected device. Read the notes in the instructions of the respective device driver!

## 3.7 Grip Workpiece – GRIP

This command executes a grip command with the gripper connected to the selected port and the selected grip preset index.

The function argument `x_bWstrEnabled` can be used to enable waiting for a state transition (WSTR) of the respective device. The command execution then waits until the state of the device at the port with the passed index has changed to HOLDING, NO PART or FAULT. If there is no change of state, the command returns with the TIMEOUT error code.

If WSTR is enabled, the device state is written to the variable `x_nDevstate` passed by reference after command execution.

The grip preset parameters can be configured via the web interface of the GRIPLINK Controller or using the “SET GRIPCFG” command (refer to section 3.11).

***Syntax***

```
grip(num x_nPort, num x_nPreset, num& x_nDevstate, bool x_bWstrEnabled)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nPreset | Index of the grip preset (valid value range depends on the device) |
| num& x_nDevstate | Reference to return variable of the device state after successful command execution and with activated WSTR. |
| bool x_bWstrEnabled | Set to „true“, to activate WSTR<br>Set to „false“, to deactivate WSTR |

***Example***

Gripping module at port 0 is to grip with grip 2. If no workpiece was found, the gripper should wait, open again and try again:

```
l_bLoop = true
while l_bLoop
    // Grip with gripper
    call griplink:grip(0, 1, nDevStatePort0, true)
    if nDevStatePort3 == 5
         logMsg("Gripper in HOLDING state", 1)
         delay(1)
         l_bLoop = false

    elseIf nDevStatePort3 == 7
         logMsg("Gripper in FAULT state", 3)
         return
```

```
    elseIf nDevStatePort3 == 4
         logMsg("Gripper in NO PART state", 2)
         delay(1)

         // Release with gripper
         call griplink:release(x_0, 2, nDevStatePort0, true)
    else
         logMsg("Gripper in invalid state", 3)
         return
    endIf
endWhile
```

Only if WSTR was activated the new device state is written into the return variable!

## 3.8 Synchronous grip of a Workpiece – MGRIP

This command executes a simultaneous grip command with the selected grip preset index with the selected gripping modules.

The function argument x_bMwaitforEnabled can be used to enable waiting for a state transition (MWAITFOR) of the respective devices. This has the consequence that the command execution waits until the state of all selected devices has changed. If no state change occurs, the command returns with the error code TIMEOUT.

The grip preset parameters can be configured via the web interface of the GRIPLINK Controller or using the “SET GRIPCFG” command (refer to section 3.11).

***Syntax***

```
mgrip(num x_nPort, num x_nPorts, bool x_bMwaitforEnabled)
```

***Parameters***

| | |
|---|---|
| num x_nPreset | Index of the grip preset (valid value range depends on the device) |
| num x_nPorts | Selected gripping modules as 32-bit vector:<br>Bit x = 1: Device at port (1 << x) is selected<br>Bit x = 0: Device at port (1 << x) is not selected<br><br>Example: Select grippers at ports 0 and 3<br>→Set bits 0 and 3<br>→Ports = (1 << 0) + (1 << 3) = 1 + 8 = 9 |

bool x_bMwaitforEnabled Set to „true“, to activate MWAITFOR
Set to „false“, to deactivate MWAITFOR

***Example***

Gripping modules at port 0 and 3 grip workpiece with grip preset 2 and then wait for state changes:

```
// Grip with grippers at ports 0 and 3
call griplink:mgrip(2, 9, true)
// Get states of the grippers
call griplink:devstate(0, l_nDevstateGripper0)
call griplink:devstate(3, l_nDevstateGripper3)
...
// Release with grippers at ports 0 and 3
call griplink:mrelease(2, 9, true)
// Get actual states of the grippers
call griplink:devstate(0, l_nDevstateGripper0)
call griplink:devstate(3, l_nDevstateGripper3)
```

## 3.9 Release Workpiece – RELEASE

This command executes a release command with the gripper connected to the selected port and the selected grip preset index.

The function argument `x_bWstrEnabled` can be used to enable waiting for a state transition (WSTR) of the respective device. The command execution then waits until the state of the device at the port with the passed index has changed to RELEASED or FAULT. If there is no change of state, the command returns with the TIMEOUT error code.

If WSTR is enabled, the device state is written to the variable `x_nDevstate` passed by reference after command execution.

The grip preset parameters can be configured via the web interface of the GRIPLINK Controller or using the "SET GRIPCFG" command (refer to section 3.11).

***Syntax***

```
release(num x_nPort, num x_nPreset, num& x_nDevstate, bool x_bWstrEnabled)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nPreset | Index of the grip preset (valid value range depends on the device) |
| num& x_nDevstate | Reference to return variable of the device state after successful command execution and with activated WSTR. |
| bool x_bWstrEnabled | Set to „true", to activate WSTR<br>Set to „false", to deactivate WSTR |

***Example***

Refer to example in 3.7

Only if WSTR was activated the new device state is written into the return variable!

## 3.10 Synchronous grip of a Workpiece – MRELEASE

This command executes a simultaneous release command with the selected grip preset index with the selected gripping modules.

The function argument `x_bMwaitforEnabled` can be used to enable waiting for a state transition (MWAITFOR) of the respective devices. This has the consequence that the command execution waits

until the state of all selected devices has changed. If no state change occurs, the command returns with the error code TIMEOUT.

The grip preset parameters can be configured via the web interface of the GRIPLINK Controller or using the “SET GRIPCFG” command (refer to section 3.11).

***Syntax***

```
mrelease(num x_nPort, num x_nPorts, bool x_bMwaitforEnabled)
```

***Parameters***

| | |
|---|---|
| num x_nPreset | Index of the grip preset (valid value range depends on the device) |
| num x_nPorts | Selected gripping modules as 32-bit vector:<br>Bit x = 1: Device at port (1 << x) is selected<br>Bit x = 0: Device at port (1 << x) is not selected<br><br>Example: Select grippers at ports 0 and 3<br>→Set bits 0 and 3<br>→Ports = (1 << 0) + (1 << 3) = 1 + 8 = 9 |
| bool x_bMwaitforEnabled | Set to „true“, to activate MWAITFOR<br>Set to „false“, to deactivate MWAITFOR |

***Example***

Refer to example in 3.8

## 3.11 Configure a grip preset – SETGRIPCFG

This command changes the parameters of a handle preset. Up to eight parameters can be set, all of which must always be sent to the device. This also applies to devices that have fewer than eight adjustable parameters.
Depending on the device, each parameter has an individual meaning, which can be taken from the corresponding GRIPLINK device driver.

***Syntax***

```
setgripcfg(num x_nPort, num x_nPreset, string x_sTag, num& x_nParams[---])
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nPreset | Index of the grip preset (valid value range depends on the device) |
| string x_sTag | Preset tag |
| num& x_nParams[---] | Array with 8 elements for the preset's parameters |

***Parameters for IO-Link gripping modules by Weiss Robotics***

| | |
|---|---|
| x_nParams[0] | No Part Limit (-300000..300000) in 1/1000 mm |
| x_nParams[1] | Release Limit (-300000..300000) in 1/1000 mm |
| x_nParams[2] | Force Factor (0..100) in 1/1000% |
| x_nParams[4..7] | Not used, set to 0 |

***Example***

For the gripping module at port 3, set the parameters of the handle with index 1 to:

No Part Limit = 31 mm →in 1/1000 mm: 31000
Release Limit = 41 mm →in 1/1000 mm: 41000
Force Factor = 59 %: →in 1/1000 %: 59000

```
// l_nGripPresetParameters defined as array of 8 integers
l_nGripPresetParameters[0] = 31000
l_nGripPresetParameters[1] = 41000
l_nGripPresetParameters[2] = 59000
call griplink:setgripcfg(3, 1, l_nGripPresetParameters)
```

## 3.12 Get current Device State – DEVSTATE

This command reads back the current device state of the selected device and stores it in the return variable.

***Syntax***

```
devstate(num x_nPort, num& x_nDevstate)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num& x_nDevstate | Reference to return variable of the device state |

***Example***

Check the state of the device at port 3 for the value DISABLED (2):

```
call griplink:devstate(3, l_nDevstatePort3)
if (l_nDevstatePort3 == 2)
     // Process state DISABLED
endIf
```

## 3.13 Get Device Value – VALUE

This command reads an indexed device value and stores it in the return variable.

The valid value range for the value index can be found in the GRIPLINK driver description of the respective device.

***Syntax***

```
value(num x_nPort, num x_nIndex, num& x_nValue)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nIndex | Index of the device value (valid value range depends on the device) |
| num& x_nValue | Reference to return variable of the value |

***Example***

Check the primary value of the device at port 3 for the minimum value 14.000:

14 => 14 · 1,000 = 14,000 (in 1/1000, refer to GRIPLINK protocol specification)

```
call griplink:value(3, l_nValue)
if (l_ nValue > 14000)
     // Value larger than 14.000
endIf
```

## 3.14 Set a Device Value – SETVALUE

This command can be used to set a device-specific value. The value can be selected via an index, which has a valid value range dependent on the device. The value to be written can have different meanings depending on the connected device.

The valid value range for the value index can be found in the GRIPLINK driver description of the respective device.

***Syntax***

```
setvalue(num x_nPort, num x_nIndex, num x_nValue)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nIndex | Index of the device value (valid value range depends on the device) |
| num x_nValue | Value (valid value range depends on the device) |

***Example***

Set in the device at port 0 the value with index 2 to 10:

10 => 10 · 1,000 = 10,000 (in 1/1000, refer to GRIPLINK protocol specification)

```
call griplink:setvalue(0, 2, 10000)
```

## 3.15 Waiting for a device value to be reached – WAITVAL

This command can be used to wait until a device-specific value has reached a transferred target value or a timeout has occurred while waiting for the target value. The value can be selected via an index, which has a valid value range dependent on the device.

The valid value range for the value index can be found in the GRIPLINK driver description of the respective device.

***Syntax***

```
waitval(num x_nPort, num x_nIndex, num x_nValue)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nIndex | Index of the device value (valid value range depends on the device) |
| num& x_nValue | Reference to return variable of the value |

***Example***

Wait until the distance sensor connected to port 2 has reached the distance value (index 0) 100:

```
// 100 => 100 · 1,000 = 100,000
call griplink:waitvalue(2, 0, 100000)
```

## 3.16 Control Force Retention – CLAMP

With the CLAMP command, gripping modules with integrated gripping force safety device can de-energize the drive in the HOLDING state to reduce the current requirement of the gripping module. The workpiece continues to be held securely.

If CLAMP is activated, the drive switches off in the HOLDING state. If CLAMP is deactivated, the gripping force control remains active in the HOLDING state and the control of the gripping module permanently regulates the gripping force to.

The CLAMP command is not available for all gripping modules.

***Syntax***

```
clamp(num x_nPort, num x_bEnabled)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_bEnabled | CLAMP state |

***Example***

Activate the clamping in the CRG 200-085 from Weiss Robotics at port 1:

```
call griplink:clamp(1, true)
```

## 3.17 Controlling the status LED – LED

This command sets the preset of the light ring of a selected CRG gripping module.

Light patterns can be configured via the web interface of the GRIPLINK controller.

The LED command is not available for all gripping modules.

***Syntax***

```
led(num x_nPort, num x_nPreset)
```

***Parameters***

| | |
|---|---|
| num x_nPort | Index of the device (0 to 31) |
| num x_nPreset | Index of the LED preset (valid value range depends on the device) |

***Example***

Perform a grip command with the gripping module at port 3 and read out the finger position. Change the color of the LED ring to light pattern 0, if the finger position is greater than or equal to 8.1 mm afterwards and to light pattern 1, if smaller than 8.1 mm:

```
// Grip with gripper at port 3 and preset 0, wait for state
transition
call griplink:grip(3, 0, true)
// Read finger position and set respective LED preset
call griplink:value(3, 0, l_nGripperPosition)
if (l_nGripperPosition >= 8100)
    call griplink:led(1, 1)
else
    call griplink:led(1, 2)
endIf
```

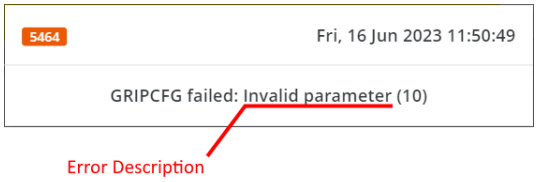
# 4 Troubleshooting

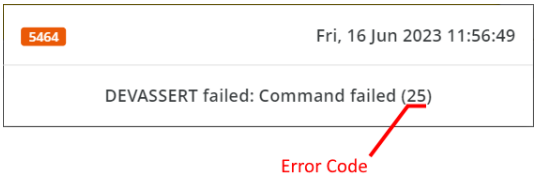
The GRIPLINK plug-in issues error messages during operation. Important messages and solutions are explained below.

## 4.1 Error messages

### 4.1.1 <Command name> failed: <Error Description> (<Error Code>)

Error messages that appear due to a protocol error. Examples:

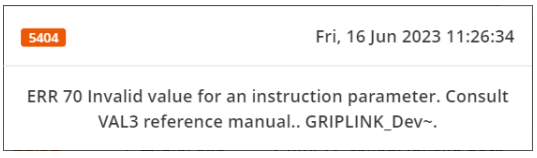



| Possible cause | Remedy |
|---|---|
| The cause of the error is displayed as a message and in brackets as an error code | This is a GRIPLINK protocol error<br>• Check error description and code (refer to document „GRIPLINK Unified Command Set Reference Manual“)<br>• Check command function call in robot program<br>• Use correct function arguments (ensure correct value ranges!) |

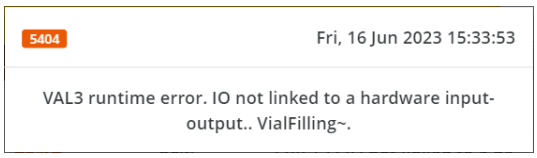
### 4.1.2 Log message „VAL3/ERR 70 invalid value for an instr. param.“

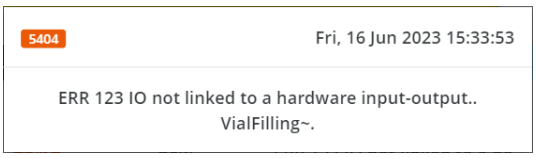
Error messages that occur during the execution of the VAL3 program.


| Possible cause | Remedy |
|---|---|
| Timeout occurred by WSTR command | • Invalid command order (e.g. RELEASE command with activated WSTR after ENABLE command for the same port)<br>→Adjust command order<br>→Deactivate WSTR for the second command |

### 4.1.3 Log message „VAL3/ERR 123 IO not linked to a hardware input-output“




| Possible cause | Remedy |
|---|---|
| The socket variable is not linked | Refer to section 2.3.2 |

# Appendix A. Device state

| Device status | Code | Description |
|---|---|---|
| NOT CONNECTED | 0 | No device connected |
| NOT INITIALIZED | 1 | Not initialized |
| DISABLED | 2 | Ready for operation, but not activated |
| RELEASED | 3 | Workpiece released |
| NO PART | 4 | No workpiece found |
| HOLDING | 5 | Workpiece is held |
| OPERATING | 6 | Ready for operation |
| FAULT | 7 | Error condition |