



GRIPLINK PLUGIN FOR EPSON

Version 5.0.0

April 2025



Contents

1	Introduction	2
1.1	Notation and symbols	2
1.2	Intended use	2
1.3	System requirements	2
1.4	License terms	3
2	Installation	4
2.1	Installing the software	4
2.2	Sample programs	4
2.3	Network configuration	4
3	Command reference	6
3.1	Establish connection - Griplink_Connect	7
3.2	Terminate connection - Griplink_Disconnect	8
3.3	Activate device - Griplink_Enable	9
3.4	Deactivate device - Griplink_Disable	10
3.5	Query device status - Griplink_DevState	11
3.6	Wait for device state change - Griplink_WaitForStateTransition	12
3.7	Reference gripping module - Griplink_Home	13
3.8	Workpiece index-based gripping - Griplink_Grip	14
3.9	Workpiece can be gripped freely parameterized - Griplink_FlexGrip	15
3.10	Release workpiece index-based - Griplink_Release	16
3.11	Release workpiece freely parameterizable - Griplink_FlexRelease	17
3.12	Read sensor value - Griplink_Value	19
3.13	Set control value - Griplink_SetVal	20
3.14	Wait for value - Griplink_WaitVal	21
3.15	Control gripping force maintenance - Griplink_Clamp	22
3.16	Control of the LED display - Griplink_LED	23
3.17	Set grip settings - Griplink_GripCfg	24
3.18	Ensure device type - Griplink_DevAssert	25
4	Error handling	26
5	Device states	28

1 Introduction

With GRIPLINK technology, servo-electric and smart pneumatic gripper modules from WEISS ROBOTICS as well as selected sensors and actuators from third-party manufacturers can be connected to robot systems from leading manufacturers via a network connection. The GRIPLINK plug-in for EPSON is the control-side link and enables the simple integration of GRIPLINK technology from WEISS ROBOTICS into robot systems from the manufacturer EPSON.



These instructions describe the functions of the GRIPLINK plug-in. Information on installation, commissioning and operation of the GRIPLINK interface converter can be found in the operating instructions for the respective module. These can be found online at www.griplink.de

1.1 Notation and symbols

The following symbols are used in these instructions to provide a better overview:



Functional or safety-relevant information. Failure to observe this may endanger the safety of personnel and the system, damage the appliance or impair the function of the appliance.



Additional information for a better understanding of the facts described.



Reference to further information.

1.2 Intended use

The "GRIPLINK-Plugin" software is intended for communication between the GRIPLINK interface converter from Weiss Robotics and a robot controller. The requirements of the applicable directives and the installation and operating instructions in this manual must be observed and complied with. Any other or additional use is considered improper use. The manufacturer is not liable for any resulting damage.

1.3 System requirements

The following EPSON products are required for operation:

- EPSON robot controller of the RC700-A or RC90-B series
- EPSON development environment RC+ 7.0 in version 7.4.5 or



Contact EPSON or your EPSON partner to purchase these products.



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

1.4 License terms

The GRIPLINK plug-in is protected by copyright. The applicable license terms are included with the software package. With the installation you accept these license terms.

2 Installation

2.1 Installing the software

To operate the GRIPLINK-ET4, the GRIPLINK plug-in provided by Weiss Robotics is required on the robot controller. To install the GRIPLINK plug-in, carry out the following steps:



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

1. Unpack the previously downloaded ZIP archive with the GRIPLINK plug-in into a directory of your choice
2. Create a new project in the EPSON RC+ robot development environment or open an existing project in which the GRIPLINK is to be used.
3. Import the unpacked files in EPSON RC+ into your new project using the menu item "File" -> "Import". Please note that the package contains several files with the extensions *.prg and *.inc. Both file types must be imported separately.
4. The necessary functions for using the GRIPLINK-ET4 in your project are now available.

2.2 Sample programs

The GRIPLINK plug-in for EPSON contains several program examples that illustrate the use of the individual GRIPLINK commands. You will find the program examples in the Example.prg file. To start the examples, call them up from the main function of your project.

2.3 Network configuration

The connection between the GRIPLINK-ET4 and the robot controller is established via a TCP/IP network connection. To do this, the robot controller must be assigned an IP address in the system settings. Please note that the IP address of the robot controller must be in the same subnet as the IP address of the GRIPLINK-ET4.

The IP address of the GRIPLINK is set to 192.168.1.40 by default and can be changed at any time via its web interface. To do this, connect the GRIPLINK to a PC or laptop and open the web interface in the browser of your choice by entering the address <http://192.168.1.40>. You can access the IP settings via the "Config" button.

The command interface of the GRIPLINK accepts incoming connections on port 10001 (TCP).



For more information on the configuration of the GRIPLINK, please refer to the corresponding user manual.

The IP address of the robot controller can be set in the EPSON RC+ development environment in the menu item "Settings" → "System settings" under "Controller" → "Configuration".

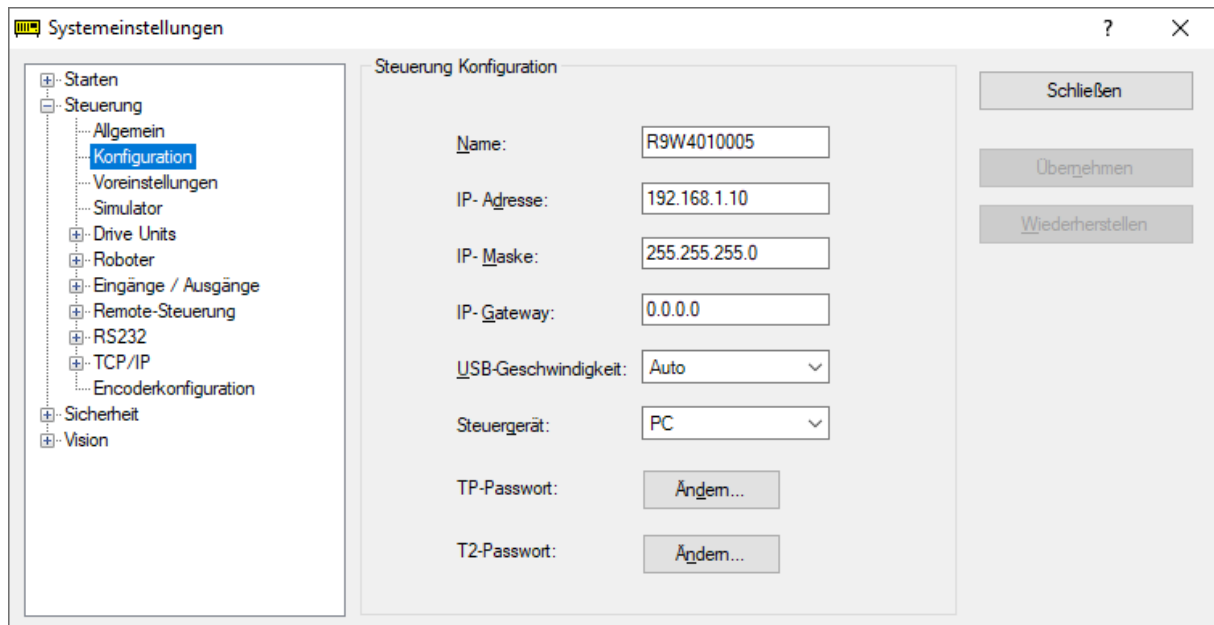


Figure1 : Setting the IP address on the robot controller

3 Command reference

The GRIPLINK plug-in provides the user with a collection of functions for controlling gripping modules or actuators and for reading out sensor readings.

The basic program sequence for gripping with the GRIPLINK plug-in is always as follows

1. Establishing a connection with CONNECT
2. Activate device with ENABLE
3. For servo gripping modules without absolute encoder: Reference gripping module with HOME
4. Gripping/releasing with GRIP/FLEXGRIP or RELEASE/FLEXRELEASE

The available commands of the GRIPLINK plug-in are described below.

Command	Description
Griplink_Connect()	Establishing a connection to the GRIPLINK
Griplink_Disconnect()	Disconnect from the GRIPLINK
Griplink_Enable()	Activate device
Griplink_Disable()	Deactivate device
Griplink_DevState()	Query device status
Griplink_WaitForStateTransition()	Waiting for a change of state
Griplink_Home()	Referencing the gripping module
Griplink_Grip()	Grip workpiece via preset
Griplink_FlexGrip()	Grip workpiece with parameters
Griplink_Release()	Release workpiece via preset
Griplink_FlexRelease()	Release workpiece with parameters
Griplink_Value()	Query sensor value
Griplink_SetVal()	Set tax value
Griplink_Waitval()	Wait until a sensor value is in a predefined window
Griplink_Clamp()	Control mechanical gripping force maintenance
Griplink_LED()	Control LED display
Griplink_GripCfg()	Change grip preset
Griplink_DevAssert()	Ensure the specified device is connected to a port

3.1 Establish connection - Griplink_Connect

This command establishes the connection between the GRIPLINK interface converter and the robot controller. For communication, one of 16 available ports (no. 201 to 216) must be selected on the robot controller.



Please note that the selected port must be available exclusively for communication with the GRIPLINK and must not be used anywhere else in the robot program.



The IP address of the GRIPLINK can be changed via the web interface.

Signature

```
Function Griplink_Connect(strIPAddr$ As String, intControllerPortNo As Integer)
```

Parameters

strIPAddr\$	IP address of the Griplink as string
intControllerPortNumber	Port number of the preconfigured TCP/IP connection. Must be between 201 and 216.

Return value

-

Example

Establish a connection between the robot controller and GRIPLINK with the IP address 192.168.1.40 using TCP/IP port no. 201.

```
Griplink_Connect("192.168.1.40", 201)
```

3.2 Terminate connection - Griplink_Disconnect

This function terminates an existing connection that was established using Griplink_Connect().

Signature

```
Function Griplink_Disconnect()
```

Parameters

-

Return value

-

Example

```
Griplink_Disconnect()
```

3.3 Activate device - Griplink_Enable

This command activates the device on the selected GRIPLINK port. The command triggers a status transition to *DS_ENABLED* and activates the connected device if necessary. It is used to reactivate the corresponding device after acknowledging an error with DISABLE.



For gripping modules from Weiss Robotics, please note that the command is no longer mandatory when starting the robot program and can in any case only be used after the homing run has been executed.

Signature

```
Function Griplink_Enable(ubDevicePort As UByte)
```

Parameters

ubDevicePort	Port index (0 to 31)
--------------	----------------------

Return value

-

Example

Activate the device port 0:

```
Griplink_Enable(0)
```

3.4 Deactivate device - Griplink_Disable

This command deactivates the device on the selected GRIPLINK port. In the case of a connected gripping module, this function can be used, for example, to de-energize the drive of the gripping module.

Signature

```
Function Griplink_Disable(ubDevicePort As UByte)
```

Parameters

ubDevicePort	Port index (0 to 31)
--------------	----------------------

Return value

-

Example

Change the gripping module at port 1:

```
'Establishing a connection between the robot and the GRIPLINK  
Griplink_Connect("192.168.1.40", 201)  
'Execute program (e.g. Pick & Place)  
'...  
'Deactivate gripping module on port 1  
Griplink_Disable(1)  
'Now the gripper module can be replaced mechanically.  
'...  
'Activate the new gripping module.  
Griplink_Enable(1)  
'Continue program with new tool  
'...
```

3.5 Query device status - Griplink_DevState

This function returns the device status of the selected device. The device status is coded as an integer value. To simplify and improve readability, constants are defined in the *Griplink.inc* file that can be used for processing device statuses.

Signature

```
Function Griplink_DevState(ubDevicePort As UByte)
```

Parameters

ubDevicePort Port index (0 to 31)

Return value

Current device status (see5

Example

Wait until the device status of the gripping module at port 2 has changed to "HOLDING" (4):

```
#include "Griplink.inc"
'Query device status of gripping module on port 2
Integer intGripState
'The constant DS_HOLDING is defined in Griplink.inc
Do While intGripState <> DS_HOLDING
    intGripState = Griplink_DevState(2)
Loop
```

3.6 Wait for device state change - Griplink_WaitForStateTransition

This function waits for a change in the device status and returns the changed status. This command can be used to wait for the end of the command after an asynchronous gripper or release command before starting a robot movement, for example.

Signature

```
Function Griplink_WaitForStateTransition(ubDevicePort As UByte) As Integer
```

Parameters

ubDevicePort Port index (0 to 31)

Return value

New device status (see5

Example

Grip asynchronously with the gripper module at port 2. Wait until the device status has changed (and thus the gripping movement has been completed) and output it:

```
#include "Griplink.inc"
Integer intGripState

'Start a grabbing movement:
Griplink_Grip(2, 0, True)

'Wait for change of state:
intGripState = Griplink_WaitForStateTransition(2)
print "Grip state changed to value " + Str$(intGripState)
```

3.7 Reference gripping module - Griplink_Home

References the selected servo gripper. The command executes a reference run of the gripping module and waits until this is completed. After the HOME command has been executed, the fingers of the gripping module are powerless and must be moved to a defined position with *Griplink_Grip()* / *Griplink_FlexGrip()* or *Griplink_Release()* / *Griplink_FlexRelease()*.



The direction of the reference run can be configured via the web interface of the GRIPLINK interface converter.

Signature

```
Function Griplink_Home(ubDevicePort As UByte)
```

Parameters

ubDevicePort	Port index (0 to 31)
--------------	----------------------

Return value

-

Example

Reference gripping module at port 3:

```
#include "Griplink.inc"
Integer intGripState
'Reference gripping module at port 3
Griplink_Home(3)
intGripState = Griplink_DevState(3)
If intGripState = DS_IDLE Then
    'Gripper module is referenced
Else
    'Gripper module is not referenced
EndIf
```

3.8 Workpiece index-based gripping - Griplink_Grip

Grips a workpiece with the selected gripper module and the selected grip. If boolBlocking is set, the command blocks until the gripping state changes to either "HOLDING", "NO PART" or "FAULT".



The gripping parameters can be configured via the web interface of the GRIPLINK interface converter.

Signature

```
Function Griplink_Grip(ubDevicePort As UByte, ubGripIndex As UByte,  
boolBlocking As Boolean )
```

Parameters

ubDevicePort	Port index (0 to 31)
ubGripIndex	Selected grip (value range depends on the gripper module)
boolBlocking	True: The command waits until the gripping module generates a state change. False: The command triggers the gripping movement and returns immediately.

Return value

-

Examples

Gripper module at port 0 should grip with preset 2. If no part is found, the program should output a message:

```
#include Griplink.inc  
Integer intGripState  
Griplink_Grip(0, 2, True)  
intGripState = Griplink_DevState(0)  
'The constants DS_NO_PART and DS_HOLDING are defined in Griplink.inc  
'defined  
If intGripState = DS_NO_PART Then  
    Print "No part found"  
ElseIf intGripState <> DS_HOLDING Then  
    Print "Unexpected grip state"  
EndIf
```

3.9 Workpiece can be gripped freely parameterized - Griplink_FlexGrip

Grips a workpiece with the selected gripper module and uses the transferred gripping parameters. If boolBlocking is set, the command blocks until the gripping status changes to either "HOLDING", "NO PART" or "FAULT".

Signature

```
Function Griplink_FlexGrip(ubDevicePort As UByte, rNoPartPos As Real,  
rForce As Real, rSpeed As Real, rAcc As Real, boolBlocking As Boolean)
```

Parameters

ubDevicePort	Port index (0 to 31)
rNoPartPos:	No part position (mm)
rForce	Gripping force (N)
rSpeed	Speed of the gripper fingers (mm/s). Set the value to 0 to use the automatically calculated speed based on the force setting.
rAcc	Finger acceleration (mm/s ²). Set the value to 0 to use the default value of the gripper.
boolBlocking	True: The command waits until the gripping module generates a state change. False: The command triggers the gripping movement and returns immediately.

Return value

-

Examples

Gripper module at port 0 is to grip at no part position 4.5 mm with 40 N gripping force. If no part is found, the program should output a message:

```
#include Griplink.inc  
Integer intGripState  
Griplink_FlexGrip(0, 4.5, 40, 0, 0, True)  
intGripState = Griplink_DevState(0)  
'The constants DS_NO_PART and DS_HOLDING are defined in Griplink.inc  
'defined  
If intGripState = DS_NO_PART Then  
    Print "No part found"  
ElseIf intGripState <> DS_HOLDING Then  
    Print "Unexpected grip state"  
EndIf
```

3.10 Release workpiece index-based - Griplink_Release

Releases the workpiece gripped with the selected gripper module, using a previously set gripping preset. If boolBlocking is set, the command blocks until the workpiece has been released or an error has occurred (status change to DS_RELEASED or DS_FAULT).



The gripping parameters can be configured via the web interface of the GRIPLINK interface converter.

Signature

```
Function Griplink_Release(ubDevicePort As UByte, ubGripIndex As UByte,  
    boolBlocking As Boolean)
```

Parameters

ubDevicePort	Port index (0 to 31)
ubGripIndex	Selected grip (value range depends on the gripper module)
boolBlocking	True: The command waits until the gripping module generates a state change. False: The command triggers the release movement and returns immediately.

Return value

-

Examples

The gripper module at port 0 should release a workpiece held with grip preset 2:

```
#include Griplink.inc  
Integer intGripState  
Griplink_Release(0, 2, True)  
intGripState = Griplink_DevState(0)  
'The constant DS_RELEASED is defined in Griplink.inc  
'defined  
If intGripState = DS_RELEASED Then  
    Print "Part released"  
EndIf
```

3.11 Release workpiece freely parameterizable - Griplink_FlexRelease

Releases the workpiece gripped with the selected gripping module and uses the transferred movement parameters. If boolBlocking is set, the command blocks until the workpiece is released or an error occurs (status change to DS_RELEASED or DS_FAULT).

FLEXRELEASE can also be used to freely pre-position the gripper fingers. Please note that FLEXRELEASE does not lead to a state transition if the gripping module is already in the DS_RELEASED state. To wait for the end of the movement, the WAITVAL command must be used after calling FLEXRELEASE!

Signature

```
Function Griplink_FlexRelease(ubDevicePort As UByte, rReleasePos As Real, rSpeed As Real, rAcc As Real, boolBlocking As Boolean)
```

Parameters

ubDevicePort	Port index (0 to 31)
rReleasePos	Release position (mm)
rSpeed	Finger speed (mm/s). Set to 0 to use gripper default value.
rAcc	Finger acceleration (mm/s ²). Set the value to 0 to use the default value of the gripper.
boolBlocking	True: The command waits until the gripping module generates a state change. False: The command triggers the release movement and returns immediately.

Return value

-

Examples

The gripping module at port 0 should release the held workpiece by opening to 19.5 mm:

```
#include Griplink.inc
Integer intGripState
Griplink_FlexRelease(0, 19.5, 0, 0, True)
intGripState = Griplink_DevState(0)
'The constant DS_RELEASED is defined in Griplink.inc
'defined
If intGripState = DS_RELEASED Then
    Print "Part released"
EndIf
```

The gripper module at port 0 should release the held workpiece by opening to 19.5 mm and then pre-position the fingers to 10 mm in order to grip the next workpiece:

```
#include Griplink.inc

Integer intGripState
```

```

Griplink_FlexRelease(0, 19.5, 0, 0, True)
intGripState = Griplink_DevState(0)
If intGripState = DS_RELEASED Then
    Print "Part released"
EndIf

'TODO: Move Robot away from released Workpiece
'Pre-position to 10 mm:
Griplink_FlexRelease(0, 10, 0, 0, False)

'Do some robot movement here

'Ensure Fingers reached target position before last step towards
next workpiece:
If Griplink_Waitval(0, 10, 0.1, 5000) <> True Then
    Print "Error: Target position not reached"
    'ERROR HANDLING HERE!
EndIf

'Continue robot movement...

```

3.12 Read sensor value - Griplink_Value

This command returns the current measured value of a connected sensor. In the case of a connected gripping module from Weiss Robotics, for example, the current finger position can be read out in millimetres. If several measured values are available, the desired measured value can be selected via the *ubIndex* parameter.

For gripping modules from Weiss Robotics, index 0 always returns the finger position in millimeters (mm).



A list of available values for the supported devices can be found at www.griplink.de/#downloads

Signature

```
Function Griplink_Value(ubDevicePort As UByte, ubIndex As Ubyte) As Real
```

Parameters

ubDevicePort	Port index (0 to 31)
ubIndex	Measured value index (value range depends on device)

Return value

Sensor value (e.g. finger position in mm)

Example

Read the finger position of gripper 0:

```
Real rPosition  
rPosition = Griplink_Value(0, 0)  
Print "Finger position is " + Str$(rPosition) + " mm"
```

3.13 Set control value - Griplink_SetVal

This command sets a value to control the gripping module. The command is non-blocking, i.e. it does not wait for the value to be accepted. The WAITVAL command can be used to ensure that the value is accepted. The command is function-dependent and not available on all devices.

Signature

```
Function Griplink_SetVal(ubDevicePort As UByte, ubIndex As UByte, rValue  
As Real)
```

Parameters

ubDevicePort	Port index (0 to 31)
ubIndex	Control value index (value range depends on device)
rValue	Control value

Return value

-

Example

Set the control value with index 1 for the device at port 0 to 42.5:

```
Griplink_SetVal(0, 1, 42.5)
```

3.14 Wait for value - Griplink_WaitVal

The function blocks until a device value is within a specified window or has exceeded it. The command is function-dependent and not available for all devices.

Signature

```
Function Griplink_WaitVal(ubDevicePort As UByte, ubIndex As UByte,  
    rThreshold As Real, rWindowSize As Real, lngTimeout As Long) as Boolean
```

Parameters

ubDevicePort	Port index (0 to 31)
ubIndex	Index of the device value (value range depends on the device)
rThreshold	Center of the target window in the unit of the indexed value
rWindowSize	Size of the window in the unit of the indexed value
lngTimeout	Maximum waiting time until the value should appear in the window

Return value

True: the measured value has reached (or exceeded) the window

False: the measured value has reached the window (timeout)

Example

Set the finger position for pre-positioning the gripper fingers on a WPG gripping module to 63 mm. After sending the FLEXRELEASE command, the gripper fingers move to an opening width of 63 mm. The FLEXRELEASE command returns immediately, i.e. the target position is not yet safely reached after the command has been executed. To ensure this, another command must be called up with WAITVAL. In this example, the target position of 63 mm is specified as the window center and a window size of 0.1 mm is selected. The timeout value is assumed to be 5000 milliseconds:

```
Griplink_FlexRelease(0, 63, 0, 0, False)  
If Griplink_WaitVal(0, 0, 63, 0.1, 5000) = True Then  
    Print "Position reached"  
Else  
    Print "Position not reached"  
EndIf
```

3.15 Control gripping force maintenance - Griplink_Clamp

The innovative gripping force safety device PERMAGRIP® designed by Weiss Robotics maintains the gripping force on the workpiece even if the power supply to the gripping module is unexpectedly interrupted. Furthermore, PERMAGRIP® enables continuous gripping without the gripper module overheating.

This command activates or deactivates the gripping force maintenance for the selected gripping module. If the gripping module is deactivated (see *Griplink_Enable()*, chapter3.2 and *Griplink_Disable()*, chapter3.4), this function can be used to allow or prevent manual movement of the gripper fingers.



This function is irrelevant for securing the gripping force in the event of an interruption in the power supply. The gripping force safety device is always active in the event of a power failure.



PERMAGRIP is not available for all gripping modules.

Signature

```
Function Griplink_Clamp(ubDevicePort As UByte, boolEnable As Boolean)
```

Parameters

ubDevicePort	Port index (0 to 31)
boolEnable	Enable (<i>True</i>) or disable (<i>False</i>) PERMAGRIP®

Return value

-

Example

Grip workpiece with grip 0 on gripper 2. Activate PERMAGRIP when a part is held:

```
#include Griplink.inc
Integer intGripState
'Grip part
intGripState = Griplink_Grip(2, 0, True)
If intGripState = DS_HOLDING Then
    'Wait briefly, then activate mechanical gripping force
    maintenance
    Wait 0.5
    Griplink_Clamp(2, True)
EndIf
```

3.16 Control of the LED display - Griplink_LED

This command changes the color and pattern of the illuminated ring of a selected gripping module of the CRG series from Weiss Robotics.



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



This function is only available for CRG series gripping modules.

Signature

```
Function Griplink_LED(ubDevicePort As UByte, ubLEDIndex As UByte)
```

Parameters

ubDevicePort	Port index (0 to 31)
ubLEDIndex	Index of the predefined light pattern (0 to 7)

Return values

-

Example

Grip with the gripper module at port 3 and other the color of the illuminated ring on the light pattern 0, if the finger position afterwards is greater than or equal to 80 mm and on light pattern 1, if smaller:

```
#include "Griplink.inc"
Integer intGripState
Long lngPos
intGripState = Griplink_Grip(3, 0)
If intGripState = DS_HOLDING Then
    lngPos = Griplink_Value(3, 0)
    If lngPos > 80000 Then
        'Set LED pattern 0
        Griplink_LED(0)
    Else
        'Set LED pattern 1
        Griplink_LED(1)
    EndIf
Else
    Print "Failed to grip part"
EndIf
```

3.17 Set grip settings - Griplink_GripCfg

Sets the gripper settings for the selected grip on the selected device. Each gripper consists of a name (character string) and three parameters. The name makes it easier to identify the handle.



Depending on the device, certain value ranges may apply to the parameters. Please refer to the documentation for the device (e.g. gripping module).

Signature

```
Function Griplink_GripCfg(ubDevicePort As UByte, ubGripIndex As UByte,  
label$ As String, ByRef rParams() As Real)
```

Parameters

ubDevicePort	Port index (0 to 31)
ubGripIndex	Selected grip (value range depends on the gripper module)
label\$	Label / name of the handle
rParams	Array of 8 configuration values



For servo grippers from Weiss Robotics, the first three parameters represent the grip configuration. Value 0 represents the no part limit in millimeters (mm), value 1 represents the release limit, also in millimeters (mm). Value 2 corresponds to the gripping force as a percentage of the nominal gripping force.

Return values

-

Example

Set the configuration for grip 2 on the gripper module at port 1:

```
#include "Griplink.inc"  
Long rParams(8)  
rParams(0) = 2 // No Part Limit 2.0 mm  
rParams(1) = 10 // Release limit 10.0 mm  
rParams(2) = 80 // Gripping force 80%  
rParams(3) = 0  
...  
rParams(7) = 0  
Griplink_GripCfg(0, 1, "Preset 1", ByRef rParams())
```

3.18 Ensure device type - Griplink_DevAssert

Ensures that the desired device is connected to the selected port.



The required IO-Link Vendor ID and Device ID parameters can be found in the description of the device.

Signature

```
Function Griplink_DevAssert(ubDevicePort As UByte, intVendorID As Integer, intDeviceID As Integer)
```

Parameters

ubDevicePort	Port index (0 to 31)
intVendorID	IO-Link vendor ID of the desired device
intDeviceID	IO-Link device ID of the desired device

Return values

-

Example

Check whether an IEG55-020 gripping module (Device ID 20) from Weiss Robotics (Vendor ID 815) is connected to port 2:

```
#include "Griplink.inc"  
Griplink_DevAssert(2, 815, 20)
```

4 Error handling

If a problem occurs within the GRIPLINK plug-in, an error is triggered on the robot controller. If this error is not intercepted by the user, the robot program is stopped. In this case, ongoing robot movements are completed before the robot stops.



The programmer is responsible for intercepting these errors and then returning the system to a safe state. The EPSON programming environment provides suitable commands for this, such as *OnErr* (see EPSON robot controller documentation).

In order to distinguish between different types of errors, the GRIPLINK plug-in uses several different error numbers from the value range of the user-defined error numbers of the robot controller. Table1 shows the predefined error numbers and their meaning. To avoid conflicts with existing robot programs, these error numbers can be adapted in the *GriplinkConfig.inc* file.

Error	Error number	Meaning
GRIPLINK_CONNECTION_ERROR	8100	Connection error between GRIPLINK interface converter and gripping module
GRIPLINK_COMMAND_ERROR	8101	Error when executing a command
GRIPLINK_DEVICE_ERROR	8102	Device error on connected gripping module (e.g. overtemperature)
GRIPLINK_PARAM_ERROR	8103	Parameter error when calling a Griplink function (e.g. value outside the permissible range)
GRIPLINK_TIMEOUT_ERROR	8104	Exceeding the permissible execution time when executing a Griplink command

Table1 : Predefined error numbers

Errors that are triggered when executing the GRIPLINK functions listed in chapter3 can be intercepted using the usual SPEL+ error handling. Example:

```
#include Griplink.inc
Integer intGripState
OnErr GoTo errorHandler
intGripState = Griplink_Grip(0, 2)
'The constants DS_NO_PART and DS_HOLDING are defined in Griplink.inc
'defined
If intGripState = DS_NO_PART Then
    Print "No part found"
ElseIf intGripState <> DS_HOLDING Then
    Print "Unexpected grip state"
```

```
EndIf  
errorHandler:  
    MsgBox "Failed to execute grip command"
```



For more information on error handling in SPEL+, please refer to the documentation for your EPSON robot system.

5 Device states

The device state can have one of the following values:

Device status	Constant	Code	Description
NOT CONNECTED	DS_NOT_CONNECTED	0	Port not connected
NOT INITIALIZED	DS_NOT_INITIALIZED	1	Device not initialized
DISABLED	DS_DISABLED	2	Ready for operation, not active
RELEASED	DS_RELEASED	3	Workpiece released
NO PART	DS_NO_PART	4	No workpiece found
HOLDING	DS_HOLDING	5	Workpiece is held
OPERATING	DS_OPERATING	6	Ready for operation, active
FAULT	DS_FAULT	7	Error state

© 2020 WEISS ROBOTICS GmbH & Co. KG. All rights reserved.

GRIPLINK and PERMAGRIP are registered trademarks of WEISS ROBOTICS GmbH & Co KG. All other trademarks are the property of their respective owners.

The specifications given in this document are subject to change without notice for the purpose of product improvement. Trademarks are the property of their respective owners. Our products are not intended for use in life support systems or for systems where misuse could result in personal injury.

