



GRIPLINK CONTROLLER UNIFIED COMMAND SET REFERENCE MANUAL

Protocol Version 1
August 2024



Content

- 1 Introduction..... 3**
- 1.1 **Connecting to the GRIPLINK Controller 3**
- 1.2 **Communicating with the GRIPLINK Controller..... 3**
- 1.3 **Port count..... 4**
- 1.4 **Error handling..... 4**
- 1.5 **Connecting to the command interface using PuTTY 5**
- 1.5.1 Connecting from a computer or laptop 5
- 1.5.2 Connecting from a robot controller 8
- 1.6 **Routing of commands 8**

- 2 Command Format Description..... 10**
- 2.1 **System Information 11**
- 2.1.1 Identify controller type – ID 11
- 2.1.2 Identify protocol version – PROTOCOL 12
- 2.1.3 Assert protocol version – PROTASSERT..... 13
- 2.1.4 Read serial number of the GRIPLINK Controller – SN 14
- 2.1.5 GRIPLINK Controller label – LABEL 15
- 2.1.6 Read firmware version of the GRIPLINK Controller – VER 16
- 2.1.7 Verbose mode – VERBOSE 17
- 2.2 **Device Information 18**
- 2.2.1 Read device Vendor-ID – DEVVID..... 18
- 2.2.2 Read device Product ID – DEVPID 19
- 2.2.3 Assert type of connected device – DEVASSERT 20
- 2.2.4 Retrieve device name – DEVNAME 21
- 2.2.5 Retrieve device vendor name – DEVVENDOR..... 22
- 2.2.6 Read serial number of a device – DEVSN 23
- 2.2.7 Read application tag of a device – DEVTAG 24
- 2.2.8 Read firmware version of a device – DEVVER..... 25
- 2.3 **Controller Commands 26**
- 2.3.1 Close connection – BYE 26
- 2.4 **Device Commands..... 27**
- 2.4.1 Enable device – ENABLE 27
- 2.4.2 Disable device – DISABLE 28
- 2.4.3 Perform a homing sequence – HOME..... 29
- 2.4.4 Grip with the selected grip preset – GRIP 30
- 2.4.5 Release with the selected grip preset – RELEASE 31
- 2.4.6 Flexible gripping using motion parameters – FLEXGRIP 32
- 2.4.7 Flexible releasing using motion parameters – FLEXRELEASE 33
- 2.4.8 Set LED visualization – LED..... 34

2.4.9	Control the force retention feature – CLAMP	35
2.4.10	Wait for State Transition and Return – WSTR.....	36
2.4.11	Set value – SETVAL	37
2.4.12	Wait for indexed value to reach/cross threshold – WAITVAL.....	38
2.5	Multi-Device Commands	42
2.5.1	Grip with selected devices – MGRIP	42
2.5.2	Release with selected devices – MRELEASE.....	43
2.5.3	Wait for state transition on multiple devices – MWAITFOR.....	44
2.6	Device status and diagnosis	45
2.6.1	Get device state – DEVSTATE	45
2.6.2	Get value – VALUE.....	46
2.7	Device Configuration.....	47
2.7.1	Grip configuration – GRIPCFG	47
3	Status and error codes	49
4	Device status codes.....	51
5	Supported commands by device class.....	52
6	List of Product and Vendor IDs	54

1 Introduction

The GRIPLINK controller is intended to connect IO-Link based sensor and actuator devices to a wide range of industrial robot controllers using a standard TCP/IP networking interface. The GRIPLINK Controller converts a generic command set into the sensor's or actuator's IO-Link commands using its unique device driver architecture. A list of all supported devices can be found at

www.griplink.de/devices

This manual describes the text-based Unified Command Protocol that is used to control the connected modules via a TCP/IP socket connection. The following chapters provide a detailed explanation of the protocol itself as well as of the GRIPLINK controller's Unified Command Set. To get started with the communication protocol, we recommend using a common Telnet client like the free PuTTY¹ for Windows as described in chapter 1.5.

Weiss Robotics already provides extensively tested GRIPLINK Plug-Ins for various robot platforms. A list of the supported robot platforms can be found at

www.griplink.de/plugins



If you intend to use GRIPLINK for a robotic system that is already supported by a GRIPLINK-Plugin, please refer to the respective GRIPLINK-Plugin Manual.

1.1 Connecting to the GRIPLINK Controller

Before connecting to the module, an appropriate IP address must be set using the GRIPLINK Controller's web interface. Connect the GRIPLINK Controller to the local network or directly to your computer's network interface and point your favorite web browser to the GRIPLINK Controller's IP address e.g., by typing the default <http://192.168.1.40> into the address bar and pressing <Enter>. Please make sure that your computer's network settings are appropriate.


The GRIPLINK Controller's IP address can be configured by choosing "Config" from the buttons on the top of the main page.

1.2 Communicating with the GRIPLINK Controller

The GRIPLINK Controller communicates with its client using a text-based protocol. The following chapters describe the general format of these commands.

¹ <http://www.putty.org/>

The module expects commands being submitted as plain ASCII strings. **Each command must be terminated by a line feed character ('\n' or ASCII code 0x0a).** Return messages are submitted in the same format and are terminated by a line feed character, too.

 Please note that the commands of the GRIPLINK Controller are not case sensitive, i.e. sending "grip(0,1)" is the same as "GRIP(0,1)" or "gRiP(0,1)". However, it is good practice to send commands in upper case notation. Response messages from the GRIPLINK Controller will always be sent in upper case notation.

1.3 Port count

As the number of ports differ by the different versions of the GRIPLINK compatible hardware used, this manual uses a generic placeholder PORTS for the number of physical device ports, see the following table for common hardware platforms.

Hardware	PORTS
GRIPLINK-ET4	4
WPG Series	1

Table 1: Number of ports depending on device

See your GRIPLINK controller's manual for any information about the number of ports, if not included in this table.

1.4 Error handling


In case of an error, the module returns a message string of the following format:

```
ERR <error_code>
```

where <error_code> represents a number referencing an error code. Chapter 3 gives an overview of all available error codes.

If verbose mode is active (cf. chapter 2.1.7), the module submits extended error messages containing an additional text string that describes the type of error:

```
ERR <error_code> <description_string>
```

 See chapter 3 for a description of the returned error code.
See chapter 2.1.7 on how to enable verbose mode.

1.5 Connecting to the command interface using PuTTY

1.5.1 Connecting from a computer or laptop

PuTTY is a free Telnet and SSH client that can be used to connect to the GRIPLINK Controller's command interface e.g., for testing purposes or to learn the communication protocol. The following chapter shows how to use PuTTY with the GRIPLINK Controller on Windows. On Unix-like systems (Linux, Mac), you can use command line tools like netcat (nc) instead.

 The default IP address depends on the selected device, see Table 2. The default TCP/IP listening port is always 10001.

Hardware	Default IP Address
GRIPLINK-ET4	192.168.1.40
WPG Series	192.168.1.50

Table 2: Default IP Addresses depending on device

Download and install PuTTY for Windows from <http://www.putty.org>.

After starting PuTTY, a new connection must be configured. Type in the IP address and port number of the module and set the connection type to "raw" (see Figure 1).

As the module does not send a carriage return character ('\r' or ASCII code 0x0d) in its response messages, PuTTY must be configured to explicitly do a carriage return on each line feed character. In the settings window, select the "Terminal" tab and enable "Implicit CR in every LF" (see Figure 2).

Now click the "Open" button to open the connection. A new and empty terminal window will appear (Figure 3), ready to type in your commands.

Typing "HOME(0)" for example, followed by <Enter>, will execute a homing sequence on the gripper device connected to port 0 (see chapter 2.4.1).

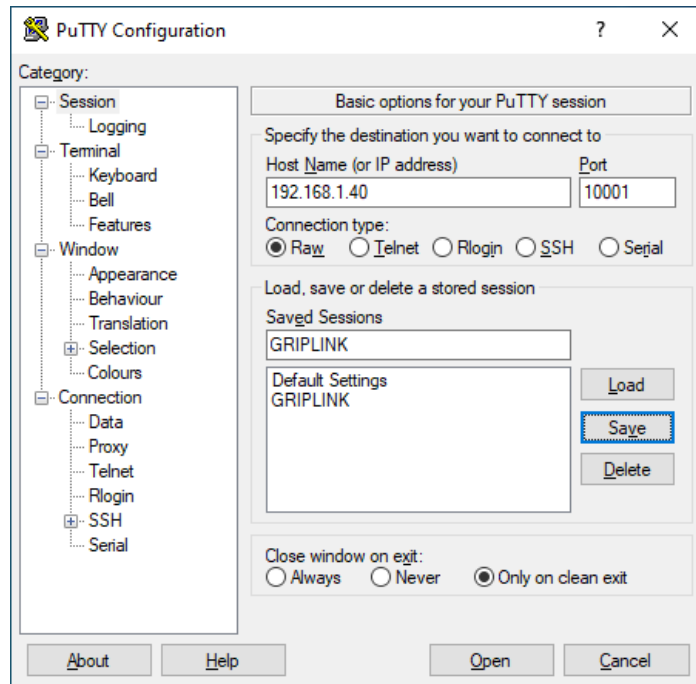


Figure 1: PuTTY session settings

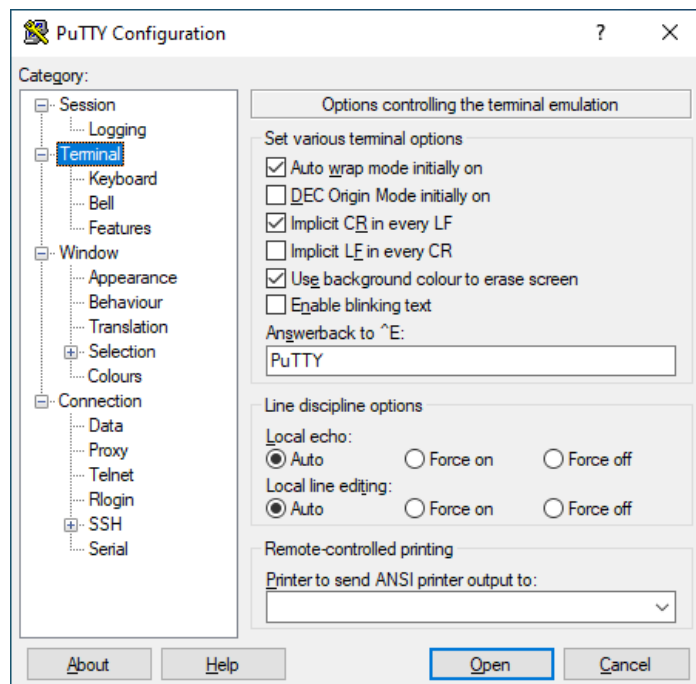
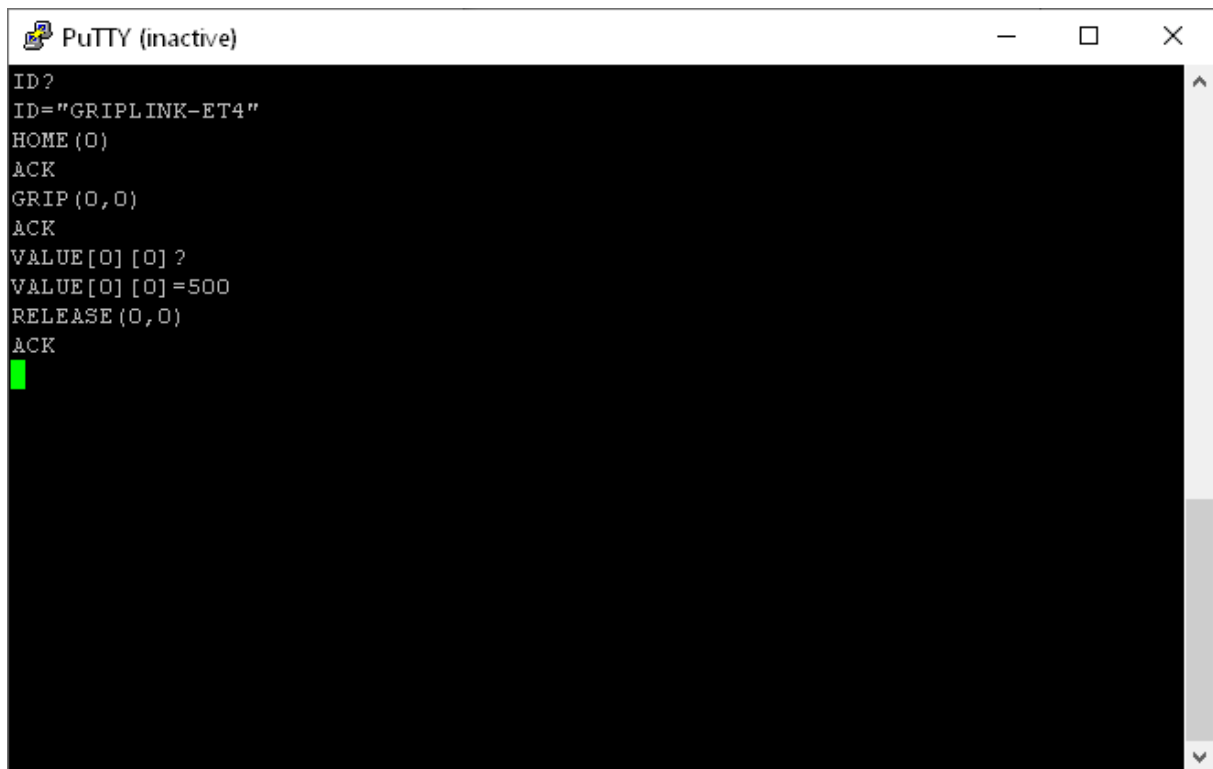


Figure 2: PuTTY terminal settings



```
PuTTY (inactive)
ID?
ID="GRIPLINK-ET4"
HOME (0)
ACK
GRIP (0,0)
ACK
VALUE [0] [0] ?
VALUE [0] [0] =500
RELEASE (0,0)
ACK
█
```

Figure 3: PuTTY terminal window

1.5.2 Connecting from a robot controller

The text-based message protocol described in this manual is intended to control devices from a robot controller. Nowadays, almost every robot controller is able to open up network connections (usually called socket connections) to exchange data with remote hosts like a computer or the GRIPLINK Controller. Please refer to the documentation of your robot controller or ask the robot manufacturer to find out how to use network connections from your robot program and how to send and receive text-based messages.

WEISS ROBOTICS also provides ready-to-use implementations for some robot systems. Please ask our sales and support teams if there is a turnkey solution for your setup.

1.6 Routing of commands

GRIPLINK Controller supports routing of incoming commands, so that multiple GRIPLINK Controllers can form a “virtual controller network” that behaves like a single large GRIPLINK controller towards the host side (i.e. the robot). The routing depends on the configuration of the GRIPLINK controller network, see the GRIPLINK Controller manual for details.

The GRIPLINK Controller can be configured to play one of three roles:

- Master
- Slave
- Stand-alone

Master mode

In Master mode, up to seven slaves can be assigned to one GRIPLINK Controller. Every slave controller has exactly 4 logical ports that will be mapped to its physical ports. If the slave controller has less than four ports available, the higher ports are left unused and show up as “INVALID” when querying their state.

Example 1

Assuming a virtual GRIPLINK Controller network consisting of one GRIPLINK-ET4 master and seven GRIPLINK-ET4 slave controllers, the ports are assignment as follows:

Master	Ports 0 to 3
Slave #1	Ports 4 to 7
Slave #2	Ports 8 to 11
Slave #3	Ports 12 to 15
Slave #4	Ports 16 to 19
Slave #5	Ports 20 to 23
Slave #6	Ports 24 to 27
Slave #7	Ports 28 to 31

The lowest numbered ports are mapped to the Controller’s own hardware, while ports with index > 3 will be routed to the pre-configured slave GRIPLINK Controllers over Ethernet.

Example 2

A network of a GRIPLINK-ET4 master and two WPG Series grippers using the GRIPLINK protocol will use the following port assignment:

Master	Ports 0 to 3
Slave #1 (WPG Series Gripper)	Port 4, Ports 5 to 7 are unused
Slave #2 (WPG Series Gripper)	Port 8, Ports 9 to 11 are unused
	Ports 12 to 31 are unused

Slave mode

In Slave mode, the GRIPLINK Controller will only accept incoming connections from its pre-configured Master GRIPLINK Controller. All other connection attempts will be rejected.

Stand-Alone mode

In Stand-Alone mode, routing is disabled and only the local ports can be accessed.

2 Command Format Description

Commands that are sent to the GRIPLINK Controller have to be terminated with a Line Feed character (LF, 0x0A). Other non-printable characters with ASCII codes 0x01 to 0x1F will be replaced by whitespace characters. Therefore, additional Carriage Return characters (CR, 0x0D) that are automatically sent by some host platforms will be ignored.

Response messages from the GRIPLINK Controller will always be terminated with a Line Feed character (0x0A).

Example

Read the identification string of the connected GRIPLINK Controller.

Command: "ID?"[CR][LF] *[CR] will be ignored!*

Response: "GRIPLINK-ET4"[LF]

Notation and Data Types

The following data types are used in the command description:

- <string> Character string. Always within double quotes ("")
- <int> Signed integer value
- <uint> Unsigned integer value
- <bool> Boolean value, represented as integer, i.e., 0 or 1.

2.1 System Information

The following commands can be used to query information about the GRIPLINK Controller.

2.1.1 Identify controller type – ID

Read device type of the GRIPLINK Controller. This command returns the identification character string of the GRIPLINK Controller and can be used to distinguish different types.

Syntax

ID?

Parameters

-

Response Message

ID=<idstring>

<idstring> String ID of this GRIPLINK Controller.

Example

Ask the connected GRIPLINK Controller for its identification string.

Command: ID?

Response: ID="GRIPLINK-ET4"

2.1.2 Identify protocol version – PROTOCOL

Read the version of the GRIPLINK protocol used by this GRIPLINK Controller to determine the capabilities of the interface.

Syntax

PROTOCOL?

Parameters

-

Response Message

PROTOCOL="GRIPLINK-V<pversion>"

<pversion> Protocol version of this GRIPLINK Controller including the protocol identifier.

Example

Ask the connected GRIPLINK Controller for its GRIPLINK protocol version.

Command: PROTOCOL?

Response: PROTOCOL="GRIPLINK-V1"

2.1.3 Assert protocol version – PROTASSERT

Assert a specific protocol with a given minimum version is running on the GRIPLINK controller.

Syntax

PROTASSERT(<protidentifier>,<minversion>)

Parameters

<protidentifier> Protocol identifier string with enclosing quotation marks (“
<minversion> Minimum version of the protocol running on the controller.

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Assert a GRIPLINK protocol with minimum version 1.

Command: PROTASSERT(“GRIPLINK”,1)

Response on success: ACK

Response if asserted protocol is not supported: ERR 20

2.1.4 Read serial number of the GRIPLINK Controller – SN

Read the serial number of GRIPLINK Controller.

Syntax

SN?

Parameters

-

Response Message

SN=<snstring>

<snstring> Serial number string

Example

Ask for the serial number of the connected GRIPLINK Controller.

Command: SN?

Response: SN="123456"

2.1.5 GRIPLINK Controller label – LABEL

Set or read the device label of GRIPLINK Controller. The device label can be used to identify a certain GRIPLINK Controller in a larger network of devices.

Syntax

Assign: LABEL=<label>

Query: LABEL?

Parameters

<label> String label (32 characters max) to be set.

Response Message

LABEL=<label>

Example

1) Ask for the label of the connected GRIPLINK Controller.

Command: LABEL?

Response: LABEL="Hello World"

2) Set the label of the connected GRIPLINK Controller to "Marvin"

Command: LABEL="Marvin"

Response: LABEL="Marvin"

2.1.6 Read firmware version of the GRIPLINK Controller – VER

Read the firmware version of GRIPLINK Controller. The version is returned as string containing a 3-element version number.

Syntax

VER?

Parameters

-

Response Message

VER=<version>

<version> Version number string

Example

Query the device version:

Command: VER?

Response: VER="4.0.0"

2.1.7 Verbose mode – VERBOSE

Decodes any returned error codes into a human readable string. Example:
The response "ERR 04" turns into "ERR 04 The device is not initialized".

Syntax

VERBOSE=<enable>

Parameters

<enable> 0: disable verbose mode
1: enable verbose mode

Response Message

VERBOSE=<enable>

Example

Enable Verbose Mode:

Command: VERBOSE=1

Response: VERBOSE=1

2.2 Device Information

The following commands can be used to gather information about the connected field devices.

2.2.1 Read device Vendor-ID – DEVVID

Read the Vendor ID of the connected device. This command returns the Vendor ID as assigned by the IO-Link Association of the device connected to the given port.

 See Chapter 6 for a list with Vendor- and Product IDs.

Syntax

DEVVID[<port>]?

Parameters

<port> Port number
 Proxy role STANDALONE: 0 to (PORTS-1)
 Proxy role MASTER: 0 to 31

Response Message

DEVVID[<uint>]=<vid>

<vid> Vendor ID as assigned by the IO-Link Association (range 1 to 65535). (-1) is returned, if no device was found at the specified port.

Example

Read the Vendor ID of the device connected to GRIPLINK Controller port 2 (Device is IEG 55 gripper with Vendor ID 815):

Command: DEVVID[2]?

Response: DEVVID[2]=815

2.2.2 Read device Product ID – DEVPID

Read the Product ID of the connected device. This command returns the Product ID as assigned by the vendor of the device connected to the given port (i.e. the IO-Link Device ID or Product ID for non-IO-Link devices).

 See Chapter 6 for a list with Vendor- and Product IDs.

Syntax

DEVPID[<port>]?

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

DEVPID[<uint>]=<pid>

<pid> Product ID as assigned by the device vendor (range 1 to 33.554.431).

Error codes

In case of an error, one of the following codes will be returned. Error codes different to this error will indicate protocol and/or parameter errors.

ERR 04	No device connected or device not supported (E_NOT_INITIALIZED)
ERR 11	Port index outside the allowed range (E_INDEX_OUT_OF_BOUNDS)

Example

Read the Product ID of the device connected to GRIPLINK Controller port 2 (Device is IEG 55 gripper with Product ID 20):

Command: DEVPID[2]?

Response: DEVPID[2]=20

2.2.3 Assert type of connected device – DEVASSERT

This function expects a device identification based on Vendor and Product ID and checks, if this device is connected to the specified port. The function returns an error, if either a different device is connected or no device is connected.

 See Chapter 6 for a list with Vendor- and Product IDs.

Syntax

DEVASSERT(<port>,<vid>,<pid>)

Parameters

<port>	Port number
	Proxy role STANDALONE: 0 to (PORTS-1)
	Proxy role MASTER: 0 to 31
<vid>	Vendor ID of the expected device. The Vendor ID is assigned to the manufacturer by the IO-Link Association.
<pid>	Product ID of the expected device. This equals the Device ID the vendor has assigned to its device.

Response Message

ACK if the specified device was found or ERR 25 (E_CMD_FAILED) if not. Error codes different to this error will indicate protocol and/or parameter errors.

Example

The robot program expects a WEISS ROBOTICS IEG 55 servo gripper (Vendor ID = 815, Product ID = 20) to be connected at port 0:

Command: DEVASSERT(0,815,20)

Response: ACK

2.2.4 Retrieve device name – DEVNAME

Retrieve the IO-Link "ProductName" identification character string as reported from the device connected to the given port.

Syntax

DEVNAME[<port>]?

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

DEVNAME[<uint>]=<name>

<name> Device name as string

Example

Read the name of the device connected to GRIPLINK Controller port 1:

Command: DEVNAME [1]?

Response: DEVNAME [1]="IEG 55-020"

2.2.5 Retrieve device vendor name – DEVVENDOR

Retrieve the IO-Link "VendorName" identification character string as reported from the device connected to the given port.

Syntax

DEVVENDOR[<port>]?

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

DEVVENDOR[<uint>]=<vendor>

<vendor> Vendor name as string

Example

Read the name of the device connected to GRIPLINK Controller port 1:

Command: DEVVENDOR [1]?

Response: DEVVENDOR [1]="WEISS ROBOTICS"

2.2.6 Read serial number of a device – DEVSN

Read the serial number (IO-Link "SerialNumber" value) of the device connected to the given port. The serial number is a device-specific string that can contain alpha-numeric characters. See the device specifications for details.

Syntax

DEVSN[<port>]?

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

DEVSN[<port>]=<snstring>

<snstring> Serial number in string format

Example

Read the name of the device connected to GRIPLINK Controller port 1:

Command: DEVSN[1]?

Response: DEVSN[1]="000042"

2.2.7 Read application tag of a device – DEVTAG

Get or set the application tag of a device. This tag can be used e.g., to identify the device in a larger assembly. The tag is transferred to the device as "Application Tag" and is stored using the IO-Link data storage mechanism. If the connected device doesn't support application tags, the function will return an error.

Syntax

Assign: DEVTAG[<port>]=<label>
Query: DEVTAG[<port>]?

Parameters

<port> Port number
Proxy role STANDALONE: 0 to (PORTS-1)
Proxy role MASTER: 0 to 31

<label> String label (32 characters max) to be set.

Response Message

DEVTAG[<port>]=<label>

Example

1) Retrieve the label of the device connected to port 1.

Command: DEVTAG[1]?
Response: DEVTAG[1]="Hello World"

2) Change the label of the device connected to port 1 to "Device 1"

Command: DEVTAG[1]="Device 1"
Response: DEVTAG[1]="Device 1"

2.2.8 Read firmware version of a device – DEVVER

Return the firmware version of a selected device. The content of the returned version string is device-specific (IO-Link parameter "SWVersion"). See the device manual for details.

Syntax

DEVVER [<port>]?

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

DEVVER[<port>]=<version>

<version> Version string

Example

Retrieve the version information of the device connected to port 3.

Command: DEVVER[3]?

Response: DEVVER[3]="1.0.1-RC2"

2.3 Controller Commands

2.3.1 Close connection – BYE

Close the connection from the client side. BYE should be sent at the end of a session to gracefully close the connection. The server will respond with an ACK and will close the connection afterwards.

Syntax

BYE()

Parameters

-

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Close current session:

Command: BYE()

Response: ACK

[Server closes connection]


[Client can close socket]

2.4 Device Commands

All device commands return ACK if the command has been received correctly. They return ERR <ERRORCODE> if an error occurred, where <ERRORCODE> indicates the specific error. See Chapter 3 for a list of error codes.

2.4.1 Enable device – ENABLE

Enables the selected device. The action performed by this command depends on the connected device, see the device documentation for details.

 Grippers by WEISS ROBOTICS: The device will be (re-)enabled automatically when executing a GRIP(), RELEASE() or HOME() command.

Syntax

ENABLE(<port>)

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Enable the device connected to port 2:

Command: ENABLE(2)

Response: ACK

2.4.2 Disable device – DISABLE

Disables the selected device. The result depends on the connected device, see the device documentation for details.

Grippers by WEISS ROBOTICS: This function will disable the actuator thus allowing to move the fingers by hand.

Syntax

DISABLE(<port>)

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Disable the device connected to port 2:

Command: DISABLE(2)

Response: ACK

2.4.3 Perform a homing sequence – HOME

Perform a homing sequence for the given device. On grippers, this command will move the fingers to either inner or outer mechanical limit to reference its position, depending on the gripper's configuration. See the device manual for further details.

The function will block until the homing sequence is completed and returns the result of that operation.

Syntax

HOME(<port>)

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Home the gripper connected to port 0:

Command: HOME(0)

Response: ACK

2.4.4 Grip with the selected grip preset – GRIP

Perform a grip with the given grip recipe/preset. The command returns immediately after starting the gripping process. To determine the result, periodically read the device state (device state polling) that will change to

DS_HOLDING	If a workpiece was found and the gripping force is applied
DS_NO_PART	If the fingers reached the "No Part Position" without gripping
DS_FAULT	In case of a fault during motion

Syntax

GRIP(<port>,<index>)

Parameters

<port>	Port number
	Proxy role STANDALONE: 0 to (PORTS-1)
	Proxy role MASTER: 0 to 31
<index>	Grip index. The number of available recipes/presets depends on the connected device.

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Grip a workpiece with the gripper on port 2 that was freshly homed using recipe 3:

Command: GRIP(2,3)

Response: ACK

Poll the device state to determine the end of the gripping process. Be aware that the start state can be different to DS_IDLE and depends on the command issued before:

```
DEVSTATE[2]?
->DEVSTATE[2]=2 (= DS_IDLE)
DEVSTATE[2]?
->DEVSTATE[2]=2 (= DS_IDLE)
...
DEVSTATE[2]?
->DEVSTATE[2]=5 (= DS_HOLDING)
```

2.4.5 Release with the selected grip preset – RELEASE

Perform release with the given grip index. The command returns immediately after starting to move the fingers. To determine the end of the release process, periodically read the device state (device state polling) that will change to

DS_RELEASED If the fingers arrived at the "Release Position"

DS_FAULT In case of a fault during motion

Syntax

RELEASE(<port>,<index>)

Parameters

<port> Port number

Proxy role STANDALONE: 0 to (PORTS-1)

Proxy role MASTER: 0 to 31

<index> Grip index. The number of available recipes/presets depends on the connected device.

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Release the workpiece previously gripped the gripper on port 2 using recipe 3:

Command: RELEASE(2,3)

Response: ACK

Poll the device state to determine the end of the release process:

DEVSTATE[2]?

->DEVSTATE[2]=5 (= DS_HOLDING)

DEVSTATE[2]?

->DEVSTATE[2]=5 (= DS_HOLDING)


...

DEVSTATE[2]?

->DEVSTATE[2]=3 (= DS_RELEASED)

2.4.6 Flexible gripping using motion parameters – FLEXGRIP

Perform a grip operation using parameters for target position, gripping force in Newton, gripping speed, and acceleration.

 This command is not available for all devices.

Syntax

FLEXGRIP(<port>,<position>,<force>,<speed>,<acceleration>)

Parameters

<port>	Port number Proxy role STANDALONE: 0 to (PORTS-1) Proxy role MASTER: 0 to 31
<position>	Target position representing the No Part Limit. Value given in 1/1000 mm.
<force>	Gripping force. Value given in 1/1000 N.
<speed>	Gripping speed. Set to 0 for using the auto-computed speed value based on the selected gripping force. Value given in 1/1000 mm/s.
<acceleration>	Gripping acceleration. Set to 0 for using the default acceleration value. Value given in 1/1000 mm/s ² .

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Grip a workpiece with 60 N which is larger than 15 mm with the gripper on port 2. Use a speed of 45 mm/s, standard acceleration:

Command: FLEXGRIP(2,15000,60000,45000,0)

Response: ACK

2.4.7 Flexible releasing using motion parameters – FLEXRELEASE

Perform a release operation using parameters for target position, gripping speed, and acceleration.

This command also can be used to pre-position grippers that support FLEX actions.

 This command is not available for all devices.

Syntax

FLEXRELEASE(<port>,<position>,<speed>,<acceleration>)

Parameters

<port>	Port number Proxy role STANDALONE: 0 to (PORTS-1) Proxy role MASTER: 0 to 31
<position>	Target position representing the Release Limit. Value given in 1/1000 mm.
<speed>	Release speed. Set to 0 for using the default release speed value. Value given in 1/1000 mm/s.
<acceleration>	Release acceleration. Set to 0 for using the default acceleration value. Value given in 1/1000 mm/s ² .

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.


Example

Preposition the gripper at port 2 to 20 mm with default motion parameters, then grip at 5 mm with 100 N using a speed of 20 mm/s. After handling the workpiece, release to 20 mm again:

```
Commands: FLEXRELEASE(2,20000,0,0)
          FLEXGRIP(2,5000,100000,20000,0)
          ...
          FLEXRELEASE(2,20000,0,0)
```

2.4.8 Set LED visualization – LED

Some devices will provide the ability to visualize a certain state. This command allows to control this visualization.

 This command is not available for all devices.

Syntax

LED(<port>,<index>)

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31
<index>	Animation index	

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example


A WEISS ROBOTICS CRG 200-085 servo gripper is installed on port 1. Enable visualization 2 on this gripper:

Command: LED(1,2)

Response: ACK

2.4.9 Control the force retention feature – CLAMP

Enables or disables the mechanical force retention feature of the connected device.

 This command is not available for all devices.

WEISS ROBOTICS grippers: The force retention feature can only be controlled manually if the device is disabled. While the device is enabled, force retention will be controlled automatically by the device.

Syntax

CLAMP(<port>, <enable>)

Parameters

<port>	Port number
	Proxy role STANDALONE: 0 to (PORTS-1)
	Proxy role MASTER: 0 to 31
<enable>	Enable (1) or disable (0) force retention

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

A WEISS ROBOTICS CRG 200-085 gripper is installed on port 1. Enable the force retention feature:

Command: CLAMP(1,1)

Response: ACK

2.4.10 Wait for State Transition and Return – WSTR

This query can be used to detect the end of the previous state-manipulating command like GRIP, RELEASE, HOME, etc. The query waits, until a device state transition occurs and returns the new device state. If no state transition occurs, the query returns with a timeout error. When issuing the same command twice, WSTR immediately returns with the current device state. WSTR only cares about manipulating commands. Non-manipulating commands like state queries or LED control are ignored.

Syntax

WSTR[<port>]?

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

WSTR[<port>]=<state>

<state> Device state, see table in chapter 4

ERR <ERRORCODE> in case of an error

Example

Wait for completion of the previous GRIP command with grip index 1 on port 0:

Commands: GRIP(0,1)
WSTR[0]?

[now GRIPLINK blocks until state transition, i.e. GRIP command was finished]

Response: WSTR[0]=5 (= DS_HOLDING)

2.4.11 Set value – SETVAL

This function can be used to set a device value. The behavior strongly depends on the connected device type and the value index.

The values selectable by the value index correspond to the values available in the VALUE query (cf. chapter 2.6.2), i.e. the VALUE query can be used to read the corresponding values before setting them with SETVAL. The SETVAL function can induce changes in the behavior of a device. On a WPG Series gripping module, for example, the function can be used to pre-position the gripper jaws.

Syntax

SETVAL(<port>, <index>, <value>)

Parameters

<port>	Port number
	Proxy role STANDALONE: 0 to (PORTS-1)
	Proxy role MASTER: 0 to 31
<index>	Value index (0..n-1)
	The number of provided values depends on the connected device.
<value>	Value to be set

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example

Pre-position the gripper jaws of a WPG 300-120 gripping module to an opening width of 100 mm:

Command: SETVAL(0, 0, 100000)


Response: ACK

[Now the WPG gripper will move to the given position]

You can use the WAITVAL query to block until the given value has been set, i. e. the desired position has been reached.

2.4.12 Wait for indexed value to reach/cross threshold – WAITVAL

This function can be used to wait until the desired value has reached or crosses a threshold window. The command returns a timeout error when the execution time exceeds the specified timeout.

 This command is not available for all devices.

Syntax

WAITVAL(<port>,<index>,<threshold>,<>window size>,<timeout>)

Parameters

<port>	Port number Proxy role STANDALONE: 0 to (PORTS-1) Proxy role MASTER: 0 to 31
<index>	Value index (0..n-1). The number of provided values depends on the connected device.
<threshold>	Target value threshold.
<window size>	Size of the window around the threshold value to check for.
<timeout>	Timeout in milliseconds. Must be set to a positive integer or 0.

Response Message

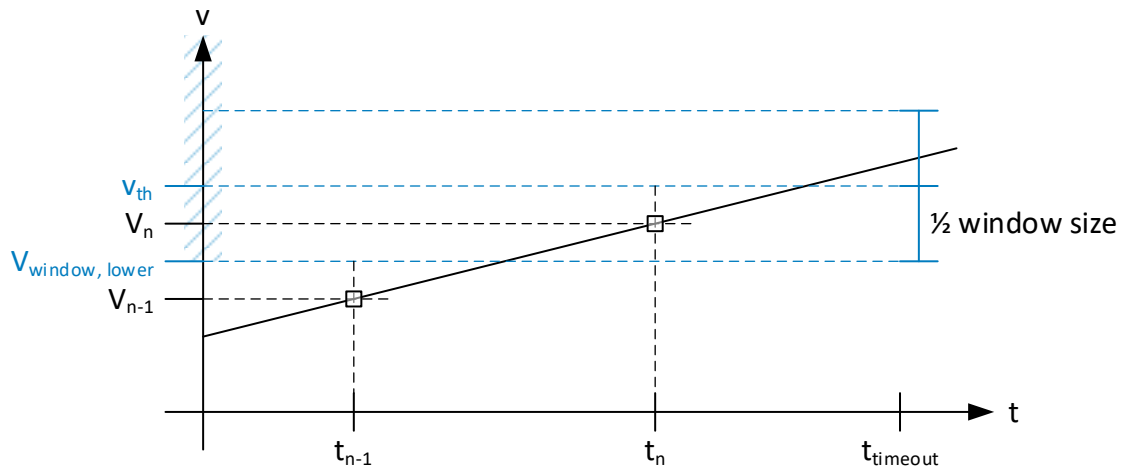
ACK on success or ERR <ERRORCODE> in case of an error.

Explanation

After calling WAITVAL, the selected device value will be polled periodically (corresponds to the values v_0, \dots, v_{n-1}, v_n).

Case 1:

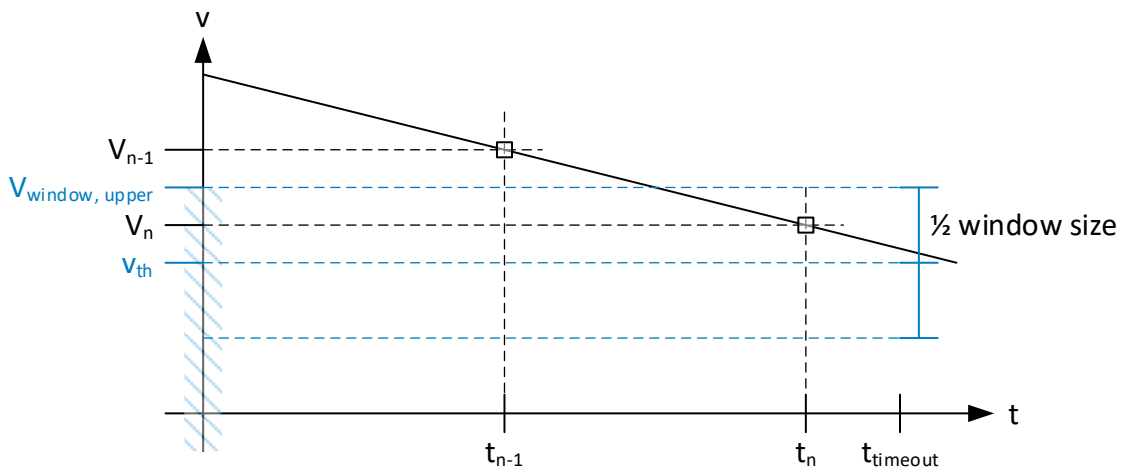
The polled values approach the window defined by threshold value v_{th} and window size at the lower boundary.



If value v_n is larger than the threshold value v_{th} minus half the window size and the timeout duration has not been reached, yet, WAITVAL will return ACK.

Case 2:

The polled values approach the window defined by threshold value v_{th} and window size at the upper boundary.



If value v_n is less than the threshold value v_{th} plus half the window size and the timeout duration has not been reached, yet, WAITVAL will return ACK.

Case 3:

The polled values do not enter the window within the given timeout duration. Then, WAITVAL will return ERR 05 (Timeout).

Example

Set value with index 0 of the device connected to port 0 to 300.

Then wait for value 0 to either be greater or less than 300 ± 50 . Wait at most 10 seconds (10,000 milliseconds).

Commands: SETVAL(0,0,300000)
 WAITVAL(0,0,300000,50000,10000)



For non-steady device values, the sampling time has to be considered in order to detect the value change reliably.

This has to be tested in the real application!

Application example 1

Task

A WPG 300-120 is used for a pick and place application with rings that have to be gripped from the outside and from the inside.

To switch from outside gripping to inside gripping, the gripper should be pre-positioned first.

Solution

1) When the gripper released the ring from the outer diameter d_{outer} and the robot moved to a safe position, the gripper is pre-positioned to a position slightly smaller than the inner diameter d_{inner} .

SETVAL is used here to perform this operation.

The set-value v_{set} is calculated as $v_{set} = (d_{inner} - d_{tolerance}) \cdot 1000$

2) Now, the robot shall wait until the target position is reached, to not collide with the workpiece
WAITVAL is used here to perform this operation.

The threshold value v_{th} is chosen to be the same as v_{th} . An appropriate window size of $\frac{1}{2} \cdot d_{tolerance} \cdot 1000$ should be used.

3) WAITVAL will block, until the position is reached or a timeout occurs

4) When WAITVAL returned ACK, the robot can safely move to the pick-position.

Application example 2

Task:

A conveyor belt transports boxes with variable heights ($50 \text{ cm} \leq h \leq 90 \text{ cm}$).

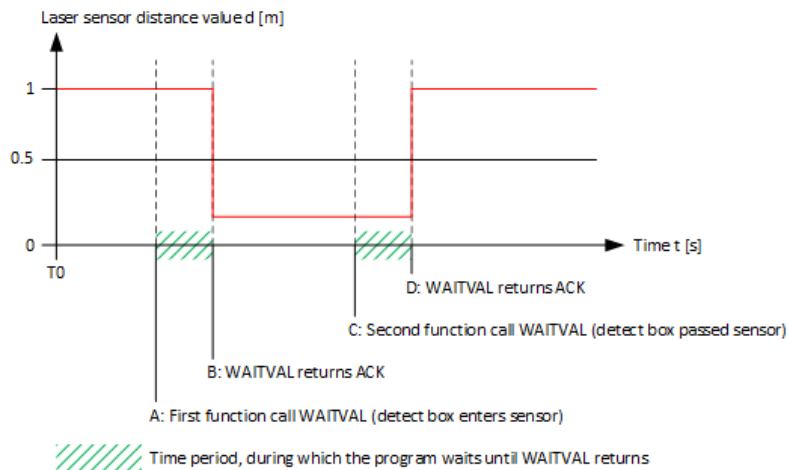
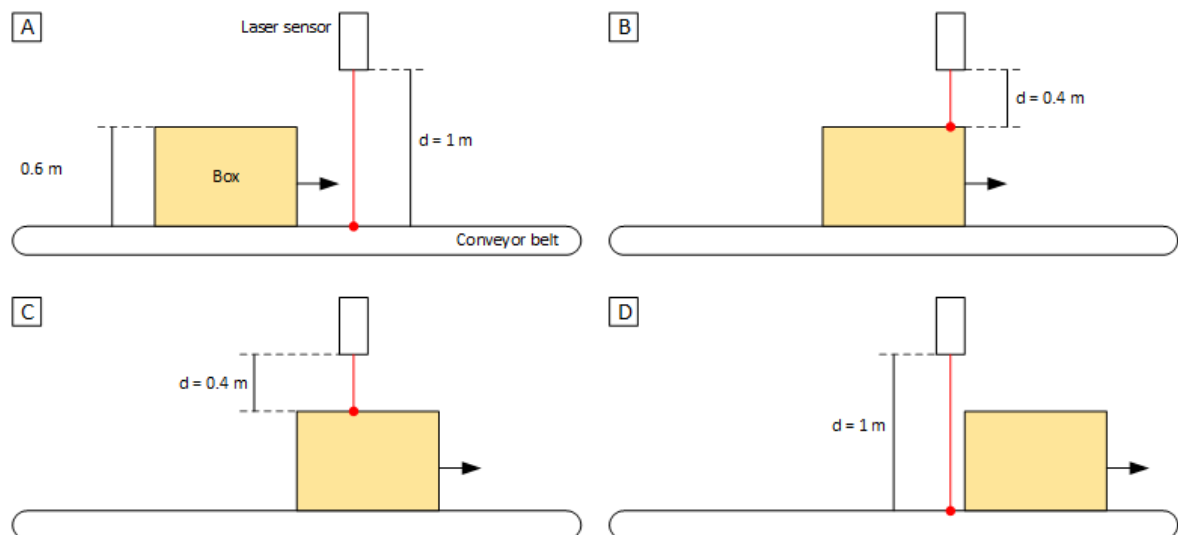
- 1) The robot program needs to detect, when a box enters and leaves a certain point at the belt.
- 2) The height of the box needs to be determined, too.

A laser distance sensor is mounted perpendicularly 1 m over the belt at the detection position and measures the distance to the belt surface. It is connected to the GRIPLINK Controller.

Solution:

With the first call of WAITVAL (image A), the program waits until the sensor value crosses the value of 0.5 m (box enters sensor spot, image B). Then, the application can process the box (time between B and C). At this point, the actual box size can be determined using the VALUE command.

With the second call of WAITVAL (image C), the program waits until the sensor value crosses the value 0.5 m again, now in opposite direction (box has left the sensor spot, image D).



2.5 Multi-Device Commands

Multi-device commands allow to run the same command for a selected list of devices.

2.5.1 Grip with selected devices – MGRIP

Perform a grip with the selected grippers and with the indexed grip. The function returns immediately after starting the gripping process. Use device state polling to determine the end of the gripping process.

Syntax

MGRIP(<index>,<sel0>, ...,<seln>)

Parameters

<index> Grip index.

<seln> Select (1) or deselect (0) device on port n.

Port number n will be 0 to (PORTS-1) in Stand-Alone mode and 0 to 31 in Master mode. Trailing 0 parameters can be omitted.

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example 1

Grip with grippers on port 0 and 1 using grip index 3:

Command: MGRIP(3,1,1,0,0) or MGRIP(3,1,1)

Response: ACK

Example 2

Grip with grippers on port 0, 1 and 5, 6 on a virtual GRIPLINK controller network of two GRIPLINK-ET4 Controllers using grip index 3:

Command: MGRIP(3,1,1,0,0,0,1,1)

Response: ACK

2.5.2 Release with selected devices – MRELEASE

Perform release with the given grip preset on the selected devices. The function returns immediately after starting the gripping process. Use device state polling to determine the end of the gripping process.

Syntax

MRELEASE(<index>,<sel0>,...,<seln>)

Parameters

<index> Grip index.

<seln> Select (1) or deselect (0) device on port n.

Port number n will be 0 to (PORTS-1) in Stand-Alone mode and 0 to 31 in Master mode. Trailing 0 parameters can be omitted.

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example 1

Release a previously gripped workpiece with grippers on port 0 and 1 using grip index 3:

Command: MRELEASE(3,1,1,0,0) or MRELEASE(3,1,1)

Response: ACK

Example 2

Release a previously gripped workpiece with grippers on port 0, 1 and 5, 6 on a virtual GRIPLINK controller network of two GRIPLINK-ET4 Controllers using grip index 3:

Command: MRELEASE(3,1,1,0,0,0,1,1)

Response: ACK

2.5.3 Wait for state transition on multiple devices – MWAITFOR

Whenever an action like GRIP or RELEASE is performed, the device state of the respective devices will be updated upon completion. The MWAITFOR command allows to wait for this state transition on multiple devices. It will return successfully after all selected devices have changed their state, regardless of the new state. If a state transition does not occur on a channel, the function returns with a timeout error.

 To determine the current state of the devices of interest, use command DEVSTATE.

Syntax

MWAITFOR(<sel0>,...,<seln>)

Parameters

<seln> Select (1) or deselect (0) device on port n.
Port number n will be 0 to (PORTS-1) in Stand-Alone mode and 0 to 31 in Master mode. Trailing 0 parameters can be omitted.

Response Message

ACK on success or ERR <ERRORCODE> in case of an error.

Example 1

Release a previously gripped workpiece with grippers on port 0 and 1 using grip index 3 and wait for completion:

Command: MRELEASE(3,1,1,0,0)

Response: ACK

Command: MWAITFOR(1,1,0,0)

Response: ACK

Example 2

Grip a workpiece with grippers on port 0, 1 and 5, 6 on a virtual GRIPLINK controller network of two GRIPLINK-ET4 Controllers using grip index 3:

Command: MGRIP(3,1,1,0,0,0,1,1)

Response: ACK

Command: MWAITFOR(1,1,0,0,0,1,1)

Response: ACK

2.6 Device status and diagnosis

2.6.1 Get device state – DEVSTATE

Each device has a certain status. This command is used to retrieve the device state of the selected device. Possible device state values are given in chapter 4. Grippers are designed in a way that with every active command (i.e. GRIP, RELEASE, HOME, DISABLE/ENABLE) always a state change occurs on completion. It is a common practice to poll the device state and continue, as soon as this state changes.

 If no device is installed on the selected port, the function returns 0 (DS_INVALID).

Syntax

DEVSTATE[<port>]?

Parameters

<port>	Port number	
	Proxy role STANDALONE:	0 to (PORTS-1)
	Proxy role MASTER:	0 to 31

Response Message

DEVSTATE[<port>]=<state>

<state> Device state, see table in chapter 4

Example


Retrieve the device state of the connected gripper on port 0:

Command: DEVSTATE[0]?

Response: DEVSTATE[0]=3 -> Device state is DS_RELEASED

2.6.2 Get value – VALUE

Read a sensor value from the given device. Devices may support one or more values that can be read. See the device specification for details on provided sensor values.

 For gripper devices, the first value (index 0) returns the finger position in Micrometers (μm).

Syntax

VALUE[<port>][<index>]?

Parameters

<port>	Port number
	Proxy role STANDALONE: 0 to (PORTS-1)
	Proxy role MASTER: 0 to 31
<index>	Value index (0..n-1). The number of provided values depends on the connected device.

Response Message

VALUE[<port>][<index>]=<val>

<val> Sensor data as integer value

 If the sensor value is out of range, the VALUE function returns 2.147.483.647 (INT32 MAX)

Example

Retrieve the finger position of the connected gripper on port 0:


Command: VALUE[0][0]?

Response: VALUE[0][0]=42000 -> Finger position is 42 mm.

2.7 Device Configuration

2.7.1 Grip configuration – GRIPCFG

Read or modify the selected grip recipe/preset. Each recipe consists of a tag string and 8 parameters that have device-specific meaning. The tag string can be used to give the recipe a meaningful name for later identification.

 The parameters may have specific limits depending on the connected device. See the device manual for details.

Syntax

Assign: GRIPCFG[<port>][<index>]=[<tag>,<param0>,<param1>,...,<param7>]
Query: GRIPCFG[<port>][<index>]?

Parameters

<port> Port number
Proxy role STANDALONE: 0 to (PORTS-1)
Proxy role MASTER: 0 to 31
<tag> String to name the specific recipe (max. 32 characters)
<param0..7> Device-specific floating-point parameter values

For **WEISS ROBOTICS gripper modules**, the parameters are used as follows:

<param0> No-Part limit in Micrometers (range is device-specific)
<param1> Release limit in Micrometers (range is device-specific)
<param2> Force factor in percent multiplied by 1000 (0 to 100000)
<param3..7> not used (set to 0)

For **WEISS ROBOTICS WPG series**, the parameters 3 to 7 can be used as follows:

<param3> Grip speed in Micrometers per second (optional, range is device-specific)
<param4> Grip acceleration in Micrometers per square second (optional, range is device-specific)
<param5> Release speed in Micrometers per second (optional, range is device-specific)
<param6> Release acceleration in Micrometers per square second (optional, range is device-specific)
<param7> not used (set to 0)

Response Message

GRIPCFG[<port>][<index>]=[<tag>,<param0>,<param1>,...,<param7>]

Example

Set the recipe 0 named "Tube" for the WEISS ROBOTICS servo gripper on port 1 with a No-Part limit of 3.5 mm, a release limit of 10.8 mm and a gripping force of 50%:

Command: GRIPCFG[1][0]=["tube",3500,10800,50000,0,0,0,0,0]

Response: GRIPCFG[1][0]=["tube",3500,10800,50000,0,0,0,0,0]

3 Status and error codes

After executing a command, the GRIPLINK Controller returns a status code describing the result of the operation.

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	Pending action / Not yet finished

4 Device status codes

The following table lists the possible return values of the DEVSTATE command. Note that some return values are reserved to a certain group of devices (e.g., a DS_RELEASED state will usually be returned by a gripper but not by a sensor).

Code	Identifier	Scope	Description
0	DS_NOT_CONNECTED	UNI	No device connected
1	DS_NOT_INITIALIZED	UNI	Device is not initialized. This state is e.g., used by grippers that need to perform a reference drive at startup.
2	DS_DISABLED	UNI	Idle state. The device is operable and waiting for a command.
3	DS_RELEASED	GRIPPER	Part released. A RELEASE command was issued and the fingers reached the release position.
4	DS_NO_PART	GRIPPER	No part found. A GRIP command was issued, but the gripper didn't detect a part during grasping.
5	DS_HOLDING	GRIPPER	Holding a part. A GRIP command was issued and the gripper is now holding the part.
6	DS_ENABLED	SENSOR	Enabled state. The device is in operating state and performs its function as intended.
7	DS_FAULT	UNI	Fault state. The device is in fault state. See the device manual for possible recovery strategy.

5 Supported commands by device class

Command	Gripper	Sensor
CLAMP	Optional	-
DEVASSERT	Yes	Yes
DEVNAME	Yes	Yes
DEVPID	Yes	Yes
DEVSN	Yes	Yes
DEVSTATE	Yes	Yes
DEVTAG	Optional	Optional
DEVVENDOR	Yes	Yes
DEVVER	Yes	Yes
DEVVID	Yes	Yes
DISABLE	Yes	Yes
ENABLE	Yes	Yes
FLEXGRIP	Optional	-
FLEXRELEASE	Optional	-
GRIP	Yes	-
GRIPCFG	Optional	-
HOME	Optional	-
LED	Optional	-
MGRIP	Yes	-
MRELEASE	Yes	-
MWAITFOR	Yes	Yes
RELEASE	Yes	-
SETVAL	Optional	Optional
VALUE	Yes	Yes

WAITVAL	Yes	Yes
WSTR	Yes	Yes

6 List of Product and Vendor IDs

The following table lists selected WEISS ROBOTICS products with their associated Vendor and Product IDs.

Product	Primary control interface	Vendor ID	Product ID
CLG 30-006	IO-Link	815	60
CRG 200-085	IO-Link	815	50
CRG 30-050	IO-Link	815	40
IEG 55-020	IO-Link	815	20
IEG 76-030	IO-Link	815	20
IEG PLUS 260-030	IO-Link	815	24
IEG PLUS 260-080	IO-Link	815	25
IEG PLUS 40-020	IO-Link	815	22
IEG PLUS 40-050	IO-Link	815	23
RPG 120-020	IO-Link	815	10
RPG 75-012	IO-Link	815	10
STG 200-030	IO-Link	815	52
STG 200-085	IO-Link	815	51
STG 40-024	IO-Link	815	53
WPG 100-090	Ethernet	815	6050
WPG 300-120	Ethernet	815	6060
ZPG 75-012	IO-Link	815	30

© 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 property of the respective owners.

The technical data given in this document are subject to change without notice for the purpose of product improvement. Our products are not intended for use in life support systems or for systems in which misconduct could lead to personal injury.

GRIP SMARTER.



WEISS ROBOTICS