WSG Series of Intelligent Servo-Electric Grippers

GCL Gripper Control Language

Reference Manual

Firmware Version 4.0

December 2016

www.weiss-robotics.com

Contents

1 Introduction

The WSG family of gripping modules can be controlled by different standard interfaces, each of which supports one or more different communication protocols. This manual describes the Weiss Robotics *Gripper Control Language (GCL)*, a text based protocol that can be used to control the module via Ethernet using standard TCP/IP or UDP/IP connections.
The following chapters provide a detailed explanation of the protocol itself as well as of the module's command set. To get started with the communication protocol, we recommend using a common Telnet client like the free PuTTY[1] for Microsoft Windows as described in chapter 1.4.

1.1 Connecting to the WSG

Before connecting to the module, an appropriate interface must be selected using the module's web interface. Connect the WSG to the local network or directly to your computer's network interface and point your favorite web browser to the module's IP address, e.g. by typing the default **http://192.168.1.20** into the address bar and pressing <Enter>. Please make sure that your computer's network settings are appropriate.
After the page has loaded, choose "*Settings*" -> *"Command Interface"* from the menu to configure the default interface. The text based command set described in this manual is available for TCP/IP and UDP/IP connections. To get started, Weiss Robotics recommends to use the TCP/IP interface. You can switch to UDP/IP later if necessary as soon as you are familiar with the basic command format.
Finally, don't forget to enable the option "Use text based interface" (cf. Figure 1).

[1] http://www.putty.org/

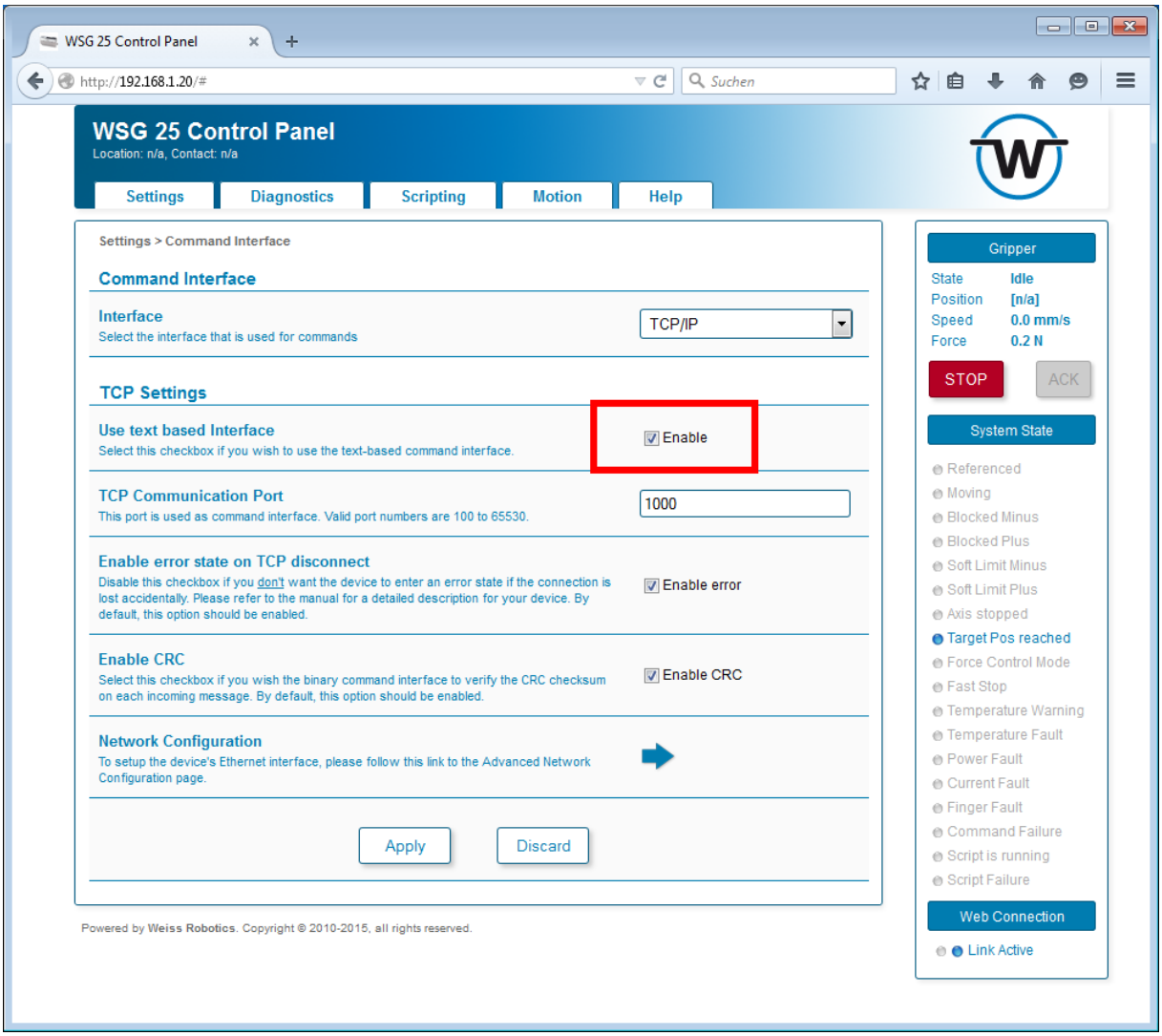


Figure 1: Enabling GCL on the web interface

1.2 Communicating with the WSG

Regardless of the interface chosen above, the WSG now communicates with its client using the text-based protocol. The following chapters describe the general format of these commands.
The module expects commands being submitted as plain ASCII strings. **Each command must be terminated by a line feed character ('\n' or ASCII code 0x0d).** Return messages are submitted in the same format and are terminated by a line feed character, too.

Please note that the commands of the GCL Gripper Control Language are not case sensitive, i.e. sending "move(50)" is the same as "MOVE(50)" or "mOVe(50)". Response messages will, however, always be sent in upper case notation.

1.3 Error Handling

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

ERR <command_name> <error_code>

where <command_name> represents the command that caused the error and <error_code> represents a number referencing an error code. Appendix A**Fehler! Verweisquelle konnte nicht gefunden erden.** gives an overview of all available error codes.
If verbose mode is active (cf. chapter 2.1.1), the module submits extended error messages containing an additional text string that describes the type of error:

ERR <command_name> <error_code> <description_string>

☞ **See Appendix A to get a description of the error codes.**

☞ **See chapter 2.1.1 on how to enable verbose mode.**

1.4 Connecting to the command interface using PuTTY

PuTTY is a free Telnet and SSH client that can be used to connect to the module's command interface. The following chapter shows how to use PuTTY with the WSG.
Before you start, please make sure the module is configured to use the TCP/IP interface and that the text based interface is selected. **The default IP address of the module is 192.168.1.20, the default TCP/IP listening port is 1000.**
Download and install PuTTY for Windows from **http://www.putty.org**. On Unix-like systems, an equivalent command line Telnet client should be available which can be used, too.
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 2).
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 3).
Now click the "Open" button to open the connection. A new and empty terminal window will appear (Figure 4), ready to type in your commands.
Typing "HOME()" for example, followed by <Enter>, will reference the gripper (see chapter 2.2.1).

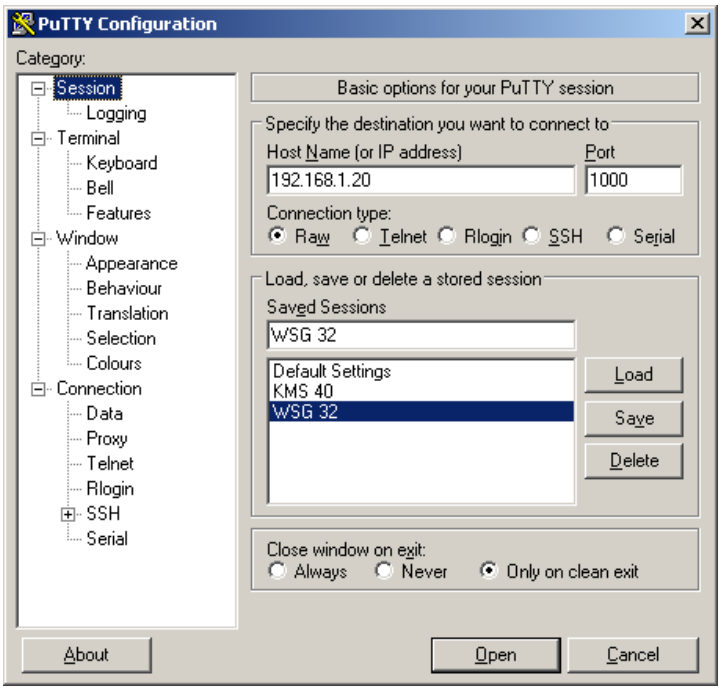


Figure 2: PuTTY session settings

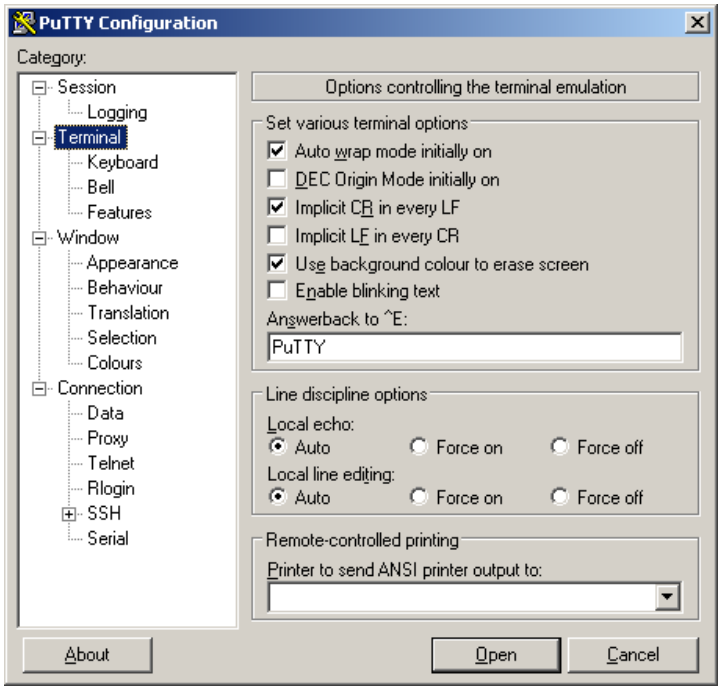


Figure 3: PuTTY terminal settings

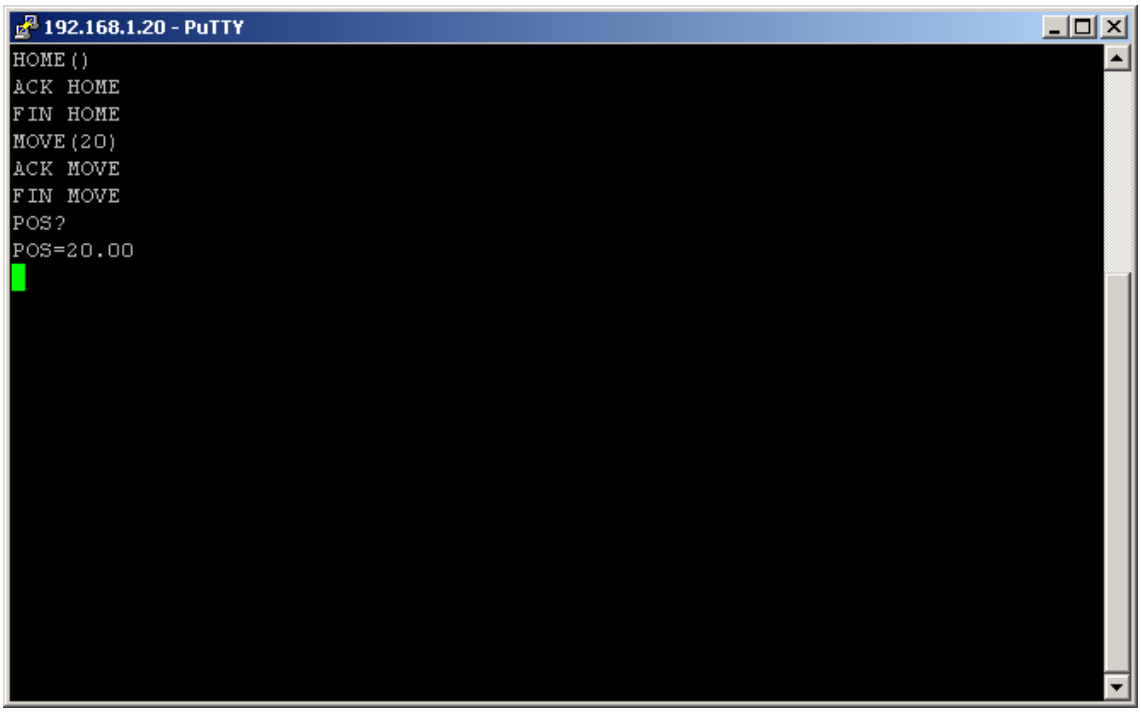


Figure 4: PuTTY terminal window

2 Basic Command Set

This chapter describes the basic command set that can be used to grip parts. For an extended set of commands providing more functionality, please refer to chapter 3.

2.1 Interface Control

2.1.1 Enable verbose mode – VERBOSE

Enables the interface's verbose mode. By default, verbose mode is turned off, meaning that the module only returns a numeral error code with its error messages. By enabling verbose mode, the module additionally returns a text string describing the error.

Syntax

VERBOSE=<integer>

Parameters

Integer value telling whether verbose mode should be enabled (1) or disabled (0).

Return String

VERBOSE=<integer>
e. g. VERBOSE=0

2.1.2 Disconnect from interface – BYE

To disconnect safely from the module, the termination of the connection must be announced before closing.

If the disconnect is not announced before closing the connection, a FAST STOP will be raised blocking all further motion commands.

Syntax

BYE()

Parameters

No parameters.

Return String

ACK BYE to acknowledge the command

2.2 Motion Control

2.2.1 Referencing the module – HOME

Execute a homing sequence to reference the gripper fingers. If no parameter is given, referencing will be done in default direction.
This command has to be executed prior to any other motion-related command. The direction of homing can be either explicitly specified or can be obtained from the gripper's configuration. During homing, the gripper moves its fingers into the specified direction until it reaches its mechanical end stop. The blocking position is used as new origin for all motion-related commands.

The best positioning performance will be achieved if homing is done into the direction you require the better positioning accuracy.

During homing soft limits are disabled!

Obstacles in the movement range of the fingers and collision with these during homing may result in a wrong reference point for the finger position!

Syntax

HOME()
HOME(<bool>)

Parameters

<bool> (optional) Direction of referencing. If 1, referencing will be done in positive direction, if 0 in negative direction

Return String

ACK HOME to immediately acknowledge command execution
FIN HOME to indicate command has completed successfully

2.2.2 Move the gripper fingers – MOVE

The MOVE command is intended to position the gripper jaws between the gripping cycles, e.g. to move the jaws quickly before softly gripping sensitive parts.
The command expects one or two parameters of which the first one indicates the target position in millimeters to which the gripper jaws should be moved and the second parameter indicates a speed limit in millimeters per second.

Do not use the MOVE command to grip or release parts. Use the GRIP and RELEASE command instead.

The gripper module has to be homed and must not be in FAST STOP state!

Syntax

MOVE(<float>)
MOVE(<float>, <float>)

Parameters

<float>	Position in mm
<float>	Speed limit in mm/s (optional)

Return String

ACK MOVE	to immediately acknowledge command execution
FIN MOVE	to indicate command has completed successfully

2.2.3 Grip part – GRIP

Grip a part. The command's behavior depends on the number of given parameters:

1. No parameter. Grip inside until a part is detected. Use default force and speed.
2. One parameter: Force. Grip inside until a part is detected. Use the given force (in N).
3. Two parameters: Force, Part width. Grip inside or outside (depending on the current position and the target position). Expect a part at the given position. Use the given force (in N).
   If the gripper detects a contact outside the part width tolerance, it is interpreted as a collision and an error is returned.
4. Three parameters: Force, Position, Speed Limit. Like 3 but use the additional speed limit.

Case 1 and 2: As no part width is given, the gripper will grip itself if there is no part between the fingers and the part detection will always set the gripper state to HOLDING. You might check the position of the gripper jaws after gripping to make sure a part has been gripped.

If a part width is passed to the GRIP command (i.e. case 3 or 4 in the list above) and the gripper can establish the desired force within the defined clamping travel, the gripper state is set to HOLDING (part detection feature) and Grip Monitoring will be enabled, i.e. force and position will be continuously checked.
If there was no part between the gripper fingers so they can fall through the clamping travel without establishing the full force, the gripper reports that no part was found.

The clamping travel and the part width tolerance can be set using the WSG's web interface. Please see the User's Manual for a detailed description of these parameters.

When gripping, the gripper state is updated with the result (either HOLDING or NO PART) as well as the gripper statistics (see chapter 3.3.2). If no part was found, the command returns an E_CMD_FAILED error.

Part Detection Feature

If a part width was passed to the GRIP command (i.e. case 3 or 4), the gripper expects a part to be found around this position, see the figure below.

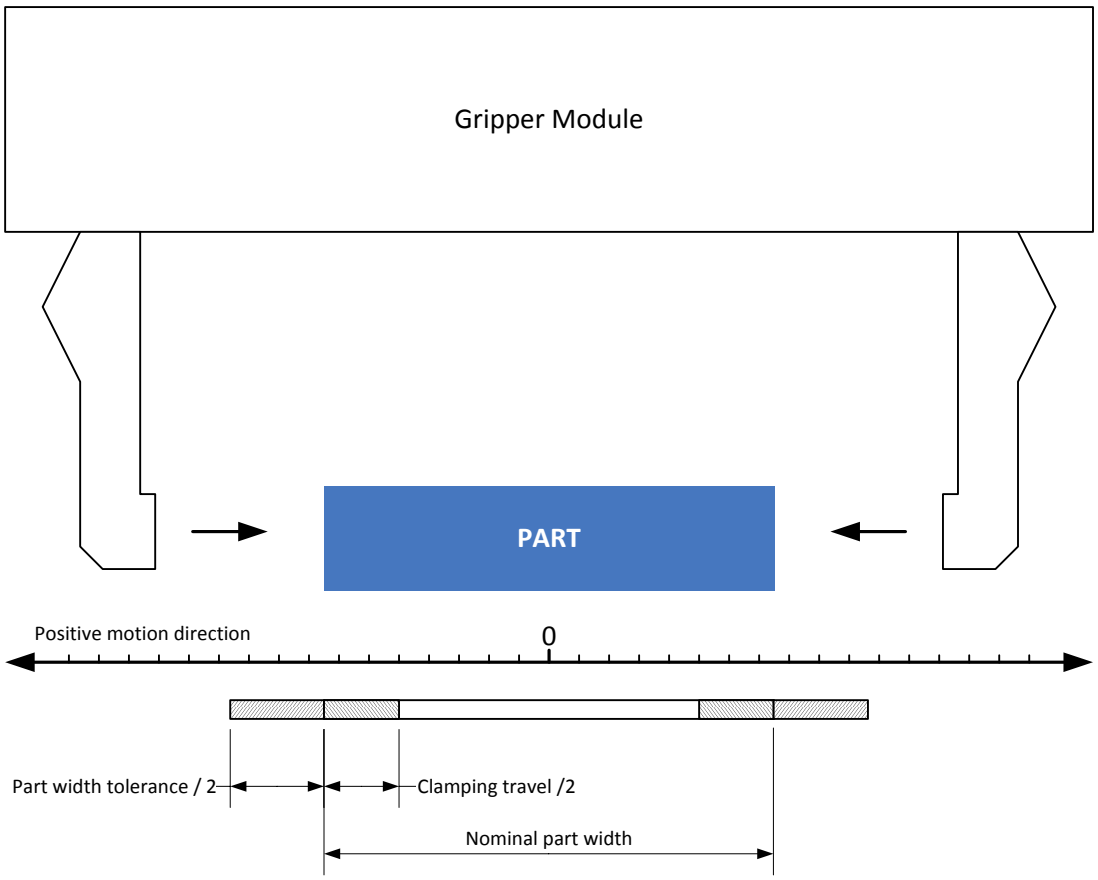


Figure 5: Part width tolerance and clamping travel

If the gripper detects a contact before reaching the part width tolerance area, this is interpreted as a collision and an E_AXIS_BLOCKED error is returned.

You may reduce the grasping speed with sensitive parts to limit the impact due to the mass of the gripper fingers and the internal mechanics.

The gripper state reflects the current state of the process. You can read it using the GRIPSTATE command (see chapter 2.3.4).

It is not possible to send a grip commands while holding a part. In general, a grip command should be always followed by a release command (see chapter 2.2.4) before the next grip command is issued.

Syntax

GRIP()
GRIP(<float>)
GRIP(<float>, <float>)
GRIP(<float>, <float>, <float>)

Parameters

<*float*>	Force in N (optional)
<*float*>	Part width in mm (optional)
<*float*>	Speed limit in mm/s (optional)

Return String

ACK GRIP	to immediately acknowledge command execution
FIN GRIP	to indicate command has completed successfully

2.2.4 Release part – RELEASE

Release a previously gripped part. The command's behavior depends on the number of given parameters:

1. No parameter. Open the gripper fingers relative to the current position by the predefined default pull back distance relative to the current position. The default pull back distance can be set via the module's web interface by choosing *"Settings"* -> *"Motion Configuration"* from the menu.
2. One parameter: Pull back distance. Open the gripper fingers by the given pull back distance relative to the current position.
3. Two parameters: Pull back distance, speed limit. Open the gripper fingers by the given pull back distance relative to the current position using the given speed limit.

Release commands are only allowed if the gripper has gripped a part before using the GRIP() command.

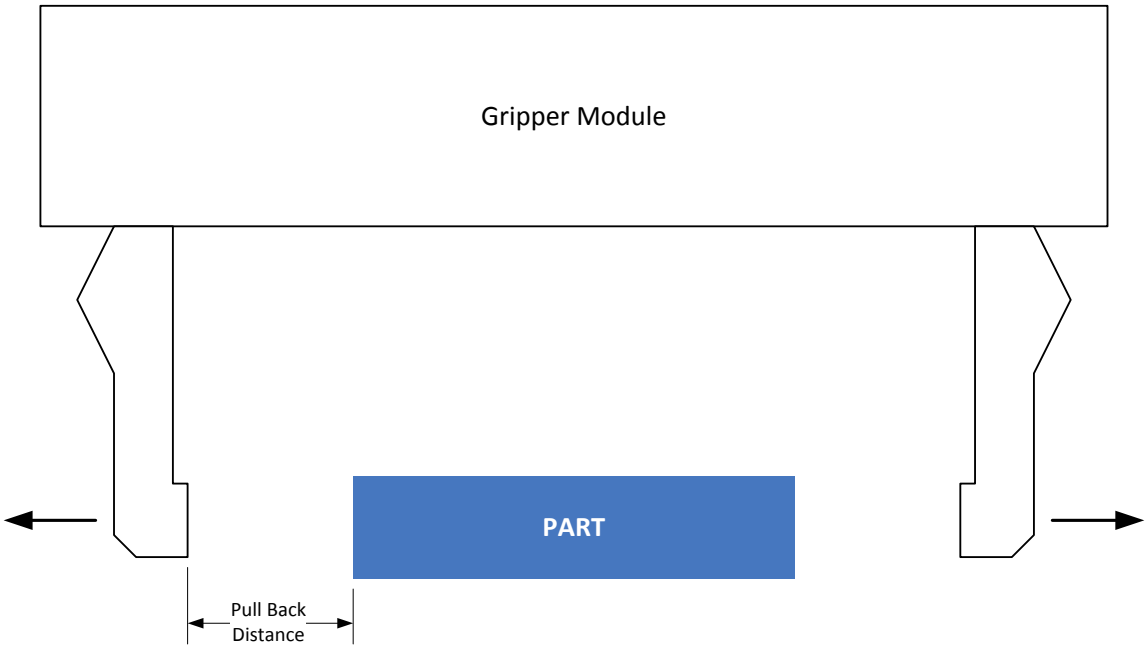


Figure 6: Pull back distance on release

Syntax

RELEASE()
RELEASE(<float>)
RELEASE(<float>, <float>)

Parameters

<float>	Pull back distance in mm (optional)
<float>	Speed limit in mm/s (optional)

Return String

ACK RELEASE	to immediately acknowledge command execution
FIN RELEASE	to indicate the command has completed successfully

2.2.5 Get or set part width tolerance – PWT

During the execution of a grip command (cf. chapter 2.2.3), the part width tolerance indicates the distance before reaching the nominal part width, within which a part is considered to be gripped correctly (cf. Figure 5). If the fingers are blocked outside this distance, the grip command returns an error.
The part width tolerance can be set globally using the gripper's web interface. The command described here can be used to override the preconfigured value, for example to dynamically adjust the part width tolerance to different parts.

The part width tolerance is changed only for the time the connection is active and the changed value only takes effect for grip commands sent via GCL. As soon as the connection is closed, the part width tolerance will be reset to the preconfigured value.

Syntax

PWT?
PWT=<float>

Parameters

<float> Part width tolerance in mm

Return string

PWT=<float>

2.2.6 Get or set clamping travel – CLT

During the execution of a grip command (cf. chapter 2.2.3), the clamping travel indicates the distance the fingers are allowed to move further after touching a part to apply the desired gripping force (cf. Figure 5). If the gripping force can't be applied within this distance, the grip command returns an error.
At the same time, the clamping travel indicates how far the fingers are allowed to move beyond the nominal part width to detect a part.
If a part is detected before reaching the nominal part width, the clamping travel is measured from that point. If the nominal part width is reached without detecting a part, the clamping travel is measured from the nominal part width.
The clamping travel can be set globally using the gripper's web interface. The command described here can be used to override the preconfigured value, for example to dynamically adjust the clamping travel to different parts.

The clamping travel is changed only for the time the connection is active and the changed value only takes effect for grip commands sent via GCL. As soon as the connection is closed, the clamping travel will be reset to the preconfigured value.

Syntax

CLT?
CLT=<float>

Parameters

<float> Clamping travel in mm

Return string

CLT=<float>

2.3 Gripper State

2.3.1 Query current position of gripper jaws – POS

Returns the current position of the gripper jaws (open width).

Syntax

POS?

Return String

POS=<float>
e. g. POS=20.0

2.3.2 Query current speed of gripper jaws – SPEED

Get current speed in mm/s.

Syntax

SPEED?

Return String

SPEED=<float>
e. g. SPEED=142.0

2.3.3 Query current gripping force – FORCE

Get current force value in N.

Syntax

FORCE?

Return String

FORCE=<float>
e. g. FORCE=23.0

2.3.4 Query gripper state – GRIPSTATE

Get current gripper state. The command returns a numeral value indicating the gripper state.

☞ **See Appendix D**Fehler! Verweisquelle konnte nicht gefunden werden. **for a description of he gripper states.**

Syntax

GRIPSTATE?

Return String

GRIPSTATE=<integer>
e. g. GRIPSTATE=4 to indicate HOLDING.

3 Extended Command Set

The following chapter describes the extended command set of the WSG in detail.

3.1 System Commands

3.1.1 Query device type – DEVTYPE

This command returns the type of the system and can be used for example to distinguish between different devices manufactured by Weiss Robotics that support this protocol.

Syntax

DEVTYPE?

Return String

DEVTYPE=<string>
The command returns the system type string, e.g. DEVTYPE=”WSG 32-068”.

3.1.2 Query firmware version – VERSION

Returns the firmware version of the module.

Syntax

VERSION?

Return String

VERSION=<string>
The command returns the firmware version string, e.g. VERSION=”1.0.0”.

3.1.3 Query serial number – SN

Returns the module’s serial number.

Syntax

SN?

Return String

SN=<integer>
The command returns the module’s serial number, eg. SN=12345678.

3.1.4 Query descriptor string (device tag) – TAG

This command can be used to read the module's descriptor string. The descriptor string, also known as device tag, can be used to identify the module.

Syntax

TAG?

Return String

TAG=<string>
The command returns the module's descriptor string, e.g. DEVTAG="My Descriptor".

3.1.5 Query system state flags – SYSFLAGS

Get system state flags. The returned value is a vector that represents the 32 available system state flags. Appendix B gives an overview of the available flags.

Syntax

SYSFLAGS?
SYSFLAGS[<index>]?
with index representing a value between 0 and 31 that indicates the desired system flag.

Return String

SYSFLAGS=[<bool>,<bool>,…<bool>]
where <bool> is 1 if the corresponding system flag is set or 0 if not.
In case an optional index is submitted, the response is
SYSFLAGS[<index>]=<bool>

3.1.6 Query temperature – TEMP

Get the current system temperature in degrees Celsius.

Syntax

TEMP?

Return String

TEMP=<float>
in degrees Celsius, e.g. TEMP=34.2

3.1.7 Enable or disable auto-sending of parameters - AUTOSEND

A number of gripper values can be submitted automatically in custom time intervals or on change. The following values are available:

Parameter Name	Description
POS	Position of the gripper fingers (open width)
SPEED	Current speed of the gripper fingers
FORCE	Current gripping force
GRIPSTATE	Gripper state
SYSFLAGS	System flags (see chapter **Fehler! Verweisquelle konnte icht gefunden werden.** for detailed description)
TEMP	Temperature of the controller board

Syntax

AUTOSEND(<string>, <integer>)
AUTOSEND(<string>, <integer>, <float>)
AUTOSEND(<string>, <integer>, <bool>)

Parameters

<string> Indicates the name of the value that should be submitted automatically.

<integer> Indicates the send interval in ms with a minimum value of 10 ms. Set this value to 0 to disable auto-sending

<float> Optional delta value. For numeric values, a delta value can be set. In this case, the value will only be auto-submitted if it has changed by that value since the last submission.

<bool> Enable auto-submit on change only. For non-numeric values (e. g. system flags vector), the third parameter can be used as <bool> to indicate whether the parameter should be generally sent only if it has changed:

Return String

ACK AUTOSEND to immediately acknowledge command execution

The periodically submitted values all begin with an '@' sign:

@POS=<float> Auto submitted position value in mm
@FORCE=<float> Auto submitted force value in N
@SPEED=<float> Auto submitted speed value in mm/s
@GRIPSTATE=<integer> Auto submitted gripper state
@SYSFLAGS=[<bool>,…,<bool>] Auto submitted system flags vector
@TEMP=<float> Auto submitted temperature value

Examples

AUTOSEND("POS",10) will send the finger position every 10 ms

AUTOSEND("FORCE",50,0.1) will send the gripping force every 50 ms if it has changed for more than 0.1 N

3.2 Extended Motion Control

3.2.1 Stop motion – STOP

Immediately stops any ongoing finger movement and maintain the current position. The command sets the SF_AXIS_STOPPED flag. The AXIS STOPPED state does not need to be acknowledged; it is cleared automatically by the next motion-related command. A running motion command (e.g. MOVE) will return ERR 19 (command aborted).

☞ **If you would like to stop the gripper module in case of an error, use the FASTSTOP command instead.**

Syntax

STOP()

Parameters

No parameters

Return String

ACK STOP to immediately acknowledge command execution

3.2.2 Issue fast stop – FASTSTOP

Issue fast stop and disable drive. This command is intended to react to error conditions within the application. This function is similar to an “Emergency Stop”. It immediately stops any finger movement the fastest way and prevents further motion-related commands from being executed. The FAST STOP state can only be left by issuing a **FAST STOP Acknowledge command**. All motion-related commands are prohibited during FAST STOP and will produce an E_ACCESS_DENIED error.
The FAST STOP state is indicated in the **System Flags** and logged in the gripper module’s log file, so this command should in general be used to react on certain error conditions.

ⓘ **To stop the current finger movement without raising an error condition, use the STOP command instead.**

Syntax

FASTSTOP()

Parameters

No parameters

Return String

ACK FASTSTOP to immediately acknowledge command execution

3.2.3 Acknowledge Fast Stop – FSACK

Acknowledge fast stop. A previously issued **FAST STOP** or a severe error condition must be acknowledged using this command to bring the gripper module back into normal operating mode.

Syntax

FSACK()

Parameters

No parameters

Return String

ACK FSACK to immediately acknowledge command execution

3.3 Extended Gripper State

3.3.1 Tare force sensor – TARE

Tare the given force measurement finger(s). If no parameter is given, all connected force measurement fingers will be tared. If a parameter is given, the force measurement finger with the given index (0 or 1) will be tared.

Syntax

TARE()
TARE(<integer>)

Parameters

<integer> Index of force measurement finger to be tared.

Return String

ACK TARE to immediately acknowledge command execution

3.3.2 Query gripper statistics – GRIPSTATS

Get gripper statistics.

Syntax

GRIPSTATS?
GRIPSTATS[<integer>]?

Return String

GRIPSTATS=[<integer>,<integer>,<integer>]
GRIPSTATS[<integer>]=<integer>

Returns an integer vector representing the gripper stats or a single integer value representing the value at the given index.

Index 0: Total number grips
Index 1: Number of grips with no part detected
Index 2: Number of grips with part lost

3.4 Finger Interface

3.4.1 Query finger data – FDATA

Get finger data. Return value depends on finger type.

Syntax

FDATA?
FDATA[<integer>]?

Index

<integer> Finger index

Return String

FDATA=[<data>,<data>]
FDATA[<integer>]=<data>
The return value depends heavily on the finger type. WSG-FMF returns the measured force value, WSG-DSA returns a vector holding measured tactile sensing values.

3.4.2 Query finger type – FTYPE

Get finger type string.

Syntax

FTYPE?
FTYPE[<integer>]?

Index

<integer> Finger index

Return String

FTYPE=[<string>,<string>]
FTYPE[<integer>]=<string>

3.4.3 Query finger flags – FFLAGS

Get finger flags. See **Fehler! Verweisquelle konnte nicht gefunden werden.** for a description of the vailable flags.

Syntax

FFLAGS?

FFLAGS[<integer>]?

Index

<*integer*> Finger index

Return string

FFLAGS=[[<bool>, …, <bool>], [<bool>, …, <bool>]]
FFLAGS[<integer>]=[<bool>,…,<bool>]

Appendix A. Status Codes

Error code	Symbol name	Description
0	E_SUCCESS	No error occurred, operation was successful
1	E_NOT_AVAILABLE	Function or data is not available
2	E_NO_SENSOR	No measurement converter is connected
3	E_NOT_INITIALIZED	Device was not initialized
4	E_ALREADY_RUNNING	The data acquisition is already running
5	E_FEATURE_NOT_SUPPORTED	The requested feature is currently not available
6	E_INCONSISTENT_DATA	One or more parameters are inconsistent
7	E_TIMEOUT	Timeout error
8	E_READ_ERROR	Error while reading data
9	E_WRITE_ERROR	Error while writing data
10	E_INSUFFICIENT_RESOURCES	No more memory available
11	E_CHECKSUM_ERROR	Checksum error
12	E_NO_PARAM_EXPECTED	A Parameter was given, but none expected
13	E_NOT_ENOUGH_PARAMS	Not enough parameters to execute the command
14	E_CMD_UNKNOWN	Unknown command
15	E_CMD_FORMAT_ERROR	Command format error
16	E_ACCESS_DENIED	Access denied
17	E_ALREADY_OPEN	Interface is already open
18	E_CMD_FAILED	Error while executing a command
19	E_CMD_ABORTED	Command execution was aborted by the user
20	E_INVALID_HANDLE	Invalid handle
21	E_ NOT_FOUND	Device or file not found
22	E_ NOT_OPEN	Device or file not open

23	E_IO_ERROR	Input/Output Error
24	E_INVALID_PARAMETER	Wrong parameter
25	E_INDEX_OUT_OF_BOUNDS	Index out of bounds
26	E_CMD_PENDING	No error, but the command was not completed, yet. Another return message will follow including an error code, if the function was completed.
27	E_OVERRUN	Data overrun
28	RANGE_ERROR	Range error
29	E_AXIS_BLOCKED	Axis blocked
30	E_FILE_EXISTS	File already exists

Appendix B. System State Flags

The WSG provides a total of 32 system state flags that can be read using the SYSFLAGS query (see chapter3.1.5). The following table lists the available flags and explains their meaning.

Index	Flag Name	Description
31..21	reserved	These bits are currently unused but may be used in a future release of the WSG firmware.
20	SF_SCRIPT_FAILURE	**Script Error.** An error occurred while executing a script and the script has been aborted. The flag is reset whenever a script is started.
19	SF_SCRIPT_RUNNING	**A script is currently running.** The flag is reset if the script either terminated normally, a script error occurred or the script has been terminated manually by the user.
18	SF_CMD_FAILURE	**Command Error.** The last command returned an error.
17	SF_FINGER_FAULT	**Finger Fault.** The status of at least one finger is different from “operating” and “not connected”. Please check the finger flags for a more detailed error description.
16	SF_CURR_FAULT	**Engine Current Error.** The engine has reached its maximum thermal power consumption. The flag will be reset automatically as soon as the engine has recovered. Then the corresponding Fast Stop can be committed.
15	SF_POWER_FAULT	**Power Error.** The power supply is outside the valid range.
14	SF_TEMP_FAULT	**Temperature Error.** The gripper hardware has reached a critical temperature level. All motion-related commands are disabled until the temperature falls below the critical level.

13	SF_TEMP_WARNING	**Temperature Warning.** The gripper hardware will soon reach a critical temperature level.
12	SF_FAST_STOP	**Fast Stop.** The gripper has been stopped due to an error condition. You have to acknowledge the error in order to reset this flag and to re-enable motion-related commands.
11..10	reserved	These bits are currently unused but may be used in a future release of the WSG firmware.
9	SF_FORCECNTL_MODE	**Force Control Mode.** True force control is currently enabled by using the installed force measurement finger (WSG-FMF). If this flag is not set, the grasping force is controlled by approximation based on the motor current.
8	SF_OVERDRIVE_MODE	**Overdrive Mode[2].** Gripper is in overdrive mode and the grasping force can be set to a value up to the overdrive force limit. If this bit is not set, the grasping force cannot be higher than the gripper's nominal grasping force value.
7	SF_TARGET_POS_REACHED	**Target position reached.** Set if the target position was reached. This flag is not synchronized with SF_MOVING, so it is possible that there is a delay between SF_MOVING being reset and SF_TARGET_POS becoming active.
6	SF_AXIS_STOPPED	**Axis stopped.** A previous motion command has been aborted using the stop command. This flag is reset on the next motion command.

[2] Overdrive mode is not supported by all WSG grippers. Please refer to the User's Manual for further information.

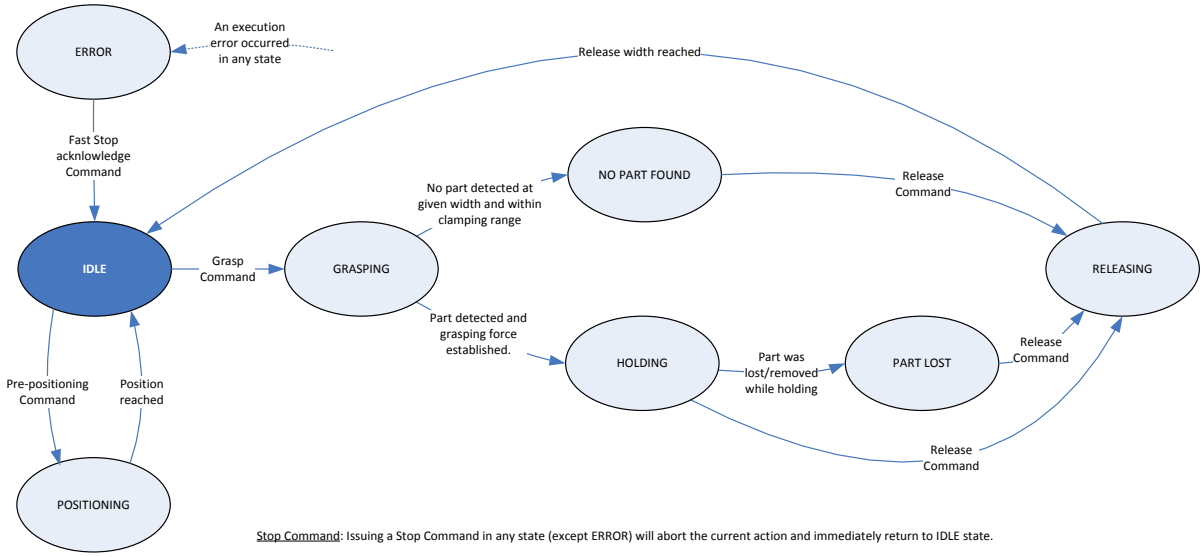
5	SF_SOFT_LIMIT_PLUS	**Positive direction soft limit reached.** The fingers reached the defined soft limit in positive moving direction. A further movement into this direction is not allowed any more. This flag is cleared if the fingers are moved away from the soft limit position.
4	SF_SOFT_LIMIT_MINUS	**Negative direction soft limit reached.** The fingers reached the defined soft limit in negative moving direction. A further movement into this direction is not allowed any more. This flag is cleared if the fingers are moved away from the soft limit position.
3	SF_BLOCKED_PLUS	**Axis is blocked in positive moving direction.** Set if the axis is blocked in positive moving direction. The flag is reset if either the blocking condition is resolved or a stop command is issued.
2	SF_BLOCKED_MINUS	**Axis is blocked in negative moving direction.** Set if the axis is blocked in negative moving direction. The flag will be reset if either the blocking condition is resolved or a stop command is issued.
1	SF_MOVING	**The Fingers are currently moving.** This flag is set whenever a movement is started (e.g. MOVE command) and reset automatically as soon as the movement stops.
0	SF_REFERENCED	**Fingers Referenced.** If set, the gripper is referenced and accepts motion-related commands.

Appendix C. Finger Flags

Index	Flag Name	Description
15..11	reserved	These flags are currently unused but may be used in a future release of the WSG firmware.
10	FF_INIT_FAULT	**Finger initialization error.** An error occurred during finger initialization.
9	FF_COMM_FAULT	**Communication Fault.** A Communication fault occurred during runtime.
8	FF_POWER_FAULT	**Power Fault.** Over-Current fault detected.
7..3	reserved	These flags are currently unused but may be used in a future release of the WSG firmware.
2	FF_COMM_OPEN	**Communication open.** Finger communication interface is open.
1	FF_CONFIG_AVAIL	**Finger configuration available.** A finger configuration descriptor could be read from the finger's memory.
0	FF_POWER_ON	**Finger powered.** Finger is powered on.

Appendix D. Gripper States

The following diagram illustrates the gripper states and transitions as intended to be used in normal operation.



The table below lists the available gripper states and their corresponding numeral representation.

No.	Flag Name	Description
0	IDLE	The module is IDLE, waiting for commands.
1	GRASPING	The gripper fingers are moving to grip a part.
2	NO PART	Couldn't detect a part when gripping.
3	PART LOST	A part has been gripped but was lost afterwards
4	HOLDING	Currently holding a part with the given force.
5	RELEASING	The gripper fingers are moving to release a part.
6	POSITIONING	The gripper fingers are moving to position the fingers.
7	ERROR	An error occurred.

Appendix E. Data Types

The following data types are used throughout this document:

<integer>	An integer value
<bool>	An integer value which can be 0 or 1
<float>	A floating point value
<string>	A string literal, always enclosed into quotes
<vector>	A vector of multiple values of the same type

www.weiss-robotics.com