Fieldbus Interface
WSG-Fieldbus Interface

Software manual - Firmware version 4.0.x

Imprint

Technical changes:

We reserve the right to make alterations for the purpose of technical improvement.

Document number: 1011122

Version: 01.00 |17/08/2017|en

Dear Customer,

thank you for trusting our products and our family-owned company, the leading technology supplier of robots and production machines.

Our team is always available to answer any questions on this product and other solutions. Ask us questions and challenge us. We will find a solution!

Best regards,

Your SCHUNK team

SCHUNK GmbH & Co. KG
Spann- und Greiftechnik

Bahnhofstr. 106 – 134
D-74348 Lauffen/Neckar

Tel. +49-7133-103-0
Fax +49-7133-103-2399

info@de.schunk.com
schunk.com

Reg. No. 003496 QM08

Reg. No. 003496 QM08

Table of contents

1 Introduction

The WSG family of grippers provides interfaces to PROFIBUS DP V0, PROFINET and/or Modbus/TCP, dependent on the device type.

PROFIBUS is a widely spread field bus protocol for industrial automation. It supports both single and multiple Master modes.

PROFINET is a new generation of fieldbus interface, designed to provide real-time communication via a standard Ethernet interface.

Every device is represented by an I/O register space that is periodically synchronized with the PROFIBUS Master, PROFINET Controller.

The following manual assumes knowledge about the PROFIBUS, PROFINET and/or Mod-bus/TCP technology and the Siemens SIMATIC software.

NOTE
PROFINET and Modbus/TCP are optional features. Licence keys can be obtained separately at SCHUNK.

NOTE
Simple demo programs for Siemens SIMATIC S7-1200 are available on the down load section of the WSG´s web interface.(☞ 5.4, Page 34).

2 Interfaces

2.1 PROFIBUS

Each PROFIBUS slave has an I/O register space that is periodically updated and read by the PROFIBUS Master. The I/O-Space of the WSG is pre-configured at master-side by using the device profile that can be found on the product CD or downloaded from the WSG's web interface.

Further information for the single I/O register see, (☞ 3, Page 9)

2.1.1 Installation of the GSD file

NOTE

The GSD-file can be installed at Siemens STEP7 v11.0 (TIA) or newer version.

The GSD-file is a compressed ZIP archive, which contains the following files:

- WEIS5555.gsd (device description file)
- WSG_D.bmp (visualization file)
- WSG_R.bmp (visualization file)
- WSG_S.bmp (visualization file)
- install.txt (installation notes)

Follow these steps to install the GSD file in Siemens STEP7 v11.0:

1. Unpack ZIP files on your hard disk.
2. Open the project view in the Siemens TIA.
3. Select *Options -> Install general station description file (GSD)*.
4. Change directory in which the files had been saved and choose the GSD file.

⇨ The gripper is displayed in the device catalog under *Other field devices-> PROFIBUS DP -> Drives* .

2.1.2 Configuration

To use the PROFIBUS interface, it must first be enabled via the device's web interface (*Settings -> Command Interface*). The PROFIBUS station address is predefined to 7. The station address can be changed via the web interface, too.

For further information see the Assembly and Operating manual of the gripper.

2.2 PROFINET

The PROFINET interface uses the same I/O register space layout that is used for PROFIBUS. Like with PROFIBUS, the I/O space is periodically updated and uses a pre-defined profile to be installed at controller side that can be found on the product CD or downloaded from the grippers web interface. Further information for the single I/O register see, (☞ 3, Page 9).

2.2.1 Installation of the GSDML file

NOTE

The GSDML-file can be installed at Siemens STEP7 v11.0 (TIA) or newer version.

The GSDML-file is a compressed ZIP archive, which contains the following files:

- GSDML-V2.31-WSG-20140401.gsdml (device description file)
- GSDML-02A2-0001-WSG.bmp (visualization file)

Follow these steps to install the GSDML file in Siemens STEP7 v11.0:

1 Unpack ZIP files on your hard disk.

2 Open the project view in the Siemens TIA.

3 Select *Options -> Install general station description file (GSD)*.

4 Change directory in which the files had been saved and choose the GSDML file.

⇨ The gripper is displayed in the device catalog under *Other field devices -> PROFINET IO -> I/O* .

2.2.2 Configuration

To use the PROFINET interface, it must first be enabled via the device's web interface. Further configuration options can be set either directly on the WSG using its web interface to change IP address or PROFINET device name. However, PROFINET also allows various configuration options to be set remotely using an engineering tool like e.g. Siemens STEP7.

NOTE
When changing the IP address settings remotely using an engineering tool, the device's web interface might become inaccessible if the PROFINET connection gets lost.
It is strongly recommended to change these settings only via the web interface.

For further information see the assembly and operating manual of the gripper.

2.3 Modbus/TCP

Although the Modbus/TCP interface is different to the PROFIBUS and PROFINET Interfaces, it uses a similar I/O register space layout as the PROFIBUS and PROFINET interfaces. The main difference is that in Modbus/TCP a register consists of two bytes, starting with the lower byte (Little Endian). So the registers are mapped to Modbus registers in the following way:

Modbus register	0								1							
Byte number	0								1							
Bit number	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

This example starts the MOVE Command and also sets the input user flag IF6 to 1:

Modbus register 0	$0000\ 0001\ 0010\ 0000_2 = 288_{10}$															
Register name	CMDFLAGS								IF							
Byte	$0000\ 0001_2 = 1_{10}$								$0010\ 0000_2 = 32_{10}$							
Bits	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1

All binary output flags (Command Flags and User Flags) can also be addressed directly as Modbus Coils and all binary input flags (Gripper State, User Flags and System State) can be accessed as Modbus Discrete Inputs. The order of the flags is then in the natural order of the bits (starting with Byte 0 and Bit 0).

2.3.1 Configuration

To use the Modbus/TCP interface, it must first be enabled via the device's web interface (*Settings -> Command Interface*). The IP address of the interface is the same as the address of the device. The port is set to the Modbus/TCP standard port 502 and can't be changed. For further information see the assembly and operating manual of the gripper.

3 Interface Description

The fieldbus interface is implemented as an 8-Byte output and a 12-Byte input register space.

3.1 Output Registers (PLC to WSG)

The output registers are transferred from the PROFIBUS Master, PROFINET Controller (e.g. PLC) or Modbus/TCP Master to the gripper. They consist of command flags, user flags and three parameters and are used to control the gripper. Due to the register-space-oriented nature of PROFIBUS, PROFINET and Modbus/TCP, only a subset of the grippers command set is available via this interface. The register arrangement is given in following table.

For Modbus/TCP, the Command Flags and User Flags are also available as Modbus Coils. For addressing of the Modbus Registers and Coils see (☞ 2.3, Page 7).

Structure output register

<table>
<tr><th>Byte number</th><th>Modbus Holding Register</th><th>Register name</th><th colspan="3">Description</th></tr>
<tr><td rowspan="12">0</td><td rowspan="12">0</td><td rowspan="12">CMDFLAGS</td><td colspan="3">Command Flags
A command is issued when changing the corresponding bit from 0 to 1 (raising edge). Further information to the commands (☞ 4, Page 17).</td></tr>
<tr><td>Bit Index</td><td>Name</td><td>Description</td></tr>
<tr><td>Bit 0:</td><td>MOVE</td><td>Initiate a pre-position movement</td></tr>
<tr><td>Bit 1:</td><td>GRIP</td><td>Grip a part</td></tr>
<tr><td>Bit 2:</td><td>RELEASE</td><td>Release a part</td></tr>
<tr><td>Bit 3:</td><td>HOMING</td><td>Home the gripper</td></tr>
<tr><td>Bit 4:</td><td>STOP/ACK</td><td>Stop, but do not turn off the motor / Acknowledge a FAST STOP</td></tr>
<tr><td>Bit 5:</td><td>FASTSTOP</td><td>Stop and turn off the motor.</td></tr>
<tr><td>Bit 6:</td><td>JOG+</td><td>Jog-Mode in positive direction</td></tr>
<tr><td>Bit 7:</td><td>JOG-</td><td>Jog-Mode in negative direction</td></tr>
<tr><td colspan="3">NOTICE! If the FAST STOP or STOP/ACK bit is set to ‘1’, the motion commands are disabled.</td></tr>
</table>

<table>
<tr><th>Byte number</th><th>Modbus Holding Register</th><th>Register name</th><th>Description</th></tr>
<tr><td>1</td><td></td><td>IF</td><td>User Flags (input)
Free programmable flags that can be used in conjunction with the Script Interpreter.
<table>
<tr><td>Bit Index</td><td>Name</td><td>Description</td></tr>
<tr><td>Bit 0:</td><td>IF1</td><td>Input User Flag 1</td></tr>
<tr><td>Bit 1:</td><td>IF2</td><td>Input User Flag 2</td></tr>
<tr><td>Bit 2:</td><td>IF3</td><td>Input User Flag 3</td></tr>
<tr><td>Bit 3:</td><td>IF4</td><td>Input User Flag 4</td></tr>
<tr><td>Bit 4:</td><td>IF5</td><td>Input User Flag 5</td></tr>
<tr><td>Bit 5:</td><td>IF6</td><td>Input User Flag 6</td></tr>
<tr><td>Bit 6:</td><td>IF7</td><td>Input User Flag 7</td></tr>
<tr><td>Bit 7:</td><td>IF8</td><td>Input User Flag 8</td></tr>
</table></td></tr>
<tr><td>2..3</td><td>1</td><td>WIDTH</td><td>Command parameter “Width”
New finger opening width in 1/100 millimeters (i.e. a value of 1220 means 12.20 mm).
Encoded as INT (signed).</td></tr>
<tr><td>4..5</td><td>2</td><td>SPEED</td><td>Command parameter “Speed”
Current movement speed in 1/100 millimeters per second (i.e. a value of 3005 means 30.05 mm/s), given as finger speed relative to each other.
Encoded as WORD (unsigned).
NOTICE! Setting this parameter to a value beyond the system limits and triggering a motion-related function using it raises a FAST STOP.</td></tr>
</table>

Byte number	Modbus Holding Register	Register name	Description
6..7	3	FORCELIMIT	Command parameter “Force Limit” New gripping force limit in 1/100 Newton (i.e. a value of 1050 means 10.50 N). Exception: Due to the higher force range for the WSG 70 the value is in 1/10 Newton (i.e. a value of 3050 means 305.0 N). The gripping force is twice the nominal force that is applied to the part to be gripped. Encoded as INT (signed), only positive values are allowed. **NOTICE! Setting this parameter to a value beyond the system limits and triggering a motion-related function using it raises a FAST STOP.**

To initiate a command, the command parameters have to be set up and the respective command flag has to be changed from 0 to 1 (i.e. a raising transition). Jog Mode flags are level-sensitive.

A detailed description of the specific commands can be found in (☞ 3.2, Page 12).

NOTE

If more than one command flag was changed simultaneously, only the command with the lowest bit number is executed. I.e. setting both MOVE and GRIP flags from 0 to 1 will result in a MOVE command.

NOTICE

Changing parameters while fingers are moving (i.e. MOVING in the system flags is 1) will result in a FAST STOP."1".

3.2 Input Registers (WSG to PLC)

The input register space (see the following table) is transferred from the gripper to the PROFIBUS Master, PROFINET Controller or Modbus/TCP Master each cycle. It contains the current gripper parameters, its operating state, the gripper state, user defined flags as well as a status code representing the result of the last command. The arrangement of the register see in the following table.

For Modbus/TCP the gripper state, the user flags and the system state are also available as Modbus Discrete Inputs. For addressing of the Modbus Registers and Discrete Inputs see (☞ 2.3, Page 7).

Structure input register

Byte number	Modbus Input Register	Register name	Description		
0	0	GSTATE	Gripping State These flags encode the current gripper state as below and are intended to control and monitor the gripping process:		
			Bit Index	Name	Description
			Bit 0:	IDLE	Waiting for new command
			Bit 1:	GRIPPING	Fingers moving towards the part
			Bit 2:	NO_PART	No part found
			Bit 3:	PART_LOST	Part was gripped but then lost
			Bit 4:	HOLDING	Holding a part
			Bit 5:	RELEASING	Fingers moving away from the part
			Bit 6:	POSITIONING	Fingers moving due to a pre-position command (MOVE)
			Bit 7:	ERROR	An error occurred
1	0	OF	User Flags (output) Freely programmable flags that can be used to interact between the PLC code and a running WSG Script.		
			Bit Index	Name	Description
			Bit 0:	OF1	Output User Flag 1
			Bit 1:	OF2	Output User Flag 2

<table>
<tr><th>Byte num-ber</th><th>Modbus Input Register</th><th>Register name</th><th colspan="3">Description</th></tr>
<tr><td></td><td></td><td></td><td>Bit 2:</td><td>OF3</td><td>Output User Flag 3</td></tr>
<tr><td></td><td></td><td></td><td>Bit 3:</td><td>OF4</td><td>Output User Flag 4</td></tr>
<tr><td></td><td></td><td></td><td>Bit 4:</td><td>OF5</td><td>Output User Flag 5</td></tr>
<tr><td></td><td></td><td></td><td>Bit 5:</td><td>OF6</td><td>Output User Flag 6</td></tr>
<tr><td></td><td></td><td></td><td>Bit 6:</td><td>OF7</td><td>Output User Flag 7</td></tr>
<tr><td></td><td></td><td></td><td>Bit 7:</td><td>OF8</td><td>Output User Flag 8</td></tr>
<tr><td>2..5</td><td>1..2</td><td>SYSSTATE</td><td colspan="3">System State
Current system state of the gripper encoded as a bit vector, (☞ 5.2, Page 29). This register is updated every bus cycle with the system state flags regardless of the currently processed command.
NOTICE! The system state flags should not be used to control the gripping process. Use the gripper state flags instead.</td></tr>
<tr><td></td><td></td><td></td><td>Bit Index</td><td colspan="2">Name</td></tr>
<tr><td></td><td></td><td></td><td>Bit 0</td><td colspan="2">REFERENCED</td></tr>
<tr><td></td><td></td><td></td><td>Bit 1</td><td colspan="2">MOVING</td></tr>
<tr><td></td><td></td><td></td><td>Bit 2</td><td colspan="2">BLOCKED_MINUS</td></tr>
<tr><td></td><td></td><td></td><td>Bit 3</td><td colspan="2">BLOCKED_PLUS</td></tr>
<tr><td></td><td></td><td></td><td>Bit 4</td><td colspan="2">SOFT_LIMIT_MINUS</td></tr>
<tr><td></td><td></td><td></td><td>Bit 5</td><td colspan="2">SOFT_LIMIT_PLUS</td></tr>
<tr><td></td><td></td><td></td><td>Bit 6</td><td colspan="2">AXIS_STOPPED</td></tr>
<tr><td></td><td></td><td></td><td>Bit 7</td><td colspan="2">TARGET_POS_REACHED</td></tr>
<tr><td></td><td></td><td></td><td>Bit 8</td><td colspan="2">OVERDRIVE_MODE[1]</td></tr>
<tr><td></td><td></td><td></td><td>Bit 9</td><td colspan="2">FORCECNTL_MODE</td></tr>
<tr><td></td><td></td><td></td><td>Bit 10</td><td colspan="2">reserved</td></tr>
<tr><td></td><td></td><td></td><td>Bit 11</td><td colspan="2">reserved</td></tr>
<tr><td></td><td></td><td></td><td>Bit 12</td><td colspan="2">FAST_STOP</td></tr>
<tr><td></td><td></td><td></td><td>Bit 13</td><td colspan="2">TEMP_WARNING</td></tr>
<tr><td></td><td></td><td></td><td>Bit 14</td><td colspan="2">TEMP_FAULT</td></tr>
<tr><td></td><td></td><td></td><td>Bit 15</td><td colspan="2">POWER_FAULT</td></tr>
<tr><td></td><td></td><td></td><td>Bit 16</td><td colspan="2">CURR_FAULT</td></tr>
<tr><td></td><td></td><td></td><td>Bit 17</td><td colspan="2">FINGER_FAULT</td></tr>
</table>

Byte num-ber	Modbus Input Register	Register name	Description	
			Bit 18	CMD_FAILURE
			Bit 19	SCRIPT_RUNNING
			Bit 20	SCRIPT_FAILURE
			Bit 21	reserved
			Bit 22	reserved
			Bit 23	reserved
			Bit 24	reserved
			Bit 25	reserved
			Bit 26	reserved
			Bit 27	reserved
			Bit 28	reserved
			Bit 29	reserved
			Bit 30	reserved
			Bit 31	reserved
6..7	3	WIDTH	Current Opening Width Current opening width of the fingers in 1/100 millimeters (i.e. a value of 1220 means 12.2 mm). Encoded as INT. This register is updated every bus cycle with the current opening width regardless of the currently processed command.	

Byte number	Modbus Input Register	Register name	Description
8..9	4	GRIPPING FORCE	Current Gripping Force Current gripping force in 1/100 Newton (i.e. a value of 405 means a gripping force of 40.5 N).. Exception: Due to the higher force range for the WSG 70 the value is in 1/10 Newton (i.e. a value of 3050 means 305.0 N). This is twice the nominal force that is currently applied to a part. Encoded as INT. This register is updated every bus cycle with the current gripping force regardless of the currently processed command. **NOTICE! If no force measurement finger is installed on the gripper, this value is approximated using the motor current.**
10..11	5	STATUS CODE	Result of the last Command This field holds its state, until a new command is issued.(☞ 5.1, Page 27).
[1]) Overdrive mode is not supported by all WSG grippers. For further information see the assembly and operating manual of the gripper.			

3.3 Diagnosis Message PROFIBUS

The gripper will send diagnosis messages containing the current system state flags as the first double word to the PROFIBUS Master (PLC) if at least one of the following error-related flags of its system state was raised (i.e. changes from 0 to 1):

- SF_SOFT_LIMIT_MINUS
- SF_SOFT_LIMIT_PLUS
- SF_FAST_STOP
- SF_TEMP_FAULT
- SF_POWER_FAULT
- SF_CURR_FAULT
- SF_FINGER_FAULT
- SF_CMD_FAILURE
- SF_SCRIPT_FAILURE

For a detailed description see (☞ 5.2, Page 29).

Format of the diagnosis messages

Byte number	Description
0..3	**Standard Diagnostic Data** Diagnostic Data as defined by the PROFIBUS Specification
4..5	**Slave Ident-No.** Slave Identification Number. This is 0x5555 for the gripper.
6	**Length of Diagnostic Message** Diagnosis messages of the gripper are always 10 Bytes = 0x0A.
7..10	**System State** Current system state of the gripper encoded as a bit vector. Same coding as SYSSTATE in table (☞ 3.2, Page 12).
10..15	**reserved** This area is reserved for future use.

4 Commands

4.1 MOVE – Move the fingers in positioning mode

This command can be used to position the gripper fingers to a defined width before issuing a grip command. The command is intended to speed up gripping of sensitive parts when the gripper fingers have to travel a larger distance due to process constraints. MOVE can only be issued if the gripper is idle, i.e. gripper state is IDLE.

Command Flag Position

Bit 0

Parameters used

WIDTH, SPEED

Status Code

The status code register is set to E_CMD_PENDING upon start of the movement and set to the command's result when it has finished.

Gripper State

The gripper state changes to POSITIONING when starting to move and back to IDLE when finished. In case of an error, the gripper state is set to ERROR.

System State

Various transitions will occur. You should use the gripper state to evaluate the current state of the gripping process, unless you have very special requirements.

4.2 GRIP – Grip a part

Grip a part using its nominal width, the speed and the force limit at which the part should be gripped. When the command is issued, the gripper moves its fingers to the nominal part width and tries to clamp the expected part with the previously set gripping force. If the gripper can establish the desired gripping force within the defined clamping travel, a part is gripped.

If the fingers fall through the clamping travel without establishing the gripping force, no part was found and the gripper state is updated accordingly. The clamping travel can be set using the grippers web interface. The gripper state is updated with the result of this operation (either HOLDING or NO_PART) as well as the gripping statistics.

If no part is found, the command returns the status code E_CMD_FAILED as result. After successfully gripping a part, the integrated part monitoring is enabled which supervises the gripping force. If a part is removed from the gripper before issuing the release command, the gripper detects it and changes the gripper state to PART_LOST.

NOTE

The impact due to the mass of the gripper fingers and the internal mechanics can be reduced by limiting the gripping speed with sensitive parts.

The gripper state reflects the current state of the gripping process. It should be checked after each command to control if the gripping process works as intended.

Command Flag Position

Bit 1

Parameters used

WIDTH, SPEED, FORCELIMIT

Status Code

The status code register is set to E_CMD_PENDING upon start of the movement and set to the command's result when it has finished. If no part was found, the status code is set to E_CMD_FAILED.

Gripper State

During finger movement, the gripper state is set to GRIPPING. If a part was found, it changes to HOLDING. If no part was found, the Gripper state is set to NO_PART. If a part was removed after it was clamped, the gripper state is set to PART_LOST. In case of an error, the gripper state is set to ERROR.

System State

Various transitions will occur. You should use the gripper state to evaluate the current state of the gripping process, unless you have very special requirements.

4.3 RELEASE – Release a part

Release a part by opening the fingers with a given speed and width. The RELEASE command does not pinch the part. This is ensured by successively increasing the internal force limit only when moving away from it. The part monitoring is disabled before releasing it. The gripper's nominal force is used for release.

Command Flag Position

Bit 3

Parameters used

WIDTH, SPEED

Status Code

The status code register is set to E_CMD_PENDING upon start of the movement and set to the command's result when it has finished.

Gripper State

During finger movement, the gripper state is set to RELEASING. When the end position is reached, the gripper state is set to IDLE. In case of an error, the gripper state is set to ERROR.

System State

Various transitions will occur. You should use the gripper state to evaluate the current state of the gripping process, unless you have very special requirements.

4.4 HOMING – Referencing the gripper

This command references the gripper by executing a homing sequence. During homing, the fingers will move to the mechanical end stop. The homing sequence has to be configured on the *Settings -> Motion Configuration* page of the WSG's web interface. You can set the direction of homing (inbound or outbound) as well as enable automatic homing on startup.

NOTE
Homing is required prior to any motion-related command. The best positioning performance will be achieved if homing is done into the direction in which the better positioning accuracy is required.

NOTE
During homing, soft limits are disabled. Obstacles in the movement range of the fingers and collisions with these during homing may result in a wrong reference point for the finger position!

Command Flag Position

Bit 3

Parameters used

none

Status Code

The status code register is immediately set to E_CMD_PENDING and to the command's result when it has finished.

Gripper State

During homing, the gripper state is POSITIONING.

System State

During movement, the MOVING flag is set to 1. If the gripper is referenced, the REFERENCED flag is set to 1.

4.5 STOP/ACK – Stop movement or acknowledge a FAST STOP

Stop any pending movement immediately without disabling the drive. When stopping during holding (i.e. the gripper state is HOLDING), the part monitor will be disabled and the gripping force will not be applied anymore.

Acknowledging a FAST STOP condition:
If the gripper is in FAST STOP mode, a transition from 0 to 1 is required on this flag to acknowledge and to return in normal operating mode. Reset the FAST STOP before acknowledging.

Command Flag Position

Bit 4

Parameters used

none

Status Code

Set to E_SUCCESS.

Gripper State

The gripper state is set to IDLE.

System State

The AXIS_STOPPED flag is set to 1. If acknowledging a FAST STOP, the FAST STOP flag is cleared.

4.6 FAST STOP – Raise a Fast Stop

This function is similar to an “Emergency Stop”. It immediately stops any movement the fastest way, disables the drive and prevents further motion-related commands from being executed. The FAST STOP state can only be left by issuing a FAST STOP Acknowledge (☞ 4.5, Page 22). 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 system’s log file.

NOTE
This command should in general be used to react on certain error conditions. To simply stop the current movement, you may want to use the STOP command instead.

NOTE
In addition to the STOP/ACK flag, the FAST STOP can be cleared interactively using the web interface, too. This will enable the drive again; however, it is required to reset the FAST STOP flag on the PROFIBUS interface to enable motion-related commands again.

Command Flag Position

Bit 5

Parameters used

none

Status Code

Set to E_SUCCESS.

Gripper State

The gripper state is set to IDLE.

System State

The FAST STOP flag is set to 1.

4.7 JOG+ and JOG- – Jog Mode

To set up a process, it may be required to move the fingers manually. This can be done using the Jog Mode Flags. These flags are evaluated level-sensitive and allow a constant speed drive of the fingers using two switches on the PLC. The flags are decoded as given in the following table:

Interpretation of the flags

JOG+	JOG-	Movement direction
0	0	Jog Mode is disabled*
1	0	positive with SPEED
0	1	negative with SPEED
1	1	Stop
*) If the Jog flags change to both 0, the Jog mode is left and the drive is stopped.		

The force limit (current controlled only) as well as the speed can be passed as parameters. You may consider using a hand wheel to control them. Be aware that high movement speed may interfere with a low force limit setting.

NOTE

In contrast to other motion-related commands, the SPEED Parameter can be set to 0 resulting in an internal clamping of the value to the minimum gripper speed.

NOTE

The Jog Mode is intended only to set up a process. Do not use the Jog Mode in normal operation of the gripper!

Command Flag Position

Bit 6 and 7

Parameters used

SPEED, FORCELIMIT

Status Code

The status code register is set to E_CMD_PENDING upon start of the movement and set to the command's result when it has finished.

Gripper State

During finger movement, the gripper state is set to RELEASING. When the end position is reached (or in case of an error), the gripper state is set to IDLE.

System State

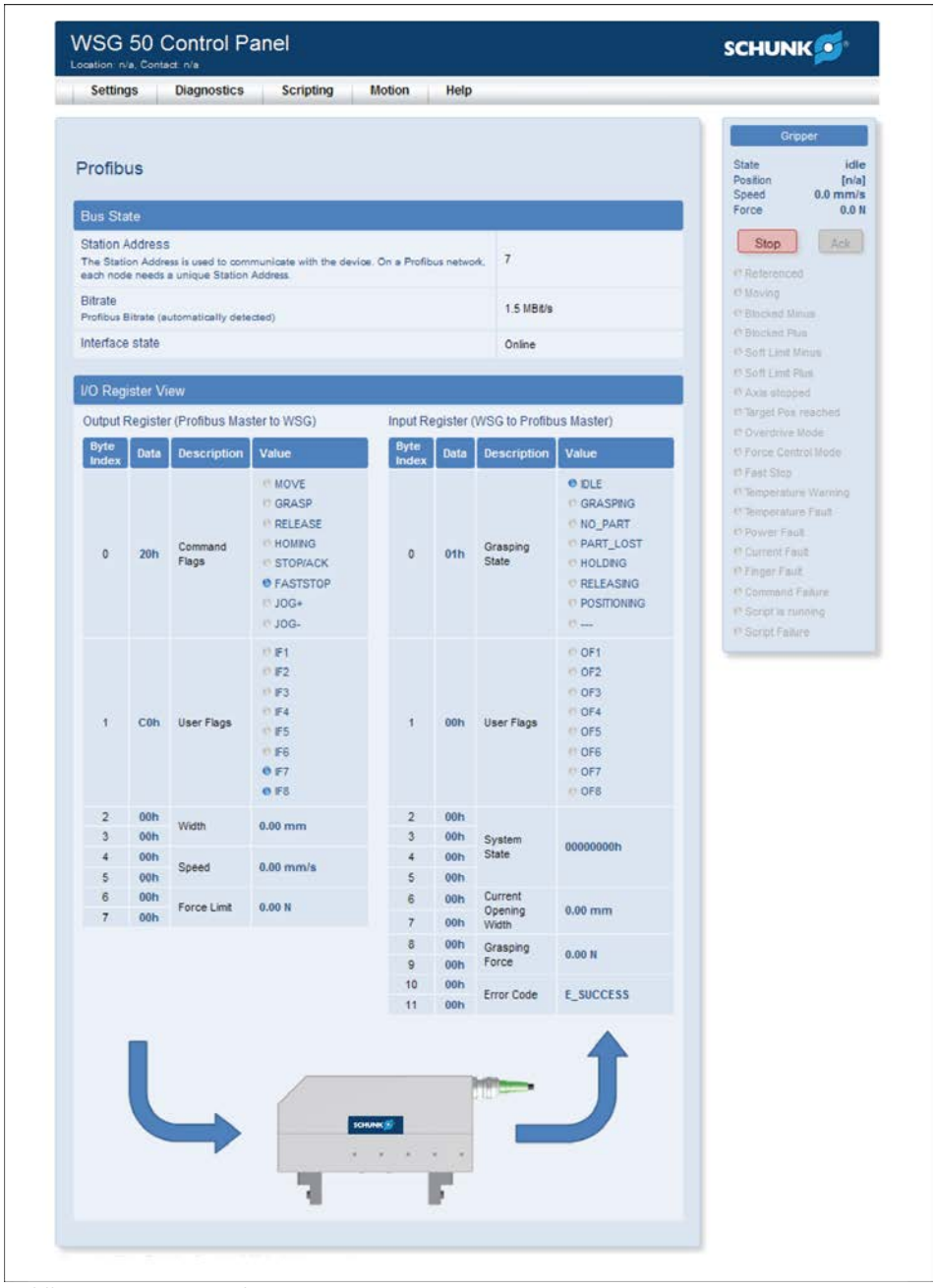
Various transitions will occur. You should use the gripper state to evaluate the current state of the gripping process, unless you have very special requirements.

5 Fieldbus monitor

The gripper has a built-in fieldbus monitor that can be accessed via the web interface *Diagnosis -> Fieldbus Monitor*.

The monitor displays the current content of the input and output registers and gives some basic information about the fieldbus state.

During the integration of the gripper into a system, the fieldbus monitor can be used to query the state of the fieldbus interface.



Fieldbus monitor, example

5.1 Appendix A: Status codes

Status 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 for executing 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 a status code, if the function has completed.
27	E_OVERRUN	Data overrun

Status Code	Symbol Name	Description
28	E_RANGE_ERROR	Range error
29	E_AXIS_BLOCKED	Axis blocked
30	E_FILE_EXISTS	File already exists

5.2 Appendix B: System State Flags

The system state flags are arranged as a 32-bit wide integer value that is provided via the PROFIBUS Input Registers.

Each bit has a special meaning listed in the table below:

Bit No.	Flag Name	Description
D31..21	reserved	These bits are currently unused.
D20	SF_SCRIPT_FAILURE	**Script Error** The flag is set, if during executing a script an error occurred and the script has been aborted. The flag is reset whenever a script is started.
D19	SF_SCRIPT_RUNNING	**A script is currently running** The flag is set, if a script is executed. The flag is reset if the script either terminated normally, a script error occurred or the script has been terminated manually by the user.
D18	SF_CMD_FAILURE	**Command Error** The flag is set, if the last command returned an error.
D17	SF_FINGER_FAULT	**Finger fault** The flag is set, if the status of at least one finger is different from OPERATING and NOT CONNECTED. Check the finger flags for a more detailed error description.
D16	SF_CURR_FAULT	**Engine current fault** The flag is set, if the engine has reached its maximum thermal power consumption. At the same time a FAST STOP is set. The flag will be reset automatically as soon as the engine has recovered. The FAST STOP has to be acknowledged.
D15	SF_POWER_FAULT	**Power Error** The flag is set, if the power supply is outside of the valid range. The power supply must be checked and if necessary adjusted.

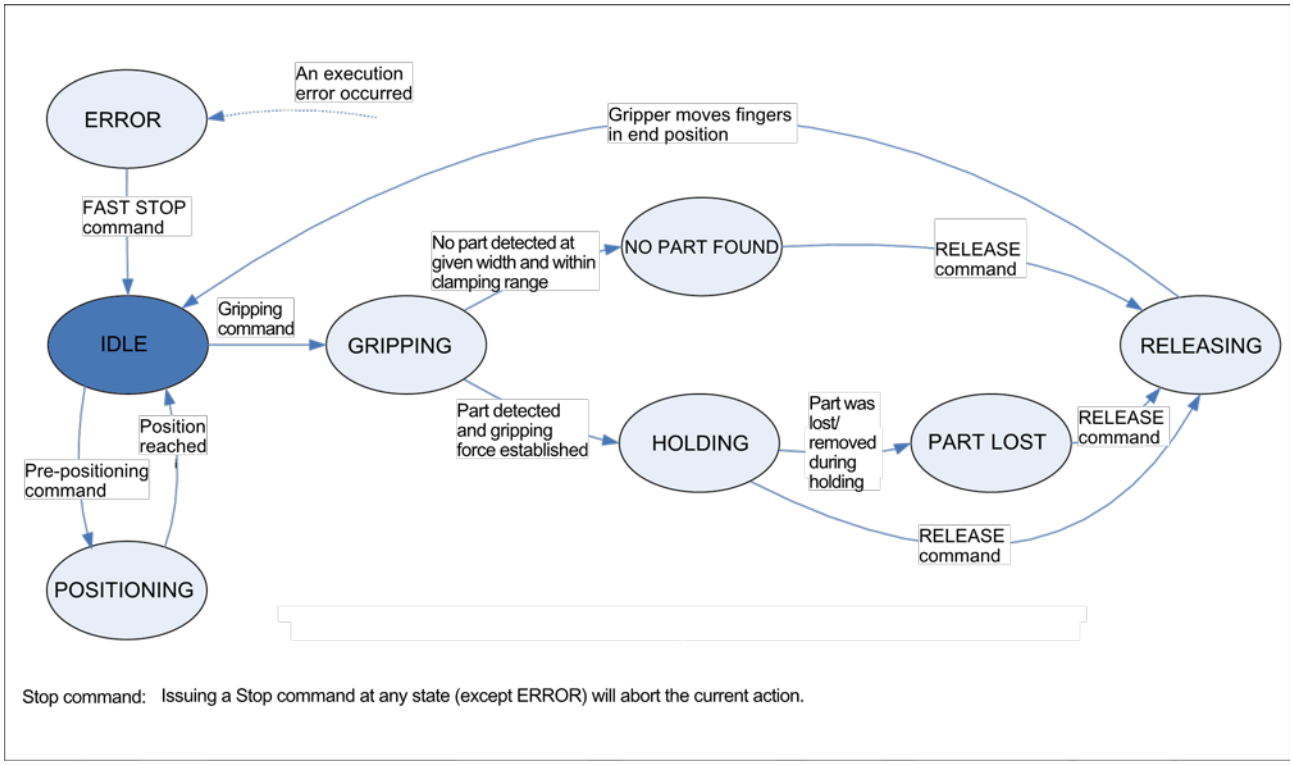
Bit No.	Flag Name	Description
D14	SF_TEMP_FAULT	**Temperature Error** The flag is set, if the gripper has reached a critical temperature level. All motion-related commands are disabled. The flag is reset, when the temperature has fallen under the critical level.
D13	SF_TEMP_WARNING	**Temperature warning** The flag is set, if the gripper will soon reach a critical temperature level.
D12	SF_FAST_STOP	**Fast Stop** The flag is set, if the gripper has been stopped due to an error condition. The drive will shorted and the execution of all motion-related commands stopped. The flag is reset, when the error was acknowledged. After acknowledging all motion-related commands will be activated again.
D11..10	reserved	These bits are currently unused.
D9	SF_FORCECNTL_MODE	**Force Control Mode** The flag is set, if the force control is currently enabled by using the installed force measurement finger (WSG-FMF). If the flag is not set, the gripping force is controlled by approximation based on the motor current.
D8	SF_OVERDRIVE_MODE	**Overdrive Mode**[1] The flag is set, if the gripper is in overdrive mode. The gripping force can be set to a value up to the overdrive force limit. If this flag is not set, the gripping force cannot be higher than the grippers nominal gripping force value. [1]) The overdrive mode is not supported by all grippers. For further information see the assembly and operating manual fo the gripper.
D7	SF_TARGET_POS_REACHED	**Target position reached** The flag is set, if the target position was reached.

Bit No.	Flag Name	Description
		The flag is not synchronized with SF_MOVING, so that a delay between the reset of SF_MOVING and the set of SF_TARGET_POS_REACHED can occur.
D6	SF_AXIS_STOPPED	**Axis stopped** The flag is set, if a motion-related command has been stopped due to a STOP/ACK command. The flag is reset due to the next motion command.
D5	SF_SOFT_LIMIT_PLUS	**Positive direction soft limit reached** The flag is set, if the fingers have reached the defined soft limits in positive moving direction. A further movement into this direction is not possible. The flag is reset, if the fingers have moved away from the soft limit position.
D4	SF_SOFT_LIMIT_MINUS	**Negative direction soft limit reached** The flag is set, if the fingers have reached the defined soft limits in negative moving direction. A further movement into this direction is not possible. The flag is reset, if the fingers have moved away from the soft limit position.
D3	SF_BLOCKED_PLUS	**Axis is blocked in positive moving direction** The flag is 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.

Bit No.	Flag Name	Description
D2	SF_BLOCKED_MINUS	**Axis is blocked in negative moving direction** The flag is set, if the axis is blocked in negative moving direction. The flag is reset if either the blocking condition is resolved or a stop command is issued.
D1	SF_MOVING	**The Fingers are currently moving** The flag is set, if a movement is stared (e.g. MOVE command). The flag is reset, when the movement stops.
D0	SF_REFERENCED	**Fingers Referenced** If this flag is set, the gripper is referenced and accepts motion-related commands.

5.3 Appendix C: Gripper states

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



5.4 Appendix D: Demo program

The gripper is provided with several simple demo projects for Siemens SIMATIC S7-1200 controls and STEP 7 TIA-Portal v12 (and higher). The projects can be downloaded from the CD or the web interface.

The projects have been implemented and tested on a CPU of type 1212C with the PROFIBUS module CM1243-5 using the Siemens STEP7 Basic v12.0 (TIA Portal) project environment. They will execute an endless loop of a simple gripping cycle, consisting of pre-positioning the gripper jaws (MOVE), gripping a part (GRIP), releasing (RELEASE) and returning to the start position (MOVE).

If a part is detected, the gripper will hold it for a short moment. In case of an error, the gripper will execute a homing sequence and restart from the beginning. The gripper must be referenced before running the program.

The PLC is configured in the project to use IP address 192.168.1.250 and PROFIBUS address 2. The gripper is expected to use PROFIBUS address 7 (default).
By using the PROFINET demo project it is expected, that the gripper is configured to the IP address 192.168.1.20 (delivery state).

NOTE
It might be useful to open the Fieldbus Monitor on the gripper´s web interface when running the program to get more information on possible problems.

NOTE
The demo project is intended for testing purposes only. Do not use it in any production environment.

SCL Source Code Listing for the gripping cycle used in the demo projects

```
1    //////////////////////////////////////
2    // Receive data type            //
3    //////////////////////////////////////
4
5    TYPE "WSG_RECEIVE"
6    VERSION : 0.1
7      STRUCT
8          STW1 : Struct
9                IDLE : Bool; 9 GRASPING : Bool;
10               GRASPING : Bool;
11               NO_PART : Bool;
12               PART_LOST : Bool;
13               HOLDING : Bool;
14               RELEASING : Bool;
15               POSITIONING : Bool;
16               ERROR : Bool;
17               OF1 : Bool;
18               OF2 : Bool;
19               OF3 : Bool;
20               OF4 : Bool;
21               OF5 : Bool;
22               OF6 : Bool;
23               OF7 : Bool;
24               OF8 : Bool;
25         END_STRUCT;
26         SYSSTATE : Struct
27               REFERENCED : Bool;
28               MOVING : Bool;
29               BLOCKED_MINUS : Bool;
30               BLOCKED_PLUS : Bool;
31               SOFT_LIMIT_MINUS : Bool;
32               SOFT_LIMIT_PLUS : Bool;
33               AXIS_STOPPED : Bool;
34               TARGET_POS_REACHED : Bool;
35               OVERDRIVE_MODE : Bool;
36               FORCECNTL_MODE : Bool;
37               RES10 : Bool;
38               RES11 : Bool;
39               FAST_STOP : Bool;
40               TEMP_WARNING : Bool;
41               TEMP_FAULT : Bool;
```

```
42              POWER_FAULT : Bool;
43              CURR_FAULT : Bool;
44              FINGER_FAULT : Bool;
45              CMD_FAILURE : Bool;
46              SCRIPT_RUN : Bool;
47              SCRIPT_FAILURE : Bool;
48              RES21 : Bool;
49              RES22 : Bool;
50              RES23 : Bool;
51              RES24 : Bool;
52              RES25 : Bool;
53              RES26 : Bool;
54              RES27 : Bool;
55              RES28 : Bool;
56              RES29 : Bool;
57              RES30 : Bool;
58              RES31 : Bool;
59          END_STRUCT;
60          WIDTH : Int;
61          FORCE : UInt;
62          ERROR_CODE : UInt;
63      END_STRUCT;
64
65  END_TYPE
66
67
68  ////////////////////////////////////
69  // Send data type                 //
70  ////////////////////////////////////
71
72  TYPE "WSG_SEND"
73  VERSION : 0.1
74      STRUCT
75          STW1 : Struct
76              MOVE : Bool;
77              GRASP : Bool;
78              RELEASE : Bool;
79              HOMING : Bool;
80              STOP_ACK : Bool;
81              FASTSTOP : Bool;
82              JOG_PLUS : Bool;
83              JOG_MINUS : Bool;
84              IF1 : Bool;
85              IF2 : Bool;
```

```
86              IF3 : Bool;
87              IF4 : Bool;
88              IF5 : Bool;
89              IF6 : Bool;
90              IF7 : Bool;
91              IF8 : Bool;
92          END_STRUCT;
93          WIDTH : Int;
94          SPEED : UInt;
95          FORCELIMIT : UInt;
96      END_STRUCT;
97
98    END_TYPE
100
101   //////////////////////////////////////
102   // Gripping Cycle FB             //
103   //////////////////////////////////////
104
105   FUNCTION_BLOCK "GrippingCycle"
106   { S7_Optimized_Access := 'TRUE' }
107   VERSION : 0.1
108   VAR DB_SPECIFIC
109     dp_data_in { S7_HMI_Visible := 'False'} : Array [1..12] of Byte;
110     receive { S7_HMI_Accessible := 'False'; S7_HMI_Visible :=
'False'} AT
111   dp_data_in : "WSG_RECEIVE";
112     dp_data_out { S7_HMI_Visible := 'False'} : Array [1..8] of Byte;
113     send { S7_HMI_Accessible := 'False'; S7_HMI_Visible := 'False'}
AT
114   dp_data_out : "WSG_SEND";
115   END_VAR
116   VAR
117         holding_active : Bool;
118         state : Int := -2;
119         timer_expired : Bool;
120         CycleConfig : Struct
121           PreposWidth : Int := 3000;
122           PreposSpeed : UInt := 40000;
123           PreposForce : UInt := 8000;
124           GraspWidth : Int := 2200;
125           GraspSpeed : UInt := 40000;
126           GraspForce : UInt := 5000; ;
127           HoldingTime : Time := T#1000ms
128           ReleaseWidth : Int := 3000;
```

```
129           ReleaseSpeed : UInt := 40000;
130           ReleaseForce : UInt := 8000;
131           StartWidth : Int := 6000;
132           StartSpeed : UInt := 40000;
133           StartForce : UInt := 8000;
134           CycleFinished : Bool := true;
135           ErrorCount : UInt := 0;
136           TimerExpired : Bool;
137        END_STRUCT;
138  END_VAR
139
140  VAR_TEMP
141        ret_val : Int;
142        do_next_step : Bool;
143  END_VAR
144
145


146  BEGIN
147  // Implementation of state machine
148
149  // Call receive function block
150  // Note: The address parameter comes from the Profibus module.
151  // Check default tag table, system constants tab, Profibus inter-
face and
152  // convert the decimal address listed there to hex and enter it
here.
153  #ret_val := DPRD_DAT( LADDR := W#16#113, RECORD => #dp_data_in );
154
155  // Initial values
156  #do_next_step := false;
157
158  // State transitions
159  CASE #state OF
160
161        // NOTE: All states <= 0 belong to ERROR HANDLING!
162
163  // Step -1 (error state)
164        -1:
165        // Reset all control flags TO get a defined #state
166              #CycleConfig.ErrorCount := #CycleConfig.ErrorCount + 1;
167              #send.STW1.FASTSTOP := false;
168              #send.STW1.GRASP := false;
169              #send.STW1.HOMING := false;
```

```
170             #send.STW1.JOG_MINUS := false;
171             #send.STW1.JOG_PLUS := false;
172             #send.STW1.MOVE := false;
173             #send.STW1.RELEASE := false;
174
175   // Set STOP/ACK flag to true to resolve error condition
176             #send.STW1.STOP_ACK := true;
177
178             // Go to next step
179             #do_next_step := true;
180
181   // Step 0 (initial start): Execute homing sequence
182         0:
183         // Error handling. State *must* be IDLE at this point.
184               IF #receive.STW1.IDLE = false THEN
185                     #state := -1;
186               END_IF;
187
188
189         // Reset STOP/ACK flag and set HOMING command flag
190               IF #receive.STW1.IDLE = true THEN
191                     #send.STW1.STOP_ACK := false;
192                     #send.STW1.HOMING := true;
193                     #do_next_step := true;
194               END_IF;
195
196   // Step 1: Check if HOMING is running
197         1:
198               // Error handling
199               IF #receive.STW1.ERROR = true THEN
200                     #state := -1;
201               END_IF;
202
203               // Check for gripper state set to POSITIONING
204               IF #receive.STW1.POSITIONING = true THEN
205                     #send.STW1.HOMING := false;
206                     #do_next_step := true;
207               END_IF;
208
209   // Step 2: Wait for gripper state to become IDLE
210   // and check if gripper is referenced
211         2:
212               IF #receive.STW1.ERROR = true THEN
213                     #state := -1;
```

```
214             END_IF;
215
 216        IF #receive.STW1.IDLE = true THEN
217             IF #receive.SYSSTATE.REFERENCED = false THEN
218                  #state := -1;
219             ELSE
220                  #do_next_step := true;
221                  END_IF;
222             END_IF;
223
224   // Step 3: When idle, move to pre-position width
225        3:
226             // Error handling
227             IF #receive.STW1.ERROR = true THEN
228                  #state := -1;
229             END_IF;
230
231        // Trigger move command to pre-position the gripper jaws
232             IF #receive.STW1.IDLE = true THEN
233                  #send.WIDTH := #CycleConfig.PreposWidth;
234                  #send.SPEED := #CycleConfig.PreposSpeed;
235                  #send.FORCELIMIT := #CycleConfig.PreposForce;
236                  #send.STW1.MOVE := true;
237                  #do_next_step := true;
238             END_IF;
239
240   // Step 4: Check if gripper state is set to POSITIONING, i.e.
gripper is
241   moving
242        4
243             // Error handling
244             IF #receive.STW1.ERROR = true THEN
245                  #state := -1;
246             END_IF;
247
248             // Reset move command flag
249             IF #receive.STW1.POSITIONING = true THEN
250                  #send.STW1.MOVE := false;
251                  #do_next_step := true;
252             END_IF;
253
254   // Step 5: When idle, start grasping
255        5:
256             IF #receive.STW1.ERROR = true THEN
```

```
257                 #state := -1;
258             END_IF;
259
260             IF #receive.STW1.IDLE = true THEN
261                 #send.WIDTH := #CycleConfig.GraspWidth;
262                 #send.SPEED := #CycleConfig.GraspSpeed;
263                 #send.FORCELIMIT := #CycleConfig.GraspForce;
264                 #send.STW1.GRASP := true;
265                 #do_next_step := true;
266                 END_IF;
267
268   // Step 6: When grasping is active, go to next step
269         6:
270             IF #receive.STW1.ERROR = true THEH
271                 #state := -1;
272                 END_IF;
273
274             #IF receive.STW1.GRASPING = true THEN
275                 #send.STW1.GRASP := false;
276                 #do_next_step := true;
277             END_IF;
278
279   // Step 7: When holding, wait. If no part found/part lost, release
immedi-
280   ately.
281         7:
282             IF #receive.STW1.ERROR = true THEN
283                 #state := -1;
284             END_IF;
285
286             // Part found. Hold for some time, then go to next step.
287             IF #receive.STW1.HOLDING = true AND #holding_active =
false THEN
288                 #holding_active := true;
289                 #ret_val := SRT_DINT( OB_NR := 20, DTIME := #Cy-
cleConfig.HoldingTime,
291   SIGN := 1 );
290             END_IF;
292
293             // No part found or part lost. Go to next step.
294             IF #timer_expired = true OR #receive.STW1.NO_PART = true
OR #re-
295   ceive.STW1.PART_LOST = true THEN
296                 #holding_active := false;
```

```
297                 #timer_expired := false;
298                 #send.WIDTH := #CycleConfig.ReleaseWidth;
299                 #send.SPEED := #CycleConfig.ReleaseSpeed;
300                 #send.FORCELIMIT := #CycleConfig.ReleaseForce;
301                 #send.STW1.RELEASE := true;
302                 #do_next_step := true;
303           END_IF;
304
305 // Step 8: When releasing is active, go to next step
306       8:
307           IF #receive.STW1.ERROR = true THEN
308                 #state := -1;
309           END_IF;
310
311           IF #receive.STW1.RELEASING = true THEN
312                 #send.STW1.RELEASE := false;
313                 #do_next_step := true;
314           END_IF;
315
316 // Step 9: When idle, move to start position
317       9:
318           IF #receive.STW1.ERROR = true THEN
319                 #state := -1;
320           END_IF;
321
322           IF #receive.STW1.IDLE = true THEN
323                 #send.WIDTH := #CycleConfig.StartWidth;
324                 #send.SPEED := #CycleConfig.StartSpeed;
325                 #send.FORCELIMIT := #CycleConfig.StartForce;
326                 #send.STW1.MOVE := true;
327                 #do_next_step := true;
328           END_IF;
329
330 // Step 10: When positioning is active, go to next step
331       10:
332           IF #receive.STW1.ERROR = true THEN
333                 #state := -1;
334           END_IF;
335
336           IF #receive.STW1.POSITIONING = true THEN
337                 #send.STW1.MOVE := false;
338                 #do_next_step := true;
339           END_IF;
340
```

```
341             // Default (state is not -1..10)
342             ELSE
343
344             // Go to error state
345             IF #receive.STW1.ERROR = true THEN
346                  #state := -1;
347             END_IF;
348
349             // Start cycle from the beginning (without homing)
350             IF #receive.STW1.IDLE = true THEN
351                  #state := 3;
352                  #CycleConfig.CycleFinished := true;
353             END_IF;
354
355        END_CASE;
356
357        // Increment state variable
358        IF #do_next_step = true THEN
359             #state := #state + 1;
360        END_IF;
361
362        // Call send function block
363        #ret_val := DPWR_DAT( LADDR := W#16#114, RECORD :=
#dp_data_out );
364
365   END_FUNCTION_BLOCK
```