WSG Serie

Dokumentation der Feldbusschnittstelle

Firmware Version 4.0

Dezember 2016

PROFIBUS® PROFINET® **Modbus/TCP**

www.weiss-robotics.com

Inhalt

1 Einleitung

Die WSG-Greiferfamilie bietet Schnittstellen für PROFIBUS DP V0 und/oder PROFINET, abhängig vom Gerätetyp. PROFIBUS ist ein weit verbreitetes Feldbus-Protokoll in der industriellen Automation. Es unterstützt Einzel- und Multimastermodi. PROFINET ist eine neue Generation von Feldbus-Schnittstelle, speziell entwickelt für Echtzeitkommunikation über Standard-Ethernet-Schnittstellen.
Jedes Gerät wird durch einen E/A-Registersatz repräsentiert, welcher periodisch mit dem PROFIBUS-Master oder einem PROFINET Controller (z.B. SPS) synchronisiert wird.
Dieses Handbuch setzt Kenntnisse über PROFIBUS- und/oder PROFINET-Technologien und Siemens SIMATIC Software voraus.

PROFINET und Modbus/TCP sind optionale Features der WSG Greifmodule. Lizenzschlüssel können separate bezogen werden. Kontaktieren Sie hierfür Ihren regionalen Vertriebspartner.

Auf der Weiss Robotics Website[1] und im "Documentation"-Bereich der WSG-Weboberfläche finden Sie im Downloadbereich einfache Demoprogramme für Siemens SIMATIC S7-1200. Weitere Informationen finden Sie in Kapitel Anhang D.

[1] http://www.weiss-robotics.com

2 PROFIBUS-Schnittstelle

Jeder PROFIBUS-Slave hat einen E/A-Registersatz, welcher periodisch ausgetauscht und vom PROFIBUS-Master gelesen bzw. geschrieben wird. Der E/A-Bereich des WSG ist auf der Masterseite mit dem Geräteprofil (GSD-Datei) konfiguriert, wie es von der Produkt-CD oder unter der WSG-Weboberfläche heruntergeladen werden kann.
Die einzelnen E/A-Register sind in Kapitel 4 detailliert beschrieben.

2.1 Installation der GSD-Datei in Siemens STEP7 v11.0 (TIA) und neuer

Die GSD-Datei wird als komprimiertes ZIP-Archiv angeboten, welches folgende Dateien enthält:

- WEIS5555.gsd (Gerätebeschreibungsdatei)
- WSG_D.bmp (Bilddatei)
- WSG_R.bmp (Bilddatei)
- WSG_S.bmp (Bilddatei)
- install.txt (Installationshinweise)

Befolgen Sie folgende Schritte zur Installation der GSD Datei in Siemens STEP7 11.0:

- Entpacken Sie das Zip-Archiv auf die Festplatte
- Öffnen Sie in Siemens TIA die Projektansicht
- Wählen Sie “Extras -> Gerätebeschreibungsdatei (GSD) installieren“
- Gehen Sie zu dem Verzeichnis, in welches Sie zuvor die Dateien entpackt haben und wählen Sie die GSD-Datei aus
- Nun finden Sie den WSG im Gerätekatalog unter:
  “Weitere Feldgeräte-> PROFIBUS DP -> Drives -> Weiss Robotics GmbH & Co. KG”

2.2 Konfiguration

Um die PROFIBUS-Schnittstelle des WSG benutzen zu können, muss diese zuerst über die Weboberfläche des Gerätes (Menüpunkt „Settings“ -> „Command Interface“) eingeschaltet werden. Die PROFIBUS-Teilnehmeradresse des WSG ist auf 7 voreingestellt, kann aber ebenfalls über die Weboberfläche geändert werden. Weitere Informationen finden Sie im Benutzerhandbuch.

3 PROFINET-Schnittstelle

Die PROFINET-Schnittstelle verwendet dieselben E/A-Register wie die PROFIBUS-Schnittstelle. Wie bei PROFIBUS wird auch hierbei der E/A-Registersatz periodisch zwischen Greifmodul und PROFINET-Controller ausgetauscht und verwendet ein auf dem Controller installiertes, vordefiniertes Geräteprofil (GSDML-Datei), welches von der Produkt-CD oder der Weboberfläche des WSG heruntergeladen werden kann. Die einzelnen E/A-Register sind in Kapitel 4 detailliert beschrieben.

3.1 Installation der GSDML Datei in Siemens STEP7 v11.0 (TIA) und neuer

Die GSD-Datei wird als komprimiertes ZIP-Archiv angeboten, welches folgende Dateien enthält:

- GSDML-V2.31-Weiss Robotics-WSG-20140401.gsdml (Gerätebeschreibungsdatei)
- GSDML-02A2-0001-WSG.bmp (Bilddatei)

Befolgen Sie folgende Schritte zur Installation der GSDML Datei in Siemens STEP7 11.0:

- Entpacken Sie das Zip-Archiv auf die Festplatte
- Öffnen Sie in Siemens TIA die Projektansicht
- Wählen Sie “Extras -> Gerätebeschreibungsdatei (GSD) installieren”
- Gehen Sie zu dem Verzeichnis, in welches Sie zuvor die Dateien entpackt haben und wählen Sie die GSDML-Datei aus
- Nun finden Sie den WSG im Gerätekatalog unter: “Weitere Feldgeräte -> PROFINET IO -> I/O -> Weiss Robotics GmbH & Co. KG”

3.2 Konfiguration

Um die WSG PROFINET-Schnittstelle verwenden zu können, muss diese vorher über die Weboberfläche des Gerätes eingeschaltet werden. Weitere Einstellungen, wie das Ändern der IP-Adresse oder der PROFINET- Gerätebezeichnung, können direkt auf dem WSG über die Weboberfläche oder über die verschiedenen Konfigurationsmöglichkeiten von PROFINET mit Hilfe von Projektierungstools wie z.B. Siemens STEP7 vorgenommen werden.

Beim Ändern der IP-Adresse des WSG mit Hilfe eines Projektierungstools kann es passieren, dass die Weboberfläche des Gerätes nicht mehr ansprechbar ist, falls die PROFINET Verbindung verloren geht. Daher ist es ratsam, diese Einstellungen ausschließlich über die Weboberfläche des WSG durchzuführen.

4 Modbus/TCP Interface

Auch wenn sich das Modbus/TCP Protokoll sich von PROFIBUS und PROFINET unterscheidet, benutzt es die gleichen E/A-Register. Der Hauptunterschied ist, dass bei Modbus/TCP ein Register aus zwei Byte besteht, wobei das niedrigwertige Byte vorne steht (Little Endian). Dies bedeutet, dass die Register wie folgt auf die Modbus Register abgebildet werden:

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

Dieses Beispiel startet den MOVE Befehl und setzt das Benutzer-Flag IF6 auf 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

Alle binären Ausgabeflags (Befehlsflags und Benutzerflags) können auch als Modbus „Coils“ angesprochen werden. Genauso können alle binären Eingabeflags (Greifzustand, Benutzerflags und Status Code) als Modbus „Discrete Inputs“ adressiert werden. Die Reihenfolge folgt dabei der natürlichen Bit-Reihenfolge (beginnend bei Byte 0 und Bit 0).

4.1 Konfiguration

Um die WSG Modbus/TCP Schnittstelle zu benutzen, muss diese zuvor über die Weboberfläche des Gerätes (Menüpunkt „Settings“ -> „Command Interface“) eingeschaltet werden.
Die IP Adresse entspricht der des Geräts. Der Port ist fest auf den Standard Port 502 für Modbus/TCP eingestellt und kann nicht geändert werden. Weitere Informationen finden sie im Benutzerhandbuch.

5 Schnittstellenbeschreibung

5.1 Schnittstellenbeschreibung

Die WSG-Feldbusschnittstelle ist als 8-Byte-Ausgabe- und 12-Byte-Eingaberegistersatz implementiert, welcher in den folgenden Abschnitten näher beschrieben wird.

5.1.1 Ausgaberegister (SPS zu WSG)

Die Ausgaberegister werden vom PROFIBUS Master, PROFINET Controller (z.B. SPS) oder Modbus/TCP Master zum WSG übertragen. Sie bestehen aus Befehlsflags, Benutzerflags und drei Parametern und werden zur Steuerung des Greifers benutzt. Aufgrund der registersatzorientierten Natur von PROFIBUS, PROFINET und Modbus steht über diese Schnittstelle nur ein Teil des Funktionsspektrums des Greifmoduls zur Verfügung. Die Registeranordnung kann Tabelle 1 entnommen werden.
Für die Modbus/TCP Schnittstelle sind die Befehlsflags und die Benutzer-Flags auch als „Coils“ verfügbar. Eine Beschreibung der Adressierung der Modbus Register und „Coils“ befindet sich im Kapitel 4.

<table>
<tr><th>Byte Nr.</th><th>Modbus Holding Register</th><th>Register Name</th><th>Beschreibung</th></tr>
<tr><td>0</td><td>0</td><td>CMDFLAGS</td><td>
Befehls Flags
Ein Befehl wird erteilt, indem das entsprechende Bit von 0 auf 1 gesetzt wird (steigende Flanke). Bitte lesen Sie die folgenden Kapitel für eine detaillierte Beschreibung.
<table>
<tr><th>Bit Index:</th><th>Name</th><th>Beschreibung</th></tr>
<tr><td>Bit 0:</td><td>MOVE</td><td>Initiiert eine Vorpositionierung</td></tr>
<tr><td>Bit 1:</td><td>GRIP</td><td>Greift ein Teil</td></tr>
<tr><td>Bit 2:</td><td>RELEASE</td><td>Lässt ein Teil los</td></tr>
<tr><td>Bit 3:</td><td>HOMING</td><td>Referenzieren des Greifers</td></tr>
<tr><td>Bit 4:</td><td>STOP/ACK</td><td>Stopp, aber ohne Abschaltung des Motors / FAST STOP bestätigen</td></tr>
<tr><td>Bit 5:</td><td>FASTSTOP</td><td>Stopp mit Abschaltung des Motors</td></tr>
<tr><td>Bit 6:</td><td>JOG+</td><td>Jog-Modus in positiver Richtung</td></tr>
<tr><td>Bit 7:</td><td>JOG-</td><td>Jog-Modus in negativer Richtung</td></tr>
</table>
Wenn das FASTSTOP oder das STOP/ACK Bit auf ‘1’ gesetzt ist, werden alle Bewegungsbefehle ignoriert.
</td></tr>
</table>

<table>
<tr><th>Byte Nr.</th><th>Modbus Holding Register</th><th>Register Name</th><th>Beschreibung</th></tr>
<tr><td>1</td><td></td><td>IF</td><td>Benutzer-Flags (Eingabe)
Frei verfügbare Flags, die in Verbindung mit dem Skript-Interpreter verwendet werden können.
<table><tr><th>Bit Index:</th><th>Name</th><th>Beschreibung</th></tr><tr><td>Bit 0:</td><td>IF1</td><td>Eingabe Benutzer-Flag 1</td></tr><tr><td>Bit 1:</td><td>IF2</td><td>Eingabe Benutzer-Flag 2</td></tr><tr><td>Bit 2:</td><td>IF3</td><td>Eingabe Benutzer-Flag 3</td></tr><tr><td>Bit 3:</td><td>IF4</td><td>Eingabe Benutzer-Flag 4</td></tr><tr><td>Bit 4:</td><td>IF5</td><td>Eingabe Benutzer-Flag 5</td></tr><tr><td>Bit 5:</td><td>IF6</td><td>Eingabe Benutzer-Flag 6</td></tr><tr><td>Bit 6:</td><td>IF7</td><td>Eingabe Benutzer-Flag 7</td></tr><tr><td>Bit 7:</td><td>IF8</td><td>Eingabe Benutzer-Flag 8</td></tr></table></td></tr>
<tr><td>2..3</td><td>1</td><td>WIDTH</td><td>Befehlsparameter “Width” (Öffnungsweite)
Anzufahrende Finger-Öffnungsweite in 1/100 Millimetern (d.h. ein Wert von 1220 bedeutet 12,20 mm). Kodiert als INT (signed).</td></tr>
<tr><td>4..5</td><td>2</td><td>SPEED</td><td>Befehlsparameter “Speed” (Geschwindigkeit)
Zu verwendende Relativgeschwindigkeit beider Finger in 1/100 Millimeter pro Sekunde (d.h. ein Wert von 3005 bedeutet 30.05 mm/s). Kodiert als WORD (unsigned).
Bitte beachten:
Das Setzen dieses Parameters auf einen Wert jenseits der Grenzwerte des Systems und Auslösen einer Bewegungs-funktion führt zu einem FAST STOP.</td></tr>
<tr><td>6..7</td><td>3</td><td>FORCELIMIT</td><td>Befehlsparameter “Force Limit” (Kraftbegrenzung)
Einzustellender Grenzwert für die Greifkraft in 1/100 Newton (d.h. ein Wert von 1050 bedeutet 10,50 N). Ausnahme: Beim WSG 70 ist die Einheit aufgrund der höheren Greifkraft 1/10 Newton (d.h. ein Wert von 3050 bedeutet 305,0 N). Die Greif-kraft ist das Doppelte der Nennkraft, die auf das zu greifende Teil ausgeübt wird.
Kodiert als INT (signed), nur positive Werte erlaubt.
Bitte beachten:
Das Setzen dieses Parameters auf einen Wert jenseits der Grenzwerte des Systems und Auslösen einer Bewegungsfunkti-on, führt zu einem FAST STOP.</td></tr>
</table>

Tabelle 1: WSG Ausgabe Register

Um einen Befehl auszuführen müssen die Befehlsparameter gesetzt und das entsprechende Befehlsflag von 0 auf 1 (d.h. steigende Flanke) gesetzt werden. Jog Modus Flags werden ihrem Pegel entsprechend ausgewertet. Eine detaillierte Beschreibung der einzelnen Befehle befindet sich im Kapitel 5.1.2.

Wenn mehr als ein Befehlsflag gleichzeitig geändert wurde, wird nur der Befehl mit der niedrigsten Bit Nummer ausgeführt (d.h. Setzen des MOVE und GRIP Flags von 0 auf 1 löst nur den MOVE Befehl aus).

Ändern der Parameter während der Fingerbewegung (d.h. das MOVING System Flag ist 1) löst einen FAST STOP aus.

5.1.2 Eingaberegister (WSG zu SPS)

Der Eingaberegistersatz (siehe Tabelle 2) wird in jedem Zyklus vom WSG zum PROFIBUS Master, PROFINET Controller oder Modbus/TCP Master übertragen. Er enthält die aktuellen Greifparameter, den Betriebs- und Greifzustand, die benutzerdefinierten Flags, sowie einen Statuscode als Rückgabewert des zuletzt ausgeführten Befehls.

Für die Modbus/TCP Schnittstelle sind der Greifzustand, die benutzerdefinierten Flags und der Statuscode auch als „Discrete Inputs“ verfügbar. Eine Beschreibung der Adressierung der Modbus Register und „Discrete Inputs“ befindet sich im Kapitel 4.

<table>
<tr><th>Byte Nr.</th><th>Modbus Input Register</th><th>Register Name</th><th>Beschreibung</th></tr>
<tr><td>0</td><td rowspan="2">0</td><td>GSTATE</td><td>Greifstatus
Diese Flags zeigen den aktuellen Status des Greifvorganges wie untenstehend an und dienen der Steuerung und Überwachung des Greifvorganges:
<table><tr><th>Bit Index:</th><th>Name</th><th>Beschreibung</th></tr><tr><td>Bit 0:</td><td>IDLE</td><td>Warte auf neuen Befehl</td></tr><tr><td>Bit 1:</td><td>GRIPPING</td><td>Finger bewegen sich auf das Teil zu</td></tr><tr><td>Bit 2:</td><td>NO_PART</td><td>Kein Teil gefunden</td></tr><tr><td>Bit 3:</td><td>PART_LOST</td><td>Das Teil wurde gegriffen, ging aber wieder verloren</td></tr><tr><td>Bit 4:</td><td>HOLDING</td><td>Teil wird gehalten</td></tr><tr><td>Bit 5:</td><td>RELEASING</td><td>Finger entfernen sich vom Teil</td></tr><tr><td>Bit 6:</td><td>POSITIONING</td><td>Finger bewegen sich aufgrund eines Vorpositionierungsbefehls (MOVE)</td></tr><tr><td>Bit 7:</td><td>ERROR</td><td>Ein Fehler ist aufgetreten</td></tr></table></td></tr>
<tr><td>1</td><td>OF</td><td>Benutzerflags (Ausgabe)
Frei verwendbare Flags zur Kommunikation zwischen SPS Code und einem laufenden WSG Skript
<table><tr><th>Bit Index:</th><th>Name</th><th>Beschreibung</th></tr><tr><td>Bit 0:</td><td>OF1</td><td>Ausgabe Benutzer Flag 1</td></tr><tr><td>Bit 1:</td><td>OF2</td><td>Ausgabe Benutzer Flag 2</td></tr><tr><td>Bit 2:</td><td>OF3</td><td>Ausgabe Benutzer Flag 3</td></tr><tr><td>Bit 3:</td><td>OF4</td><td>Ausgabe Benutzer Flag 4</td></tr><tr><td>Bit 4:</td><td>OF5</td><td>Ausgabe Benutzer Flag 5</td></tr><tr><td>Bit 5:</td><td>OF6</td><td>Ausgabe Benutzer Flag 6</td></tr><tr><td>Bit 6:</td><td>OF7</td><td>Ausgabe Benutzer Flag 7</td></tr><tr><td>Bit 7:</td><td>OF8</td><td>Ausgabe Benutzer Flag 8</td></tr></table></td></tr>
<tr><td>2..5</td><td>1..2</td><td>SYSSTATE</td><td>System Status
Aktueller Systemzustand des Greifers als Bitvektor kodiert. Die Beschreibung der einzelnen Bits finden Sie in Anhang B. Dieses Register wird bei jedem Buszyklus, ungeachtet laufender Befehle, aktualisiert. Bitte beachten Sie, dass die Systemzustandsflags nicht zur Steuerung des Greifvorganges verwendet werden sollten. Benutzen Sie stattdessen den Greifstatus.</td></tr>
</table>

Byte Nr.	Modbus Input Register	Register Name	Beschreibung

Bit Index:	Name
Bit 0	REFERENCED
Bit 1	MOVING
Bit 2	BLOCKED_MINUS
Bit 3	BLOCKED_PLUS
Bit 4	SOFT_LIMIT_MINUS
Bit 5	SOFT_LIMIT_PLUS
Bit 6	AXIS_STOPPED
Bit 7	TARGET_POS_REACHED
Bit 8	OVERDRIVE_MODE[2]
Bit 9	FORCECNTL_MODE
Bit 10	reserviert
Bit 11	reserviert
Bit 12	FAST_STOP
Bit 13	TEMP_WARNING
Bit 14	TEMP_FAULT
Bit 15	POWER_FAULT
Bit 16	CURR_FAULT
Bit 17	FINGER_FAULT
Bit 18	CMD_FAILURE
Bit 19	SCRIPT_RUNNING
Bit 20	SCRIPT_FAILURE
Bit 21	reserviert
Bit 22	reserviert
Bit 23	reserviert
Bit 24	reserviert
Bit 25	reserviert
Bit 26	reserviert
Bit 27	reserviert
Bit 28	reserviert
Bit 29	reserviert
Bit 30	reserviert
Bit 31	reserviert

[2] Der Overdrive-Modus wird nicht von allen WSG-Greifmodulen unterstützt.

Byte Nr.	Modbus Input Register	Register Name	Beschreibung
6..7	3	WIDTH	*Aktuelle Öffnungsweite* Aktuelle Öffnungsweite der Finger in 1/100 Millimetern (d.h. ein Wert von 1220 bedeutet 12,2 mm). Kodiert als INT. Dieses Register wird bei jedem Buszyklus, ungeachtet laufender Befehle, aktualisiert.
8..9	4	Gripping Force	*Aktuelle Greifkraft* Aktuelle Greifkraft in 1/100 Newton (d.h. ein Wert von 405 bedeutet eine Greifkraft von 40,5 N). Ausnahme: Beim WSG 70 ist die Einheit aufgrund der höheren Greifkraft 1/10 Newton (d.h. ein Wert von 3050 bedeutet 305,0 N). Die Greifkraft ist das Doppelte der Nennkraft, die auf das zu greifende Teil ausgeübt wird. Kodiert als INT. Dieses Register wird bei jedem Buszyklus, ungeachtet laufender Befehle, aktualisiert. *Bitte beachten:* Wenn kein Messfinger auf dem WSG installiert ist, wird diese Größe anhand des Motorstromes approximiert.
10..11	5	Status Code	*Ergebnis des letzten Befehls* Dieses Feld behält seinen Zustand, bis ein neuer Befehl ausgeführt wird. Eine Beschreibung der möglichen Statuscodes entnehmen Sie bitte dem Anhang A.

Tabelle 2: WSG Eingabe Register

5.2 PROFIBUS Diagnosemitteilungen

Der WSG sendet Diagnosemitteilungen, welche die aktuellen Systemzustandflags enthalten, als das erste Double Word zum PROFIBUS Master (SPS), wenn mindestens eines der folgenden fehlerbezogenen Flags des Systemzustand Registers von 0 auf 1 wechselt:

- 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

Eine detaillierte Beschreibung dieser Flags finden Sie im Anhang B.
Das Format der Diagnosemitteilungen ist wie folgt:

Byte Nummer	Beschreibung
0..3	*Standard Diagnosedaten* Diagnosedaten, wie sie in der PROFIBUS Spezifikation definiert sind.
4..5	*Slave Ident-Nr.* Slave Identifikationsnummer. Diese ist 0x5555 für den WSG.
6	*Länge der Diagnosemitteilung* Diagnosemitteilungen des WSG haben immer eine Länge 10 Bytes = 0x0A.
7..10	*System Status* Aktueller Systemzustand des Greifers als Bitvektor kodiert. Gleiche Kodierung wie bei SYSSTATE in Tabelle 2.
10..15	*reserviert* Dieser Bereich ist für die zukünftige Verwendung reserviert.

Tabelle 3: WSG Diagnosemitteilung

6 Befehle

6.1 Bewegen der Finger im Positionierungsmodus (MOVE)

Dieser Befehl kann verwendet werden, um die Greiferfinger vor dem Auslösen eines Greifvorganges auf eine bestimmte Öffnungsweite zu fahren. Der Befehl soll das Greifen empfindlicher Teile beschleunigen, wenn die Greiferfinger vorher aufgrund von Prozessbedingungen größere Distanzen zurücklegen müssen. MOVE kann nur ausgeführt werden, wenn der Greifer untätig ist, d.h. der Greifstatus IDLE ist.

Position des Befehlsflags:

Bit 0

Verwendete Parameter:

WIDTH, SPEED

Statuscode

Das Statuscode-Register wird zu Beginn der Bewegung auf E_CMD_PENDING und nach Ausführung des Befehls auf das entsprechende Ergebnis gesetzt.

Greifstatus

Der Greifstatus wechselt zu POSITIONING, wenn die Bewegung beginnt und kehrt zurück zu IDLE nachdem der Befehl erfolgreich abgearbeitet wurde. Im Fall eines Fehlers wird der Greifstatus auf ERROR gesetzt.

Systemstatus

Es treten verschiedene Übergänge auf. Sofern es keine sehr speziellen Anforderungen gibt, sollte der aktuelle Zustand des Greifvorganges, über das Greifstatusregister ausgewertet werden.

6.2 Greifen eines Teils (GRIP)

Greift ein Teil mit seiner Nennweite, unter Berücksichtigung der vorgegebenen Geschwindigkeits- und Kraftbegrenzung. Wenn der Befehl ausgelöst wird, bewegt der Greifer die Finger zur Nennweite und versucht das erwartete Teil mit der vorher gesetzten Greifkraft einzuspannen. Wenn der Greifer die gewünschte Greifkraft innerhalb des festgelegten Klemmweges aufbauen kann, wurde das Teil gegriffen.
Wenn die Finger den Klemmweg durchfahren, ohne die Greifkraft aufzubauen, wurde kein Teil gefunden und der Greifstatus wird entsprechend aktualisiert. Der Klemmweg kann über die Weboberfläche des Greifmoduls gesetzt werden. Der Greifstatus wird mit dem entsprechenden Ergebnis der Operation aktualisiert (entweder HOLDING oder NO_PART).
Wenn kein Teil gefunden wurde, gibt der Befehl den Statuscode E_CMD_FAILED als Ergebnis zurück.

Nach erfolgreichem Greifen eines Teils ist die integrierte Greifteilüberwachung aktiv, welche die Greifkraft kontinuierlich überwacht. Wenn ein Teil aus dem Greifer entfernt wird, bevor der Release Befehl aufgerufen wurde erkennt der Greifer dies und setzt den Greifstatus auf PART_LOST.

Sie können die Greifgeschwindigkeit bei empfindlichen Teilen reduzieren, um den Aufprallimpuls von der Masse der Finger und der internen Mechanik zu begrenzen.

Der Greifstatus spiegelt den aktuellen Zustand des Greifvorganges wider. Dieser sollte nach jedem Befehl ausgewertet werden um zu überprüfen, ob der Greifvorgang wie erwartet ausgeführt wurde.

Position des Befehlsflags:

Bit 1

Verwendete Parameter:

WIDTH, SPEED, FORCELIMIT

Statuscode

Das Statuscode-Register wird zum Beginn der Bewegung auf E_CMD_PENDING und nach Ausführung des Befehls auf das entsprechende Ergebnis gesetzt. Wenn kein Teil gefunden wurde, wird der Statuscode auf
E_CMD_FAILED gesetzt.

Greifstatus

Während der Bewegung steht das Greifstatusregister auf GRIPPING. Wenn ein Teil gefunden wurde, wechselt es zu HOLDING. Wenn kein Teil gefunden wurde, wird es auf NO_PART gesetzt. Wenn das Teil entfernt oder verloren wird, nachdem es erfolgreich gegriffen wurde, wechselt der Greifstatus zu PART_LOST. Im Fall eines Fehlers wird das Register auf ERROR gesetzt.

Systemstatus

Es treten verschiedene Übergänge auf. Sofern es keine sehr speziellen Anforderungen gibt, sollte der aktuelle Zustand des Greifvorganges über das Greifstatusregister ausgewertet werden.

6.3 Loslassen eines Teils (RELEASE)

Lässt ein Teil durch Öffnen der Finger mit einer vorgegebenen Geschwindigkeit und Öffnungsweite los. Der RELEASE-Befehl quetscht das Teil nicht ein. Dies ist gewährleistet durch sukzessives Erhöhen der internen Kraftgrenze beim Entfernen der Finger vom Teil. Die Greifteilüberwachung wird vor dem Loslassen abgeschaltet. Zum Loslassen wird die Nennkraft des Greifers verwendet.

Position des Befehlsflags:

Bit 3

Verwendete Parameter:

WIDTH, SPEED

Statuscode

Das Statuscode-Register wird zu Beginn der Bewegung auf E_CMD_PENDING und nach Ausführung des Befehls auf das entsprechende Ergebnis gesetzt.

Greifstatus

Während der Bewegung steht das Greifstatusregister auf RELEASING. Bei Erreichen der Endposition wechselt das Register auf IDLE. Im Fehlerfall wird es auf ERROR gesetzt.

Systemstatus

Es treten verschiedene Übergänge auf. Sofern es keine sehr speziellen Anforderungen gibt, sollte der aktuelle Zustand des Greifvorganges über das Greifstatusregister ausgewertet werden.

6.4 Referenzieren des Greifers (HOMING)

Dieser Befehl referenziert den Greifer durch Ausführen einer Homing-Sequenz. Während des Homings fahren die Finger bis zum mechanischen Anschlag. Die Homing-Sequenz muss vorher auf der WSG- Weboberfläche konfiguriert werden (Menüpunkt "*Settings -> Motion Configuration*"). Sie können die Richtung der Referenzierung (nach innen oder außen) sowie das automatische Homing beim Starten des Greifers einstellen.

Das Homing ist vor dem Ausführen aller Bewegungsbefehle erforderlich. Das beste Positionierergebnis wird erreicht, wenn das Homing in die Richtung erfolgt, in der die höhere Positioniergenauigkeit erforderlich ist.

Während des Homings sind die Softlimits abgeschaltet. Kollidieren die Finger unterwegs mit einem Hindernis, kann dies einen falschen Referenzpunkt und somit eine falsche Positionierung zur Folge haben.

Position des Befehlsflags:

Bit 3

Verwendete Parameter:

keine

Statuscode

Das Statuscode-Register wird sofort auf E_CMD_PENDING und nach Abschluss des Befehls auf das entsprechende Ergebnis gesetzt.

Greifstatus

Während der Homingsequenz steht das Greifstatus Register auf POSITIONING.

System Status

Während der Bewegung ist das MOVING Flag auf 1 gesetzt. Wenn der Greifer referenziert wurde, wird das REFERENCED Flag auf 1 gesetzt.

6.5 Stoppen der Bewegung oder Quittieren eines FAST STOP (STOP/ACK)

Stoppt jede anstehende Bewegung ohne den Antrieb auszuschalten. Wenn während des Haltens eines Teils gestoppt wird (d.h. der Greifstatus steht auf HOLDING) wird die Greifteilüberwachung abgeschaltet und die eingestellte Greifkraft wird nicht länger aufgebracht.
Quittieren eines FAST STOP Zustands:
Wenn der WSG im FAST STOP Modus ist, muss ein Übergang dieses Flags von 0 auf 1 erfolgen, um den Zustand zu quittieren und in den normalen Betriebsmodus zurückzukehren. Vor dem Quittieren muss das FASTSTOP-Flag zurückgesetzt werden!

Position des Befehlsflags:

Bit 4

Verwendete Parameter:

keine

Statuscode

Wird auf E_SUCCESS gesetzt.

Greifstatus

Das Register wird auf IDLE gesetzt.

Systemstatus

Das AXIS_STOPPED Flag ist auf 1 gesetzt. Durch Quittieren eines FAST STOP, wird das FASTSTOP Flag gelöscht.

6.6 Auslösen eines Fast Stop (FAST STOP)

Diese Funktion ähnelt einem “Notstopp“. Sie stoppt unverzüglich jede Bewegung auf dem kürzesten Weg, schaltet den Antrieb ab und unterbindet die Ausführung weiterer Bewegungsbefehle. Der FAST STOP Zustand kann ausschließlich durch Quittieren desselben (siehe Kapitel 6.5) verlassen werden. Alle Bewegungsbefehle sind während eines FAST STOP verboten und rufen einen E_ACCESS_DENIED Fehler hervor. Der FAST STOP Zustand wird in den System Flags angezeigt und in der System-Logdatei aufgezeichnet. Daher sollte dieser Befehl in der Regel verwendet werden, um auf bestimmte Fehler-bedingungen zu reagieren.

Um lediglich die aktuelle Bewegung zu stoppen, sollten Sie stattdessen den STOP Befehl verwenden.

☞ **Neben dem STOP/ACK Flag kann ein FAST STOP auch interaktiv über die Weboberfläche zurückgesetzt werden. Dies reaktiviert den Antrieb; jedoch ist es erforderlich, das FAST STOP Flag der PROFIBUS Schnittstelle zurückzusetzen, um bewegungsbezogene Befehle wieder freizuschalten.**

Position des Befehlsflags:

Bit 5

Verwendete Parameter:

keine

Statuscode

Wird auf E_SUCCESS gesetzt.

Greifstatus

Wird auf IDLE gesetzt.

Systemstatus

Das FASTSTOP Flag wird auf 1 gesetzt.

6.7 Jog Modus (JOG+ und JOG-)

Um einen Prozess einzurichten, kann es erforderlich sein, die Finger des WSG manuell zu bewegen. Dies kann mit Hilfe der Jog Modus Flags realisiert werden. Die Flags werden ihrem Pegel entsprechend ausgewertet und ermöglichen den Antrieb der Finger mit konstanter Geschwindigkeit unter Verwendung zweier Taster an der SPS. Die Flags werden wie in der folgenden Tabelle interpretiert:

JOG+	JOG-	Movement direction
0	0	Jog Mode is disabled*
1	0	positive with SPEED
0	1	negative with SPEED
1	1	Stop

*) Wenn beide Jog Flags auf 0 wechseln, wird der Jog Modus verlassen und der Antrieb gestoppt.
Die Kraftgrenze (nur stromgesteuert), sowie die Geschwindigkeit können als Parameter übergeben werden.
Es könnte ein Stellrad verwendet werden, um sie zu steuern. Es ist zu beachten, dass eine Bewegung mit hoher Geschwindigkeit, durch das Einstellen einer zu niedrigen Kraftgrenze behindert werden kann.

ⓘ **Im Gegensatz zu anderen Bewegungsbefehlen kann der SPEED Parameter auf 0 gesetzt werden, was dazu führt, dass der Wert intern auf die minimal zulässige Geschwindigkeit gesetzt wird.**

Der Jog Modus ist lediglich zum Einrichten gedacht und sollte nicht für den normalen Betrieb des Greifers benutzt werden!

Position des Befehlsflags:

Bit 6 und 7

Verwendete Parameter:

SPEED, FORCELIMIT

Statuscode

Das Statuscode-Register wird auf E_CMD_PENDING und nach Abschluss des Befehls auf das entsprechende Ergebnis gesetzt.

Greifstatus

Während der Fingerbewegung wird das Register auf RELEASING gesetzt. Wenn die Endposition erreicht wurde (oder im Fall eines Fehlers), wird der Greifstatus auf IDLE gesetzt.

Systemstatus

Es treten verschiedene Übergänge auf. Sofern es keine sehr speziellen Anforderungen gibt, sollte der aktuelle Zustand des Greifvorganges über das Greifstatus Register ausgewertet werden.

7 WSG-Feldbusmonitor

Der WSG verfügt über einen integrierten Feldbusmonitor, auf welchen über die Weboberfläche zugegriffen werden kann (“*Diagnosis -> Fieldbus Monitor*”). Der Feldbusmonitor zeigt den aktuellen Inhalt der Eingabe- und Ausgaberegister des WSG und gibt einige Grundinformationen zum Buszustand wieder. Der Feldbusmonitor kann genutzt werden, um während der Integration des Greifers in eine Anlage den Zustand der Feldbusschnittstelle abzufragen.

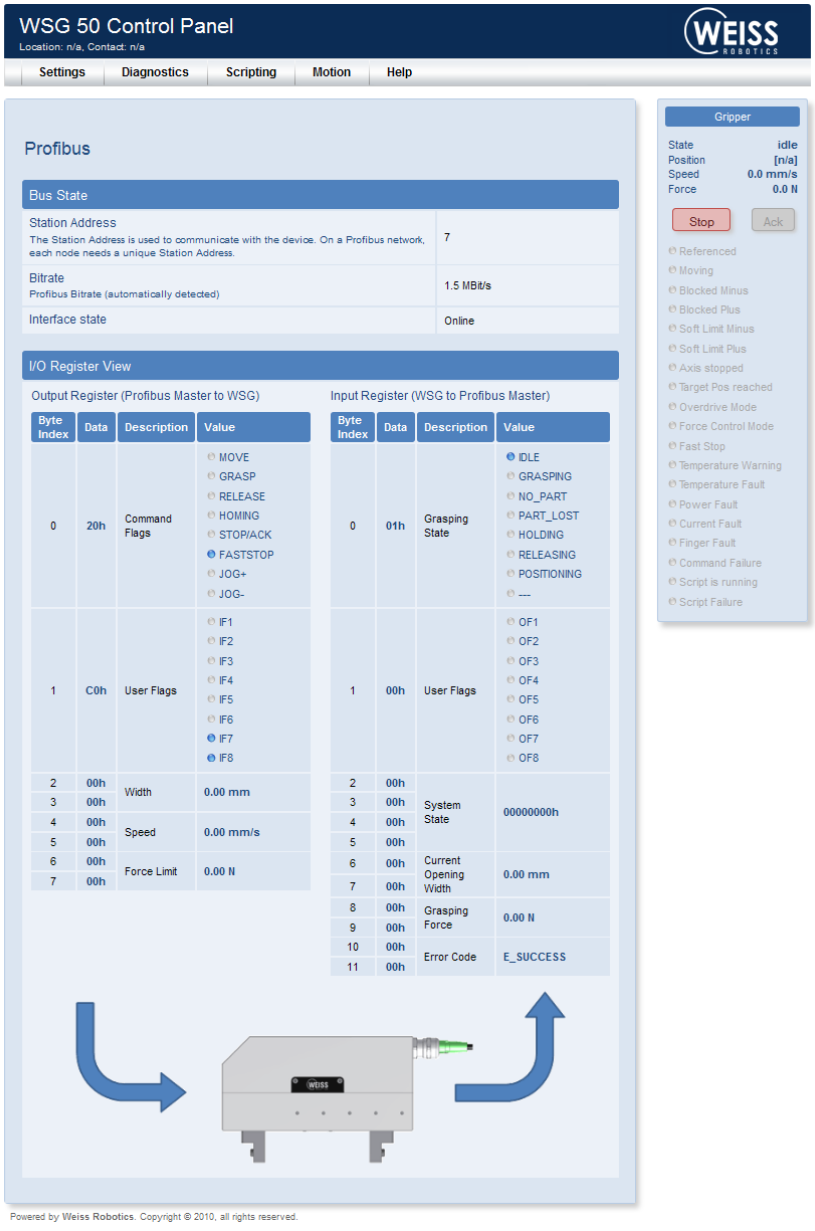


Abbildung 1: Bildschirmfoto des WSG Feldbus Monitors

Anhang A. Statuscodes

Status Code	Symbol Name	Beschreibung
0	E_SUCCESS	Kein Fehler aufgetreten, Vorgang erfolgreich
1	E_NOT_AVAILABLE	Funktion oder Daten nicht verfügbar
2	E_NO_SENSOR	Kein Messumformer angeschlossen
3	E_NOT_INITIALIZED	Gerät nicht initialisiert
4	E_ALREADY_RUNNING	Datenerfassung wird bereits ausgeführt
5	E_FEATURE_NOT_SUPPORTED	Die Funktion ist momentan nicht verfügbar
6	E_INCONSISTENT_DATA	Einer oder mehrere Parameter sind inkonsistent
7	E_TIMEOUT	Zeitüberschreitungs-Fehler
8	E_READ_ERROR	Fehler beim Lesen von Daten
9	E_WRITE_ERROR	Fehler beim Schreiben von Daten
10	E_INSUFFICIENT_RESOURCES	Nicht genügend Speicher vorhanden
11	E_CHECKSUM_ERROR	Prüfsummenfehler
12	E_NO_PARAM_EXPECTED	Parameter übergeben, obwohl keiner erwartet
13	E_NOT_ENOUGH_PARAMS	Zu wenige Parameter für den Befehl übergeben
14	E_CMD_UNKNOWN	Unbekannter Befehl
15	E_CMD_FORMAT_ERROR	Befehlsformat Fehler
16	E_ACCESS_DENIED	Zugriff verweigert
17	E_ALREADY_OPEN	Schnittstelle ist bereits geöffnet
18	E_CMD_FAILED	Fehler während der Ausführung eines Befehls
19	E_CMD_ABORTED	Befehlsausführung vom Benutzer abgebrochen
20	E_INVALID_HANDLE	ungültiges Handle
21	E_NOT_FOUND	Gerät oder Datei nicht gefunden
22	E_NOT_OPEN	Gerät oder Datei nicht geöffnet

23	E_IO_ERROR	Eingabe/Ausgabe Fehler
24	E_INVALID_PARAMETER	Falscher Parameter
25	E_INDEX_OUT_OF_BOUNDS	Index außerhalb des zulässigen Bereichs
26	E_CMD_PENDING	Kein Fehler, aber der Befehl wurde noch nicht vollständig ausgeführt. Eine Rückmeldung mit Statuscode folgt nach Ausführung des Befehls.
27	E_OVERRUN	Datenüberlauf
28	E_RANGE_ERROR	Bereichsfehler
29	E_AXIS_BLOCKED	Achse blockiert
30	E_FILE_EXISTS	Datei existiert bereits

Anhang B. Systemstatus-Flags

Die System Status Flags sind in einem 32-Bit breiten Integer angeordnet, der über das PROFIBUS Eingabe Register bereitgestellt wird. Jedes Bit hat eine spezielle Bedeutung, die in folgender Tabelle beschrieben ist.

Bit Nr.	Flag Name	Beschreibung
D31..21	reserviert	Diese Bits werden aktuell nicht verwendet, können aber in zukünftig veröffentlichter WSG Firmware verwendet werden.
D20	SF_SCRIPT_FAILURE	**Skript Fehler.** Während der Ausführung eines Skripts ist ein Fehler aufgetreten und das Skript wurde abgebrochen. Das Flag wird zurückgesetzt, sobald ein Skript gestartet wird.
D19	SF_SCRIPT_RUNNING	**Momentan wird ein Skript ausgeführt.** Das Flag wird zurückgesetzt wenn das Skript normal beendet wurde, ein Skriptfehler auftrat oder es vom Benutzer manuell beendet wurde.
D18	SF_CMD_FAILURE	**Befehl gescheitert.** Der letzte Befehl hat einen Fehler zurückgegeben.
D17	SF_FINGER_FAULT	**Finger Störung.** Der Status mindestens eines Fingers ist von „operating“ und „not connected“ verschieden. Bitte prüfen Sie die Finger Flags für eine genauere Fehlerbeschreibung.
D16	SF_CURR_FAULT	**Zu hoher Motorstrom.** Der Motor hat die maximal zulässige Wärmeverlustleistung erreicht. Das Flag wird automatisch zurück-gesetzt, sobald sich der Motor erholt hat. Danach kann der dadurch bedingte Fast Stop angegangen werden.
D15	SF_POWER_FAULT	**Energieversorgung fehlerhaft.** Die Versorgungsspannung ist außerhalb des zulässigen Bereichs. Bitte prüfen Sie die Stromversorgung.

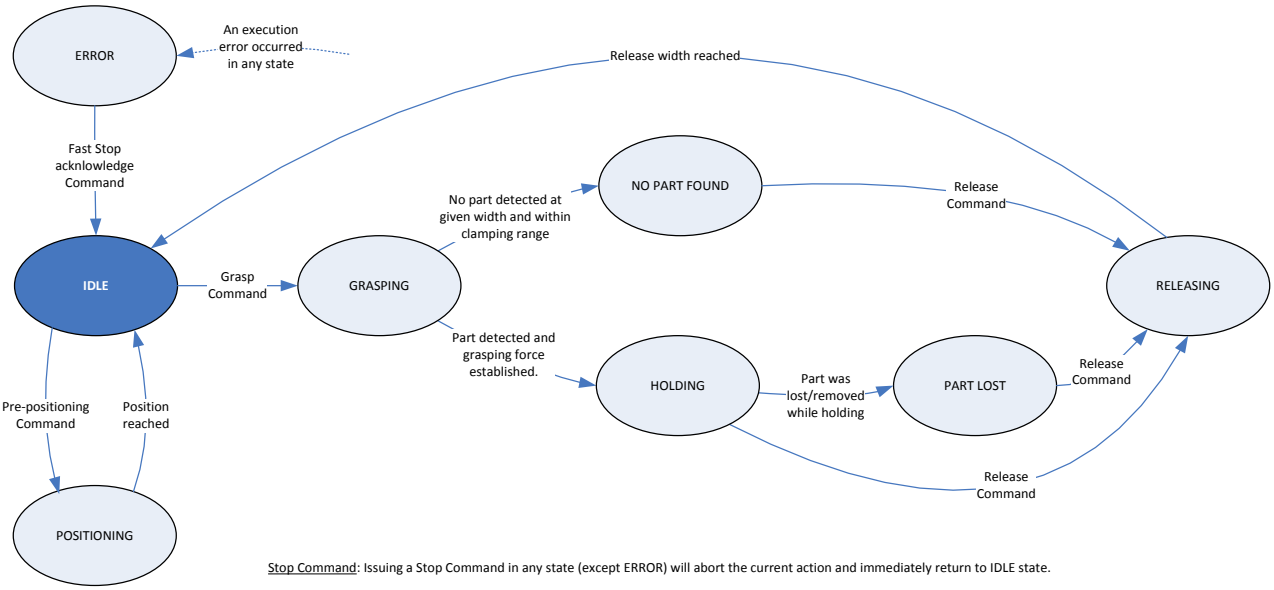
D14	SF_TEMP_FAULT	**Temperaturfehler.** Die Greiferhardware hat eine kritische Temperatur erreicht. Alle Bewegungsbefehle werden unterbunden, bis die Temperatur wieder unter die kritische Grenze gefallen ist.
D13	SF_TEMP_WARNING	**Temperaturwarnung.** Die Greiferhardware wird bald eine kritische Temperatur erreichen.
D12	SF_FAST_STOP	**Fast Stop.** Der Greifer wurde aufgrund eines Fehlers gestoppt. Um das Flag zurückzusetzen und die Bewegungsbefehle wieder zu aktivieren, muss der Fehler quittiert werden.
D11..10	reserviert	Diese Bits werden aktuell nicht verwendet, können aber in zukünftig veröffentlichter WSG Firmware verwendet werden.
D9	SF_FORCECNTL_MODE	**Kraftgeregelter Modus.** Die Kraftregelung ist momentan aktiv unter Verwendung des installierten Kraftmessfingers (WSG-FMF). Wenn dieses Flag nicht gesetzt ist, wird die Greifkraft auf Basis des Motorstroms näherungsweise geregelt.
D8	SF_OVERDRIVE_MODE	**Overdrive Modus[3].** Der Greifer befindet sich im Overdrive Modus und die Greifkraft kann bis auf den Wert der Overdrive Kraftgrenze erhöht werden. Wenn dieses Bit nicht gesetzt ist, kann die Greifkraft nicht höher als die Nennkraft des Greifers gesetzt werden.
D7	SF_TARGET_POS_REACHED	**Zielposition erreicht.** Wird gesetzt, sobald die Zielposition erreicht wurde. Das Flag ist nicht mit SF_MOVING synchronisiert, sodass eine Verzögerung zwischen dem Zurücksetzen von SF_MOVING und dem Setzen von SF_TARGET_POS auftreten kann.

[3] Der Overdrive Modus wird nicht von allen WSG Greifern unterstützt. Bitte beachten Sie die Bedienungsanleitung für weitere Informationen.

D6	SF_AXIS_STOPPED	**Achse gestoppt.** Ein vorheriger Bewegungsbefehl wurde durch den Stop-Befehl abgebrochen. Das Flag wird durch den nächsten Bewegungsbefehl wieder zurückgesetzt.
D5	SF_SOFT_LIMIT_PLUS	**Soft Limit in positiver Richtung erreicht.** Die Finger haben die definierten Soft Limits in positiver Bewegungsrichtung erreicht. Eine weitere Bewegung in diese Richtung ist nicht erlaubt. Das Flag wird gelöscht, wenn sich die Finger wieder von der Position entfernen.
D4	SF_SOFT_LIMIT_MINUS	**Soft Limit in negativer Richtung erreicht.** Die Finger haben die definierten Soft Limits in negativer Bewegungsrichtung erreicht. Eine weitere Bewegung in diese Richtung ist nicht erlaubt. Das Flag wird gelöscht, wenn sich die Finger wieder von der Position entfernen.
D3	SF_BLOCKED_PLUS	**Achse ist in positiver Bewegungsrichtung blockiert.** Wird gesetzt, wenn die Achse in positiver Bewegungsrichtung blockiert ist. Das Flag wird zurückgesetzt, wenn die Blockade gelöst oder ein Stop-Befehl erteilt wurde.
D2	SF_BLOCKED_MINUS	**Achse ist in negativer Bewegungsrichtung blockiert.** Wird gesetzt, wenn die Achse in negativer Bewegungsrichtung blockiert ist. Das Flag wird zurückgesetzt, wenn die Blockade gelöst oder ein Stop-Befehl erteilt wurde.
D1	SF_MOVING	**Die Finger sind gerade in Bewegung.** Das Flag wird gesetzt, sobald eine Bewegung gestartet wurde (z.B. MOVE Befehl) und wird automatisch zurückgesetzt, wenn die Bewegung stoppt.
D0	SF_REFERENCED	**Finger sind referenziert.** Ist dieses Flag gesetzt, dann ist der Greifer referenziert und akzeptiert Bewegungsbefehle.

Anhang C. Greifzustände

Das folgende Diagramm illustriert die Greifzustände und Übergänge, die im normalen Betrieb vorgesehen sind.



Anhang D. Demo Programm

Im Lieferumfang des WSG befinden sich einige einfache Demoprojekte für SPSen der Siemens SIMATIC S7-1200 Serie sowie STEP 7 TIA-Portal v12 (und höher).
Die Demoprojekte können auf der Weiss Robotics Website[4], der beiliegenden Produkt-CD oder von der WSG-Weboberfläche heruntergeladen werden.
Die Demoprojekte wurden auf einer CPU vom Typ 1212C mit dem PROFIBUS Modul CM1243-5 unter Verwendung der Siemens STEP7 Basic v12.0 (TIA Portal) Projektierungsumgebung implementiert und getestet. Sie vollführen eine Endlosschleife einfacher Greifzyklen, bestehend aus Vorpositionieren der Greiferbacken (MOVE), Greifen eines Teils (GRIP), Loslassen (RELEASE) und Zurückkehren zur Startposition (MOVE). Wenn ein Teil erkannt wurde, wird es vom Greifmodul für einen kurzen Moment festhalten. Im Fall eines Fehlers führt das Greifmodul eine Homingsequenz aus beginnt wieder von vorn. Bitte beachten Sie, dass der WSG vor Ausführung des Programms referenziert sein muss.

Die SPS ist in dem Projekt auf die IP-Adresse 192.168.1.250 und die PROFIBUS-Adresse 2 konfiguriert. Der WSG wird auf der PROFIBUS-Adresse 7 (Standard) erwartet.
Bei Verwendung des PROFINET-Demoprogramms wird erwartet, dass das Greifmodul auf die IP-Adresse 192.168.1.20 eingestellt ist (Auslieferungszustand).

Es könnte nützlich sein, den Feldbusmonitor in der WSG-Weboberfläche zu öffnen (siehe Kapitel 7), um mehr Informationen zu möglichen Problemen zu erhalten.

Das Demoprojekt ist nur für Testzwecke gedacht. Setzen Sie es nicht in einer Produktionsumgebung ein.

[4] http://www.weiss-robotics.com/

SCL-Quelltext des Greifzyklus-Bausteins in den Demo-Projekten

```
1  ////////////////////////////////////
2  // Receive data type               //
3  ////////////////////////////////////
4
5  TYPE "WSG_RECEIVE"
6  VERSION : 0.1
7     STRUCT
8        STW1 : Struct
9           IDLE : 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
99
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 interface 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 THEN
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 := #CycleConfig.HoldingTime,
290   SIGN := 1 );
291             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
```

www.weiss-robotics.com

© Weiss Robotics GmbH & Co. KG. Alle Rechte vorbehalten.

Die in diesem Dokument angegebenen technischen Daten können zum Zwecke der Produktverbesserung ohne Vorankündigung geändert werden. Warenzeichen sind Eigentum des jeweiligen Eigentümers. Unsere Produkte sind nicht für den Einsatz in lebenserhaltenden Systemen oder für Systeme, bei denen ein Fehlverhalten zu Personenschäden führen könnte, vorgesehen.