



GRIPLINK-PLUGIN FÜR STÄUBLI

Version 4.0.0

April 2026



Inhalt

1	Einführung.....	3
1.1	Notation und Symbole	3
1.2	Bestimmungsgemäße Verwendung.....	3
1.3	Systemvoraussetzungen	4
1.4	Lizenzbestimmungen	4
2	Änderungen gegenüber vorigen Versionen.....	5
2.1	Fehlerbehandlung – x_nErrorCode.....	5
2.2	Entfernung der impliziten Gerätezustand-Ausgabe in GRIP, RELEASE und HOME.....	5
2.3	Umbenennung des Befehls devstate in deviceState	5
2.4	Umstellung auf Fließkomma-Parameter.....	6
2.5	Neue Befehle – FLEXGRIP und FLEXRELEASE	6
2.6	GRIP und RELEASE – Parameter x_bWstrEnabled durch x_nDoBlock ersetzt	7
2.7	WAITVAL – Erweiterte Parameter	7
2.8	CLAMP – Parametertyp geändert	7
2.9	Weiterhin verfügbare (veraltete) Befehle	7
2.10	Hinweis zur Migration.....	8
3	Installation	9
3.1	Vorbereitung des Roboters.....	9
3.2	Installation der Software via STAUBLI Robotics Suite (SRS)	10
3.3	Einrichten des Socket-Anschlusses für den GRIPLINK-Controller	11
3.3.1	Teach Pendant	11
3.3.2	STAUBLI Robotics Suite (SRS).....	12
3.4	Übertragung der VAL3-Anwendung mit dem Transfer Manager	15
3.4.1	Verbindung zur CS9-Steuerung konfigurieren	15
3.4.2	Verbindung im Transfer Manager herstellen	15
3.4.3	Zielsteuerung prüfen	16
3.4.4	Benötigte Komponenten auswählen	16
3.4.5	Anwendung auf dem SP2-Panel neu laden.....	17
3.5	Verhalten im Fehlerfall	18
3.6	Not-Halt	19
4	Befehlsreferenz.....	20
4.1	Beispielprogramme – GRIPLINK_Examples.....	21
4.2	Verbindung aufbauen – CONNECT	23
4.3	Verbindung trennen – DISCONNECT.....	24
4.4	Prüfung des angeschlossenen Geräts – DEVASSERT.....	25
4.5	Gerät aktivieren – ENABLE.....	26

4.6	Gerät deaktivieren – DISABLE	27
4.7	Greifmodul referenzieren – HOME	29
4.8	Werkstück greifen – GRIP	30
4.9	Werkstück freigeben – RELEASE	32
4.10	Werkstück über Parameter greifen – FLEXGRIP	33
4.11	Werkstück über Parameter freigeben – FLEXRELEASE	34
4.12	Parametrierung eines Griff-Presets – SETGRIPCFG	35
4.13	Aktuellen Gerätezustand abfragen – DEVICESTATE	37
4.14	Warten auf eine Änderung des Gerätezustands – WSTR	38
4.15	Abfrage eines Gerätewertes – VALUE	40
4.16	Setzen eines Gerätewertes – SETVALUE	41
4.17	Warten auf einen Gerätewert – WAITVAL	42
4.18	Greifkraftherhaltung steuern – CLAMP	43
4.19	Ansteuerung der Status-LED – LED	44
4.20	Veraltete Befehle	45
4.20.1	Gleichzeitiges Greifen von Werkstücken – MGRIP	45
4.20.2	Gleichzeitiges Freigeben von Werkstücken – MRELEASE	46
4.20.3	Warten auf die Änderung mehrerer Gerätezustände – MWAITFOR	48
5	Fehlersuche	49
5.1	Fehlermeldungen	49
5.1.1	GRIPLINK-Fehlerantwort mit Statuscode	49
5.1.2	Socket-Variable nicht mit Hardware-I/O verknüpft	49
5.1.3	Instruction timeout bei fehlender Socket-Kommunikation	50
5.1.4	Socket-Kommunikation während des Betriebs unterbrochen	50
Anhang A	Gerätezustand	52
Anhang B	Statuscodes	53

1 Einführung

Mit der GRIPLINK-Technologie können IO-Link kompatible Automationskomponenten über eine Netzwerkverbindung mit Robotersystemen führender Hersteller verbunden werden. Das GRIPLINK-Plugin für Stäubli ist das steuerungsseitige Bindeglied und ermöglicht die einfache Einbindung der GRIPLINK-Technologie von WEISS ROBOTICS in Robotersysteme des Herstellers Stäubli.

Der Begriff „GRIPLINK-Controller“ wird in dieser Anleitung stellvertretend für alle Produkte verwendet, die die GRIPLINK-Technologie integriert haben.



Dazu zählen sowohl eigenständige GRIPLINK-Controller (z. B. GRIPLINK-ET4) als auch Geräte mit eingebetteter GRIPLINK-Funktionalität, wie etwa WPG- oder INTRAPAL-Servogreifer von Weiss Robotics. Alle diese Produkte implementieren dasselbe GRIPLINK-Protokoll und können daher über das Plugin auf identische Weise angesteuert werden.



Diese Anleitung beschreibt die Funktionen des GRIPLINK-Plugins. Informationen über Montage, Inbetriebnahme und Betrieb des GRIPLINK-Controllers entnehmen Sie der Betriebsanleitung des jeweiligen Moduls. Diese finden Sie online unter **www.griplink.de**

1.1 Notation und Symbole

Zur besseren Übersicht werden in dieser Anleitung folgende Symbole verwendet:



Funktions- oder sicherheitsrelevanter Hinweis. Nichtbeachtung kann die Sicherheit von Personal und Anlage gefährden, das Gerät beschädigen oder die Funktion des Geräts beeinträchtigen.



Zusatzinformation zum besseren Verständnis des beschriebenen Sachverhalts.



Verweis auf weiterführende Informationen.

1.2 Bestimmungsgemäße Verwendung

Die Software „GRIPLINK-Plugin“ ist zur Kommunikation zwischen dem GRIPLINK-Controller von WEISS ROBOTICS und einer Robotersteuerung bestimmt. Sie ermöglicht die Integration und Steuerung von GRIPLINK-kompatiblen Geräten über die standardisierte GRIPLINK-Kommunikationsschnittstelle.

Die Anforderungen der zutreffenden Richtlinien sowie die Installations- und Betriebshinweise in dieser Anleitung müssen beachtet und eingehalten werden. Eine andere oder darüberhinausgehende Verwendung gilt als nicht bestimmungsgemäß. Für hieraus resultierende Schäden haftet der Hersteller nicht.

1.3 Systemvoraussetzungen

Dieses Plugin ist kompatibel mit der **GRIPLINK Protokollversion 3** und höher.

Zum Betrieb wird eine der folgenden Stäubli Robotersteuerungen benötigt:

- CS9 mit Softwarestand s8.10.6 oder höher

Folgende Roboter-Optionen werden zum Betrieb der Software benötigt:

- Stäubli SRS



Kontaktieren Sie Ihren Stäubli Händler zum Bezug der Roboter-Optionen.



Die IP-Adresse des GRIPLINK-Controllers muss im selben Subnetz liegen wie die der Robotersteuerung. In der Anleitung des GRIPLINK-Controllers ist der genaue Vorgang beschrieben, wie Sie die IP-Adresse ändern.

1.4 Lizenzbestimmungen

Das GRIPLINK-Plugin ist urheberrechtlich geschützt. Die jeweils gültigen Lizenzbestimmungen liegen dem Softwarepaket bei. Mit der Installation akzeptieren Sie diese Lizenzbestimmungen.

2 Änderungen gegenüber vorigen Versionen

Mit Version 4.0.0 wurde das GRIPLINK-Plugin grundlegend überarbeitet. Die Änderungen betreffen die Parametrierung, die Befehlssignaturen und vor allem das Verhalten im Fehlerfall.



Diese Version ist nicht vollständig abwärtskompatibel; bestehende Anwendungen müssen überprüft und angepasst werden.

2.1 Fehlerbehandlung – x_nErrorCode

Ab Version 4.0.0 liefert jeder Befehl seinen Status über den per Referenz übergebenen Parameter x_nErrorCode zurück:

- -1: keine bzw. nicht auswertbare Antwort des GRIPLINK-Controllers.
- 0 (E_SUCCESS): Befehl wurde erfolgreich ausgeführt.
- 1 – 27: Statuscodes des GRIPLINK-Controllers (siehe Anhang B).



Ein Verbindungsverlust oder Socket-Timeout wird als VAL3-Laufzeitfehler gemeldet (z. B. ERR 60, ERR 123, ERR 125).

Die Fehlerreaktion ist im Anwenderprogramm zu implementieren. Eine vollständige Beschreibung findet sich in Abschnitt 3.5.

2.2 Entfernung der impliziten Gerätezustand-Ausgabe in GRIP, RELEASE und HOME

In Version 3.x.x lieferten grip(), release() und home() den Gerätezustand des Greifers als per Referenz übergebenen Parameter x_nDevstate zurück. Ab Version 4.0.0 entfällt diese Rückgabe.



Wird der Gerätezustand nach grip(), release() oder home() benötigt, ist er anschließend explizit über deviceState() abzufragen.

Bis Version 3.x.x	Ab Version 4.0.0
<code>grip(port, preset, devstate&, bWstrEnabled:bool)</code>	<code>grip(port, preset, nDoBlock, errorCode&)</code>
<code>release(port, preset, devstate&, bWstrEnabled:bool)</code>	<code>release(port, preset, nDoBlock, errorCode&)</code>
<code>home(port, devstate&)</code>	<code>home(port, errorCode&)</code>

2.3 Umbenennung des Befehls devstate in deviceState

Der Befehl devstate() wurde in Version 4.0.0 auf deviceState() umbenannt. Aufrufe der Form `call griplink:devstate(...)` sind durch `call griplink:deviceState(...)` zu ersetzen.

Bis Version 3.x.x	Ab Version 4.0.0
call griplink:devstate(...)	call griplink:deviceState(...)

2.4 Umstellung auf Fließkomma-Parameter

Die bisher verwendeten skalierten Ganzzahlparameter wurden durch Fließkommawerte ersetzt. In früheren Versionen wurden physikalische Werte (z. B. Positionen oder Geschwindigkeiten) mit einem Skalierungsfaktor (typischerweise 1000) übertragen.



Ab Version 4.0.0 erfolgt die Übergabe direkt in physikalischen Einheiten.

Diese Änderung betrifft die folgenden Befehle:

- SETGRIPCFG
- VALUE
- SETVALUE
- WAITVAL

Auswirkungen

- Skalierungsfaktoren (z. B. Faktor 1000) dürfen nicht mehr verwendet werden
- Parameterwerte müssen auf reale physikalische Werte umgestellt werden (z. B. mm statt mm × 1000)

Beispiel

Bis Version 3.x.x	Ab Version 4.0.0
POSITION = 19500	POSITION = 19,5

2.5 Neue Befehle – FLEXGRIP und FLEXRELEASE

Mit Version 4.0.0 wurden die folgenden neuen Befehle eingeführt:

- **FLEXGRIP:** Greifvorgang mit frei wählbaren Bewegungsparametern (Position, Kraft, Geschwindigkeit, Beschleunigung).
- **FLEXRELEASE:** Freigabe eines gegriffenen Werkstücks mit frei wählbaren Bewegungsparametern (Position, Geschwindigkeit, Beschleunigung).

2.6 GRIP und RELEASE – Parameter x_bWstrEnabled durch x_nDoBlock ersetzt

Bei den Befehlen GRIP und RELEASE wurde der bisherige Parameter x_bWstrEnabled (bool) durch x_nDoBlock (num) ersetzt; das Blockierverhalten wird nun direkt im Befehl gesteuert. Der eigenständige Befehl WSTR bleibt davon unberührt und kann weiterhin zur Überwachung von Zustandsänderungen verwendet werden – insbesondere in Kombination mit x_nDoBlock = 0 für nicht blockierende Aufrufe.

Zusätzlich wurde der per Referenz übergebene Parameter von x_nDevstate (Gerätezustand) entfernt. Wird der Gerätezustand nach einem blockierenden GRIP/RELEASE benötigt, ist er explizit über DEVSTATE abzufragen.

Bis Version 3.x.x	Ab Version 4.0.0
<code>grip(port, preset, devstate&, bWstrEnabled:bool)</code>	<code>grip(port, preset, nDoBlock, errorCode&)</code>
<code>release(port, preset, devstate&, bWstrEnabled:bool)</code>	<code>release(port, preset, nDoBlock, errorCode&)</code>

2.7 WAITVAL – Erweiterte Parameter

Der Befehl WAITVAL wurde erweitert. Neben der Umstellung auf Fließkommawerte wurden neue Parameter für die Überwachung eines Schwellenwerts mit Toleranzfenster und Timeout eingeführt.

Bis Version 3.x.x	Ab Version 4.0.0
<code>waitval(port, index, value)</code>	<code>waitval(port, index, threshold, windowsize, timeout, errorCode&)</code>

2.8 CLAMP – Parametertyp geändert

Der Parameter x_bEnable (bool) wurde durch x_nEnable (num) ersetzt. Bestehende Aufrufe mit true/false müssen auf 1/0 umgestellt werden.

Bis Version 3.x.x	Ab Version 4.0.0
<code>clamp(port, bEnable:bool)</code>	<code>clamp(port, nEnable, errorCode&)</code>

2.9 Weiterhin verfügbare (veraltete) Befehle

Die folgenden Befehle sind weiterhin verfügbar, werden jedoch als **veraltet (deprecated)** eingestuft:

- MGRIP
- MRELEASE
- MWAITFOR

Diese Befehle bleiben aus Gründen der Abwärtskompatibilität erhalten, sollten für neue Projekte aber nicht mehr genutzt werden. Für neue Anwendungen wird empfohlen, die folgenden Befehle zu verwenden:

- GRIP/RELEASE in Kombination mit DOBLOCK = 0 anstelle von MGRIP/MRELEASE
- WSTR zur Zustandsüberwachung anstelle von MWAITFOR

2.10 Hinweis zur Migration

Durch die Änderungen in Version 4.0.0 sind bestehende Programme **nicht vollständig abwärtskompatibel**. Vorhandene Anwendungen müssen insbesondere hinsichtlich der folgenden Punkte überprüft und angepasst werden:

- Entfernung von Skalierungsfaktoren
- Umstellung von x_bWstrEnabled auf x_nDoBlock bei GRIP und RELEASE
- Anpassung der WAITVAL-Aufrufe auf die neue Signatur
- Umstellung von true/false auf 1/0 beim CLAMP-Parameter



Es wird empfohlen, nach der Migration eine vollständige Funktionsprüfung der Anwendung durchzuführen.

3 Installation



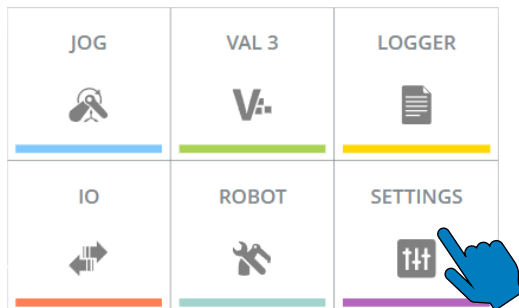
Stellen Sie sicher, dass Sie die aktuelle Version des GRIPLINK-Plugins verwenden. Die aktuelle Version kann unter www.griplink.de/software heruntergeladen werden.

3.1 Vorbereitung des Roboters

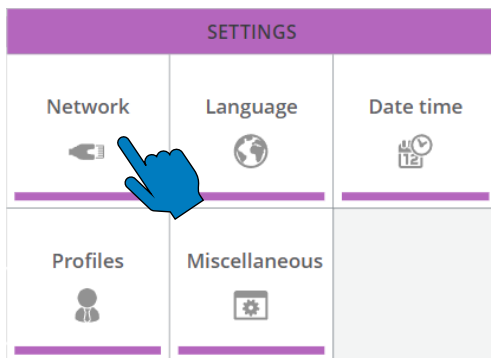
Schalten Sie den Roboter ein und richten Sie die IP-Adresse ein (z.B. 192.168.1.X). Stellen Sie dabei sicher, dass Roboter und der GRIPLINK-Controller im gleichen Netzwerk sind.

Führen Sie dazu folgende Schritte aus:

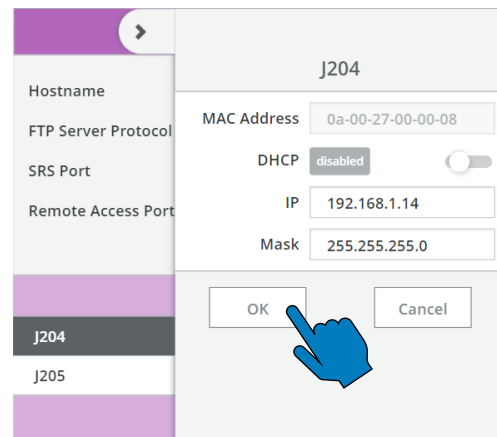
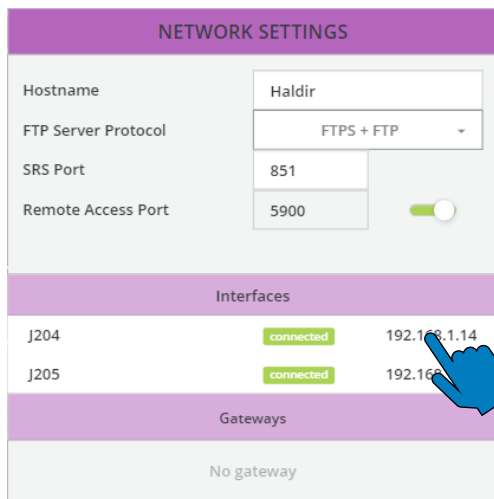
1. Klicken Sie auf 
2. Navigieren Sie zu „SETTINGS“



3. Navigieren Sie zu „Network“

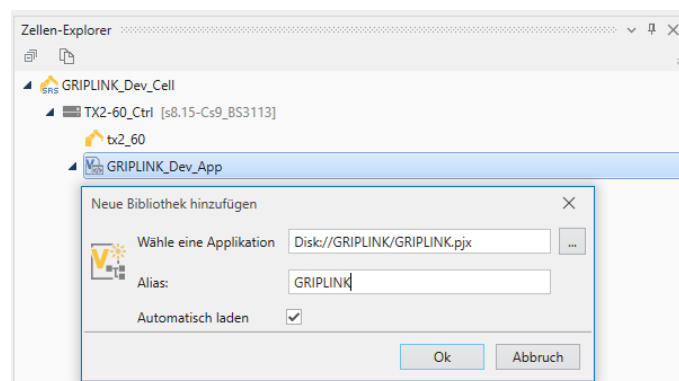


4. Stellen Sie die IP-Adresse der verwendeten Netzwerkschnittstelle auf eine gültige Adresse



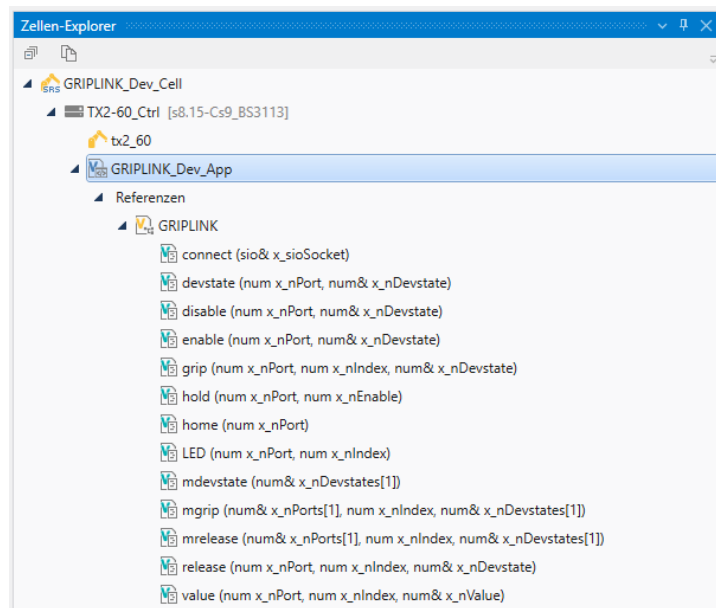
3.2 Installation der Software via STAUBLI Robotics Suite (SRS)

1. Laden Sie die Plugin-Datei „griplink_staubli_<Version>.zip“ herunter.
2. Entpacken Sie das ZIP-Archiv in das Verzeichnis „usrapp“:
„.../<Zellenname>/<Controllernamen>/usr/usrapp“
3. Öffnen Sie die Zelle in SRS und navigieren Sie zum Zellen-Explorer.
4. Fügen Sie mit Rechtsklick auf das VAL3-Programm, in dem das Plugin verwendet werden soll, das Plugin-Programm „GRIPLINK.pjx“ als Bibliothek hinzu.



Geben Sie in das Feld „Alias“ einen eindeutigen Namen ein, mit dem Sie die Plugin-Funktionen in den Unterprogrammen Ihrer Applikation verwenden.

5. Im Zellen-Explorer erscheint nun ein Programm mit dem Namen „GRIPLINK“



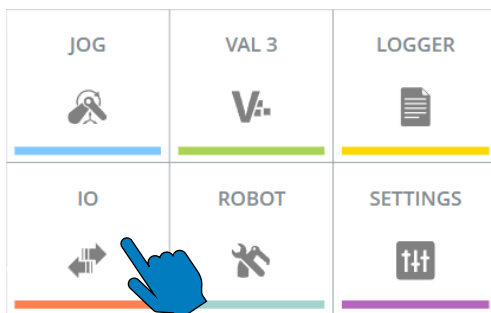
3.3 Einrichten des Socket-Anschlusses für den GRIPLINK-Controller

Die Netzwerkschnittstelle wird über Socket-Variablen angesprochen, die zuvor konfiguriert werden müssen.

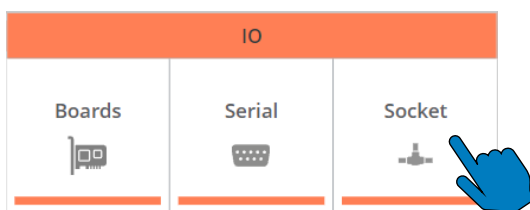
3.3.1 Teach Pendant

Führen Sie dazu folgende Schritte aus:

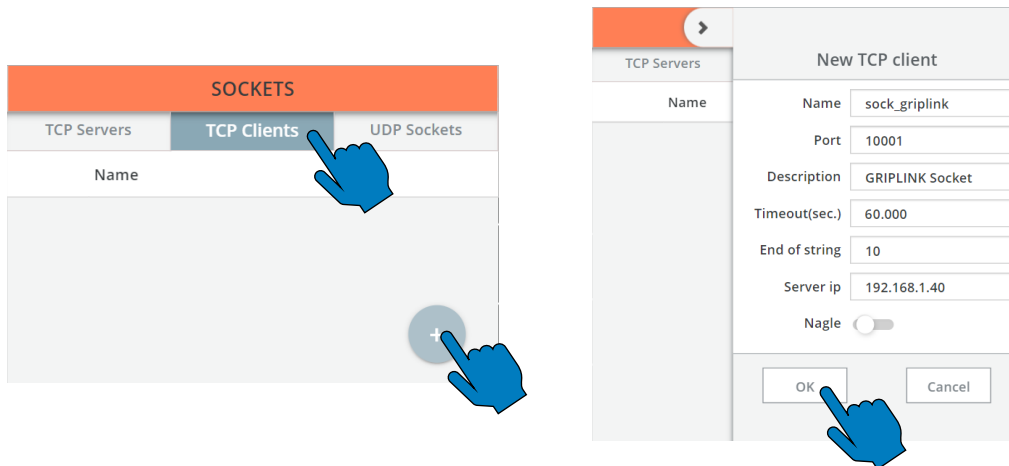
1. Klicken Sie auf 
2. Navigieren Sie zu „IO“



3. Navigieren Sie zu „Socket“



4. Wählen Sie „TCP Clients“ und öffnen Sie mit Klick auf  die Eingabemaske. Geben Sie den Namen der Socket-Verbindung und die übrigen Parameter der Schnittstelle ein. Bestätigen Sie mit einem Klick auf „OK“.



Stellen Sie folgende Werte ein:

Feld	Wert	Einheit
Server ip	IP-Adresse des GRIPLINK-Controllers	-
Port	10001	-
Timeout	60	Sekunden
End of string	10	-



Achten Sie darauf, dass Sie den korrekten Namen eingeben, den Sie auch im Roboter-Programm verwenden.

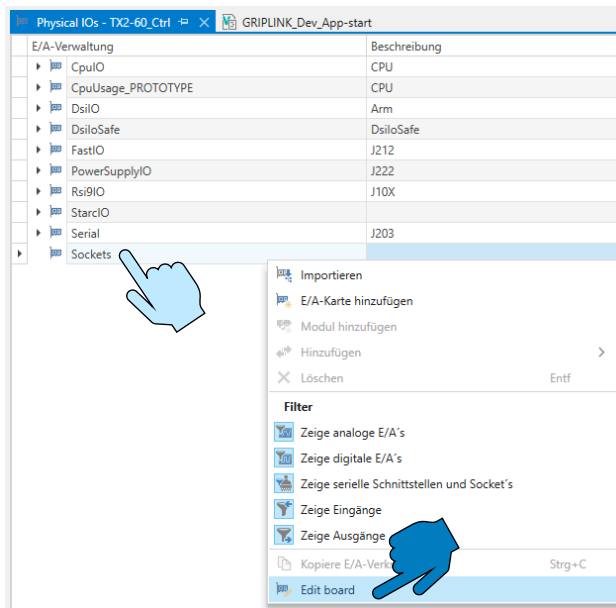


Andere Werte können zu Fehlverhalten führen!

3.3.2 STAUBLI Robotics Suite (SRS)

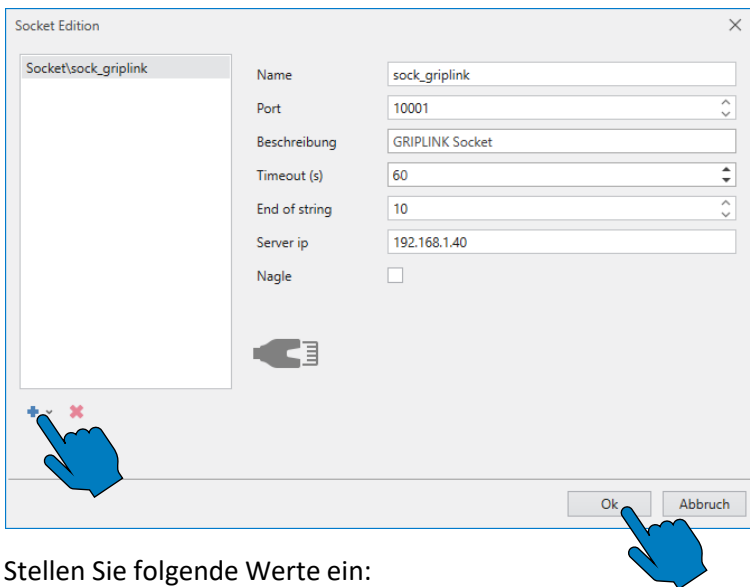
Führen Sie dazu folgende Schritte aus:

1. Öffnen Sie in SRS die E/A-Verwaltung des Roboter-Controllers
2. Öffnen Sie die Eingabemaske für eine neue Socket-Verbindung durch Rechtsklick auf „Sockets“



3. Fügen Sie mit Klick auf  einen neuen Socket hinzu. Geben Sie den Namen der Socket-Verbindung und die übrigen Parameter der Schnittstelle ein. Bestätigen Sie mit einem Klick auf „OK“.

4.



Stellen Sie folgende Werte ein:

Feld	Wert	Einheit
Server ip	IP-Adresse des GRIPLINK-Controllers	-
Port	10001	-
Timeout	60	Sekunden
End of string	10	-



Achten Sie darauf, dass Sie den korrekten Namen eingeben, den Sie auch im Roboter-Programm verwenden.



Andere Werte können zu Fehlverhalten führen!

5. Die Socket-Verbindung erscheint nun in der Tabelle.

Sockets		
	sock_griplink	GRIPLINK Socket Socket\sock_griplink

6. Die Socket-Verbindung kann nun einer Socket-Variable im Roboterprogramm zugewiesen werden

- 7.

The screenshot shows the 'Physical I/Os - TX2-60_Ctrl' configuration window. The 'Sockets' section is expanded, showing a table with columns: E/A-Verwaltung, Beschreibung, Format, Logical Name, and E/A-Verknüpfu... The 'sock_griplink' entry is selected. The 'Globale Variablen' panel on the right shows the 'sioGriplink[0]' variable being assigned to the selected socket.

Index	lo
0	Socket\sock_griplink

Beim Laden von VAL3-Applikationen aus der STAUBLI Robotics Suite auf die Steuerung kann es sein, dass die SIO-Variable neu über den Applikation Manager verlinkt werden muss. Erstellen Sie dafür einen SIO Socket Anschluss (siehe Abschnitt 3.3.1).



Klicken Sie und öffnen Sie den Applikations-Manager.
 Klicken Sie auf „VAL3 Applikation“ → „Festplatte“ → „Programm“
 → „Globale Variablen“ → „sio“ → <Variablenname>.
 Klicken Sie auf „Link (F2)“ → „Edit (F6)“ → Socket → „TCP Clients“
 → „Socket Name“ → „OK (F8)“ → „OK (F8)“ → „Save (F8)“

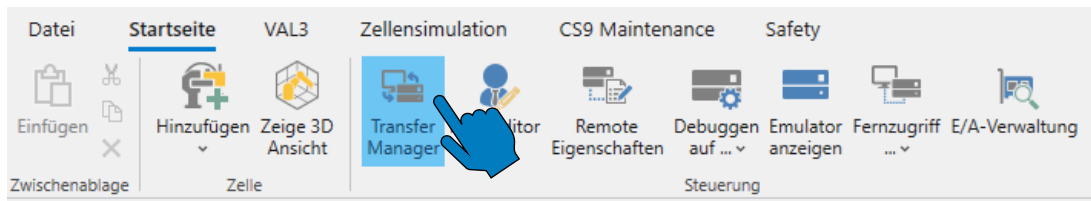
3.4 Übertragung der VAL3-Anwendung mit dem Transfer Manager

Die VAL3-Anwendungen und die benötigten Konfigurationen werden mit dem Transfer Manager von Stäubli SRS auf die CS9-Steuerung übertragen.

Die Ethernet-Schnittstellen müssen getrennt konfiguriert werden. Beide Schnittstellen müssen sich in unterschiedlichen Netzwerken befinden:

- J204 (Transfer Manager): 172.31.0.1
- J205 (GRIPLINK TCP/IP-Kommunikation): 192.168.1.1

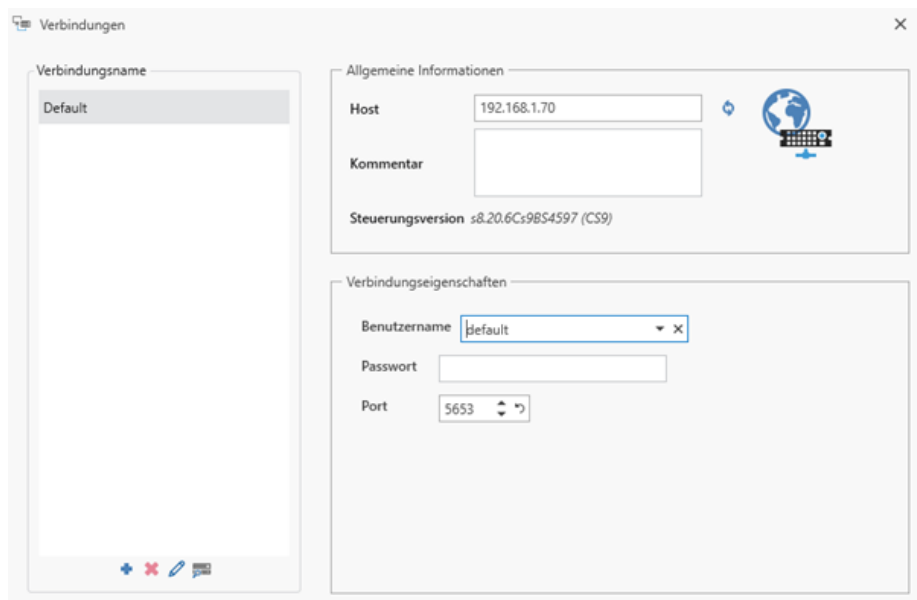
In Stäubli SRS wird der Transfer Manager über die obere Menüleiste geöffnet:



3.4.1 Verbindung zur CS9-Steuerung konfigurieren

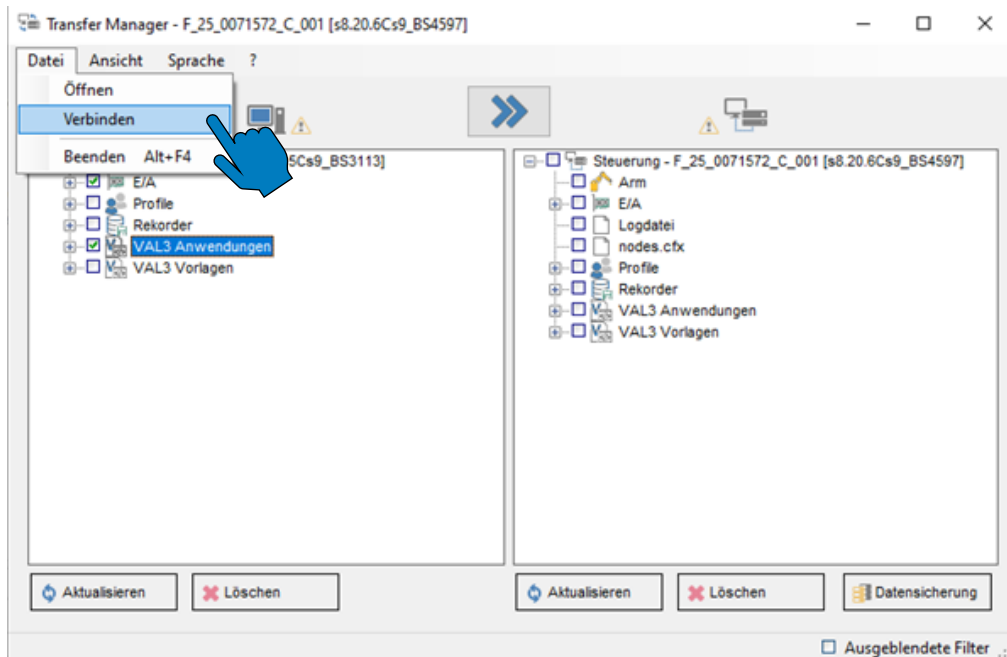
Im Verbindungsdialog wird die Verbindung zur CS9-Steuerung eingerichtet. Als Host wird die IP-Adresse der Steuerung eingetragen.

Die Steuerungsversion wird nach erfolgreichem Verbindungsaufbau angezeigt:



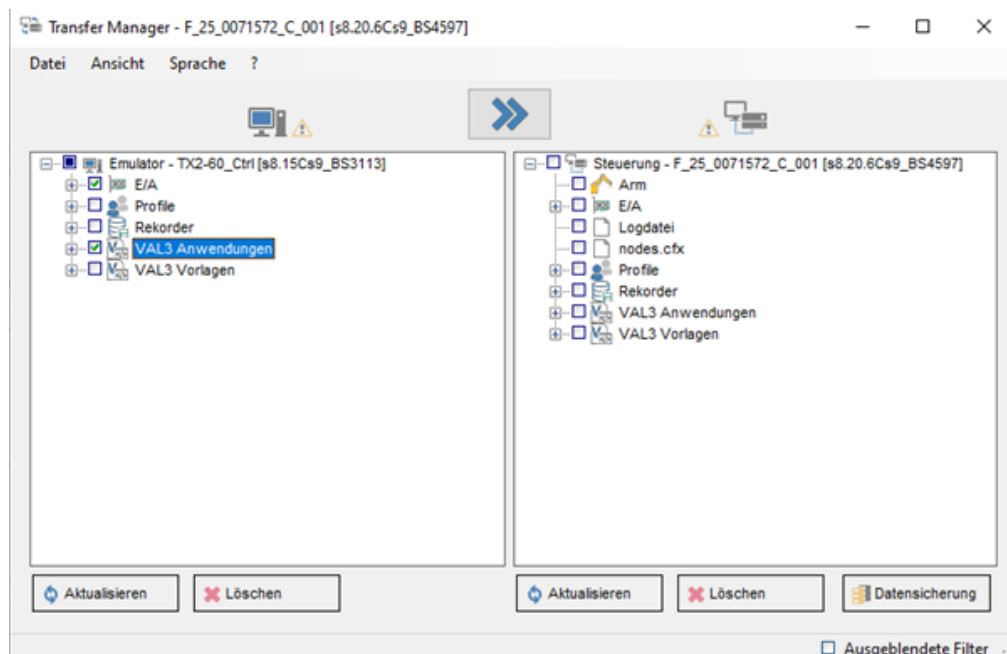
3.4.2 Verbindung im Transfer Manager herstellen

Nach dem Öffnen des Transfer Managers wird über Datei → Verbinden die Verbindung zur CS9-Steuerung hergestellt. Auf der linken Seite wird das lokale Projekt bzw. der Emulator angezeigt, auf der rechten Seite die verbundene CS9-Steuerung.



3.4.3 Zielsteuerung prüfen

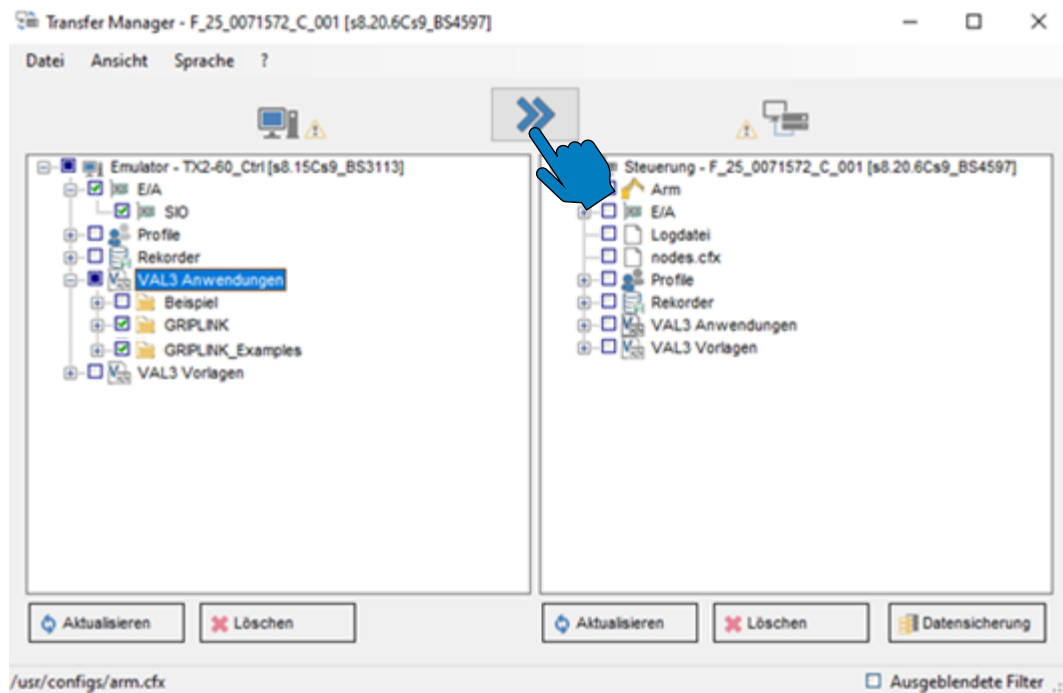
Nach erfolgreicher Verbindung ist die CS9-Steuerung im rechten Bereich des Transfer Managers sichtbar. Die Steuerung wird mit ihrem Softwarestand angezeigt. Damit ist bestätigt, dass der Transfer Manager mit der Zielsteuerung verbunden ist.



3.4.4 Benötigte Komponenten auswählen

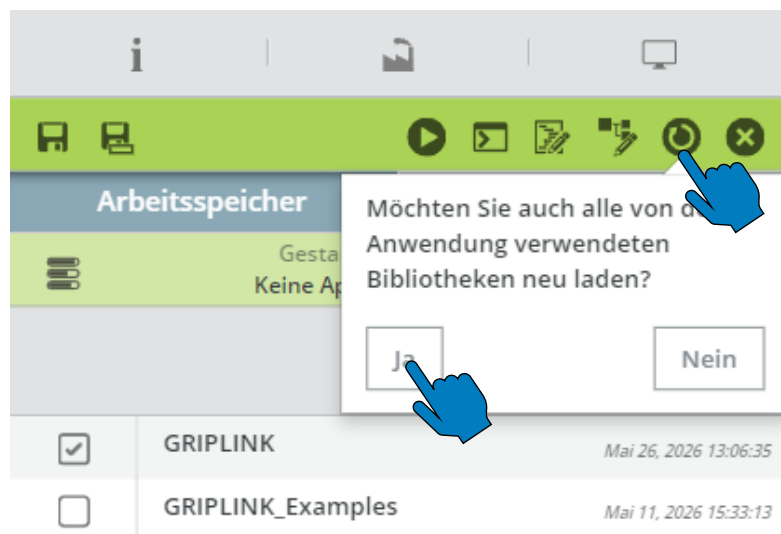
Im Transfer Manager werden die für die Übertragung benötigten Komponenten ausgewählt. Diese Komponenten werden auf die Disk der CS9-Steuerung übertragen.

- E/A (SIO)
- VAL3-Anwendung GRIPLINK
- VAL3-Anwendung GRIPLINK_Examples



3.4.5 Anwendung auf dem SP2-Panel neu laden

Nach der Übertragung befinden sich die VAL3-Anwendungen zunächst nur auf der Disk der CS9-Steuerung. Damit die Anwendung ausgeführt werden kann, wird sie auf dem SP2-Panel neu geladen. Dadurch werden die Anwendung und die verwendeten Bibliotheken von der Disk in den Arbeitsspeicher der Steuerung übernommen.





Die Übertragung speichert die Dateien zunächst auf der Disk der Steuerung. Damit die Anwendung ausgeführt werden kann, muss sie anschließend auf dem SP2-Panel neu geladen werden.

3.5 Verhalten im Fehlerfall

Jeder Befehl des GRIPLINK-Plugins liefert seinen Statuscode über den per Referenz übergebenen Parameter `x_nErrorCode` zurück. Der Wert ist nach jedem Aufruf auszuwerten und im Fehlerfall im Anwenderprogramm zu behandeln.

Mögliche Werte des Statuscodes:

- -1: keine bzw. nicht auswertbare Antwort des GRIPLINK-Controllers.
- 0 (E_SUCCESS): Befehl wurde erfolgreich ausgeführt.
- 1 – 27: Statuscodes des GRIPLINK-Controllers (siehe Anhang B).



Ein Verbindungsverlust oder Socket-Timeout wird als VAL3-Laufzeitfehler gemeldet (z. B. ERR 60, ERR 123, ERR 125).

Empfohlenes Aufrufmuster:

Nach jedem GRIPLINK-Aufruf `x_nErrorCode` prüfen und bei einem Wert ungleich 0 die Routine `errorHandling()` aufrufen.

Die Vorlage `errorHandling.pgx` im Projekt `GRIPLINK_Examples` dekodiert den Statuscode in einen Log-Eintrag und bietet ein Muster an, an der die anwendungsspezifische Fehlerreaktion (`stopMove`, `movej`, `taskSuspend`, `taskKill`) ergänzt werden kann.

```
call griplink:grip(x_nPortGripper, 0, 1, l_nErrorCode)
if l_nErrorCode != 0
  call errorHandling(l_nErrorCode)
  return
endif
```

Eine Übersicht über typische Fehlermeldungen, deren Ursachen sowie mögliche Lösungsansätze finden Sie in Abschnitt 5.1.

3.6 Not-Halt

Der aktuelle Not-Halt-Zustand kann über die VAL3-Funktion `esStatus()` (siehe VAL3-Referenzhandbuch) abgefragt werden. Je nach Anwendung kann die geeignete Reaktion z. B. das Deaktivieren des Greifers, das Belassen im Zustand HOLDING oder eine Meldung an den Anlagenbetreiber sein.



Ein Not-Halt unterbricht die Bewegung des Roboters. Die TCP/IP-Verbindung zum GRIPLINK-Controller wird in VAL3 dabei jedoch nicht abgebaut.



Die geeignete Reaktion auf einen Not-Halt richtet sich nach dem Sicherheitskonzept der Anwendung. Die Entscheidung, wie die TCP/IP-Verbindung und der Greifer im Not-Halt reagieren, ist im Anwenderprogramm zu implementieren.

4 Befehlsreferenz

Das GRIPLINK-Plugin stellt eine Reihe von Funktionen zur Ansteuerung und Überwachung von GRIPLINK-kompatiblen Geräten bereit. Über diese Befehle können Greifmodule aktiviert, gesteuert sowie Zustände und Prozesswerte abgefragt werden.

Es stehen sowohl Einzel- als auch Mehrfachbefehle zur Verfügung. Die Befehle werden mit dem vom Anwender definierten Alias der Bibliotheksreferenz aufgerufen.

In dieser Befehlsreferenz werden Parameter, wie in VAL3, in zwei Formen unterschieden:

Elementwert

Ein Parameter per Elementwert übergibt nur den Wert einer Variablen an den Befehl. Änderungen im Befehl wirken sich nicht auf die ursprüngliche Variable im aufrufenden Programm aus.

Beispiele:

- num x_nPort
- num x_nIndex

Elementreferenz

Ein Parameter per Elementreferenz ist am Zeichen „&“ hinter dem Typ erkennbar. Dabei wird nicht nur ein Wert übergeben, sondern die Variable selbst. Der Befehl kann den Inhalt dieser Variablen direkt schreiben oder ändern. Auf diese Weise werden beim GRIPLINK-Plugin Ergebniswerte bereitgestellt.

Beispiele:

- num& x_nValue
- num& x_nDeviceState
- sio& x_sioSocket

Mehrfachbefehle

Mit den Mehrfachbefehlen (Präfix M) können mehrere Geräte gleichzeitig angesprochen werden. Diese Befehle eignen sich insbesondere für die Handhabung großer oder biegeschlaffer Werkstücke mit mehreren Greifmodulen.

Statuscode als Rückgabeparameter

Jeder Befehl (mit Ausnahme von DISCONNECT) liefert seinen Ausführungsstatus über den letzten Parameter x_nErrorCode (per Referenz, num) zurück. Die Semantik ist für alle Befehle identisch und in Abschnitt 3.5 beschrieben.

Prinzipieller Programmablauf

1. Verbindung mit dem GRIPLINK-Controller herstellen mit **CONNECT**.
Dieser Befehl muss einmal zu Beginn des Programms ausgeführt werden.
2. Optional: Geräte initialisieren oder referenzieren.

Bei bestimmten Gerätetypen, zum Beispiel Servogreifern ohne Absolutgeber, kann eine Referenzfahrt mit **HOME** erforderlich sein. Andere Gerätetypen, zum Beispiel Sensoren, benötigen keine Referenzierung.

3. Gerät aktivieren mit **ENABLE** und konfigurieren, sofern dies vom jeweiligen Gerät unterstützt wird.
4. Gerätefunktionen ausführen, zum Beispiel Greifen oder Freigeben bei Greifmodulen mit **GRIP** und **RELEASE** oder Zustände und Werte von Geräten abfragen.
5. Verbindung zum GRIPLINK-Controller trennen mit **DISCONNECT**, wenn die Schnittstelle physikalisch getrennt werden muss, zum Beispiel bei einem Werkzeugwechsel oder wenn das Programm beendet wird.

Um eine zuverlässige Kommunikation und die bestmögliche Leistung zu erreichen, sollte die Anwendung so umgesetzt werden, dass die Verbindung zum GRIPLINK-Controller während der gesamten Programmlaufzeit bestehen bleibt. In diesem Fall wird **CONNECT** einmal zu Beginn des Programms ausgeführt und **DISCONNECT** nur bei Programmende oder im Fehlerfall aufgerufen.

Beispiel: Minimalprogramm

```
// Connect to GRIPLINK controller
call griplink:connect(sioGriplink, l_nErrorCode)

// Enable gripper
call griplink:enable(0, l_nGripperState, true, l_nErrorCode)

// Grip part
call griplink:grip(0, 0, 1, l_nErrorCode)

// Release part
call griplink:release(0, 0, 1, l_nErrorCode)

// Disconnect
call griplink:disconnect()
```

Im Folgenden sind die verfügbaren Befehle des GRIPLINK-Plugins beschrieben. Für die Codebeispiele wird der Bibliothekspräfix „griplink“ verwendet (siehe Abschnitt 3.2).

4.1 Beispielprogramme – GRIPLINK_Examples

Zusätzlich zu den in dieser Befehlsreferenz aufgeführten Aufrufmustern liegt dem Plugin das vollständige VAL3-Projekt GRIPLINK_Examples bei. Es enthält ausführbare Beispielprogramme, die zeigen, wie die Befehle in realen Abläufen kombiniert werden.

- **SimpleGripRelease:** minimaler Greifzyklus aus connect, enable, home, grip, release.

- **AsyncPreposition:** nicht-blockierendes Vor-Positionieren des Greifers parallel zur Roboterbewegung (release mit x_nDoBlock = 0 und anschließend wstr).
- **CheckWorkpiece:** Greifzyklus mit Werkstückprüfung über value() zur Auswertung von IO/NIO.
- **FlexgripCycle:** Greifvorgang mit frei wählbaren Bewegungsparametern (Position, Kraft, Geschwindigkeit, Beschleunigung).
- **MultiGripRelease:** synchrones Greifen und Lösen mehrerer Greifer mit grip()/release() statt der veralteten M-Befehle.
- **FaultRecoveryViaDisable:** Quittieren im Fehlerzustand FAULT durch disable().
- **GRIPLINKDemo:** vollständiger Demoablauf, der mehrere der oben genannten Muster kombiniert.
- **errorHandling:** Vorlage für die Fehlerbehandlung (siehe Abschnitt 3.5). Diese Routine wird aus den anderen Beispielprogrammen aufgerufen, wenn x_nErrorCode einen Fehlerwert liefert.

Es wird empfohlen, neue Anwendungen auf Basis dieser Beispielprogramme zu erstellen, anstatt die einzelnen Aufrufmuster aus dieser Referenz von Grund auf nachzubilden. Die Beispielprogramme sind so aufgebaut, dass sie als Vorlage in das eigene VAL3-Projekt übernommen und an die jeweilige Anwendung angepasst werden können.

4.2 Verbindung aufbauen – CONNECT

Dieser Befehl stellt die Verbindung zwischen dem GRIPLINK-Controller und der Robotersteuerung her. Dazu wird eine Socket-IO-Variable per Elementreferenz übergeben, über die die weiteren Befehle ausgeführt werden. Diese Variable muss zuvor mit einem TCP/IP-Socket verknüpft worden sein (siehe Abschnitt 3.3).

Der Befehl prüft weiterhin, ob der angeschlossene GRIPLINK-Controller die Mindestanforderungen des GRIPLINK Protokolls erfüllt (siehe Abschnitt 1.3).

Syntax

```
connect(sio& x_sioSocket, num& x_nErrorCode)
```

Parameter per Elementreferenz

sio& x_sioSocket	Referenz auf die Socket-Variable des verwendeten TCP/IP-Sockets
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Häufige Fehlerursachen

Socket-Variable nicht korrekt initialisiert

→ Einstellungen prüfen

GRIPLINK-Controller nicht mit Roboter verbunden

→ Netzwerkanschluss und Spannungsversorgung prüfen

GRIPLINK-Controller anders konfiguriert

→ Netzwerkeinstellungen des GRIPLINK-Controllers über die Weboberfläche prüfen

Protokollversion des GRIPLINK-Controllers passt nicht zum GRIPLINK-Plugin

→ Firmware des GRIPLINK-Controllers aktualisieren

Beispiel

Stellt die Verbindung zwischen der Robotersteuerung und dem GRIPLINK-Controller über die Socket-Variable sioGriplink her:

```
call griplink:connect(sioGriplink, l_nErrorCode)
```



Der Befehl CONNECT muss vor jedem anderen GRIPLINK-Befehl aufgerufen werden. Wenn GRIPLINK-Befehle vor einem CONNECT ausgeführt werden, löst dies den Fehler „ERR 123 IO-Variable nicht mit einem Ein- oder Ausgang verknüpft“ aus.

4.3 Verbindung trennen – DISCONNECT

Dieser Befehl trennt die Verbindung zwischen Robotersteuerung und GRIPLINK-Controller. Dies kann notwendig werden, wenn über einen Werkzeugwechsler die Netzwerkverbindung zum GRIPLINK-Controller unterbrochen wird.



Das Trennen einer bestehenden Verbindung ist im normalen Programmablauf nicht erforderlich und sollte nur mit Bedacht verwendet werden!

Syntax

```
disconnect()
```

Parameter

keine

Beispiel

Verbindung zwischen Roboter und dem GRIPLINK-Controller trennen, wenn ein Werkzeugwechsel erforderlich ist:

```
call griplink:disconnect()
```

```
// Werkzeugwechsel hier durchführen
```

```
call griplink:connect(sioGriplink, l_nErrorCode)
```

4.4 Prüfung des angeschlossenen Geräts – DEVASSERT

Dieser Befehl prüft, ob am ausgewählten Port das vorgesehene Gerät angeschlossen ist. Besitzt das am Port angeschlossene Gerät nicht die angegebene Vendor- und Product-ID, so wird ein Statuscode in den Parameter x_nErrorCode geschrieben.

Syntax

```
devassert(num x_nPort, num x_nVid, num x_nPid, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nVid	In der IODD des erwarteten Geräts spezifizierte Vendor-ID
num x_nPid	In der IODD des erwarteten Geräts spezifizierte Product-ID
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

Stelle sicher, dass an Port 0 das Gerät STERIGRIP 200-030 von Weiss Robotics (VID 815, PID 52) angeschlossen ist:

```
call GRIPLINK:devassert(0, 815, 52, l_nErrorCode)
```



Die Vendor- und Product-ID entnehmen Sie der vom Hersteller zur Verfügung gestellten Gerätebeschreibung bzw. der IODD.

4.5 Gerät aktivieren – ENABLE

Dieser Befehl aktiviert das am gewählten Port angeschlossene Gerät. IO-Link Greifmodule wechseln dabei in den Zustand RELEASED und führen die Freigabeaktion des zuletzt gewählten Griff-Presets aus.

Mit dem Parameter `x_bWstrEnabled` kann das Warten auf einen Zustandsübergang (WSTR) aktiviert werden. Wenn WSTR aktiviert ist, wartet die Befehlsausführung, bis der Gerätezustand am gewählten Port auf ENABLED oder FAULT wechselt. Erfolgt kein Zustandswechsel, endet die Befehlsausführung mit TIMEOUT.

Wenn WSTR aktiviert ist, wird der Gerätezustand nach der Befehlsausführung in die per Elementreferenz übergebene Variable `x_nDeviceState` geschrieben.

Syntax

```
enable(num x_nPort, num& x_nDeviceState, bool x_bWstrEnabled, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
bool x_bWstrEnabled	Auf „true“ setzen, um WSTR zu aktivieren Auf „false“ setzen, um WSTR nicht zu nutzen

Parameter per Elementreferenz

num& x_nDeviceState	Enthält nach erfolgreicher Befehlsausführung den Gerätezustand, sofern WSTR aktiviert ist
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

Aktiviert das Gerät an Port 0 und schreibt den neuen Gerätezustand in `l_nDeviceState`:

```
call GRIPLINK:enable(0, l_nDeviceState, true, l_nErrorCode)
```



WSTR sollte nur verwendet werden, wenn sich das Gerät zuvor im Zustand DISABLED befindet, da es ansonsten zum Timeout-Fehler bei der Befehlsausführung kommt. Nur wenn WSTR aktiviert wurde wird der neue Gerätezustand in die Rückgabevariable geschrieben!



Das Verhalten kann je nach angeschlossenem Gerät unterschiedlich sein. Beachten Sie die Hinweise in den Anleitungen des jeweiligen Gerätetreibers!

4.6 Gerät deaktivieren – DISABLE

Dieser Befehl deaktiviert das am gewählten Port angeschlossene Gerät. Der Befehl kann beispielsweise genutzt werden, um die Finger eines Greifmoduls kraftfrei zu schalten oder den Laserstrahl eines Distanzsensors temporär zu deaktivieren.

Mit dem Parameter `x_bWstrEnabled` kann das Warten auf einen Zustandsübergang (WSTR) aktiviert werden. Wenn WSTR aktiviert ist, wartet die Befehlsausführung, bis der Gerätezustand am gewählten Port auf DISABLED oder FAULT wechselt; passiert das nicht, endet sie mit einem TIMEOUT-Fehler.

Fehlerquittierung mittels DISABLE

Mit DISABLE können auch Gerätefehler quittiert werden, beispielsweise wenn das angeschlossene Gerät sich im FAULT-Zustand befindet.

Syntax

```
disable(num x_nPort, num& x_nDeviceState, bool x_bWstrEnabled, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
bool x_bWstrEnabled	Auf „true“ setzen, um WSTR zu aktivieren Auf „false“ setzen, um WSTR nicht zu nutzen

Parameter per Elementreferenz

num& x_nDeviceState	Enthält nach erfolgreicher Befehlsausführung den Gerätezustand, sofern WSTR aktiviert ist
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

An Port 0 ist ein Servogreifer angeschlossen. Beim Systemstart soll vor dem Referenzieren auf einen etwaigen Fehlerzustand geprüft und dieser quittiert werden:

```
// Read current device state
call griplink:deviceState(0, l_nGripperState, l_nErrorCode)

// Check for FAULT state (7)
if l_nGripperState == 7

    // Gripper is in FAULT state
    call griplink:disable(0, l_nGripperState, true, l_nErrorCode)
    delay(1)
    call griplink:enable(0, l_nGripperState, true, l_nErrorCode)
endif
```

```
call griplink:home(0, l_nErrorCode)
```



WSTR sollte nur verwendet werden, wenn sich das Gerät zuvor nicht im Zustand DISABLED befindet, da es ansonsten zur Zeitüberschreitung bei der Befehlsausführung kommt.



Nur wenn WSTR aktiviert wurde wird der neue Gerätezustand in die Rückgabevariable geschrieben!

4.7 Greifmodul referenzieren – HOME

Referenziert das ausgewählte Gerät. Der Befehl führt die integrierte Referenzierungs-Funktion des Geräts aus und wartet, bis diese abgeschlossen ist.

Syntax

```
home(num x_nPort, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
-------------	-----------------------------

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Referenziert den Greifer an Port 0:

```
call GRIPLINK:home(0, l_nErrorCode)
```



Das Verhalten kann je nach angeschlossenem Gerät unterschiedlich sein. Beachten Sie die Hinweise in den Anleitungen des jeweiligen Gerätetreibers!

4.8 Werkstück greifen – GRIP

Der Befehl GRIP führt einen Griff mit dem angegebenen Griff-Preset aus. Über den Parameter `x_nDoBlock` kann festgelegt werden, ob der Befehl unmittelbar nach dem Start der Bewegung oder erst nach deren Abschluss zurückkehrt:

`x_nDoBlock = 0` → Der Befehl kehrt sofort nach Start der Bewegung zurück (nicht blockierend).

`x_nDoBlock = 1` → Der Befehl wartet bis der Greifvorgang abgeschlossen ist (blockierend).

Bei nicht blockierender Ausführung (`x_nDoBlock = 0`) kann zusätzlich mit dem WSTR-Befehl auf eine Zustandsänderung gewartet werden, um das Ende der Freigabe zu erkennen.



Die Greifparameter können über die Weboberfläche des GRIPLINK-Controllers oder über den Befehl „SET GRIPCFG“ (siehe Abschnitt 4.12) konfiguriert werden.

Syntax

```
grip(num x_nPort, num x_nPreset, num x_nDoBlock, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
num x_nDoBlock	Wenn 0, kommt der Befehl direkt nach Start der Bewegung zurück. Wenn 1 wartet GRIPLINK, bis das Greifen abgeschlossen wurde

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Greifmodul an Port 0 soll mit Griff-Preset 2 greifen. Wurde kein Werkstück gefunden, soll der Greifer warten, wieder öffnen und es erneut versuchen:

```
l_bLoop = true
while l_bLoop
    // Grip
    call griplink:grip(0, 2, 1, l_nErrorCode)
    call griplink:deviceState(0, l_nDeviceStatePort0, l_nErrorCode)

    if l_nDeviceStatePort0 == 5
        // Holding a part
        logMsg("Gripper in HOLDING state", 1)
```

```
        delay(1)
        l_bLoop = false

    elseif l_nDeviceStatePort0 == 4
        logMsg("Gripper in NO PART state", 2)
        delay(1)

        // Release
        call griplink:release(0, 2, 1, l_nErrorCode)

    elseif l_nDeviceStatePort0 == 7
        logMsg("Gripper in FAULT state", 3)
        return
    else
        logMsg("Gripper in invalid state", 3)
        return
    endif
endWhile
```

4.9 Werkstück freigeben – RELEASE

Der Befehl RELEASE gibt ein zuvor gegriffenes Werkstück mit den Parametern des angegebenen Griff-Presets frei. Über den optionalen Parameter x_nDoBlock kann gesteuert werden, ob der Befehl unmittelbar nach dem Start der Bewegung oder erst nach deren Abschluss zurückkehrt:

x_nDoBlock = 0 → Der Befehl kehrt sofort nach Start der Bewegung zurück (nicht blockierend).

x_nDoBlock = 1 → Der Befehl wartet bis die Freigabe abgeschlossen ist (blockierend).

Bei nicht blockierender Ausführung (x_nDoBlock = 0) kann zusätzlich mit dem WSTR-Befehl auf eine Zustandsänderung gewartet werden, um das Ende der Freigabe zu erkennen.



Die Greifparameter können über die Weboberfläche des GRIPLINK-Controllers oder über den Befehl „SET GRIPCFG“ (siehe Abschnitt 4.12) konfiguriert werden.

Syntax

```
release(num x_nPort, num x_nPreset, num x_nDoBlock, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
num x_nDoBlock	Wenn 0, kommt der Befehl direkt nach Start der Bewegung zurück. Wenn 1, wartet GRIPLINK, bis die Freigabebewegung abgeschlossen wurde

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Siehe Beispiel in 4.8

4.10 Werkstück über Parameter greifen – FLEXGRIP

Der Befehl FLEXGRIP führt einen Greifvorgang mit frei wählbaren Bewegungsparametern aus.

Im Gegensatz zum GRIP-Befehl wird hier keine vordefinierte Griffkonfiguration (Preset) verwendet, sondern der Bewegungsablauf vollständig über die Befehlsparameter bestimmt.

Damit kann der Anwender individuelle Zielpositionen, Kräfte, Geschwindigkeiten und Beschleunigungen vorgeben, ohne auf ein gespeichertes Griff-Preset zurückgreifen zu müssen.

Der Befehl kann wahlweise blockierend oder nicht blockierend ausgeführt werden.

Bei blockierender Ausführung kehrt der Befehl erst nach Abschluss des Greifvorgangs zurück.

Nach Aufruf des Befehls wird der Greifvorgang mit den angegebenen Bewegungsparametern gestartet. Wird der Parameter `x_nDoBlock = 0` gesetzt, kehrt der Befehl sofort zurück, während die Bewegung im Hintergrund ausgeführt wird. Bei `x_nDoBlock = 1` wartet der Befehl, bis der Vorgang abgeschlossen ist (Werkstück gegriffen oder kein Werkstück erkannt).

Syntax

```
flexgrip(num x_nPort, num x_nPosition, num x_nForce,  
num x_nSpeed, num x_nAcceleration, num x_nDoBlock, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nPosition	Zielposition für „No Part Limit“ in mm
num x_nForce	Greifkraft in N
num x_nSpeed	Greifgeschwindigkeit in mm/s (0 = nutze automatische Berechnung)
num x_nAcceleration	Fingerbeschleunigung in mm/s ² (0 = nutze Standardwert)
num x_nDoBlock	0 = nicht blockierend, 1 = blockierend

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Greifen eines Werkstücks mit dem Greifer an Port 0 und 60 N Kraft, No-Part-Limit = 15 mm, Geschwindigkeit = 45,5 mm/s, Standardbeschleunigung, nicht blockierend:

```
call griplink:flexgrip(0, 15.0, 60, 45.5, 0, 0, l_nErrorCode)
```

Blockierende Variante:

```
call griplink:flexgrip(0, 15.0, 60, 45.5, 0, 1, l_nErrorCode)
```

4.11 Werkstück über Parameter freigeben – FLEXRELEASE

Der Befehl FLEXRELEASE gibt ein zuvor gegriffenes Werkstück mit frei wählbaren Bewegungsparametern aus. Er kann sowohl für eine einfache Freigabe als auch für eine gezielte Vorpositionierung eines Servogreifers verwendet werden. Die Bewegungsparameter werden vollständig über die Befehlsargumente definiert. Nach Aufruf des Befehls öffnet das Greifmodul die Finger bis zur angegebenen Zielposition.

Bei nicht blockierender Ausführung kehrt der Befehl sofort zurück, während die Bewegung im Hintergrund läuft. Bei blockierender Ausführung wartet der Befehl bis zum Abschluss der Bewegung. Der Zustand ändert sich nach Abschluss typischerweise zu DS_RELEASED.

Syntax

```
flexrelease(num x_nPort, num x_nPosition, num x_nSpeed, num x_nAcceleration, num  
            x_nDoBlock, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nPosition	Zielposition (Release-Limit) in mm
num x_nSpeed	Öffnungsgeschwindigkeit in mm/s (0 = nutze Standardwert)
num x_nAcceleration	Fingerbeschleunigung in mm/s ² (0 = nutze Standardwert)
num x_nDoBlock	0 = nicht blockierend, 1 = blockierend

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Vorpositionierung des Greifers an Port 0 auf 19,5 mm mit Standardparametern, nicht blockierend:

```
call griplink:flexrelease(0, 19.5, 0, 0, 0, l_nErrorCode)
```

Blockierende Variante:

```
call griplink:flexrelease(0, 19.5, 0, 0, 1, l_nErrorCode)
```

4.12 Parametrierung eines Griff-Presets – SETGRIPCFG

Der Befehl SETGRIPCFG dient zur Anpassung der Parameter eines Griff-Presets. Ein Griff-Preset definiert die Bewegungs- und Kraftparameter, die ein Greifmodul beim Ausführen eines GRIP- oder RELEASE-Befehls verwendet.

Für jedes Griff-Preset können bis zu acht Parameter konfiguriert werden.



Die Bedeutung und zulässigen Wertebereiche dieser Parameter sind gerätespezifisch und müssen der jeweiligen Geräte- oder Treiberdokumentation entnommen werden.

Syntax

```
setgripcfg(num x_nPort, num x_nPreset, string x_sTag, num& x_nParams[8],  
num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
string x_sTag	Preset-Tag

Parameter per Elementreferenz

num& x_nParams[8]	Array mit 8 Elementen für die gerätespezifische Preset-Parameter 0 bis 7.
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel: Parameter für Greifmodule von Weiss Robotics über GRIPLINK-ET4

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
string x_sTag	Preset-Tag
x_nParams[0]	No Part Limit (Wertebereich hängt von Greifermodell ab, in mm)
x_nParams[1]	Release Limit (Wertebereich hängt von Greifermodell ab, in mm)
x_nParams[2]	Force Factor (Wertebereich hängt von Greifermodell ab, in % [0..100])
x_nParams[4..7]	Nicht benutzt

Beispiel

Setze beim IEG PLUS Greifmodul an Port 3 des GRIPLINK-ET4 Controllers die Parameter des Griffs mit dem Index 1 auf:

Preset-Tag= "bearing"

No Part Limit = 31 mm

Release Limit = 41 mm

Force Factor = 59 %

```
// l_nGripPresetParameters defined as array of 8 integers  
l_nGripPresetParameters[0] = 31  
l_nGripPresetParameters[1] = 41  
l_nGripPresetParameters[2] = 59  
call griplink:setgripcfg(3, 1, "bearing", l_nGripPresetParameters,  
l_nErrorCode)
```

4.13 Aktuellen Gerätezustand abfragen – DEVICESTATE

Dieser Befehl liest den aktuellen Gerätezustand und schreibt ihn in einen per Elementreferenz übergebenen Parameter.

Syntax

```
deviceState(num x_nPort, num& x_nDeviceState, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
-------------	-----------------------------

Parameter per Elementreferenz

num& x_nDeviceState	Referenz auf Rückgabe-Variable des Gerätezustands
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

Warte, bis der Gerätezustand von Greifmodul an Port 2 „HOLDING“ (5) ist:

```
// Grip with gripper
call griplink:grip(2, 0, 0, l_nErrorCode)
while l_nGripperState != 5
    call griplink:deviceState(2, l_nGripperState, l_nErrorCode)
    delay(0.1)
endWhile
```

4.14 Warten auf eine Änderung des Gerätezustands – WSTR

Der Befehl WSTR wartet auf eine Änderung des Gerätezustands am angegebenen Port. Er kann beispielsweise verwendet werden, um zu prüfen, ob eine asynchron ausgeführte Freigabebewegung (RELEASE mit x_nDoBlock=0) abgeschlossen ist. Der Befehl blockiert, bis am angegebenen Gerät eine Zustandsänderung auftritt. Anschließend wird der neue Zustand in Parameter per Elementreferenz x_nDeviceState zurückgegeben.



Tritt die Zustandsänderung nicht in einem vorgegebenen Zeitintervall ein, wird ein Timeout-Fehler ausgelöst und die Programmausführung unterbrochen.

Syntax

```
wstr(num x_nPort, num& x_nDeviceState, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
-------------	-----------------------------

Parameter per Elementreferenz

num& x_nDeviceState	Referenz auf Rückgabe-Variable des Gerätezustands
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

Positioniere die Greiffinger mit Preset 2 vor, während der Roboter über die Aufnahme position fährt. An dieser angekommen, wird gewartet, bis die Finger geöffnet sind, um dann an die Aufnahme position zu fahren, um das Werkstück zu greifen.

```
begin
  num l_nGripperState

  // Start pre-positioning the fingers
  call griplink:release(0, 2, 0, l_nErrorCode)

  // Move to approach point
  movej()
  waitEndMove()

  // Wait until the gripper changed its state
  call griplink:wstr(0, l_nGripperState, l_nErrorCode)
  if l_nGripperState != 3
    logMsg("Gripper not in RELEASED state", 3)
    return
  endif

  // Move to pick point
```

```
movej()  
waitEndMove()  
  
// Grip workpiece  
call griplink:grip(0, 2, 1, l_nErrorCode)  
end
```

4.15 Abfrage eines Gerätwertes – VALUE

Dieser Befehl liest einen indizierten Gerätwert aus und speichert ihn im Parameter per Elementreferenz.



Der zulässige Wertebereich für den Wertindex kann in der GRIPLINK-Treiber-Beschreibung des jeweiligen Geräts nachgelesen werden.

Syntax

```
value(num x_nPort, num x_nValueIndex, num& x_nValue, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nValueIndex	Index des Gerätwertes (zulässiger Wertebereich ist geräteabhängig)

Parameter per Elementreferenz

num& x_nValue	Enthält nach erfolgreicher Ausführung den gelesenen Gerätwert
num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

Lese vom Gerät an Port 3, einem Servogreifer, den Primärwert (Value-Index 0, hier: Fingerposition) aus und speichere den Wert in der lokalen Variable l_nPartWidth. Nutze diesen für eine IO/NIO-Prüfung:

```
l_nGripperState = 0
l_nPartWidth     = 0

// Grip workpiece
call griplink:grip(3, 0, 1, l_nErrorCode)

// Get part width
call griplink:value(3, 0, l_nPartWidth, l_nErrorCode)

// Is part width 30 +/- 0.1?
if l_nPartWidth < 29.9 or l_nPartWidth > 30.1
  // Part is NIO -> Move to Drop position
  movej()
  call griplink:release(3, 0, 1, l_nErrorCode)
else
  // Part is IO -> Move to Use position
  movej()
  call griplink:release(3, 0, 1, l_nErrorCode)
endif
```

4.16 Setzen eines Gerätewertes – SETVALUE

Mit diesem Befehl kann ein gerätespezifischer Wert gesetzt werden. Der Wert ist über einen Index auswählbar, welcher einen vom Gerät abhängigen gültigen Wertebereich hat. Der zu schreibende Wert kann je nach angeschlossenem Gerät verschiedene Bedeutungen haben.



Wertebereiche von Wertindex und zu schreibendem Wert entnehmen Sie der Treiber-Anleitung des jeweiligen Geräts.

Syntax

```
setvalue(num x_nPort, num x_nValueIndex, num x_nValue, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nValueIndex	Index des Gerätewertes (zulässiger Wertebereich ist geräteabhängig)
num x_nValue	Wert (zulässiger Wertebereich ist geräteabhängig)

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Setze im Gerät an Port 0 den Wert mit dem Index 2 auf 10.5:

```
call griplink:setvalue(0, 2, 10.5, l_nErrorCode)
```

4.17 Warten auf einen Gerätewert – WAITVAL

Der Befehl WAITVAL wartet, bis ein bestimmter Gerätewert den zuvor gesetzten Sollwert erreicht oder innerhalb eines definierten Toleranzbereichs liegt. Er kann beispielsweise verwendet werden, um zu prüfen, ob ein Sensor- oder Positionswert einen Zielbereich erreicht hat, bevor das Programm fortgesetzt wird.

Der Befehl blockiert, bis eine der folgenden Bedingungen erfüllt ist:

- Der überwachte Wert liegt innerhalb des vorgegebenen Toleranzfensters, oder
- das Timeout wurde überschritten.



Der Befehl WAITVAL ist mit der GRIPLINK-Protokollversion 3 kompatibel. Der überwachte Wert entspricht dem über VALUE auslesbaren Indexwert.



Erreicht der zu überwachende Wert das Detektionsfenster nicht im vorgegebenen TIMEOUT-Zeitintervall, wird ein Timeout-Fehler ausgelöst und die Programmausführung unterbrochen.

Syntax

```
waitval(num x_nPort, num x_nIndex, num x_nThreshold, num x_nWindowSize, num  
x_nTimeout, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nIndex	Index des Gerätewertes (zulässiger Wertebereich ist geräteabhängig)
num x_nThreshold	Zielwert, auf den gewartet werden soll
num x_nWindowSize	Größe des Toleranzfensters um den Zielwert
num x_nTimeout	Timeout in ms

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Warte, bis der an Port 2 angeschlossene Distanzsensor am Werteindex 0 den Zielwert 100,0 innerhalb eines Toleranzfensters von $\pm 0,5$ erreicht. Nach 5000 ms wird mit Timeout abgebrochen.

```
call griplink:waitval(2, 0, 100.0, 0.5, 5000, l_nErrorCode)
```

4.18 Greifkraftherhaltung steuern – CLAMP

Mit dem CLAMP-Befehl können Greifmodule mit integrierter Greifkraftsicherung im Zustand HOLDING den Antrieb stromlos schalten, um den Strombedarf des Greifmoduls zu reduzieren. Das Werkstück wird weiterhin sicher gehalten.

Wird CLAMP aktiviert, schaltet sich der Antrieb im Zustand HOLDING ab. Wird CLAMP deaktiviert, bleibt im Zustand HOLDING die Greifkraftregelung aktiv und die Steuerung des Greifmoduls regelt die Greifkraft permanent nach.



Der CLAMP Befehl ist nicht bei allen Greifmodulen verfügbar.

Syntax

```
clamp(num x_nPort, num x_nEnable, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nEnable	CLAMP-Zustand

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Aktiviere die Klemmung im CRG 200-085 von Weiss Robotics an Port 1:

```
call griplink:clamp(1, 1, l_nErrorCode)
```

4.19 Ansteuerung der Status-LED – LED

Dieser Befehl aktiviert ein vordefiniertes LED-Preset der Status-LED eines angeschlossenen Geräts. Dadurch kann der Betriebszustand oder der Fortschritt eines Prozesses direkt am Endeffektor visuell signalisiert werden.



LED-Presets können über die Weboberfläche des GRIPLINK-Controllers konfiguriert werden.



Der LED-Befehl wird nicht von allen Geräten unterstützt.

Syntax

```
led(num x_nPort, num x_nPreset, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des LED-Presets (zulässiger Wertebereich ist geräteabhängig)

Parameter per Elementreferenz

num& x_nErrorCode	Liefert den Statuscode zurück: -1 bei nicht auswertbarer oder leerer Controller-Antwort; 0 bei Erfolg; 1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).
-------------------	--

Beispiel

Führe mit dem Greifmodul an Port 2 einen Greifbefehl aus und lese die Greifposition aus. Ändere die Farbe des Leuchtrings auf das Leuchtmuster 0, wenn die Fingerposition danach größer gleich 8,1 mm und auf Leuchtmusters 1, wenn kleiner:

```
// Grip with gripper at port 3 and preset 0, wait for state transition  
call griplink:grip(2, 0, 1, l_nErrorCode)
```

```
// Read finger position and set respective LED preset  
call griplink:value(2, 0, l_nGripperPosition, l_nErrorCode)
```

```
if (l_nGripperPosition >= 8.1)  
    call GRIPLINK:led(2, 1, l_nErrorCode)  
else  
    call GRIPLINK:led(2, 2, l_nErrorCode)  
endif
```

4.20 Veraltete Befehle

In diesem Abschnitt sind Befehle aufgeführt, die aus Gründen der Abwärtskompatibilität weiterhin im GRIPLINK-Plugin enthalten sind, deren Verwendung jedoch nicht mehr empfohlen wird.

Diese Befehle wurden durch funktional gleichwertige, jedoch flexiblere Mechanismen ersetzt. Insbesondere ermöglichen die neueren Befehle eine feinere Steuerung des Programmablaufs sowie eine bessere Anpassung an unterschiedliche Anwendungen und Gerätetypen.

Für neue Programme sollte daher auf die Verwendung dieser veralteten Befehle verzichtet werden. Stattdessen wird empfohlen, die entsprechenden aktuellen Befehle und Programmiermuster zu verwenden.

4.20.1 Gleichzeitiges Greifen von Werkstücken – MGRIP

Dieser Befehl führt mit den ausgewählten Greifmodulen einen simultanen Greifbefehl mit dem gewählten Griff-Preset-Index aus.

Mit dem Funktionsargument `x_bMwaitforEnabled` kann das Warten auf einen Zustandsübergang (MWAITFOR) der jeweiligen Geräte aktiviert werden. Dies hat zur Folge, dass die Befehlsausführung wartet, bis sich der Zustand aller gewählten Geräte geändert hat. Erfolgt kein Zustandswechsel, kommt der Befehl mit dem Fehlercode `TIMEOUT` zurück.



Dieser Befehl ist veraltet und wird nur aus Kompatibilitätsgründen bereitgestellt. Für neue Anwendungen wird empfohlen, stattdessen GRIP mit `x_nDoBlock = 0` in Kombination mit WSTR zu verwenden.

Syntax

```
mgrip(num x_nPreset, num x_nPorts, bool x_bMwaitforEnabled, num&
x_nErrorCode)
```

Parameter per Elementwert

num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
num x_nPorts	Ausgewählte Greifmodule als 32-Bit-Vektor: Bit x = 1: Gerät an Port x ist selektiert Bit x = 0: Gerät an Port x ist nicht selektiert

Beispiel: Greifer an Ports 0 und 3 selektieren

→ Bit 0 und 3 setzen

→ $\text{Ports} = (1 \ll 0) + (1 \ll 3) = 1 + 8 = 9$

bool x_bMwaitforEnabled Auf „true“ setzen, um MWAITFOR zu aktivieren

Auf „false“ setzen, um MWAITFOR nicht zu nutzen

Parameter per Elementreferenz

num& x_nErrorCode Liefert den Statuscode zurück:
-1 bei nicht auswertbarer oder leerer Controller-Antwort;
0 bei Erfolg;
1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

Greifmodule an Port 0 und 3 greifen Werkstück mit Griff-Preset 2 und anschließend auf Zustandsänderungen warten:

```
// Grip with grippers at ports 0 and 3
call griplink:mgrip(2, 9, true, l_nErrorCode)

// Get states of the grippers
call griplink:deviceState(0, l_nDeviceStateGripper0, l_nErrorCode)
call griplink:deviceState(3, l_nDeviceStateGripper3, l_nErrorCode)
...
// Release with grippers at ports 0 and 3
call griplink:mrelease(2, 9, true, l_nErrorCode)

// Get actual states of the grippers
call griplink:deviceState(0, l_nDeviceStateGripper0, l_nErrorCode)
call griplink:deviceState(3, l_nDeviceStateGripper3, l_nErrorCode)
```

4.20.2 Gleichzeitiges Freigeben von Werkstücken – MRELEASE

Dieser Befehl führt mit den ausgewählten Greifmodulen einen simultanen Freigabebefehl mit dem gewählten Griff-Preset-Index aus.

Mit dem Funktionsargument x_bMwaitforEnabled kann das Warten auf einen Zustandsübergang (MWAITFOR) der jeweiligen Geräte aktiviert werden. Dies hat zur Folge, dass die Befehlsausführung wartet, bis sich der Zustand aller gewählten Geräte geändert hat. Erfolgt kein Zustandswechsel, kommt der Befehl mit dem Fehlercode TIMEOUT zurück.



Dieser Befehl ist veraltet und wird nur aus Kompatibilitätsgründen bereitgestellt. Für neue Anwendungen wird empfohlen, stattdessen RELEASE mit x_nDoBlock = 0 in Kombination mit WSTR zu verwenden.

Syntax

```
mrelease(num x_nPreset, num x_nPorts, bool x_bMwaitforEnabled, num&
x_nErrorCode)
```

Parameter per Elementwert

num x_nPreset Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)

num x_nPorts

Ausgewählte Greifmodule als 32-Bit-Vektor:

Bit x = 1: Gerät an Port x ist selektiert

Bit x = 0: Gerät an Port x ist nicht selektiert

Beispiel: Greifer an Ports 0 und 3 selektieren

→ Bit 0 und 3 setzen

→ Ports = $(1 \ll 0) + (1 \ll 3) = 1 + 8 = 9$

bool x_bMwaitForEnabled Auf „true“ setzen, um WSTR zu aktivieren

Auf „false“ setzen, um WSTR nicht zu nutzen

Parameter per Elementreferenz

num& x_nErrorCode

Liefert den Statuscode zurück:

-1 bei nicht auswertbarer oder leerer Controller-Antwort;

0 bei Erfolg;

1..27 = Statuscodes des GRIPLINK-Controllers (siehe Anhang B).

Beispiel

Siehe Beispiel in 4.20.1

4.20.3 Warten auf die Änderung mehrerer Gerätezustände – MWAITFOR

Der Befehl MWAITFOR wartet darauf, dass mehrere Geräte ihren Zustand ändern.



Tritt die Zustandsänderung bei mindestens einem ausgewählten Port nicht in einem vorgegebenen Zeitintervall ein, wird ein Timeout-Fehler ausgelöst und die Programmausführung unterbrochen.



Dieser Befehl ist veraltet und wird nur aus Kompatibilitätsgründen bereitgestellt. Für neue Anwendungen wird empfohlen, stattdessen WSTR zu verwenden.

Syntax

```
mwaitfor(num x_nPorts, num& x_nErrorCode)
```

Parameter per Elementwert

num x_nPorts	Ausgewählte Greifmodule als 32-Bit-Vektor: Bit x = 1: Gerät an Port x ist selektiert Bit x = 0: Gerät an Port x ist nicht selektiert
--------------	--

Beispiel: Greifer an Ports 0 und 3 selektieren

→ Bit 0 und 3 setzen

→ Ports = $(1 \ll 0) + (1 \ll 3) = 1 + 8 = 9$

Beispiel

```
// Wait for devices at port 0 and 3 to change their state  
call griplink:mwaitfor(9, l_nErrorCode)
```

5 Fehlersuche

Das GRIPLINK-Plugin gibt im Betrieb Fehlermeldungen aus. Im Folgenden werden wichtige Meldungen, mögliche Ursachen und Lösungswege erläutert.

5.1 Fehlermeldungen

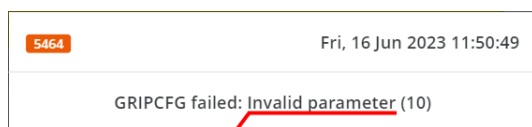
5.1.1 GRIPLINK-Fehlerantwort mit Statuscode

Der GRIPLINK-Controller meldet fehlerhafte Befehlsausführungen mit einer ERR-Antwort. Die Bedeutung der einzelnen Statuscodes ist in **Anhang B** beschrieben.

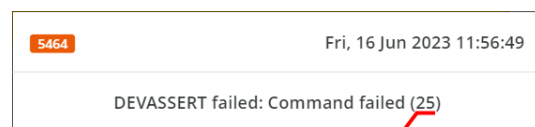
Das GRIPLINK-Plugin wertet diese Antwort aus und zeigt sie im Logger in folgender Form an:

<Befehlsname> failed: <Fehlerbeschreibung> (<Statuscode>)

Meldungen im Logger:



Fehlerbeschreibung



Fehlercode

Mögliche Ursachen

- Der GRIPLINK-Controller hat den Befehl mit einer ERR-Antwort abgelehnt.
- Der Befehlsaufruf im Roboterprogramm ist fehlerhaft.
- Ein Funktionsargument ist ungültig oder liegt außerhalb des zulässigen Wertebereichs.
- Der Befehl ist im aktuellen Gerätezustand nicht zulässig.

Behebung

- Statuscode und Fehlerbeschreibung prüfen, siehe **Anhang B**.
- Befehlsaufruf im VAL3-Programm prüfen.
- Korrekte Funktionsargumente verwenden und zulässige Wertebereiche beachten.
- Gerätezustand prüfen, z. B. ob das Gerät referenziert, verbunden und betriebsbereit ist.

5.1.2 Socket-Variable nicht mit Hardware-I/O verknüpft

Diese Meldung erscheint, wenn die verwendete Socket-Variable in VAL3 nicht korrekt mit einem Hardware-I/O verknüpft wurde.

5404	Fri, 16 Jun 2023 15:33:53
VAL3 runtime error. IO not linked to a hardware input-output.. VialFilling~.	

5404	Fri, 16 Jun 2023 15:33:53
ERR 123 IO not linked to a hardware input-output.. VialFilling~.	

Mögliche Ursache

- Die Socket-Variable ist nicht verlinkt.

Behebung

- Socket-Variable korrekt verlinken, siehe Abschnitt 3.3.2.

5.1.3 Instruction timeout bei fehlender Socket-Kommunikation

Diese Meldung erscheint, wenn während eines Befehlsaufrufs keine erfolgreiche Socket-Kommunikation mit dem GRIPLINK-Controller zustande kommt oder die Antwort nicht innerhalb der erwarteten Zeit empfangen wird.

5404	Tue, 05 May 2026 10:37:22
ERR 60 Instruction timeout. GRIPLINK_Dev~.	

5404	Tue, 05 May 2026 10:37:22
VAL3 runtime error. Instruction timeout. GRIPLINK_Dev~.	

Mögliche Ursachen

- Der GRIPLINK-Controller ist über die Socket-Verbindung nicht erreichbar.
- Die Socket-Verbindung wurde nicht erfolgreich geöffnet.
- Die IP-Adresse oder der Port ist falsch konfiguriert.
- Der GRIPLINK-Controller antwortet nicht innerhalb der erwarteten Zeit.
- Die Kommunikation wurde während der Befehlsausführung unterbrochen.

Behebung

- Ethernet-Verbindung, IP-Adresse und Port prüfen.
- Socket-Konfiguration und Verbindungsaufbau im VAL3-Programm prüfen.
- Spannungsversorgung und Betriebszustand des GRIPLINK-Controllers prüfen.
- Verbindung erneut herstellen und Programm neu starten.

5.1.4 Socket-Kommunikation während des Betriebs unterbrochen

Diese Meldung erscheint, wenn zwischen Robotersteuerung und GRIPLINK-Controller bereits eine Socket-Verbindung aufgebaut wurde, die Kommunikation während des Betriebs jedoch unterbrochen wird.

5404	Tue, 05 May 2026 11:18:18
VAL3 runtime error. No answer from input-output.. GRIPLINK_Dev~.	

5404	Tue, 05 May 2026 11:18:18
ERR 125 No answer from input-output.. GRIPLINK_Dev~.	

Mögliche Ursache

- Die Netzwerkverbindung wurde während des Betriebs unterbrochen.
- Das Netzkabel wurde abgezogen oder ist beschädigt.
- Eine Steckverbindung ist lose oder nicht korrekt eingerastet.
- Die Spannungsversorgung des GRIPLINK-Controllers wurde unterbrochen oder ist instabil.

Behebung

- Ethernet-Kabel und Steckverbindungen prüfen.
- Kabel auf festen Sitz, Kabelbruch oder Beschädigung prüfen.
- Spannungsversorgung prüfen und sicherstellen, dass der GRIPLINK-Controller dauerhaft versorgt wird.
- Nach einer Unterbrechung die Verbindung erneut herstellen.
- Programm neu starten.

Anhang A Gerätezustand

Gerätezustand	Code	Beschreibung
NOT CONNECTED	0	Greifmodul nicht verbunden
NOT INITIALIZED	1	Greifmodul nicht initialisiert
DISABLED	2	Betriebsbereit, nicht aktiv
RELEASED	3	Werkstück freigegeben
NO PART	4	Kein Werkstück gefunden
HOLDING	5	Werkstück wird gehalten
ENABLED	6	Sensor betriebsbereit
FAULT	7	Fehlerzustand

Anhang B Statuscodes

Tritt während der Ausführung eines GRIPLINK-Befehls ein Fehler auf, wird ein entsprechender Statuscode ausgegeben. Dieser beschreibt die Ursache des Fehlers und kann zur Diagnose im Roboterprogramm oder während der Inbetriebnahme herangezogen werden. Die folgende Tabelle enthält eine Übersicht der möglichen Statuscodes sowie deren Bedeutung.

Status code	Identifier	Description
-1	-	No response from the controller, or the response could not be processed. Set by the plugin.
0	E_SUCCESS	No error. Command successfully executed.
1	E_OVERRUN	Data overrun
2	E_RANGE_ERROR	Value out of range
3	E_NOT_AVAILABLE	Function or data not available
4	E_NOT_INITIALIZED	Device not initialized
5	E_TIMEOUT	Timeout
6	E_INSUFFICIENT_RESOURCES	Not enough memory available
7	E_CHECKSUM_ERROR	Checksum error
8	E_ACCESS_DENIED	Access denied
9	E_INVALID_HANDLE	Invalid handle
10	E_INVALID_PARAMETER	Invalid parameter
11	E_INDEX_OUT_OF_BOUNDS	Index out of bounds
12	E_IO_ERROR	Generic I/O error
13	E_READ_ERROR	Read error
14	E_WRITE_ERROR	Write error
15	E_NOT_FOUND	Resource not found
16	E_NOT_OPEN	File or device not open
17	E_EXISTS	Resource already exists
18	E_NO_COMM	Connection error
19	E_STATE_CONFLICT	Invalid state

20	E_NOT_SUPPORTED	Command or function not supported
21	E_INCONSISTENT_DATA	Data inconsistent
22	E_CMD_SYNTAX	Syntax error
23	E_CMD_UNKNOWN	Unknown command
24	E_CMD_ABORTED	Command aborted
25	E_CMD_FAILED	Command failed
26	E_AXIS_BLOCKED	Axis is blocked
27	E_PENDING	Pending action / Not yet finished

© 2026 WEISS ROBOTICS GmbH & Co. KG. Alle Rechte vorbehalten.

GRIPLINK und PERMAGRIP sind eingetragene Marken der WEISS ROBOTICS GmbH & Co. KG. Alle weiteren Marken sind Eigentum ihrer jeweiligen Inhaber.

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.