WEISS ROBOTICS

GRIPLINK-PLUGIN FÜR STÄUBLI

Version 3.0.0

Juni 2023

Inhalt

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.

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 Gerätes 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. 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 GRIPLINK ab Firmwarestand 4.1.0.
Zum Betrieb wird eine der folgenden Stäubli Robotersteuerungen benötigt:

- CS9 (ab s8.10.6)

Folgende Roboter-Optionen werden zum Betrieb der Software benötigt:

- Stäubli SRS

Kontaktieren Sie Ihren FANUC 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

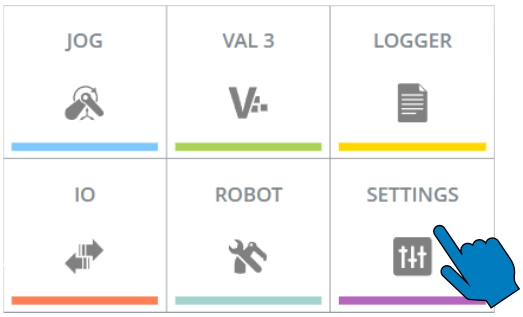
Das GRIPLINK-Plugin ist urheberrechtlich geschützt. Die jeweils gültigen Lizenzbestimmungen liegen dem Softwarepaket bei. Mit der Installation akzeptieren Sie diese Lizenzbestimmungen.

2 Installation

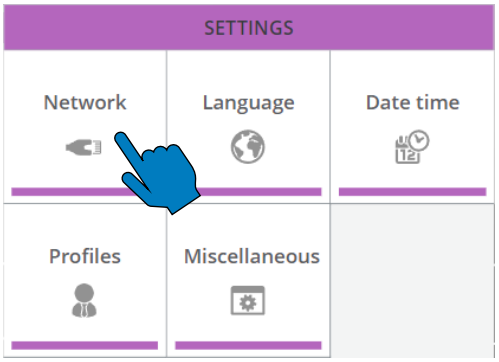
2.1 Vorbereitung des Roboters

Schalten Sie den Roboter ein und richten Sie die IP-Adresse ein (z.B. 192.168.1.14). Stellen Sie dabei sicher, dass Roboter und der GRIPLINK Controller im gleichen Netzwerk sind.
Führen Sie dazu folgende Schritte aus:

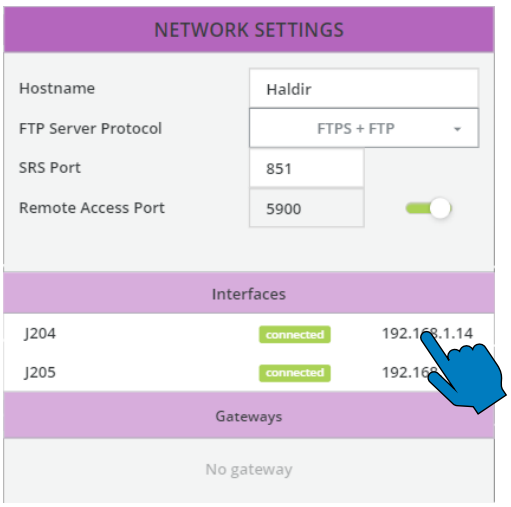
1. Klicken Sie auf 
2. Navigieren Sie zu „SETTINGS“

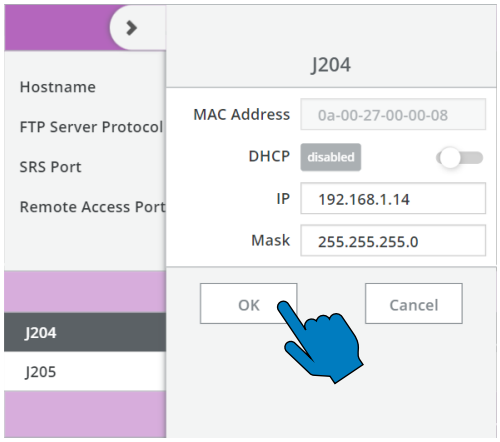


3. Navigieren Sie zu „Network“



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

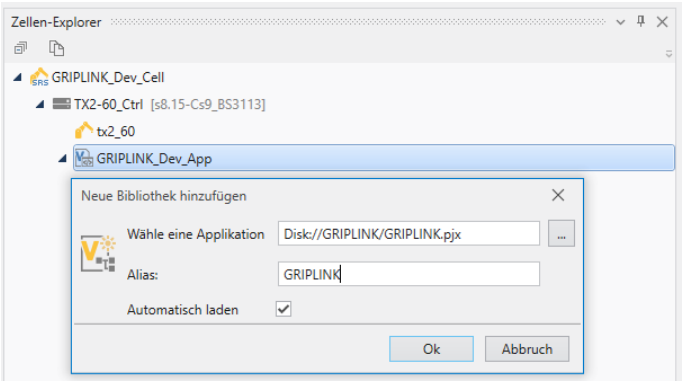




2.2 Installation der Software via STAUBLI Robotics Suite (SRS)

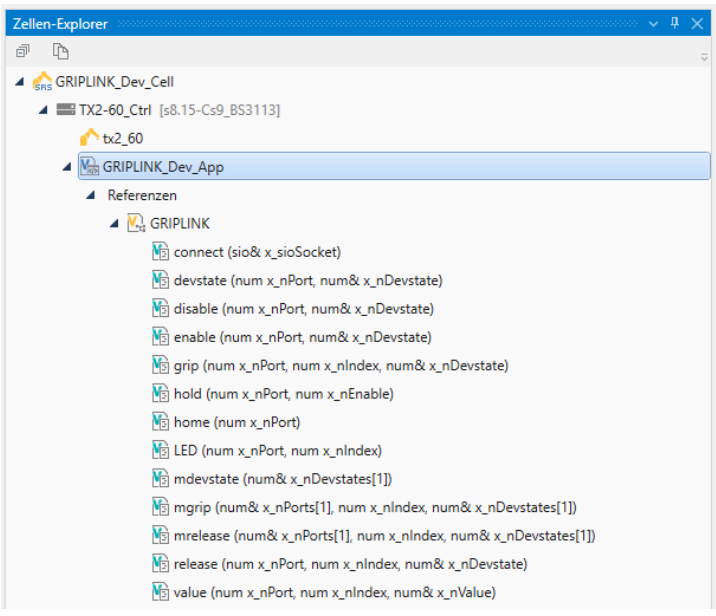
Stellen Sie sicher, dass Sie die aktuelle Version des GRIPLINK-Plugins verwenden. Die aktuelle Version kann unter **www.griplink.de/software** heruntergeladen werden.

1. Laden Sie die Plugin-Datei „griplink_staubli_<Version>.zip“ herunter.
2. Entpacken Sie das zuvor heruntergeladene ZIP-Archiv mit dem GRIPLINK-Plugin in das Verzeichnis „usrapp“:
   „…/<Zellenname>/<Controllername>/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“



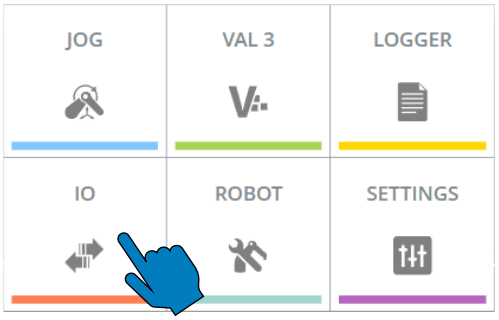
2.3 Einrichten des Socket-Anschlusses für den GRIPLINK Controller

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

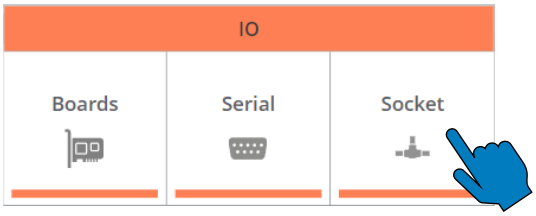
2.3.1 Teach Pendant

Führen Sie dazu folgende Schritte aus:

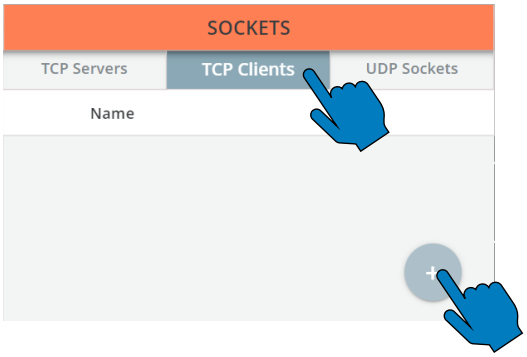
1. Klicken Sie auf 
2. Navigieren Sie zu „IO“

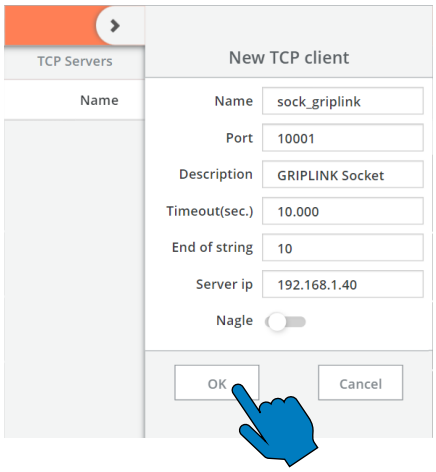


3. Navigieren Sie zu „Socket“



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	10	Sekunden
End of string	10	-

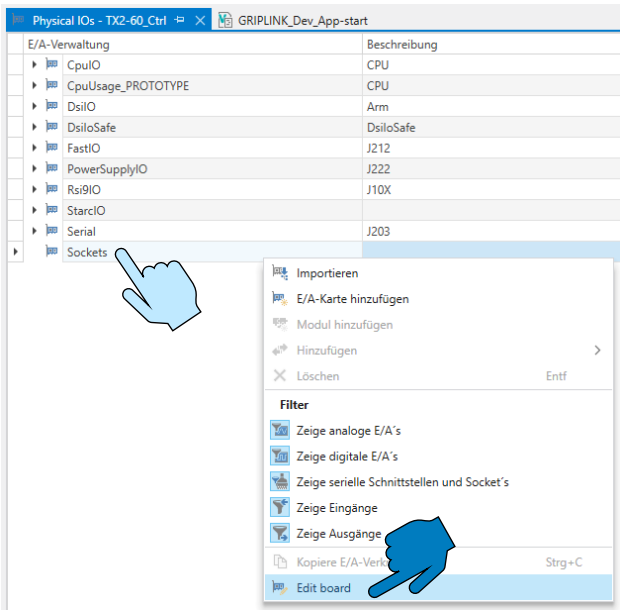
Achten Sie darauf, dass Sie den korrekten Namen eingeben, den Sie auch im Roboter-Programm verwenden.

Andere Werte können zu Fehlverhalten führen!

2.3.2 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. 

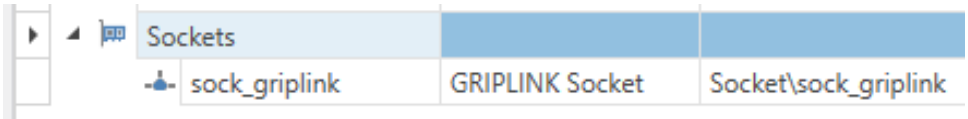
Stellen Sie folgende Werte ein:

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

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

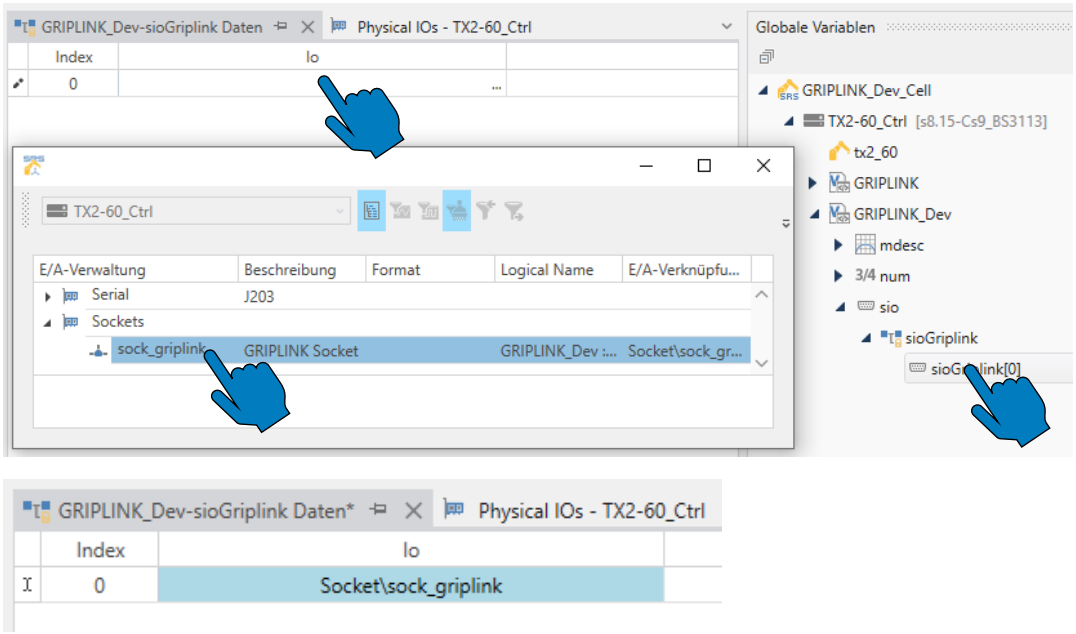
Andere Werte können zu Fehlverhalten führen!

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



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

7.



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 (siehe Abschnitt 4.1.3).
Erstellen Sie dafür einen SIO Socket Anschluss (siehe Abschnitt 2.3.1).

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)“

2.4 Verhalten im Fehlerfall

Tritt innerhalb des GRIPLINK-Plugins oder bei der Kommunikation mit dem GRIPLINK Controller ein Fehler auf, so wird grundsätzlich eine Fehlermeldung in den Logger geschrieben. In der Regel werden auch die laufenden Bewegungen des Roboters abgebrochen. Gleiches gilt, wenn das angesprochene Gerät sich im FAULT Zustand befindet oder aufgrund eines Befehls dahin wechselt.

Die häufigsten Fehlerursachen und mögliche Lösungsansätze sind in Abschnitt 4.1 aufgeführt.

3 Befehlsreferenz

Das GRIPLINK-Plugin stellt dem Anwender eine Sammlung an Greifmodul-spezifischen Funktionen bereit. Es stehen sowohl Einzel- als auch Mehrfachbefehle zur Verfügung. Die Befehle werden mit dem vom Anwender definierten Alias der Bibliotheksreferenz aufgerufen.

Rückgabewerte

Etwaige Rückgabewerte eines Befehls werden in Variablen geschrieben, die beim Befehlsaufruf als „Pass by Reference“ übergeben werden.

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.

Prinzipieller Programmablauf

1. Verbindung mit GRIPLINK Controller herstellen mit griplink:connect()
2. Bei Servogreifmodulen ohne Absolutgeber: Greifmodul referenzieren mit griplink:home()
3. Gerät aktivieren mit griplink:enable()
4. Greifen/Freigeben mit griplink:grip()/griplink:mgrip() bzw. griplink:release()/griplink:mrelease()
5. Vor Beenden des Programmes Verbindung trennen mit griplink:bye()

Der prinzipielle Programmablauf muss eingehalten werden, damit das Plugin und angeschlossene Geräte normal funktionieren.

Im Folgenden sind die verfügbaren Befehle des GRIPLINK-Plugins beschrieben. Für die Codebeispiele wird der Bibliothekspräfix „griplink“ verwendet (siehe Abschnitt 2.2).

3.1 Verbindung aufbauen – connect()

Dieser Befehl stellt die Verbindung zwischen GRIPLINK Controller und der Robotersteuerung her. Als Parameter wird die Socket-IO-Variable übergeben, mit der die weiteren Befehle ausgeführt werden. Diese muss zuvor entsprechend mit einem Socket verknüpft worden sein (siehe Abschnitt 2.3).

Wurde die Verbindung erfolgreich aufgebaut prüft das Plugin automatisch, ob der verbundene GRIPLINK Controller die Version des verwendeten GRIPLINK-Protokoll unterstützt.

Syntax

```
connect(sio& x_sioSocket)
```

Parameter

sio& x_sioSocket	Referenz auf Socket-Variable des verwendeten TCP/IP-Sockets

Beispiel

Verbindung zwischen Roboter und dem GRIPLINK über Socket-Variable `sioGriplink` herstellen:

```
call GRIPLINK:connect(sioGriplink)
```

Wenn GRIPLINK-Befehle vor einem CONNECT ausgeführt werden, wird eine Log-Nachricht geschrieben und die Programmausführung angehalten.

Der CONNECT-Befehl sollte nur ein einziges Mal zu Beginn des Roboterprogramms aufgerufen werden.

3.2 Verbindung trennen – BYE

Dieser Befehl trennt die Verbindung zwischen Robotersteuerung und GRIPLINK Controller.

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

Syntax

```
bye()
```

Beispiel

Verbindung zum GRIPLINK Controller trennen:

```
call GRIPLINK:bye()
```

3.3 Prüfung des angeschlossenen Geräts – DEVASSERT

Dieser Befehl prüft, ob am ausgewählten Port das zu erwartende Gerät angeschlossen ist. Besitzt das am Port angeschlossene Gerät nicht die erwartete Vendor- und Product-ID, so wird das Roboterprogramm sofort angehalten.

Syntax

```
devassert(num x_nPort, num x_nVID, num x_nPID)
```

Parameter

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

Beispiel

Versichere, dass an Port 3 das Gerät IEG 55-020 von Weiss Robotics (VID 815, PID 20) angeschlossen ist:

```
call GRIPLINK:devassert(3,815,20)
```

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

3.4 Gerät aktivieren – ENABLE

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

Mit dem Funktionsargument `x_bWstrEnabled` kann das Warten auf einen Zustandsübergang (WSTR) des jeweiligen Geräts aktiviert werden. Dies hat zur Folge, dass die Befehlsausführung wartet, bis sich der Zustand des Geräts am Port mit dem übergebenen Index zu ENABLED oder FAULT geändert hat. Erfolgt kein Zustandswechsel, kommt der Befehl mit dem Fehlercode TIMEOUT zurück.

Ist WSTR aktiviert, wird der Gerätezustand nach Befehlsausführung in die als Referenz übergebene Variable `x_nDevstate` geschrieben.

Syntax

```
enable(num x_nPort, num& x_nDevstate, bool x_bWstrEnabled)
```

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num& x_nDevstate	Referenz auf Rückgabe-Variable des Gerätezustands nach erfolgreicher Befehlsausführung und bei aktiviertem WSTR
bool x_bWstrEnabled	Auf „true“ setzen, um WSTR zu aktivieren Auf „false“ setzen, um WSTR nicht zu nutzen

Beispiel

Aktiviere das Gerät an Port 0 und speichere den neuen Gerätezustand in die Variable `l_nDevstate`:

```
call GRIPLINK:enable(0, l_nDevstate, true)
```

WSTR sollte nur verwendet werden, wenn sich das Gerät zuvor 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!

3.5 Gerät deaktivieren – DISABLE

Dieser Befehl deaktiviert das am gewählten Port angeschlossene Gerät.

Mit dem Funktionsargument `x_bWstrEnabled` kann das Warten auf einen Zustandsübergang (WSTR) des jeweiligen Geräts aktiviert werden. Dies hat zur Folge, dass die Befehlsausführung wartet, bis sich der Zustand des Geräts am Port mit dem übergebenen Index zu DISABLED oder FAULT geändert hat. Erfolgt kein Zustandswechsel, kommt der Befehl mit dem Fehlercode TIMEOUT zurück.

Ist WSTR aktiviert, wird der Gerätezustand nach Befehlsausführung in die als Referenz übergebene Variable `x_nDevstate` geschrieben.

Syntax

```
disable(num x_nPort, num& x_nDevstate, bool x_bWstrEnabled)
```

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num& x_nDevstate	Referenz auf Rückgabe-Variable des Gerätezustands nach erfolgreicher Befehlsausführung und bei aktiviertem WSTR
bool x_bWstrEnabled	Auf „true“ setzen, um WSTR zu aktivieren Auf „false“ setzen, um WSTR nicht zu nutzen

Beispiel

Deaktiviere das Gerät an Port 0 und speichere den neuen Gerätezustand in die Variable `l_nDevstate`:

```
call GRIPLINK:disable(0, l_nDevstate, true)
```

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!

3.6 Greifmodul referenzieren – HOME

Dieser Befehl referenziert den am gewählten Port angeschlossenen Greifer. Der Befehl ist blockierend und wartet, bis der Referenziervorgang abgeschlossen oder eine Zeitüberschreitung aufgetreten ist.

Der Gerätezustand wird nach Befehlsausführung abgefragt und in die als Referenz übergebene Variable `x_nDevstate` geschrieben.

Syntax

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

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num& x_nDevstate	Referenz auf Rückgabe-Variable des Gerätezustands nach erfolgreicher Befehlsausführung

Beispiel

Referenziere den Greifer an Port 0 und speichere den neuen Gerätezustand in der Variable `l_nDevstate`:

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

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

3.7 Werkstück greifen – GRIP

Dieser Befehl führt mit dem am gewählten Port angeschlossenen Greifer und dem gewählten Griff-Preset-Index einen Greifbefehl aus.

Mit dem Funktionsargument `x_bWstrEnabled` kann das Warten auf einen Zustandsübergang (WSTR) des jeweiligen Geräts aktiviert werden. Dies hat zur Folge, dass die Befehlsausführung wartet, bis sich der Zustand des Geräts am Port mit dem übergebenen Index zu HOLDING, NO PART oder FAULT geändert hat. Erfolgt kein Zustandswechsel, kommt der Befehl mit dem Fehlercode TIMEOUT zurück.

Ist WSTR aktiviert, wird der Gerätezustand nach Befehlsausführung in die als Referenz übergebene Variable `x_nDevstate` geschrieben.

Die Greifparameter können über die Weboberfläche des GRIPLINK Controllers oder über den Befehl „SET GRIPCFG“ (siehe Abschnitt 3.11) konfiguriert werden.

Syntax

```
grip(num x_nPort, num x_nPreset, num& x_nDevstate, bool x_bWstrEnabled)
```

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
num& x_nDevstate	Referenz auf Rückgabe-Variable des Gerätezustands nach erfolgreicher Befehlsausführung und bei aktiviertem WSTR
bool x_bWstrEnabled	Auf „true“ setzen, um WSTR zu aktivieren Auf „false“ setzen, um WSTR nicht zu nutzen

Beispiel

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

```
l_bLoop = true
while l_bLoop
    // Grip with gripper
    call griplink:grip(0, 1, nDevStatePort0, true)
    if nDevStatePort3 == 5
         logMsg("Gripper in HOLDING state", 1)
         delay(1)
         l_bLoop = false

    elseIf nDevStatePort3 == 7
         logMsg("Gripper in FAULT state", 3)
         return
    elseIf nDevStatePort3 == 4
         logMsg("Gripper in NO PART state", 2)
```

```
            delay(1)

            // Release with gripper
            call griplink:release(x_0, 2, nDevStatePort0, true)
        else
            logMsg("Gripper in invalid state", 3)
            return
        endIf
    endWhile
```

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

3.8 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.

Die Greifparameter können über die Weboberfläche des GRIPLINK Controllers oder über den Befehl „SET GRIPCFG“ (siehe Abschnitt 3.11) konfiguriert werden.

Syntax

```
mgrip(num x_nPort, num x_nPorts, bool x_bMwaitforEnabled)
```

Parameter

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 (1 << x) ist selektiert Bit x = 0: Gerät an Port (1 << x) ist nicht selektiert Beispiel: Greifer an Ports 0 und 3 selektieren →Bit 0 und 3 setzen →Ports = (1 << 0) + (1 << 3) = 1 + 8 = 9
bool x_bMwaitforEnabled	Auf „true“ setzen, um MWAITFOR zu aktivieren Auf „false“ setzen, um MWAITFOR nicht zu nutzen

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)
// Get states of the grippers
call griplink:devstate(0, l_nDevstateGripper0)
call griplink:devstate(3, l_nDevstateGripper3)
...
// Release with grippers at ports 0 and 3
call griplink:mrelease(2, 9, true)
// Get actual states of the grippers
call griplink:devstate(0, l_nDevstateGripper0)
call griplink:devstate(3, l_nDevstateGripper3)
```

3.9 Werkstück freigeben – RELEASE

Gibt ein Werkstück mit dem ausgewählten Greifmodul und dem ausgewählten Griff-Preset frei.

Mit dem Funktionsargument `x_bWstrEnabled` kann das Warten auf einen Zustandsübergang (WSTR) des jeweiligen Geräts aktiviert werden. Dies hat zur Folge, dass die Befehlsausführung wartet, bis sich der Zustand des Geräts am Port mit dem übergebenen Index zu RELEASED oder FAULT geändert hat. Erfolgt kein Zustandswechsel, kommt der Befehl mit dem Fehlercode TIMEOUT zurück.

Ist WSTR aktiviert, wird der Gerätezustand nach Befehlsausführung in die als Referenz übergebene Variable `x_nDevstate` geschrieben.

Die Greifparameter können über die Weboberfläche des GRIPLINK Controllers oder über den Befehl „SET GRIPCFG“ (siehe Abschnitt 3.11) konfiguriert werden.

Syntax

```
release(num x_nPort, num x_nPreset, num& x_nDevstate, bool x_bWstrEnabled)
```

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
num& x_nDevstate	Referenz auf Rückgabe-Variable des Gerätezustands nach erfolgreicher Befehlsausführung und bei aktiviertem WSTR
bool x_bWstrEnabled	Auf „true“ setzen, um WSTR zu aktivieren Auf „false“ setzen, um WSTR nicht zu nutzen

Beispiel

Siehe Beispiel in 3.7

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

3.10 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.

Die Greifparameter können über die Weboberfläche des GRIPLINK Controllers oder über den Befehl „SET GRIPCFG“ (siehe Abschnitt 3.11) konfiguriert werden.

Syntax

```
mrelease(num x_nPort, num x_nPorts, bool x_bMwaitforEnabled)
```

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num x_nPreset	Index des Griff-Presets (zulässiger Wertebereich ist geräteabhängig)
num& x_nDevstate	Referenz auf Rückgabe-Variable des Gerätezustands nach erfolgreicher Befehlsausführung und bei aktiviertem WSTR
bool x_bWstrEnabled	Auf „true“ setzen, um WSTR zu aktivieren Auf „false“ setzen, um WSTR nicht zu nutzen

Beispiel

Siehe Beispiel in 3.8

3.11 Parametrierung eines Griff-Presets – SETGRIPCFG

Dieser Befehl ändert die Parameter eines Griff-Presets. Bis zu acht Parameter sind dabei einstellbar, von denen immer alle zum Gerät gesendet müssen. Dies gilt auch für Geräte die über weniger als acht einstellbare Parameter verfügen.
Je nach Gerät besitzt jeder Parameter eine individuelle Bedeutung, die dem entsprechenden GRIPLINK-Gerätetreiber zu entnehmen ist.

Syntax

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

Parameter

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
num& x_nParams[---]	Array mit 8 Elementen für die Preset-Parameter

Parameter IO-Link-Greifmodule von Weiss Robotics

x_nParams[0]	No Part-Limit (-300000..300000) in 1/1000 mm
x_nParams[1]	Release-Limit (-300000..300000) in 1/1000 mm
x_nParams[2]	Force Factor (0..100) in 1/1000%
x_nParams[4..7]	Nicht verwendet, auf 0 setzen

Beispiel

Setze beim Greifmodul an Port 3 die Parameter des Griffs mit dem Index 1 auf:

No Part Limit = 31 mm →in 1/1000 mm: 31000
Release Limit = 41 mm →in 1/1000 mm: 41000
Force Factor = 59 %: →in 1/1000 %: 59000

```
// l_nGripPresetParameters defined as array of 8 integers
l_nGripPresetParameters[0] = 31000
l_nGripPresetParameters[1] = 41000
l_nGripPresetParameters[2] = 59000
call griplink:setgripcfg(3, 1, l_nGripPresetParameters)
```

3.12 Aktuellen Gerätezustand abfragen – DEVSTATE

Dieser Befehl liest den aktuellen Gerätezustand des ausgewählten Geräts zurück und speichert ihn in der Rückgabevariable.

Syntax

```
devstate(num x_nPort, num& x_nDevstate)
```

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num& x_nDevstate	Referenz auf Rückgabe-Variable des Gerätezustands

Beispiel

Prüfe den Zustand des Gerätes an Port 3 auf den Wert DISABLED (2):

```
call griplink:devstate(3, l_nDevstatePort3)
if (l_nDevstatePort3 == 2)
  // Process state DISABLED
endIf
```

3.13 Abfrage eines Gerätewertes – VALUE

Dieser Befehl liest einen indizierten Gerätewert aus und speichert ihn in der Rückgabevariable.

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

Syntax

```
value(num x_nPort, num x_nIndex, num& x_nValue)
```

Parameter

num x_nPort	Index des Geräts (0 bis 31)
num x_nIndex	Index des Gerätewertes (zulässiger Wertebereich ist geräteabhängig)
num& x_nValue	Referenz auf Rückgabe-Variable des Wertes

Beispiel

Prüfe den Primärwert des Gerätes an Port 3 auf den Mindestwert 14,000:

14 => 14 · 1.000 = 14.000 (in 1/1000, siehe GRIPLINK Protokoll-Spezifikation)

```
call griplink:value(3, l_nValue)
if (l_ nValue > 14000)
  // Value larger than 14.000
endIf
```

3.14 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_nIndex, num x_nValue)
```

Parameter

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

Beispiel

Setze im Gerät an Port 0 den Wert mit dem Index 2 auf 10:

10 => 10 · 1.000 = 10.000 (in 1/1000, siehe GRIPLINK Protokoll-Spezifikation)

```
call griplink:setvalue(0, 2, 10000)
```

3.15 Warten auf Erreichen eines Gerätewertes – WAITVAL

Mit diesem Befehl kann gewartet werden, bis ein gerätespezifischer Wert einen übergebenen Zielwert erreicht hat oder eine Zeitüberschreitung beim Warten auf den Zielwert aufgetreten ist. Der Wert ist über einen Index auswählbar, welcher einen vom Gerät abhängigen gültigen Wertebereich hat.

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

Syntax

```
waitval(num x_nPort, num x_nIndex, num x_nValue)
```

Parameter

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

Beispiel

Warte, bis der an Port 2 angeschlossene Distanzsensor den Distanzwert (Index 0) 100 erreicht hat:

```
// 100 => 100 · 1,000 = 100,000
call griplink:waitvalue(2, 0, 100000)
```

3.16 Greifkrafterhaltung 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_bEnabled)
```

Parameter

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

Beispiel

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

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

3.17 Ansteuerung der Status-LED – LED

Dieser Befehl setzt den Preset des Leuchtrings eines selektierten CRG-Greifmoduls.

Leuchtmuster können über die Weboberfläche des GRIPLINK Controllers konfiguriert werden.

Der LED-Befehl ist nicht bei allen Greifmodulen verfügbar.

Syntax

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

Parameter

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

Beispiel

Führe mit dem Greifmodul an Port 3 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(3, 0, true)
// Read finger position and set respective LED preset
call GRIPLINK:value(3, 0, l_nGripperPosition)
if (l_nGripperPosition >= 810)
    call GRIPLINK:led(1, 1)
else
    call GRIPLINK:led(1, 2)
endIf
```

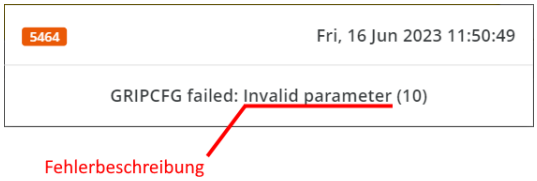
4 Fehlersuche

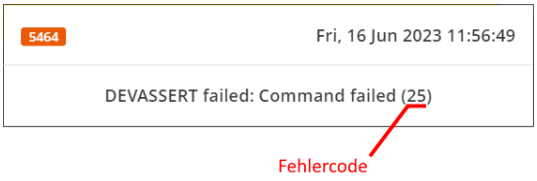
Das GRIPLINK-Plugin gibt im Betrieb Fehlermeldungen aus. Im Folgenden werden wichtige Meldungen und Lösungswege erläutert.

4.1 Fehlermeldungen

4.1.1 <Befehlsname> failed: <Fehlerbeschreibung> (<Fehlercode>)

Fehlermeldungen, die aufgrund eines Protokollfehlers erscheinen. Beispiele:

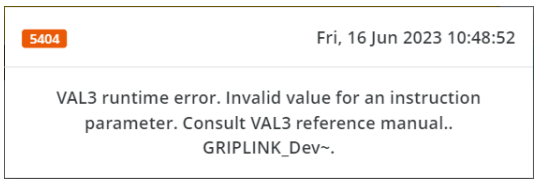


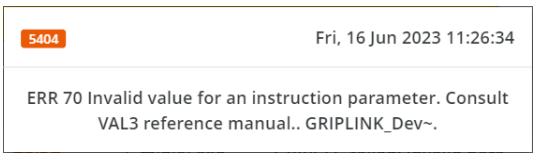


Mögliche Ursache	Behebung
Die Fehlerursache wird als Meldung und in Klammern als Fehlercode angezeigt	Es handelt sich um einen GRIPLINK-Protokollfehler • Fehlerbeschreibung und -code prüfen (siehe Dokument „GRIPLINK Unified Command Set Reference Manual“) • Befehlsaufruf im Roboterprogramm prüfen • Korrekte Funktionsargumente verwenden (Wertebereiche beachten!)

4.1.2 Log-Meldung „VAL3/ERR 70 invalid value for an instr. param.“

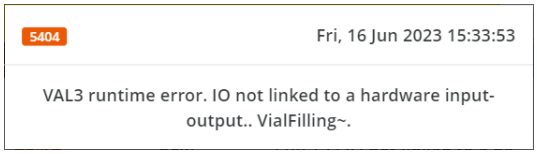
Fehlermeldungen, die während der Ausführung des VAL3-Programms auftreten.

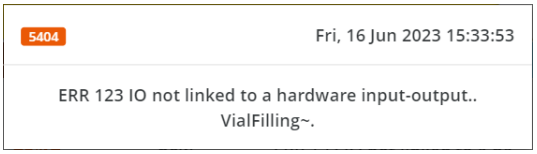




Mögliche Ursache	Behebung
Timeout durch WSTR-Befehl aufgetreten	• Falsche Befehlsreihenfolge verwendet (z.B. RELEASE-Befehl mit WSTR nach ENABLE-Befehl für den selben Port) →Befehlsreihenfolge anpassen →WSTR für den zweiten Befehl deaktivieren

4.1.3 Log-Meldung „VAL3/ERR 123 IO not linked to a hardware input-output“





Mögliche Ursache	Behebung
Die Socket-Variable ist nicht verlinkt	siehe Abschnitt 2.3.2

Anhang A Gerätezustand

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