Softwarehandbuch
2D Grasping-Kit
Smart Grasping – Natives Protokoll V3

Original-Softwarehandbuch

Hand in hand for tomorrow

Impressum

Technische Änderungen:
Änderungen im Sinne technischer Verbesserungen sind uns vorbehalten.

Dokumentennummer: 1591390

Auflage: 01.00 | 22.04.2024 | de

Sehr geehrte Kundin,

sehr geehrter Kunde,

vielen Dank, dass Sie unseren Produkten und unserem Familienunternehmen als führendem Technologieausrüster für Roboter und Produktionsmaschinen vertrauen.

Unser Team steht Ihnen bei Fragen rund um dieses Produkt und weiteren Lösungen jederzeit zur Verfügung. Fragen Sie uns und fordern Sie uns heraus. Wir lösen Ihre Aufgabe!

Mit freundlichen Grüßen

Ihr SCHUNK-Team

Customer Management
Tel. +49-7133-103-2503
Fax +49-7133-103-2189
cmg@de.schunk.com

Betriebsanleitung bitte vollständig lesen und produktnah aufbewahren.

Inhaltsverzeichnis

1 Natives Protokoll – Allgemeines

Low-Level-Pipeline-Schnittstelle für Robotersteuerungen.

Das native Protokoll ist ein Kommunikationsprotokoll, das auf dem TCP/IP-Stapel aufsetzt. Das Protokoll ist einfach und überschaubar aufgebaut, so dass es selbst auf veralteten Robotern und Industriesteuerungen implementiert werden kann.

1.1 Grundsätze

- **Port**: Der Server wartet auf TCP-Port 42001
- **Byte-Reihenfolge**: Die Reihenfolge von Multibyte-Feldern ist Big-Endian
- **Gleitkommazahlen**: Einfache IEEE-Norm (IEC-60559)
- **Ganzzahlen mit Vorzeichen**: Ganzzahlen mit Vorzeichen werden als Zweierkomplement dargestellt
- **Maßeinheiten**: Wenn nicht anders angegeben, werden Längen in Meter [m] und Winkel in Bogenmaß [rad] gemessen.

1.2 Datentypen

Folgende Datentypen sind definiert:

- **Ganzzahlen mit Vorzeichen**: int8, int16, int32
- **Ganzzahlen ohne Vorzeichen**: uint8, uint16, uint32
- **Gleitkommazahlen**: float32

Das Zahlensuffix eines Datentyps gibt seine Größe in Bits an (z. B. int16 ist zwei Bytes lang).

1.3 Versionierung

Der Server ist abwärtskompatibel.

Clients, die frühere Protokollversionen verwenden, werden wie erwartet bedient und bemerken nicht einmal die Versionsabweichung. Der Server antwortet immer mit derselben Versionsnummer im Präfix, die auch in den Anfragen verwendet wird.

Ist ein Server jedoch so veraltet, dass seine Protokollversion niedriger ist als die des Clients, antwortet der Server immer mit einer leeren Meldung. Das Längenfeld des Meldungspräfixes wird auf Null und das Versionsfeld auf die höchste unterstützte Protokollversion des Servers gesetzt. In diesem Fall werden die Clients darauf hingewiesen, den Anwender darauf aufmerksam zu machen, dass ihr Server veraltet und mit ihrer aktuellen Clientversion nicht kompatibel ist.

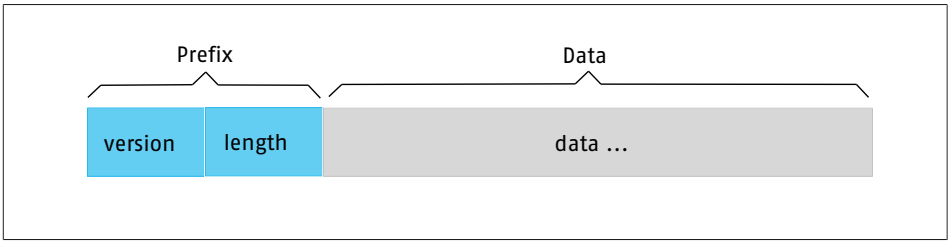
1.4 Aufbau

Jeder Meldungsrahmen besteht aus zwei Teilen: einem unveränderlichen Präfix, das über verschiedene Protokollversionen hinweg unveränderlich ist, und einem versionsspezifischen Teil. Einzelheiten zu diesen Teilen werden in den jeweiligen Kapiteln erläutert.

2 Natives Protokoll – Präfix

Unveränderlicher Teil der Low-Level-Pipeline-Schnittstelle für Robotersteuerungen.

2.1 Aufbau



2.2 Präfix

Alle Meldungen beginnen mit einem *Präfix*, das die Protokollversion und die Größe der restlichen Meldung definiert.

Das Präfix ist unveränderlich und bleibt in verschiedenen Protokollversionen gleich.

	Byte-Offset	Name	Typ
Präfix	0	Version	uint16
	2	Länge	uint32
Daten	6	...	...

- **Version**: Protokollversion.
- **Länge**: Die Größe der restlichen Meldung in Byte (max. 4 GB), ohne das *Präfix* selbst (6 B).

2.3 Daten

Eigentlicher Inhalt der Meldung. Die Struktur ist versionsspezifisch und wird in den Versionskapiteln beschrieben.

3 Natives Protokoll – V3

Version 3 der Low-Level-Pipeline-Schnittstelle für Robotersteuerungen.

3.1 Allgemeines

- Größe der Meldung: Jede Meldung ist 80 Byte lang. Immer. Daten
- Typen: Nur Ganzzahlen (mit und ohne Vorzeichen). Keine Gleitkommazahlen.

3.2 Aufbau

- Größe der Meldung: Jede Meldung ist 80 Byte lang. Immer. Daten
- Typen: Nur Ganzzahlen (mit und ohne Vorzeichen). Keine Gleitkommazahlen.

Byte-Offset	Daten
0	Präfix
6	Kopfzeile
10 – 79	Rumpf

Anfragen und Antworten folgen einem einheitlichen Aufbau, was eine statische Zuordnung zwischen Feldern und ihren Byte-Indizes in einem Rahmen ermöglicht. Je nach Kontext der Meldung tragen jedoch nur relevante Abschnitte des Rumpfes sinnvolle Werte, während andere Teile zu vernachlässigen sind. Die zu vernachlässigenden Teile werden als *Offset* oder Padding bezeichnet.

	Anfrage			*Antwort*		
	Byte-Offset	Daten	Typ	Byte-Offset	Daten	Typ
Präfix	0	Version	unit16	0	Version	uint16
	2	Länge	unit32	2	Länge	uint32
Kopf-zeile	6	Kommunikations-typ	unit8	6	Kommunikations-typ	uint8
	7	Antwortcode	unit8	7	Antwortcode	uint8
	8	Antwortzähler	unit8	8	Antwortzähler	uint8
Rumpf	9	Meldungstyp	unit8	9	Meldungstyp	uint8
	10	Client	unit8	10	Version	uint16
	11	Greifmodus	unit8	12	Status	uint8
	12	Objektklasse	uint16	13	Greifmodus	uint8
	14	Werkzeug	unit8	14	Objektklasse	uint16

Anfrage			*Antwort*		
Byte-Offset	**Daten**	**Typ**	**Byte-Offset**	**Daten**	**Typ**
15	Posenformat	unit8	16	Objektinstanz	uint16
16	Greifrückmeldung	unit8	18	Hub	int32
17	*Offset*	–	22	Winkelversatz	int32
18	Roboterpose	int32[7]	26	Mittenversatz	int32[3]
46	Projektindex	uint32	38	Werkzeug	unit8
50 – 79	*Padding*	–	39	Posenformat	unit8
			40	Greifpose	int32[7]
			68	Objektanzahl	unit16
			70	Kandidatenanzahl	unit16
			72 – 79	*Padding*	–

3.2.1 Präfix

	Version	Länge
Wert	3	74

Bemerkung: Das Längenfeld gibt die verbleibende Meldungsgröße an, die 74 Byte beträgt.

Layout siehe ▸ 2.1 [🗋 6].

3.2.2 Kopfzeile

Die *Kopfzeile* ist der Teil, der den Meldungstyp und seine Bedeutung kodiert.

	Byte-Offset	Daten	Typ
Präfix	0	…	…
Kopfzeile	6	▸ Kommunikationstyp [🗋 9]	uint8
	7	▸ Antwortcode [🗋 9]	uint8
	8	▸ Antwortzähler [🗋 9]	uint8
	9	▸ Meldungstyp [🗋 9]	uint8
Rumpf	10 – 79	…	…

3.2.2.1 Kommunikationstyp

Wert	Beschreibung
01	Anfrage
02	Antwort

3.2.2.2 Antwortcode

Der Antwortcode gibt den Ergebnisstatus der bearbeiteten Anfrage an.

Clients sollten immer den Antwortcode in jeder einzelnen Antwortmeldung überprüfen und nur fortfahren, wenn der Antwortcode auf SUCCESS gesetzt ist. Wenn der Antwortcode nicht auf SUCCESS gesetzt ist, wird der Rest der Meldung ignoriert. Der Antwortzähler wird jedoch weiterhin hochgezählt.

Wert	Name	Beschreibung
Allgemeines		
1	SUCCESS	Der Server hat die Anfrage erfolgreich bearbeitet.
2	ERROR	Der Server ist bei der Bearbeitung der Anfrage auf einen Fehler gestoßen.
Meldungsspezifisch		
3	NO_OBJECT	Kein Objekt gefunden.
4	NO_GRASP	Keinen Griff gefunden.
5	INVALID_OBJECT_CLASS	Die Objektklasse ist entweder ungültig oder existiert nicht.

HINWEIS

Der Antwortcode wird bei Anfragen ignoriert.

3.2.2.3 Antwortzähler

Ein Zähler, der sich mit jeder Antwort erhöht und auf Null zurückgesetzt wird, wenn er 256 erreicht.

HINWEIS

Der Antwortcode wird bei Anfragen ignoriert.

3.2.2.4 Meldungstyp

siehe Kapitel ▶ 3.3.1 [🗋 10]

3.3 Meldungen

3.3.1 Meldungstypen

In der folgenden Tabelle sind alle verfügbaren Meldungstypen zusammengefasst.

Wert	Meldungstyp	Beschreibung
Allgemeines		
0	reserviert/unbenutzt	
1	▸ GET_PROTOCOL_VERSION [🗋 11]	Protokollversion des Servers abfragen.
2	▸ GET_STATE [🗋 12]	Aktuellen Status des Servers abfragen.
3	▸ REGISTER_CLIENT [🗋 13]	Robotersystem des Clients auf dem Server registrieren.
4	▸ SET_PROJECT [🗋 14]	Aktives Projekt auf dem Server festlegen.
Greifen		
16	▸ GET_GRASP [🗋 15]	Griff für die aktuelle Szene erhalten.
17	▸ GRASP_FEEDBACK [🗋 18]	Server eine Rückmeldung über das Ergebnis des letzten Greifvorgangs geben.
Erkennung		
32	▸ GET_OBJECT_COUNT [🗋 19]	Anzahl der Objekte in der aktuellen Szene abrufen.
Roboterstatus		
48	▸ ROBOT_POSE [🗋 20]	Server über die aktuelle Roboterpose benachrichtigen.

3.3.2 Allgemeines

3.3.2.1 GET_PROTOCOL_VERSION

Gibt die höchste unterstützte Protokollversion des Servers zurück. Dies ist nützlich, um festzustellen, ob die Version des Clients veraltet ist oder nicht. Sollte dies der Fall sein, wird dem Client empfohlen, den Anwender entsprechend zu informieren.

Anfrage:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
Kopfzeile	2	74	uint32
	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	1	uint8
Rumpf	10–79	*Padding*	–

Antwort:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
Kopfzeile	2	74	uint32
	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	1	uint8
Rumpf	10	Version	uint16
	12–79	*Padding*	–

Version: Höchste unterstützte Protokollversion des Servers.

3.3.2.2 GET_STATE

Ruft den Status des Servers ab.

Anfrage:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
Kopfzeile	2	74	uint32
	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	2	uint8
Rumpf	10–79	*Padding*	–

Antwort:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
Kopfzeile	2	74	uint32
	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	2	uint8
Rumpf	10	*Offset*	–
	12	Status	uint8
	13–79	*Padding*	–

Status:

Wert	Name	Beschreibung
1	INIT	Der Server wird initialisiert.
2	OPERATIONAL	Der Server ist betriebsbereit und bereit, Anfragen zu bearbeiten.
3	STOPPED	Der Server hat den Betrieb gestoppt.
4	ERROR	Auf dem Server ist ein kritischer Fehler aufgetreten.

3.3.2.3 REGISTER_CLIENT

Registriert das Robotersystem des Clients (zu Informationszwecken).

Diese Meldung sollte unmittelbar nach dem Verbindungsaufbau gesendet werden. Es gibt jedoch keine Einschränkungen hinsichtlich des Zeitpunkts und der Häufigkeit des Versands der Meldung. Wenn das Robotersystem des Clients nicht offiziell unterstützt wird (siehe Clienttabelle in den Anfragedetails), diese Meldung einfach ignorieren.

Anfrage:

Enthält Informationen über das Robotersystem des Clients.

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	3	uint8
Rumpf	10	Client	uint8
	11 – 79	*Padding*	–

Client:

Wert	Name	Anbieter
1	UR	Universal Robots
2	KUKA	KUKA
3	Yaskawa	Yaskawa
4	FANUC	FANUC
5	ABB	ABB
6	HORST	fruitcore robotics
128	Siemens PLC	Siemens

Antwort:

Der Antworttext ist leer.

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	3	uint8
Rumpf	10–79	*Padding*	–

3.3.2.4 SET_PROJECT

Legt das aktive Projekt auf dem Server fest.

Anfrage:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	4	uint8
Rumpf	10	*Offset*	–
	46	Projektindex	uint32
	50 – 79	*Padding*	–

Projektindex: Der Index des zu aktivierenden Projekts.

Antwort:

Der Antworttext ist leer. Der Antwortcode gibt an, ob der Arbeitsgang erfolgreich war oder nicht.

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	4	uint8
Rumpf	10 – 79	*Padding*	–

3.3.3 Greifen

3.3.3.1 GET_GRASP

Einen Griff für die aktuelle Szene anfordern.

Anfrage:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	16	uint8
Rumpf	10	*Offset*	–
	11	Greifmodus	uint8
	12	Objektklasse	uint16
	14	Werkzeug	uint8
	15	Posenformat	uint8
	16 – 79	*Padding*	–

- *Greifmodus*:

Wert	Name	Beschreibung
1	ACTIVE_GRASP	Der zurückgegebene Griff muss mit der anwenderdefinierten Definition des aktiven Griffs des Zielobjekts übereinstimmen.
2	ANY_GRASP	Der zurückgegebene Griff muss mit einer beliebigen anwenderdefinierten Griffdefinition des Zielobjekts übereinstimmen.
3	AUTO_GRASP	ANY_GRASP-Modus mit modellfreier Griffplanung zur Sicherheit.

- *Objektklasse*: Klassen-ID des Zielobjekts. Ist der Wert Null, wird die Klassen-ID ignoriert und das Zielobjekt nach dem Zufallsprinzip ausgewählt.

- *Werkzeug*: Das Werkzeug, das für das Greifen verwendet wird.

Wert	Name	Beschreibung
1	EXTERIOR	2-Finger-Parallel-Außengreifer.
2	INTERIOR	2-Finger-Parallel-Innengreifer.
3	CONTACT	Ansaugung, Magnet oder Haftung mit einer Kontaktfläche.

- *Posenformat*: Gibt das Posenformat an, das in der Antwortmeldung verwendet werden soll. Im Abschnitt „Posenformate“ finden Sie alle unterstützten Formate.

Antwort:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	16	uint8
Rumpf	10	*Offset*	-
	13	Greifmodus	uint8
	14	Objektklasse	uint16
	16	Objektinstanz	uint16
	18	Hub	int32
	22	Winkelversatz	int32
	26	Mittenversatz	int32[3]
	38	Werkzeug	uint8
	39	Posenformat	uint8
	40	Greifpose	int32[7]
	68	Objektanzahl	uint16
	70	Kandidatenanzahl	uint16
	72 – 79	*Padding*	-

- *Greifmodus*: Der verwendete Greifmodus, siehe Anfragedetails für die Aufzählungswerte.

  Bemerkung: Dieser Greifmodus ist nicht unbedingt mit dem gewünschten Greifmodus identisch. Wenn zum Beispiel ein automatischer Griff angefordert wird, wird das System

weiterhin anwenderdefinierte Griffe bevorzugen und nur dann auf automatische Griffe zurückgreifen, wenn ein solcher entweder nicht existiert oder nicht anwendbar ist.

- *Objektklasse*: Klassen-ID des Zielobjekts.
- *Objektinstanz*: Instanz-ID des Zielobjekts.
- *Hub*: Abstand zwischen beiden Fingern in Mikrometern, der vor der Annäherung an das Objekt eingestellt werden muss. Dieses Feld ist nur für Greifer relevant und kann bei anderen Werkzeugtypen ignoriert werden.

  Bemerkung: Der Greifer befindet sich in der Position Null, wenn sich beide Finger berühren.
- *Winkelversatz*: Winkelversatz in Mikrograd zwischen dem Objektmodell und dem Roboterflansch an der Position Null.

  Wenn ein Objekt in einer vordefinierten Ausrichtung platziert werden muss, den Roboterflansch durch Anpassen seiner Position an den Winkelversatz ausrichten (nachdem das Objekt gegriffen wurde). Bei Bedarf außerdem einen vom Anwender definierten, anwendungsfallspezifischen Versatz anwenden.

  Bemerkung: Der Winkelversatz ist nicht anwendbar, wenn es sich um verformbare Objekte handelt oder wenn automatisch gegriffen wird.
- *Mittenversatz*: Translatorischer [x,y,z]-Versatz in Mikrometern von der Objektmitte zum Greifpunkt, ausgedrückt im Basiskoordinatensystem des Roboters.
- *Werkzeug*: Der Typ des in der Anfrage verwendeten Werkzeugs (gleicher Aufzählungswert).
- *Posenformat*: Für das Posenfeld verwendetes Posenformat. Im Abschnitt „Posenformate“ finden Sie alle unterstützten Formate.
- *Greifpose*: Greifpose im Basiskoordinatensystem des Roboters. Der Typ dieses Feldes hängt von dem Wert im Feld „Posenformat“ ab.
- *Objektanzahl*: Die Anzahl aller erkannten Objekte (einschließlich des Zielobjekts selbst).
- *Kandidatenanzahl*: Die Anzahl der erkannten Objekte der angeforderten Klassen-ID (einschließlich des Zielobjekts selbst).

  Bemerkung: Wenn die angeforderte Klassen-ID 0 (= zufällig) war, werden alle Objekte gezählt.

3.3.3.2 GRASP_FEEDBACK

Gibt dem Server eine Rückmeldung über das Ergebnis des letzten Greifvorgangs.

Anfrage:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	17	uint8
Rumpf	10	*Offset*	–
	16	Greifrückmeldung	uint8
	17 – 79	*Padding*	–

- *Greifrückmeldung*: Rückmeldung über das letzte Greifergebnis.

Wert	Name	Beschreibung
1	OK	Der Griff war OK
2	BAD	Der Griff war nicht OK

Antwort:

Der Antworttext ist leer.

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	17	uint8
Rumpf	10 – 79	*Padding*	–

3.3.4 Erkennung

3.3.4.1 GET_OBJECT_COUNT

Ruft die Anzahl der Objekte in der aktuellen Szene ab.

Anfrage:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	32	uint8
Rumpf	10	*Offset*	–
	12	Objektklasse	uint16
	14 – 79	*Padding*	–

- *Objektklasse*: Klassen-ID der zu zählenden Objekte. Ist der Wert Null, werden alle Objekte aller Klassen gezählt.

Antwort:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	17	uint8
Rumpf	10	*Offset*	–
	14	Objektklasse	uint16
	16	*Offset*	–
	68	Objektanzahl	uint16
	70	Kandidatenanzahl	uint16
	72 – 79	*Padding*	–

- *Objektklasse*: Die angeforderte Klassen-ID der zu zählenden Objekte. Ist der Wert Null, werden alle Objekte aller Klassen gezählt.
- *Objektanzahl*: Die Anzahl der erkannten Objekte aller Klassen.
- *Kandidatenanzahl*: Die Anzahl der erkannten Objekte der angeforderten Klasse.

  Bemerkung: Wenn die angeforderte Klassen-ID Null war, werden alle Objekte gezählt.

3.3.5 Roboterstatus

3.3.5.1 ROBOT_POSE

Informiert den Server über die aktuelle Pose des Endeffektors im Basiskoordinatensystem des Roboters.

Anfrage:

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	1	uint8
	7	0	uint8
	8	0	uint8
	9	48	uint8
Rumpf	10 – 14	*Offset*	–
	15	Posenformat	uint8
	16	*Offset*	–
	18	Roboterpose	int32[7]
	46 – 79	*Padding*	–

- *Posenformat*: Gibt das Posenformat der Roboterpose an. Im Abschnitt „Posenformate“ finden Sie alle unterstützten Formate.
- *Roboterpose*: Die an den Server zu übertragende Pose.

  Untergeordneter Rahmen: Endeffektor (TCP), übergeordneter Rahmen: Roboterbasis.

Antwort:

Der Antworttext ist leer.

	Byte-Offset	Daten	Typ
Präfix	0	3	uint16
	2	74	uint32
Kopfzeile	6	2	uint8
	7	Antwortcode	uint8
	8	Zähler	uint8
	9	48	uint8
Rumpf	10 – 79	*Padding*	–

4 Visualisierung

Eine Visualisierung des letzten Erfassungsergebnisses wird als Bildressource bereitgestellt, auf die über HTTP zugegriffen werden kann:

`http://<Server-IP>/monitor/latest_result`

5 Posenformate

Die folgenden Posenformate werden unterstützt:

Wert	Name	Typ	Aufbau	Beschreibung
Allgemeine Formate				
1	Quaternion	int32[7]	Position: x, y, z; Orientierung: qx, qy, qz, qw	Position und nicht normalisiertes Quaternion. *Bemerkung: Um das Quaternion zu normalisieren, das Quaternion durch 10^6 teilen.* *Einheiten*: Mikrometer und Bogenmaß.
2	Achse-Winkel	int32[7]	Position: x, y, z; Orientierung: x, y, z, –	Position und nicht normierter Rotationsvektor. *Einheiten*: Mikrometer und Mikro-Bogenmaß. *Anbieter*: Universal Robots
Anbieterspezifische Formate				
16	WPR	int32[7]	Position: x, y, z; Orientierung: W, P, R, –	Position und Ausrichtung WPR mit W, das sich um die feste x-Achse dreht, dann P, das sich um die feste y-Achse dreht, dann R, das sich um die feste z-Achse dreht. *Rotationskonvention*: x-y-z (extrinsisch). *Einheiten*: Mikrometer und Mikrograd. *Anbieter*: Fanuc, Yaskawa
17	ABC	int32[7]	Position: x, y, z; Orientierung: A, B, C, –	Position und Ausrichtung ABC mit A, das sich um die z-Achse dreht, dann B, das sich um die y'-Achse dreht, dann C, das sich um die x''-Achse dreht. *Rotationskonvention*: x-y-z (intrinsisch). *Einheiten*: Mikrometer und Mikrograd. *Anbieter*: Kuka

6 Beispiele

Gehen wir davon aus, dass wir eine TCP-Verbindung zum Server erfolgreich aufgebaut haben.

Für einfache Greifaufgaben lautet die einfachste Abfolge von Meldungen:

1. ▸ GET_PROTOCOL_VERSION [🗋 23]
2. ▸ GET_STATE [🗋 24]
3. ▸ REGISTER_CLIENT [🗋 25]
4. ▸ SET_PROJECT [🗋 26]
5. ▸ GET_GRASP [🗋 27]

Die Schritte 1 bis 4 werden nur einmal ausgeführt. Schritt 5 wird in der Regel in einer Schleife durchlaufen.

6.1 GET_PROTOCOL_VERSION

Nach dem Verbindungsaufbau ist als Erstes die höchste unterstützte Protokollversion des Servers zu prüfen.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	Version	3	unit16	0	Version	3	uint16
	2	Länge	74	unit32	2	Länge	74	uint32
Kopf-zeile	6	Kommunikati-onstyp	1	unit8	6	Kommunikati-onstyp	2	uint8
	7	Antwortcode	0	unit8	7	Antwortcode	1	uint8
	8	Antwortzähler	0	unit8	8	Antwortzähler	0	uint8
Rumpf	9	Meldungstyp	1	unit8	9	Meldungstyp	1	uint8
	10 – 79	*Padding*	0	–	10	Version	3	uint16
					12 – 79	*Padding*	0	–
	Daten				**Daten**			

```
000 003 000 000 000 074 001 000
000 001 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

```
000 003 000 000 000 074 002 001
000 001 000 003 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

In diesem Beispiel antwortet der Server mit Version = 3. Dies bedeutet, dass der Client auf dem neuesten Stand ist.

6.2 GET_STATE

Bevor weitere Anfragen an den Server gesendet werden, ist der Status des Servers zu überprüfen.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	Version	3	unit16	0	Version	3	uint16
	2	Länge	74	unit32	2	Länge	74	uint32
Kopf-zeile	6	Kommunikati-onstyp	1	unit8	6	Kommunikati-onstyp	2	uint8
	7	Antwortcode	0	unit8	7	Antwortcode	1	uint8
	8	Antwortzähler	0	unit8	8	Antwortzähler	1	uint8
Rumpf	9	Meldungstyp	2	unit8	9	Meldungstyp	1	uint8
	10 – 79	*Padding*	0	–	10	*Offset*	0	–
					12	Status	2	
					13 – 79	*Padding*	0	–
	Daten				**Daten**			

```
000 003 000 000 000 074 001 000
000 002 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

```
000 003 000 000 000 074 002 001
001 002 000 002 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

In diesem Beispiel antwortet der Server mit Status = 2. Das bedeutet, dass der Server funktionsfähig ist und der Vorgang fortgesetzt werden kann.

6.3 REGISTER_CLIENT

Das Client-System registrieren, das in diesem Beispiel eine Siemens PLC ist.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	Version	3	unit16	0	Version	3	uint16
	2	Länge	74	unit32	2	Länge	74	uint32
Kopf-zeile	6	Kommunikati-onstyp	1	unit8	6	Kommunikati-onstyp	2	uint8
	7	Antwortcode	0	unit8	7	Antwortcode	1	uint8
	8	Antwortzähler	0	unit8	8	Antwortzähler	2	uint8
Rumpf	9	Meldungstyp	3	unit8	9	Meldungstyp	3	uint8
	10	Client	128	unit8	10 – 79	*Padding*	0	–
	11 – 79	*Padding*	0	–				
	Daten				**Daten**			

```
000 003 000 000 000 074 001 000
000 003 128 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

```
000 003 000 000 000 074 002 001
002 003 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

Der Antworttext ist leer.

6.4 SET_PROJECT

Projekt mit Index 5 aktivieren, das die zu erkennenden und zu ergreifenden Objekte enthält.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	Version	3	unit16	0	Version	3	uint16
	2	Länge	74	unit32	2	Länge	74	uint32
Kopf-zeile	6	Kommunikati-onstyp	1	unit8	6	Kommunikati-onstyp	2	uint8
	7	Antwortcode	0	unit8	7	Antwortcode	1	uint8
	8	Antwortzähler	0	unit8	8	Antwortzähler	3	uint8
Rumpf	9	Meldungstyp	4	unit8	9	Meldungstyp	4	uint8
	10	*Offset*	0	–	10 – 79	*Padding*	0	uint16
	46	Projektindex	5	unit32				
	50 – 79	*Padding*	0	–				
	Daten				**Daten**			

```
000 003 000 000 000 074 001 000
000 004 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 005 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

```
000 003 000 000 000 074 002 001
003 004 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

In diesem Beispiel war die Aktivierung des Projekts mit Index 5 erfolgreich.

Der Antworttext ist leer.

Bemerkung: Den Antwortcode in der Kopfzeile der Antwort überprüfen und nur fortfahren, wenn der Antwortcode 1 ist (was auf Erfolg hindeutet).

6.5 GET_GRASP

Anforderung eines automatischen Griffs mit folgenden Parametern: Die Zielobjektklasse ist zufällig, das Werkzeug ist ein Außengreifer, und die Greifpose soll in Quaternion-Darstellung zurückgegeben werden.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	Version	3	unit16	0	Version	3	uint16
	2	Länge	74	unit32	2	Länge	74	uint32
Kopf-zeile	6	Kommunikati-onstyp	1	unit8	6	Kommunikati-onstyp	2	uint8
	7	Antwortcode	0	unit8	7	Antwortcode	1	uint8
	8	Antwortzähler	0	unit8	8	Antwortzähler	3	uint8
Rumpf	9	Meldungstyp	16	unit8	9	Meldungstyp	16	uint8
	10	*Offset*	0	–	10	*Offset*	0	–
	11	Greifmodus	3	unit8	13	Greifmodus	2	uint8
	12	Objektklasse	0	unit16	14	Objektklasse	3	uint16
	14	Werkzeug	1	unit8	16	Objektinstanz	0	uint8
	15	Posenformat	1	unit8	18	Hub	86000	int32
	16 – 79	*Padding*	0	–	22	Winkelversatz	-25210142	int32
					26	Mittenversatz	[-911, -310, 68]	int32[3]
					38	Werkzeug	1	uint8
					39	Posenformat	1	uint8
					40	Greifpose	[853798, 1111998, 502790, 775990, 630744, 0, 0]	int32[7]
					68	Objektanzahl	2	uint16
					70	Kandidatenan-zahl	2	uint16
					72 – 79	*Padding*	0	–

Anfrage				*Antwort*			
Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Daten				**Daten**			

```
000 003 000 000 000 074 001 000
000 016 000 003 000 000 001 001
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

```
000 003 000 000 000 074 002 001
004 016 000 000 000 002 000 002
000 000 000 001 079 240 254 127
082 226 255 255 252 113 255 255
254 202 000 000 000 068 001 001
000 013 007 038 000 016 247 190
000 007 172 006 000 011 215 054
000 009 159 216 000 000 000 000
000 000 000 000 000 002 000 000
000 000 000 000 000 000 000 000
```

In diesem Beispiel ist der zurückgegebene Griff ein anwenderdefinierter Griff (Greifmodus = 2). Das System gibt anwenderdefinierten Griffen den Vorzug vor automatischen Griffen und verwendet automatische Griffe nur dann, wenn keine anwenderdefinierten Griffe verfügbar sind oder wenn sie für die jeweilige Situation nicht geeignet sind. Die Antwort zeigt an, dass die Anzahl der Kandidaten 2 beträgt, was bedeutet, dass nach der Anwendung des Griffs noch ein Kandidatenobjekt in der Szene übrig ist.

Bemerkung: Die Anforderung eines Griffs kann aus verschiedenen Gründen fehlschlagen, z. B. wenn die Kamera nicht kalibriert ist, das Objekt nicht lokalisiert werden kann oder der Griff unpraktisch ist. Den Antwortcode in der Kopfzeile der Antwort unbedingt überprüfen und nur fortfahren, wenn der Antwortcode 1 ist (was auf Erfolg hindeutet).

Die Schritte 1 bis 4 werden nur einmal ausgeführt. Schritt 5 wird in der Regel in einer Schleife durchlaufen.

SCHUNK SE & Co. KG
Spanntechnik | Greiftechnik | Automatisierungstechnik

Bahnhofstr. 106 – 134
D-74348 Lauffen/Neckar
Tel. +49-7133-103-0
info@de.schunk.com
schunk.com

Folgen Sie uns | *Follow us*

Wir drucken nachhaltig | *We print sustainable*

04-2024

01.00 | 2D Grasping-Kit | Softwarehandbuch | de | 1591390