ASTORINO

Anleitung zur AS-Sprache

EINFÜHRUNG

Diese Anleitung beschreibt Grundsätze und Funktionen der AS-Sprache, die beim "ASTORINO"-Roboter und bei der mit ihr verbundenen "astorino"-Software, die ein Teil des Lieferumfangs ist, angewandt wurde.

ASTORINO ist ein Bildungsroboter, der eigens für Bildungsstellen entwickelt worden ist. Schüler und Studenten können ASTORINO nutzen, um roboterunterstützte Automatisierung in Praxis zu erproben.

Diese Anleitung gilt ab der Robotersoftware in der Version 3.8.3 und 3.8.3B

Bei weiteren Fragen kontaktieren Sie bitte die technische Hilfe von Kawasaki Robotics.

Kontakt:

Astor Sp. z o.o.

astorino@astor.com.pl

INHALTSVERZEICHNIS

1 AS-SPRACHE

1.1 EINFÜHRUNG

Diese Anleitung beschreibt die AS*-Sprache, die in Steuereinheiten der ASTORINO-Roboter angewandt wird. Das Ziel dieser Anleitung ist es, eingehende Informationen über die ganze AS-Sprache, ihre grundlegende Anwendungen, Typen von Daten, Steuerung mit der Trajektorie des Roboters und über alle Befehle und Anweisungen, die diese Sprache effizient nutzen lassen, zu liefern. Diese Anleitung enthält keine Prozeduren für die Bedienung des Roboters, die in der Bedienungsanleitung enthalten sind. Man soll sich mit dem Inhalt dieser Anleitung wie auch mit sonstigen im Folgenden genannten Anleitungen in Kenntnis setzen.

Die Nutzung des Roboters ist ausschließlich dann gestattet, wenn die Anleitung genau durchgelesen und verstanden wurde.

Im Inhalt dieser Anleitung wurde angenommen, dass der Roboter nach Anforderungen, die in der Bedienungsanleitung enthalten sind, montiert und angeschlossen wurde. Bei jedweden Fragen oder Bedenken zum Betrieb des Roboters bitten wir mit der Firma Astor Kontakt aufzunehmen.

Die Version der AS-Sprache, die bei ASTORINO-Robotern zur Anwendung kommt, ist eine vereinfachte Version der vollen AS-Sprache, die bei Robotern Kawasaki Robotics der Firma Kawasaki Heavy Industries Ltd. eingesetzt wird.

Die grundlegenden Differenzen sind:

- Mangel an den meisten optionalen Ausdrücken in Funktionen

Anmerkung* AS ist wie im Amerikanischen Englisch stimmhaft [az] auszusprechen.

---

1. Diese Anleitung darf nicht als eine Garantie für Systeme, in welchen die Roboter zum Einsatz kommen, wahrgenommen werden. Die Firma Astor haftet ebenfalls nicht für jegliche Unfälle, Schäden und/oder Verletzungen der Urheberrechte, die aus Nutzung solcher Systeme resultieren.

2. Es wird empfohlen, alle Mitarbeiter, die für Inbetriebnahme, Programmierung, Wartung oder Kontrolle des Roboters zuständig sind, zuerst in Kursen, welche durch die Firma Astor angeboten werden, einzuweisen.

3. Die Firma Astor behält sich das Recht vor, Änderungen, Korrekturen und Aktualisierungen dieser Anleitung ohne vorherige Mitteilung einzuführen.

4. Die Anleitung ist aufzuheben und an einem leicht zugänglichen Ort aufzubewahren. Falls der Einsatzort des Roboters gewechselt, er in eine andere Niederlassung überlassen oder verkauft wird, ist es erforderlich, ihm diese Anleitung beizufügen. Sollte diese Anleitung verloren gehen oder zerstört werden, nehmen Sie bitte Kontakt mit der Firma Astor auf.

2 BESCHREIBUNG DER AS-SPRACHE

Im AS-System wird der Roboter gesteuert und arbeitet programmmäßig. Das Programm wird früher vorbereitet und beschreibt Aufgaben, die ausgeführt werden sollen. (Methode zur Wiederherstellung des erlernten Programms).

Kommandos, die in der AS-Sprache verfügbar sind, können in zwei Gruppen eingeteilt werden: Konsolenbefehle und Programmbefehle.

Die Konsolenbefehle dienen zur Speicherung, Editierung und Ausführung von Programmen. Diese werden ins Terminal in der astorino- oder astorinoIDE-Software nach dem Zeichen (>) eingepflegt und sofort ausgeführt.

Manche von diesen Befehlen werden auch in Programmen als Programmbefehle angewandt.

Programmbefehle: diese werden in Programmen zum Definieren von Bewegungen des Roboters, zur Überwachung und Steuerung mit Außensignalen usw. angewandt. Das Programm ist eine Sammlung von Programmkommandos.

Befehle, die in dieser Anleitung am Bildschirm eingepflegt werden, nennt man Befehle und die Programmkommandos nennt man Kommandos.

Die vereinfachte AS-Sprache besitzt eine Reihe von einzigartigen Eigenschaften:

1. Zwei Koordinatensysteme: Das globale Koordinatensystem am Robotersockel und das Werkzeugkoordinatensystem, welches mit dem am Ende des Armes befestigten Werkzeug verbunden ist.

Den Roboter kann man in beliebigem Koordinatensystem programmieren.

2. In der Betriebsart Teachen oder Wiedergeben kann man den Roboter auf der linearen Bahn verfahren. In der Betriebsart Teachen kann gleichzeitig die Orientierung des Werkzeugs beibehalten werden.

3. Die Bezeichnungen der Programme können beliebig sein und ihre Anzahl ist allein durch die verfügbare Speicherkapazität des Roboters begrenzt.

4. Jeder Bewegungsablauf kann als Programm definiert werden.

5. Den Roboter kann man mithilfe des persönlichen Computers mit gestarteter astorino- lub astorinoIDE-Software programmieren.

3 AUSDRÜCKE IN DER AS-SPRACHE

In diesem Kapitel wurden Arten von Daten und Variablen, die in vereinfachter AS-Sprache gebraucht werden, beschrieben.

3.1 NOTATION UND KONVENTIONEN

1. Große und kleine Zeichen

Um die Textverständnis zu erleichtern, wurden in dieser Anleitung die nachfolgend beschriebenen Grundsätze für die Anwendung von Klein- und Großbuchstaben angenommen. Alle Schlüsselwörter der AS-Sprache (Befehle, Kommandos, etc.) werden großgeschrieben. Variable und andere einzupflegenden Daten werden kleingeschrieben. Nicht desto weniger hat die Größe der Buchstaben bei der Eingabe vom AS-Terminal keine Bedeutung.

2. Leerschritt, Tabulator

Zwischen einem Parameter (oder einem Kommando) und einem Parameter muss sich mindestens ein Leerschritt oder ein Tabulator* befinden. Ein Leerschritt oder ein Tabulator ist ebenfalls zwischen Parameter zu setzen, die mit einer Komma oder einem anderen begrenzenden Zeichen nicht getrennt sind. Die überschüssigen Leerschritte oder Tabulatoren werden durch das System ignoriert.

Anmerkung* Der Parameter sind Daten, die zur richtigen Ausführung des Kommandos oder einer anderen Funktion erforderlich sind. Beispielsweise ist für die SPEED-Anweisung ein Parameter anzugeben, der die Geschwindigkeit des Roboters bestimmt. Wenn Kommandos oder eine Funktion viele Parameter erfordern, müssen diese mit einem Komma oder mit einem Leerschritt getrennt werden.

Beispiel *SPEED 50*

3. ENTER-Taste

Die meisten Befehle oder Kommandos des Programms werden nach Betätigung der ENTER-Taste ausgeführt. In dieser Anleitung wird die ENTER-Taste mit dem Symbol ↲ gekennzeichnet.

Numerische Werte

Die Werte werden im Dezimalsystem angegeben, soweit nicht anders vermerkt wurde. Die mathematischen Ausdrücke dienen der Setzung von Parametern in Bildschirmbefehlen und in Kommandos der AS-Programme. Nicht desto weniger sind Begrenzungen betreffend diese Werte zu beachten. Im Folgenden wurden die Grundsätze der Interpretation von Werten in verschiedenen Kontexten angegeben.

- Abstand

dient zur Definierung der Länge der Roboterbewegung zwischen zwei Punkten. Die Längenmaßeinheit ist ein Millimeter (mm); diese wird bei der Eingabe weggelassen. Sowohl negative als auch positive Werte sind zulässig.

- Winkel

beschreiben die Orientierung der Werkzeugachse und die Winkellage mithilfe von entsprechend 3 Eulerschen Drehwinkeln (O – A – T). Sowohl die negativen als auch positiven Werte sind zulässig und der zulässige Maximalwert des Winkels beträgt 180

- Achsennummer

Die Achsennummer ist der Gesamtwert von 1 bis zu der Anzahl von verfügbaren Achsen (standardmäßig hat der Roboter 6 Achsen). Die Achsen werden fortlaufend, von der Achse des Sockels beginnend, nummeriert. (Gewöhnlich werden diese als JT1, JT2,… bezeichnet).

- Signalnummern

Die Signalnummern dienen zur Identifikation der binären Signale (ON/OFF). Das sind ganze Werte. Die zulässigen Wertbereiche bestimmt die nachfolgende Tabelle.

	Standardbereich
Externe Ausgangssignale (MODBUS TCP) Ausgänge am Roboterarm*	1-8 (9-56) 57-58
Interne Eingangssignale (MODBUS TCP) Eingänge am Roboterarm *	1001-1008 (1009-1056) 1057-1058
Interne Signale	2001-2016

*B-Version des astorino-Roboters

Die Signalnummer mit negativem Wert bedeutet den Zustand OFF.

- Schlüsselwörter

Im Allgemeinen kann man beliebige Bezeichnungen der Variablen im AS-System anwenden. Nicht desto weniger sind gewisse Wörter wie Bezeichnungen der Befehle, der Kommandos etc. vorbehalten und man kann diese zur Bezeichnung von Daten über Position, als Bezeichnungen der Variablen etc., nicht anwenden.

4 DATEN DER POSITIONEN, NUMERISCHE DATEN UND TEXTDATEN

Im vereinfachten AS-System sind drei Typen von Daten: Daten über die Position(point), numerische(real) und Textdaten(string) verfügbar.

4.1 POSITIONSDATEN

Die Positionsdaten dienen zur Bestimmung der Lage und Orientierung des Roboters im Arbeitsraum. Die Lage und die Orientierung des Roboters beziehen sich auf die Lage des Werkzeugmittelpunkts (TCP) und auf die Orientierung des Werkzeugs (Koordinaten), soweit nicht anders vermerkt wurde. Die Lage und die Orientierung des Roboters bestimmen zusammen die Position des Roboters.

Die Position bestimmt den Ort, an welchem sich der Roboter befindet sowie wie er orientiert ist, also im Falle der Angabe eines Bewegungskommandos werden diese beiden Angaben bestimmt.

1. Der Roboter verfährt den Werkzeugmittelpunkt (TCP) in die bestimmte Lage.

2. Das Koordinatensystem des Roboterwerkzeugs wird zu der bestimmten Orientierung gedreht.

Die Daten zur Position werden durch die Angabe des Verfahrenswertes in Achsen oder der Koordinaten der Transformation bestimmt:

1. Verfahrenswerte der Achsen

Die Position des Roboters wird durch die Angabe des linearen Verfahrens oder des Winkelverfahrens in jedem von Achsenkoordinatensystemen des Roboters bestimmt. Die Werte für die Winkelachsen werden in Graden und die Werte für lineare Achsen in Millimetern angegeben. Nach der Ausführung der angegebenen Bewegungen in Achsen sind die Lage und die Orientierung des Werkzeugmittelpunktes eindeutig festgelegt.

Beispiel: Die Achsennummern sind gekennzeichnet als JT1,…, JT6, und die Verfahrenswerte wurden unter den Achsennummern angegeben.

	JT1[°]	JT2[°]	JT3[°]	JT4[°]	JT5[°]	JT6[°]
#pose=	0.00,	33.00,	-15.00,	0.00,	-40.00,	30

2. Transformationswerte (X,Y,Z,O,A,T)

Diese Werte beschreiben Transformationen der Koordinaten in Bezug auf das Bezugskoordinatensystem. Soweit nicht anders vermerkt wurde, werden Transformationswerte des Koordinatensystems vom Werkzeug gegenüber dem Basiskoordinatensystem des Roboters angegeben. Die Lage wird mittels der XYZ-Werte gegenüber dem Basiskoordinatensystem und die Orientierung mittels Eulerscher Winkel (O-A-T) des Werkzeugs gegenüber dem Basiskoordinatensystem bestimmt.

Beispiel X Y Z O A T

	X[mm]	Y[mm]	Z[mm]	O[°]	A[°]	T[°]
pose=	0.00,	1434.00,	300.00,	0.00,	90.00,	70.00

Wenn der Roboter mehr als sechs Achsen hat, wird die Lage der Zusatzachse mit den Transformationswerten angegeben.

Beispiel X Y Z O A T JT7

	X[mm]	Y[mm]	Z[mm]	O[°]	A[°]	T[°]	JT7[mm]
pose=	0.00,	1434.00,	300.00,	0.00,	90.00,	70.00,	1000.0

Die Nutzung der Achsenverfahrenswerte und der Koordinaten der Transformation ist mit gewissen Vor- und Nachteilen verbunden. Es ist eine von diesen Methoden, je nach Bedarf, zu wählen.

	Verfahrenswerte der Achsen	Koordinaten der Transformation
Vorteile	• große Präzision der Wiedergabe der und keine Mehrdeutigkeit der Roboterkonfiguration in der jeweiligen Position	• die Mitte des Koordinatensystems des Werkzeugs, in der Betriebsart Wiedergabe angewandt, wird sogar nach dem Werkzeugwechsel nicht geändert. (Verschiebung des Null-Koordinatensystems des Werkzeugs). • Möglichkeit, die relativen Koordinaten (z. B. Koordinaten im System des Gegenstands) zu nutzen. • Verarbeitungskomfort, da die Angaben als XYZOAT-Werte angegeben sind.
Nachteile	• Das Werkzeugmittelpunkt (TCP) wird geändert, wenn das Werkzeug gewechselt wird (das Null-Werkzeugkoordinatensystem bleibt dasselbe). • keine Möglichkeit, die relativen Koordinaten (z. B. Koordinaten des Gegenstands) zu nutzen	• die Koordinaten unterliegen der Änderung entsprechend zu den Transformations-koordinaten des Basissystems oder des Werkzeugsystems, also ist die völlige Verfolgung zur Beurteilung des Einflusses jedweder Änderung auf die Sicherheit erforderlich
Empfohlene Anwendungen	• Definierung der Anfangsposition im Programm • Einstellen der Roboterkonfigurierung kurz vor oder in der Position, die mittels Transformations-koordinaten beschrieben ist • Anwendung für andere, oft angewandte Positionen	• Beschreibung der relativen Koordinaten, beispielsweise der Koordinaten des Gegenstands • Beschreibung der Position, die mithilfe der numerischen Werte und der SHIFT-Funktion geändert werden soll • Beschreibung der Position, die in Anlehnung an vom Sensor gelieferten Informationen geändert werden soll

Für den variablen Wert der Gelenkverschiebung ist die Variable mit einer Bezeichnung zu definieren, die mit # beginnt.

Im Falle eines variablen Umwandlungswertes ist die Variable ohne #-Zeichen zu definieren.

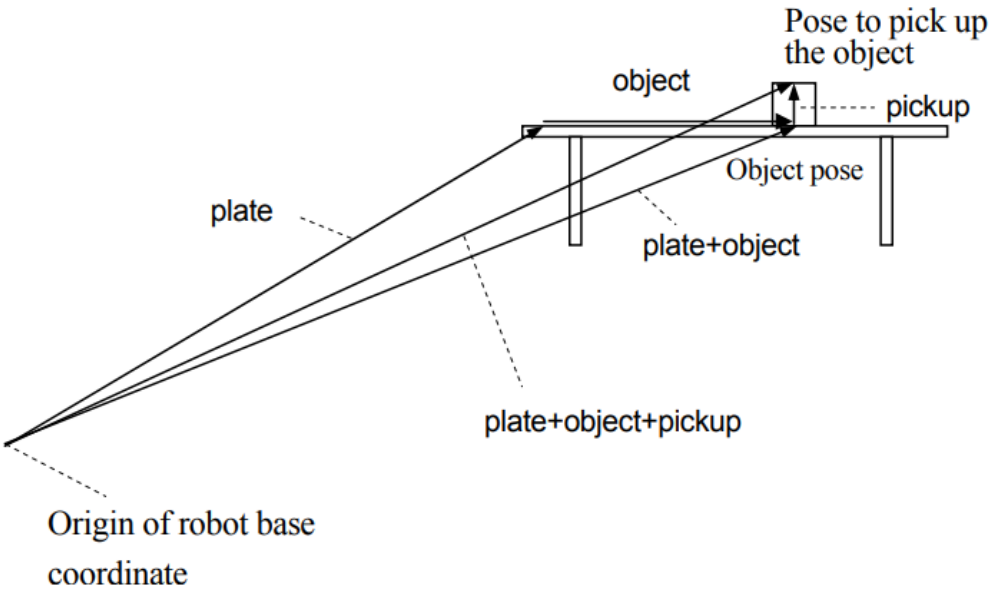
4.1.1 VERWENDUNG DER KOMPLEXEN TRANSFORMATIONSWERTE

Die Transformationswerte zwischen zwei Koordinaten kann man als eine Kombination des Umwandlungswertes zwischen zwei oder einer größeren Anzahl der Übergangskoordinaten darstellen. Eine solche Maßnahme kann man als eine Beziehung der Umwandlungswerte oder relative Umwandlungswerte bezeichnen.

Angenommen, dass „plate" die Bezeichnung einer Variablen ist, die durch Umwandlungswerte gegenüber den Basiskoordinaten definiert ist, die die Koordinaten am Tisch beschreiben, auf welchem sich das Objekt befindet.

Demnächst, wenn die Position des Objektes im Verhältnis zur Position der „plate" als „object" definiert ist, kann man den Bezug des Objekttransformationswertes gegenüber den Koordinaten des Robotersockels als „plate+object" beschreiben.

Im nachfolgenden Beispiel, sogar wenn sich die Position „plate" verändert (z. B. wenn sich der Tisch bewegt), so werden nur die Umwandlungswerte für „plate" eine Änderung erfordern und der Rest kann ohne Änderungen verwendet werden.



Im Falle einer mehrmaligen Anwendung des Wertes einer komplexen Umwandlung ist der Befehl POINT anzuwenden, um die Zeit für die Berechnung des Wertes der komplexen Transformation zu vermindern. Zum Beispiel, um an die Position „pickup" heranzugehen und dann, um zu dieser Position zu gelangen, kann man schreiben:

POINT x = plate+object+pickup - die Bestimmungspose berechnen

JAPPRO x, 100 - nähert sich 100 mm über dem Ziel

LMOVE x – lineare Bewegung zum Ziel

[**VORSICHT**]

Die Reihenfolge, in welcher die relative Transformation vorgegeben wird, ist nicht zu ändern. Wenn zum Beispiel der Umwandlungswert der Variablen der „b"-Lage gegenüber dem Umwandlungswert der variablen Position „a" definiert ist, ergibt „a+b" das erwartete Ergebnis, „b+a" aber liefert ein abweichendes Ergebnis.

Im Falle der Roboter mit 7 Achsen sind folgende Probleme zu beachten:

- verwendet man den POINT-Befehl, ist der Wert JT7 zu notieren. Zum Beispiel bei POINT p=p1+p2 ist der zu „p" zugeordnete Wert JT7, der Wert JT7 für „p2".
- wird ein bestimmter Wert zu JT7 zugeordnet, ist „/7" am Ende von POINT hinzuzufügen. Zum Beispiel ordnet der POINT/7 p = TRANS(0,0,0,0,0,0,Wert) den „Wert" der Variablen „p" als den Wert JT7 zu.

4.1.2 ANWENDUNG DER DIREKTEN UND DER INVERSEN KINEMATIK

Die AS-Sprache erlaubt die direkte und die inverse Kinematik zu Berechnungen von Punktvariablen anzuwenden. Das System erlaubt die Punkt-zu-Punkt-Variablen zu Transformationspunktvariablen zu wechseln und funktioniert auch in die umgekehrte Richtung.

Beispiel:
POINT P0 = #P0 Wechsel einer Punkt-zu-Punkt-Variablen zu einer Transformationspunktvariablen
POINT #P0 = P0 Wechsel einer Transformationspunktvariablen zu einer Punkt-zu-Punkt-Variablen

4.2 NUMERISCHE DATEN

Im AS-System ist es erlaubt, numerische Werte und Formeln anzuwenden. Die numerische Formel ist ein mithilfe von Zahlen, Variablen sowie linearen Operatoren und Funktionen angegebene Wert. Numerische Formeln werden nicht nur in mathematischen Berechnungen, sondern auch als Parameter der Konsolenbefehle und Programmkommandos angewandt.

Beispielsweise ist es für den Befehl DRIVE erforderlich, drei Parameter, die Achsennummer, die Verfahrensgröße und die Geschwindigkeit anzugeben. Diese Parameter kann man in Form von numerischen Werten oder Formeln angeben, wie es im nachfolgenden Beispiel gezeigt wurde:

DRIVE 3,45,75 Verschiebung der Achse Nr. 3 um einen Winkel von 45, mit einer Geschwindigkeit von 75%

DRIVE joint, (start+30)/2, 75 Wenn joint=2, start=30, so wird die Achse 2 um +30 mit einer Geschwindigkeit von 75% verschoben.

Die numerischen Werte im AS-System teilen sich in drei Typen:

1. Reelle Zahlen

Die reellen Zahlen können ganze Zahlen oder Bruchzahlen sein. Es sind positive und negative Werte aus dem Bereich −3.4 E+38 bis 3.4 E+38 (−3.4×1038 bis 3.4×1038) sowie die Null zulässig.

Die reellen Werte ohne Bruchteile sind ganze Werte. Der zulässige Bereich der Werte beträgt von −16 777 216 bis +16 777 215. Die ganzen Werte werden im Dezimalsystem eingepflegt. Das Trennzeichen, welches den Teil der ganzen Zahl von dem Bruchteil trennt, ist der Punkt „.", z. B. 8.5 .

2. Logische Werte

Die logischen Werte dürfen nur zwei Zustände haben: TRUE und FALSE. Der Wert 1.0 steht dem Zustand TRUE gleich und der Wert 0 (oder 0.0) steht dem Zustand FALSE gleich. TRUE und FALSE sind Wörter, die in der AS-Sprache vorbehalten sind.

Logischer Wert True = TRUE, 1.0

Logischer Wert False = FALSE, 0.0

4.3 TEXTAUSDRÜCKE

Die Textausdrücke können aus Ziffern, Variablen, Funktionen und aus anderen Textausdrücken bestehen, die mit linearen Operatoren verbunden sind. Man kann diese als Parameter der Konsolenbefehle oder Kommandos anwenden. Die Interpretation des Wertes hängt vom Kontext ab, in welchem der jeweilige Ausdruck verwendet wird. Dem Textausdruck kommt das Zeichen „***$***" voraus, z. B. ***$TEXT = „HELLO WORLD"***

Die Textausdrücke kann man hinzufügen, z. B. ***$TEST = „HELLO " + „WORLD"***

5 VARIABLEN

In der vereinfachten AS-Sprache kann man Bezeichnungen den Daten über die Position, numerischen Daten und Textdaten zuordnen. Das sind die sog. Variablen. Die Variablen, welche Daten über die Position enthalten, werden entsprechend als reelle Variablen der Position bezeichnet. Nachdem die Variable definiert wurde, wird diese im Speicher mit dem ihr zugeordneten Wert gespeichert. Man kann dann diese Variable im Programm nutzen.

5.1 TYPEN DER VARIABLEN

5.1.1 GLOBALE VARIABLEN

Die globalen Variablen werden im Bereich des ganzen Robotersystems gesehen, man kann diese im Terminal und in verschiedenen Programmen nutzen,
z. B. ***Wiederholungen = 10***

5.1.2 LOKALE VARIABLEN

Die lokalen Variablen werden nur im Bereich eines Roboterprogramms gesehen, diese kann man nicht im Terminal nutzen. Die Variable wird nach abgeschlossener Arbeit des Programms gelöscht. Man definiert diese durchs Hinzufügen des Punktes („.") vor Bezeichnung der Variablen,
z. B. ***.wieviel = 10***

5.2 BEZEICHNUNGEN DER VARIABLEN

Die Bezeichnungen der Variablen müssen mit einem Alphabetzeichen beginnen und können Zeichen, Buchstaben und Ziffern enthalten. Die Anwendung sowohl der kleinen als auch der großen Zeichen ist zulässig.
Im Folgenden wurden Beispiele der unzulässigen Variablenbezeichnungen angegeben:
3p2a - das Zeichen darf keine Ziffer sein
part#2 – das Zeichen "#" darf nicht in der Mitte der Variablenbezeichnung stehen.
random – vorbehaltenes Wort

5.3 DEFINIEREN DER POSITIONSVARIABLEN

Variablen, welche die Informationen über die Position enthalten, werden Positionsvariablen genannt. Die Positionsvariable ist ausschließlich dann definiert, wenn ihre Bezeichnung und ihr Wert angegeben wird. Wird ihr Wert nicht angegeben, bleibt sie weiterhin nicht definiert und die Ausführung eines Programms mit einer nicht definierten Variablen hat zur Folge, dass ein Fehler generiert wird.

In der vereinfachten Version der AS-Sprache kann man zwei Typen der Positionsvariablen verwenden.

- Variablen mit Bezeichnungen P1..P99 und #P1..#P99 – Variablen, die im Dauerspeicher der Steuereinheit gespeichert und nach Abschaltung der Speisung nicht gelöscht werden
- Variablen mit beliebiger Bezeichnung, z. B. Herunterladen.

Die Positionsvariablen sind aus vielen Gründen eine komfortable Lösung:

Man kann dieselben Daten über die Position vielmals nutzen, ohne die Position jedes Mal neu einzuteachen.
Die definierte Positionsvariable kann in verschiedenen Programmen eingesetzt werden.
Die definierte Positionsvariable kann zur Definierung einer anderen Position eingesetzt werden.
Man kann direkt numerische Werte einpflegen, um eine Position zu bezeichnen, anstelle eines sehr zeitaufwändigen Einteachens der Position.
Die Positionsvariablen können beliebige Bezeichnungen haben, was die Übersichtlichkeit der Programme verbessert.

5.4 ARRAY-VARIABLEN

Die Variablen von jedem Typ: Zahlen-, Text- und Punktvariablen kann man in Form von Deklarationen anwenden. Die Deklarationen werden durchs Hinzufügen des Zeichens „ [„ nach der Bezeichnung der Variablen und demnächst durch die Angabe des Deklarationsindex‘ und durchs Beenden mit dem Zeichen „]” definiert. Beispiel:

x[0] = 10

x[1] = 12

x[2] = x[0] + x[1]

POINT test[0] = P0

LMOVE test[0]

$text[0] = „Hello”

$text[1] = „World”

5.5 DEFINIEREN DER POSITIONEN MITHILFE DER BILDSCHIRMBEFEHLE

1. Der Befehl HERE, im Terminal verwendet, erlaubt die aktuelle Position des Roboters in veränderter Position mit der angegebenen Bezeichnung speichern.
2. Der Befehl POINT, im Terminal verwendet, erlaubt neue Daten, sei es durch eine Funktion oder durch eine andere Positionsvariable der Positionsvariablen zuzuordnen.

Beispiel 1 Nutzung der Werte von Achsenwinkeln
Die Bezeichnung der Variablen muss mit dem Zeichen # beginnen, damit sie von Koordinaten der Transformation unterschieden werden kann. Nach dem Befehl werden die Verfahrenswerte in Achsen für die aktuelle Position angezeigt:
> ***HERE #pose*** ↲
Beispiel 2 Nutzung der Transformationskoordinaten
>***HERE pose*** ↲

- Ersetzen eines Wertes einer bereits definierten Variablen

>***POINT pose = P1*** ↲

6 NUMERISCHE FORMELN

Die numerischen Formeln können aus Ziffern, Variablen, Funktionen und anderen numerischen Formeln, die mit Operatoren verbunden sind, bestehen. Alle numerischen Formeln, die durch das System festgesetzt werden, sind reelle Werte. Numerische Formeln kann man immer anstelle der numerischen Werte angeben. Man kann diese als Parameter der Konsolenbefehle oder Anweisungen handhaben. Die Interpretation der Werte hängt vom Kontext, in welchem die jeweilige Formel gebraucht wird, ab. Wenn beispielsweise eine Formel anstelle eines logischen Wertes eingepflegt wurde, hat sie den logischen Wert False, wenn sie als Ergebnis 0 oder True, wenn sie als Ergebnis den Wert 1 liefert.

7 OPERATOREN

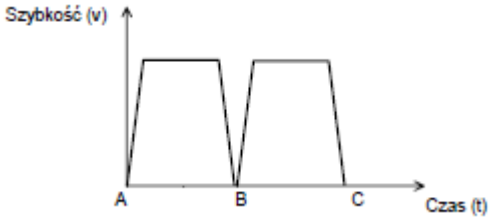
In den Ausdrücken kann man arithmetische und logische Operatoren einsetzen. Alle Operatoren nutzen zwei Werte und liefern einen Ergebniswert. Die unten befindliche Tabelle enthält eine Aufstellung von Operatoren.

arithmetische Operatoren	+ – * / ^ MOD	Addieren Subtrahieren oder Negation Multiplizieren Dividieren Potenzieren Rest nach Dividieren
Vergleichsoperatoren	< <=, =< == <> >=, => >	kleiner kleiner oder gleich gleich anders größer oder gleich größer
logische Operatoren	AND NOT OR XOR	logisches AND logische Ergänzung logische OR logische Konjunktion OR

8 ROBOTERBEWEGUNG

Die Beschleunigung für das zweite Segment beginnt nach Beendigung der Ausführung vom ersten Segment, wenn sich die aktuelle Position am Bestimmungspunkt einfindet. Die Neigung des Geschwindigkeitsanstiegs wird durch den ACCEL-Parameter und die Abbremslinie durch den DECEL-Parameter bestimmt.

Geschwindigkeit (v)



Zeit (t)

Der Astorino-Roboter kann sich auf drei verschiedene Arten bewegen. Diese Arten werden Interpolationen genannt. Wir können nennen:

1. die lineare Interpolation
2. die Punkt-zu-Punkt-Interpolation
3. die Kreisinterpolation

In anthropomorphischen Armen des Roboters (6 Achsen) gibt es gewisse Positionen, die Singularitäten genannt werden. Eine singuläre Position ist diese Position, bei welcher das Problem einer unkontrollierten Lage vorkommen kann. Eine solche Situation kommt dann vor, wenn zum Beispiel JT4 und JT6 parallel zueinander oder wenn JT1 und JT6 parallel zueinander sind. Diese Konfigurationen ergeben viele mathematische Lösungen der inversen Kinematik, daher kann die Bewegung über diese Punkte unvorhersehbar sein und viele sehr schnelle Achsenbewegungen einführen.

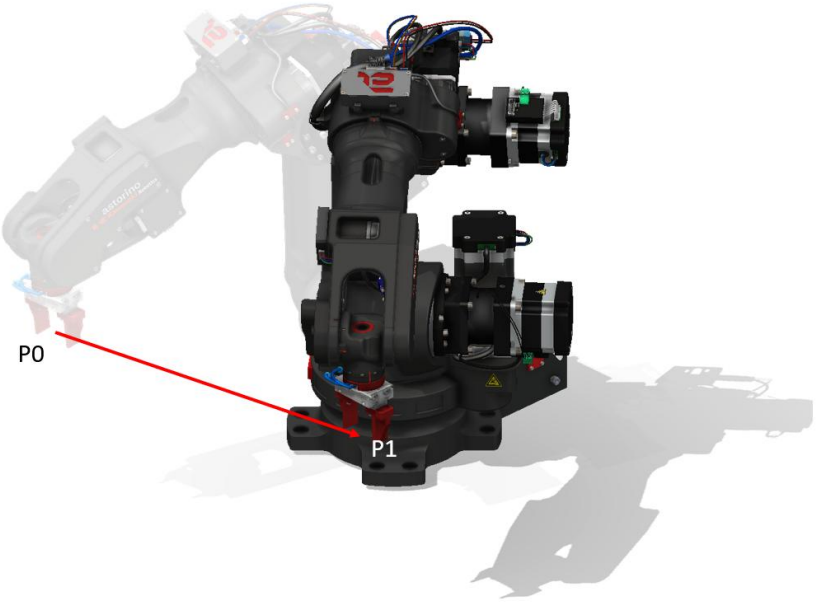
Beispiele der singulären Positionen

Achsen 6 und 4 sind parallel

Achsen 1 und 6 sind parallel

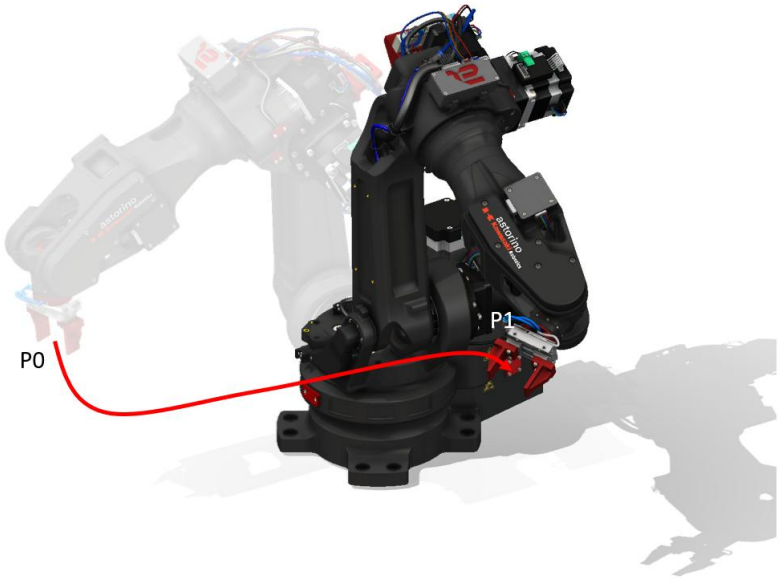
8.1 LINEARE INTERPOLATION

Bei der Interpolation dieser Art bewegt sich der Roboter von der aktuellen Position auf den Bestimmungsort so hin, dass sich TCP geradlinig im 3D-Raum bewegt.



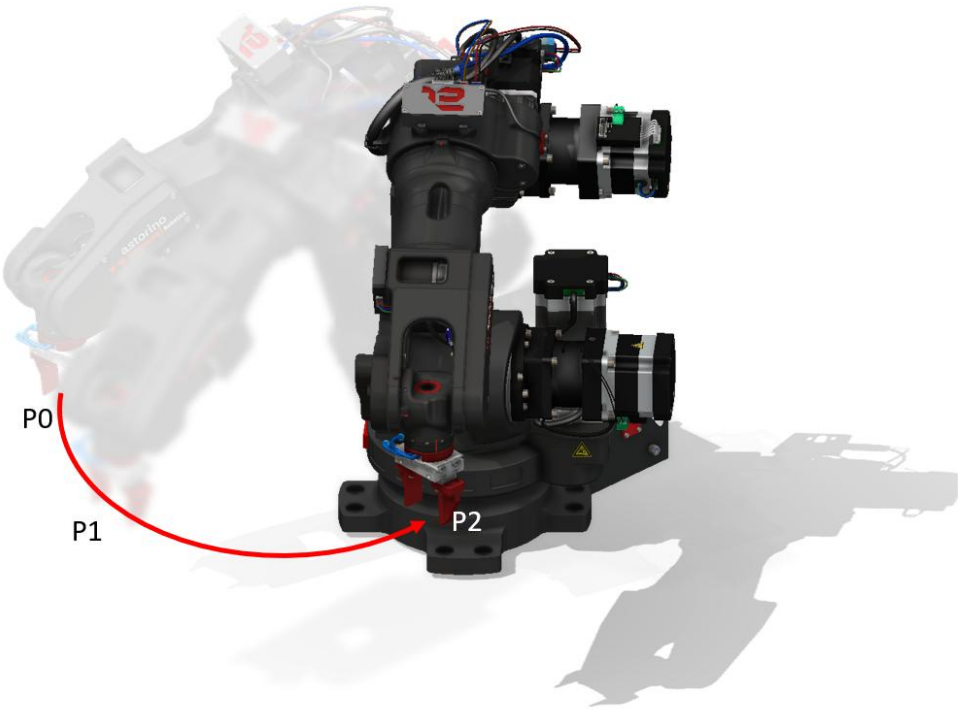
8.2 PUNKT-ZU-PUNKT-INTERPOLATION

Bei diesem Interpolationstyp bewegt sich der Roboter von der aktuellen Position auf den Bestimmungsort so hin, dass alle Achsen die Bewegung zu demselben Zeitpunkt enden. Diese Bewegung bildet einen unvorhersehbaren TCP-Pfad im 3D-Raum. Diese Bewegung erlaubt nicht singuläre Punkte zu vermeiden.



8.3 KREISINTERPOLATION

Bei diesem Bewegungstyp bewegt sich Roboter von der aktuellen Position auf den Bestimmungsort über den Mittelpunkt so hin, dass TCP im 3D-Raum einen Bogen bildet.



9 ABLAUF DER PROGRAMMAUSFÜHRUNG

Die Programmkommandos werden in der Reihenfolge von oben nach unten des Programms ausgeführt.

Das Kommando **CALL** ruft auf und führt ein anderes Programm aus, aber das verändert nicht die Reihenfolge des Ablaufs. Nach der Ausführung des Kommandos **RETURN** oder nach Beendigung der Subroutine kehrt die Verarbeitung zum aufrufenden Programm zurück und nimmt es erneut an dem Ort wieder auf, an dem sie herauskam.

Das Kommando **WAIT** hält das Programm vorm Übergang zum nächsten Schritt an und ist bis zur bestimmten Bedingung erfüllt. Die Kommandos PAUSE und HALT halten das Programm auf der Etappe an, in welchem sich diese Kommandos befinden.

9.1 SUBROUTINEN (SUBPROGRAMME)

Das Hauptprogramm kann vorläufig angehalten werden und ein anderes Programm, Subprogramm genannt, kann aufgerufen und ausgeführt werden. Nutzt man ein Subprogramm, kann man das Programm in eine modulare Struktur, die einfacher zu verstehen ist, umwandeln. Zum Beispiel ein Programm mit der Bezeichnung „**INIT**", das man dann am Anfang anderer Programme aufrufen kann. Im „**INIT**"-Programm kann man Geschwindigkeiten bestimmen, Signale und Variablen auf null setzen.

Beispiel:

```
.PROGRAM MAIN
  CALL INIT
  LMOVE P1
.END

.PROGRAM INIT
  SPEED 50 MM/S ALWAYS
  X = 10
  SIGNAL -1
.END
```

10 BEI ASTORINO VERFÜGBARE FUNKTIONEN DER AS-SPRACHE

10.1 BEFEHLE ZUR PROGRAMMSTEUERUNG

SPEED	stellt die Überwachungsgeschwindigkeit ein
PRIME	bereitet ein Programm zur Ausführung vor
EXECUTE	führt ein Programm aus
HOLD	hält die Ausführung eines Programms an
CONTINUE	nimmt die Ausführung eines Programms wieder auf
DO	führt den Einzelbefehl einer Bewegung aus
HALT	hält die Ausführung eines Programms an
RUN	nimmt die Ausführung eines Programms wieder auf
CONTINUE NEXT	übergeht einen blockierenden Befehl im Programm
REP_ONCE	stellt eine Programmschleife ein
STP_ONCE	stellt ständige oder schrittweise Funktion eines Programms ein
STPNEXT	führt einen weiteren Programmschritt aus
ZIOACTIVE	stellt den Stand des I/O-Moduls ein
STOP	schaltet das aktuell funktionierende Programm ab

SPEED Überwachungsgeschwindigkeit

Funktion

stellt die Überwachungsgeschwindigkeit, in Prozent angegeben, ein.

Parameter

Überwachungsgeschwindigkeit

stellt die Geschwindigkeit in Prozent ein. Wenn dieser Parameter einen Wert 100 hat, wird die Geschwindigkeit auf 100 % der maximalen Geschwindigkeit eingestellt. Ist der Wert gleich 50, wird die Geschwindigkeit auf eine Hälfte der maximalen Geschwindigkeit eingestellt.

Zusatzinformationen

Die Geschwindigkeit der Roboterbewegung ist ein Produkt der Überwachungsgeschwindigkeit (die mithilfe dieses Befehls eingestellt wird) und der Programmgeschwindigkeit (die im Programm mithilfe des SPEED-Befehls eingestellt wird). Beispielsweise, wenn die Überwachungsgeschwindigkeit auf 50 und die Geschwindigkeit im Programm auf 60 eingestellt ist, beträgt die maximale Geschwindigkeit des Roboters 30%.

WARNHINWEIS

Wenn das Produkt der Überwachungsgeschwindigkeit und der Programmgeschwindigkeit einen Wert von mehr als 100 hat, wird die maximale Geschwindigkeit des Roboters automatisch auf 100 % eingestellt.

Der voreingestellte Wert der Überwachungsgeschwindigkeit beträgt 80 %. Dieser Befehl hat auf die Geschwindigkeit der Bewegung, die zurzeit umgesetzt wird, keinen Einfluss. Die neu eingepflegte Geschwindigkeit wird nach Ende der aktuellen Bewegung und nach der geplanten Bewegung angewandt.

Beispiel

Wenn die Geschwindigkeit des Programms auf 100 % eingestellt ist:

>***SPEED 30*** ↲ Die Geschwindigkeit der Roboterbewegung ist auf 30 % der maximalen Geschwindigkeit eingestellt.

>***SPEED 50*** ↲ Die Geschwindigkeit der Roboterbewegung ist auf 50 % der maximalen Geschwindigkeit eingestellt.

>***SPEED 100*** ↲ Die Geschwindigkeit der Roboterbewegung ist auf 100 % der maximalen Geschwindigkeit eingestellt.

>***SPEED 200*** ↲ Die Geschwindigkeit der Roboterbewegung ist auf 100 % der maximalen Geschwindigkeit eingestellt.

PRIME *Programmname*

Funktion

bereitet das System so vor, damit das Programm mithilfe der CYCLE START-Taste ausgeführt werden kann. Dieser selbständig angewandte Befehl bewirkt die Ausführung des Programm nicht.

Parameter

Programmname
wählt ein Programm aus, welches zur Ausführung vorbereitet werden soll.

Zusatzinformationen

Dieser Befehl bereitet nur das System zur Ausführung des Programms vor. Er hat die Ausführung des Programms nicht zur Folge. Nachdem das System durch die PRIME-Befehle vorbereitet worden ist, kann das Programm mithilfe des EXECUTE-Befehls ausgeführt werden. Das Programm kann auch mithilfe der CYCLE START-Taste ausgeführt werden.

EXECUTE *Programmbezeichnung*

Funktion

führt das Programm des Roboters aus.

Parameter

Programmname
wählt das auszuführende Programm aus.

Beispiel

>**EXECUTE** test ↲	führt das Programm mit der Bezeichnung "test" stets aus. (Das Programm wird stets bis zum Anhalten, z. B. mithilfe des HOLD-Befehls oder bis zum Auftreten eines Fehlers ausgeführt.)

HOLD

Funktion

hält die Ausführung des Roboterprogramms sofort an.

Zusatzinformationen

Die Bewegung des Roboters wird sofort angehalten. Im Gegensatz zur Anwendung des EMERGENCY STOP-Schalters, erfolgt ein sofortiges Anhalten der Bewegung nicht.

CONTINUE

Funktion

nimmt die Ausführung des mit dem Befehl HOLD angehaltenen Programms wieder auf.

DO *Bewegungskommando*

Funktion

führt ein einzelnes Programmkommando aus. (Gewisse Programmkommandos können zusammen mit dieser Anweisung nicht genutzt werden.)

Parameter

Programmkommando
führt das vorgegebene Programmkommando aus. Falls dieser Parameter nicht angegeben wird, wird ein Kommando, das beim letzten Aufrufen des DO-Kommandos angegeben wurde, ausgeführt.

Zusatzinformationen

Die Programmkommandos werden in der Regel ins Programm eingepflegt und als Programmschritte ausgeführt.
Nicht desto weniger, ermöglicht das DO-Kommando die Ausführung eines einzelnen Kommandos, ohne dass es erforderlich ist, zu diesem Zweck ein Programm zu erstellen.

Beispiel

>***DO JMOVE safe*** ↲	Der Übergang des Roboters zur Position "safe" in Bewegung mit axialer Interpolation.
>***DO HOME*** ↲	Der Übergang des Roboters zur Home-Position in Bewegung mit axialer Interpolation.

HALT

Funktion

hält die Arbeit und Ausführung des Programms an.
Zusatzinformationen

Die Roboterbewegung wird sofort angehalten. Im Gegensatz zur Anwendung des NOT-HALT-Tasters, wird die Bewegung nicht plötzlich angehalten.

RUN

Funktion

nimmt die Ausführung des Programms, welches durch das Kommando HOLD/HALT angehalten wurde, wieder auf.

CONTINUE NEXT

Funktion

übergeht das Kommando wie SWAIT, TWAIT etc., welches das Programm sperrt.
Zusatzinformationen

Nach der Eingabe springt das Programm zur nächsten Zeile über.

REP_ONCE *Zustand*

Funktion

stellt den Zustand der Programmschleife des ausgeführten Programms ein.
Parameter

Zustand [ON/OFF]
schaltet die Programmschleife ein oder ab.

STP_ONCE *Zustand*

Funktion

stellt den Schritt- oder Dauermodus der Programmausführung ein.
Parameter

Zustand [ON/OFF]
schaltet den Schrittmodus ein oder ab.

STPNEXT

führt den nächsten Schritt aus, wenn der Schrittmodus aktiv ist.

ZIOACTIVE *Zustand*

Funktion

schaltet den I/O-Modus im Roboter ein oder ab.
Parameter

Zustand [0/1]
schaltet das I/O-Modul im Roboter [0] aus oder [1] ein

STOP

Funktion

hält die Ausführung des Roboterprogramms sofort an.

10.2 BEDIENUNGSBEFEHLE FÜR POSITIONSVARIABLEN

HERE	ordnet die aktuelle Position der angegeben Variablen zu.
POINT	definiert die Positionsvariable.
POINT/X	bestimmt den X-Wert der Positionsvariablen.
POINT/Y	bestimmt den Y-Wert der Positionsvariablen.
POINT/Z	bestimmt den Z-Wert der Positionsvariablen.
POINT/OAT	bestimmt die OAT-Werte der Positionsvariablen.
POINT/O	bestimmt den O-Wert der Positionsvariablen.
POINT/A	bestimmt den A-Wert der Positionsvariablen.
POINT/T	bestimmt den T-Wert der Positionsvariablen.
POINT/7	bestimmt den Wert 7 der Positionsvariablen.
POINT/8	bestimmt den Wert 8 der Positionsvariablen (Förderband 1).
POINT/9	bestimmt den Wert 9 der Positionsvariablen (Förderband 2).
DECOMPOSE	teilt Elemente der Positionsvariablen der Array-Variablen zu.
FRAME	bildet den neuen Referenzpunkt des Koordinatensystems
NULL	gibt die Null-Transformation wieder
TRANS	gibt die Transformationskoordinaten, berechnet aufgrund der vorgegebenen Bestandteile, wieder
#PPOINT	gibt die Werte der Achsenverschiebungen, berechnet aufgrund der vorgegebenen Bestandteile, wieder
SHIFT	gibt die Transformationskoordinate, die durch Verschiebung der originalen Position erzielt wurde, wieder
RX,RY,RZ	gibt den Wert der Transformation, welche die Drehung um die Achse ausdrückt, wieder.
TRADD	gibt den Wert des X-Bestandteils mit dem addierten Wert der Achse 7 wieder.
TRSUB	gibt den Wert des X-Bestandteils mit dem abgezogenen Wert der Achse 7 wieder.

HERE *Positionsvariable*

#HERE Positionsvariable

Funktion

Dieses Kommando ordnet die aktuelle Position der vorgegebenen Positionsvariablen zu. Die Position kann in Transformationswerten oder in Achsenverschiebungen angegeben werden.

Parameter

Positionsvariable
Der Wert der Variablen kann mithilfe der Transformationskoordinaten oder Achsenverschiebungen angegeben werden.

Zusatzinformationen

Die Position kann in Transformationswerten, Achsenverschiebungen oder in komplexen Transformationswerten angegeben werden.

Die Werte der Variablen werden am Terminal dargestellt
Wenn die Variable durch die Angabe der Achsenwinkel eingegeben wird (der Name der Variablen beginnt mit dem Symbol #), werden die Achsenwerte für die aktuelle Position dargestellt. Wenn die Variable Transformationskoordinaten enthält, werden die Werte XYZ OAT dargestellt. Die XYZ-Werte bestimmen die Lage des Endpunktes (TCP) des Koordinatensystems gegenüber dem Basiskoordinatensystem. Die OAT-Werte bestimmen die Orientierung des Koordinatensystems des Werkzeugs.

Beispiel

>**HERE #pick** ↲	ordnet die aktuelle Position des Roboters der Variablen "#pick" (Achsenwinkel) zu.
>**HERE place** ↲	ordnet die aktuelle Position des Roboters der Variablen "place" (Transformationskoordinaten) zu.
>**POINT PICK = HERE** ↲	ordnet die aktuelle Position des Roboters der Variablen "pick" (Transformationskoordinaten) zu.

POINT *Name der Positionsvariablen =Positionswerte*

Funktion

ordnet Informationen über die Position rechts von "=" zur Positionsvariablen links von "=" zu.

Parameter

1. ***Name der Positionsvariablen***

legt die Bezeichnung der zu bestimmenden Positionsvariablen (sie kann in Werten der Achsenverschiebung oder der Transformation bestimmt werden) fest.

2. ***Positionswerte***

Die Punktvariable oder eine Funktion, welche die Werte der Transformation oder der Winkel enthält oder wiedergibt.

[**VORSICHT**]

Wenn sich Typen der Werte rechts und links vom Zeichen ”=” unterscheiden, funktioniert der Befehl folgendermaßen:

1. POINT trans=#joint (Transformationswerte = Achsenwinkelwerte)

Die Werte der Achsenverschiebung rechts werden in Transformationswerte transformiert und der Variablen links zugeteilt.

2. POINT #joint =trans (Achsenwinkelwerte = Transformationswerte)

Die Transformationswerte rechts werden in Achsenwinkelwerte transformiert und der Variablen links zugeteilt. Die Transformationswerte werden durch den Roboter in seiner aktuellen Konfiguration transformiert.

Beispiel:

POINT TEST = P0

POINT #P0 = TEST

POINT TEST[0] = TRANS(0,0,0,0,0,0,0)

POINT #TEST = #PPOINT(0,90,90,45,0,0)

POINT/X *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/Y *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/Z *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/OAT *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/O *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/A *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/T *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/7 *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/8 *Bezeichnung der Positionsvariablen* =*Positionswerte*

POINT/9 *Bezeichnung der Positionsvariablen* =*Positionswerte*

Funktion

ordnet Informationen über die Einstellung rechts von "=" zur Positionsvariablen links von "=" zu.

Parameter

1. ***Bezeichnung der Positionsvariablen***

legt die Bezeichnung der zu bestimmenden Positionsvariablen (sie kann in Werten der Achsenverschiebung, Transformation oder in komplexen Transformationswerten bestimmt werden) fest.

2. ***Positionswerte***

Wenn es nicht bestimmt wurde, wird das Zeichen "=" übergangen.

Erklärung

dem Transformationswert werden ausschließlich bestimmte Elemente (X, Y, Z, O, A, T) zugeordnet.

Beispiel:

POINT/OAT a1 = a2 – dem OAT-Wert des Punktes a1 werden Werte des Punktes a2 zugeordnet

DECOMPOSE *Array-Variable [Nummer des Elementes] = Positionsvariable*

Funktion

speichert jedes Element der bestimmten Positionsvariablen als Element der Array-Variablen (X, Y, Z, O, A, T für den Transformationswert; JT1, JT2, JT3, JT4, JT5, JT6 für Werte der Achsenwinkel).

Parameter

1. ***Array-Variable – Bezeichnung der Array-Variablen***

bestimmt die Bezeichnung der Array-Variablen, in welcher Werte jeden Elements gespeichert werden sollen.

2. ***Nummer des Elementes***

bestimmt den ersten Array-Index, in welchem Elemente mit der Positionsvariablen gespeichert werden sollen.

3. ***Positionsvariable***

bestimmt die Bezeichnung der Positionsvariablen, aus welcher jedes Element (Transformationswerte, Werte der Achsenverschiebung) ausgesondert werden soll.

Beispiel

DECOMPOSE X[0] = part	ordnet Elemente der Transformationswerte "part" den ersten 9 Elementen in der Array-Variable **"x"** zu.
DECOMPOSE ang[4] = #pick	ordnet Elemente des Wertes der Variablen des Punktes "#pick" dem Index Nummer 4 und den ihm folgenden Indexen in der Array-Variablen "ang" zu.

Z. B., in dem vorgenannten Kommando, wenn die Werte #pick gleich 10, 20, 30, 40, 50, 60 sind, so:

ang[4]=10 ang[7]=40

ang[5]=20 ang[8]=50

ang[6]=30 ang[9]=60

FRAME(*Änd. Pos.1, Änd. Pos.2, Änd. Pos.3, Änd. Pos.4*)

Funktion

gibt Transformationswerte der (relativen) FRAME-Koordinaten gegenüber BASE-Koordinaten wieder. Es ist zu bemerken, dass nur translative Bestandteile der Transformationswerte von Positionsvariablen zur Berechnung der Koordinaten eines neuen FRAME-Systems verwendet werden.

Parameter

Wert der Positionsvariablen 1, Wert der Positionsvariablen 2

bestimmen die Transformationswerte zur Festlegung der Richtung der X-Achse. Die X-Achse der FRAME-Koordinaten ist so eingestellt, dass sie durch diese zwei Positionen durchgeht. Die positive Richtung der X-Achse ist eingestellt in Richtung von der Lage, die mit Transformationswerten der Variablen 1 bestimmt ist, zu der Lage, die mit Transformation des Wertes der Variablen 2 bestimmt ist.

Wert der Positionsvariablen 1, Wert der Positionsvariablen 3

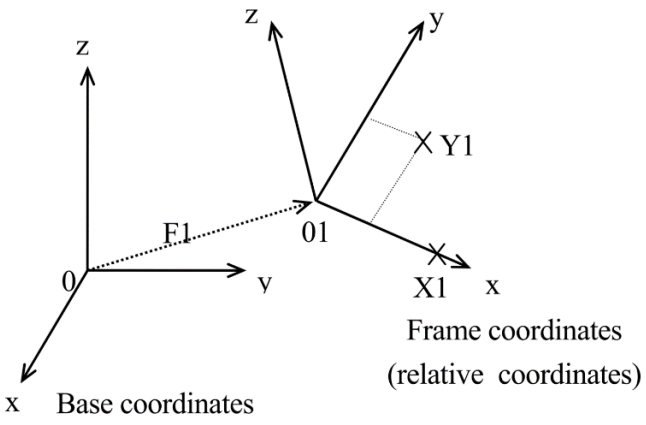
bestimmen die Transformationswerte zur Festlegung der Richtung der Y-Achse. Die Y-Achse der FRAME-Koordinaten ist so eingestellt, dass sie durch diese zwei Positionen durchgeht. Die positive Richtung der Y-Achse ist eingestellt in Richtung von der Lage, die mit Transformationswerten der Variablen 1 bestimmt ist, zu der Lage, die mit Transformation des Wertes der Variablen 3 bestimmt ist.

Wert der Positionsvariablen 4

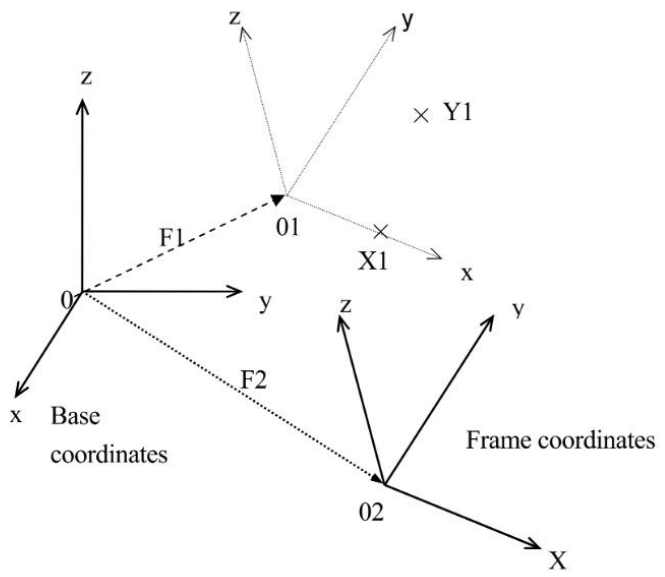
bestimmt die Variable des Transformationswertes, damit der Anfang der FRAME-Koordinaten festgelegt werden kann, der den Werten X,Y,Z, die von dieser Funktion wiedergegeben werden, gleich ist.

Erklärung

POINT F1 = FRAME(O1,X1,Y1,O1)



POINT F2 = FRAME(O1,X1,Y1,O2)



VORSICHT!

Die Punkte beim Definieren eines neuen FRAME-Systems sind zu beachten

POINT F = FRAME(a1,a2,a3,a1)

Die Achsen Y und Z sind in den vorstehenden Beispielen in verschiedene Richtungen gerichtet, je nachdem, wo der Punkt a3 eingeteacht wurde. Im Zusammenhang damit, wenn der Punkt a3 zu nahe am Punkt a1 eingeteacht ist, so kann die Achse Z des Systems falsch berechnet werden. Für ein besseres Ergebnis der Berechnungen sind die Punkte a2 und a3 weit vom Punkt a1 (mindestens 50mm) einzuteachen.

NULL

Funktion

gibt die Nulltransformation des Punktes wieder

Erklärung

gibt einen Punkt wieder, dessen Koordinaten lauter Nullen sind (Z=Y=Z=0, O=A=T=0)

Beispiel

POINT new = SHIFT(NULL BY 100,0,0)

Dist = DISTANCE(NULL,P1)

TRANS (Komp. X, Komp. Y, Komp. Z, Komp. O, Komp. A, Komp. T, JT7)

Funktion

gibt Transformationskoordinaten für vorgegebene Bestandteile, welche Verschiebungen und Drehungen bestimmen, wieder. Die Komponente JT7 kann weggelassen werden.

Parameter

Komponente X, Komponente Y, Komponente Z

Bestandteile der Verschiebungen X, Y, Z. Werden diese Parameter nicht angegeben, werden die Werte 0 angenommen.

Komponente O, Komponente A, Komponente T

bestimmen die Bestandteile der Drehungen O, A, T. Werden diese Parameter nicht angegeben, werden die Werte 0 angenommen.

Komponente JT7

bestimmt den Bestandteil der Verschiebung der JT7-Achse in mm, wird dieses Argument nicht angegeben, werden die Werte 0 angenommen.

Erklärung

Dieses Kommando berechnet die Transformationskoordinaten auf der Grundlage der Werte, die mithilfe der Parameter angegeben werden. Neue Transformationskoordinaten kann man zur Definierung der Positionsvariablen, der Koordinaten der komplexen Transformation oder in Bewegungskommandos verwenden. Die in Rede stehende Funktion ist in Verbindung mit dem Kommando DECOMPOSE (das Beispiel wurde bei Beschreibung der Funktion #PPOINT angegeben) sehr nützlich.

Beispiel

POINT temppos = TRANS(a, b+100,c, d, e, f)

#PPOINT (*jt1, jt2, jt3, jt4, jt5, jt6, jt7*)

Funktion

gibt Werte der Achsenverschiebung wieder.

Parameter

jt1, jt2, jt3, jt4, jt5, jt6, jt7

bestimmt den Wert jeden Winkels (in Graden für Drehachsen, in mm für verschiebbare Achsen).

Erklärung

Diese Funktion gibt die Werte der Achsenverschiebung wieder. Diese Werte repräsentieren die Verschiebungen in jeder von Achsen, von der Achse 1 bis zur letzten Achse (nicht unbedingt der sechsten).

WARNHINWEIS

Die Funktion #PPOINT nutzt ausschließlich Werte der Achsenwinkel. Aus diesem Grunde ist das Präfix "#" am Anfang dieser Funktion immer anzugeben.

Beispiel

POINT #temp = #PPOINT (0, a, a/2, 0, 0,0)

SHIFT(*Transformationsvariable* BY *Versch. X, Versch. Y, Versch. Z*)

Funktion

gibt die Position, die infolge einer Variablenverschiebung mithilfe der Transformationskoordinaten erhalten wurde, um den vorgegebenen Abstand gegenüber jeder Achse des Basiskoordinatensystems (X,Y,Z), wieder.

Parameter

Variable der Transformationskoordinaten
Die Variable der Transformationskoordinaten, welche die zu verschiebende Position bestimmt.
Verschiebung X, Verschiebung Y, Verschiebung Z
Verschiebungswerte in Achsen X, Y, Z des Basiskoordinatensystems.

Erklärung

Die Verschiebungswerte in Achsen X, Y und Z werden zu den Bestandteilen der Verschiebung X, Y, Z der vorgegebenen Variablen, die mithilfe der Transformationskoordinaten bestimmt ist, addiert. Das Ergebnis wird als Transformationskoordinaten wiedergegeben.

Beispiel

Wenn die Variable der Transformationskoordinaten **x** die Werte (200,150,100, 10,20,30) hat, so nach Ausführung des Kommandos

POINT y=SHIFT(x BY 5, -5, 10)
wird die Position **x** um bestimmte Werte zur Position (205,145,110,10,20,30) verschoben, die demnächst der Variablen **y** zugeordnet wird.

RX (*Winkel*)
RY (*Winkel*)
RZ (*Winkel*)

Funktion
gibt die Transformationswerte, welche die Drehung um die bestimmte Achse repräsentieren, wieder.
Parameter
Winkel
bestimmt den Drehungswert in Grad.

Erklärung
X, Y, Z repräsentieren in dieser Funktion die Koordinatenachsen. Hier wird der Drehungswert um die bestimmte Achse wiedergegeben. Die Translationswerte werden durch diese Funktion (X, Y, Z = 0) nicht wiedergegeben.

Beispiel

POINT x_rev =RX(30)	gibt den Transformationswert, der die Drehung um 30° um die Achse X repräsentiert, wieder und ordnet den Wert "x_rev" zu.

TRADD (Transformationsvariable)

Funktion

gibt die Summe der Werte der Achse 7 und des Bestandteils X der bestimmten Variablen der Transformationswerte wieder.

Parameter

Transformationsvariable
bestimmt die Bezeichnung der Variablen des Transformationswertes, an dessen Komponente X der Wert der Achse 7 addiert wird.

TRSUB (Transformationsvariable)

Funktion

gibt den durch Abzug des Wertes der Achse 7 und der Komponente X der bestimmten Variablen des Transformationswertes erreichten Wert wieder.

Parameter

Transformationsvariable
bestimmt die Bezeichnung der Variablen des Transformationswertes, von dessen Komponente X der Wert der Achse 7 abgezogen wird, wieder.

10.3 BEFEHLE ZUM SYSTEMMANAGEMENT

TOOL	zeigt das aktuell genutzte TOOL-System an
POINT	zeigt die Angaben des Punktes an
SETHOME	konfiguriert die Hausposition
ERESET	quittiert den Fehler
ZZERO	beginnt den Prozess, Achse auf null zu setzen
ZPOWER	schaltet Antriebe ein oder ab
CPUTEMP	zeigt die aktuelle Betriebstemperatur von CPU an
FREE	zeigt die aktuelle freie RAM-Speicherkapazität an
Z_OUTSOURCE	stellt die Betriebsart der 3,3V-Ausgänge ein
Z_INPULL	stellt die Betriebsart der 3,3V-Eingänge ein
Z_USER	stellt den Modus Nutzerzugang ein
STATUS	zeigt den aktuellen Status des Roboters an
WHERE	zeigt die aktuelle Position des Roboters an

TOOL

Funktion

Der Befehl zeigt Transformationskoordinaten für das Werkzeug, welche die Position des Koordinatensystems des Werkzeugs gegenüber dem Null-Koordinatensystem bestimmen, an.

Beispiel

>***TOOL*** ↲ eingegeben ins Terminal zeigt:
z. B. 0,0,100,90,0,0 an

SETHOME *Position*

Funktion

Dieser Befehl stellt die Position HOME ein und zeigt sie an.

Parameter

Position
Die Punktvariable oder der Befehl HERE (definiert die aktuelle Position als HOME-Position).

Beispiel

>***SETHOME HERE*** ↲ konfiguriert die aktuelle Position als HOME-Position.

ERESET

Funktion

quittiert den Fehler. Die Funktion dieses Befehls ist mit dem Drücken der RESET-Taste in der astorino- oder astorinoIDE-Software identisch.

Zusatzinformationen

Zum Zeitpunkt der Ausführung des ERESET-Befehls wird das RESET-Signal gesendet. Nicht desto weniger hat der Befehl jedoch keine Wirkung, wenn gleichzeitig ein Fehler eintritt. Die Funktion kann man auch in einer gekürzten Form durch Eintragung von „***ERE***" nutzen.

STATUS

Funktion

zeigt am Terminal den aktuellen Status des Roboters an.

WHERE

Funktion

zeigt am Terminal die aktuelle Position des Roboters in Achsenwinkel und in der Transformation an (XYZOAT JT7)

10.4 BEFEHLE FÜR BINÄRE SIGNALE

RESET	schaltet alle externen Signale Ein/Aus ab
SIGNAL	schaltet Ausgangssignale ein (oder ab)
PULSE	stellt den Signalzustand für bestimmte Zeit ein
DLYSIG	stellt den Signalzustand nach festgelegter Dauer ein
OPENI	stellt vordefinierte Signale in einen bestimmten Zustand ein
CLOSEI	stellt vordefinierte Signale in einen bestimmten Zustand ein
BITS	stellt die Gruppe von Signalen zu einem bestimmten Wert (max. 16 Signale) ein

RESET

Funktion

schaltet alle externen Ausgangssignale ab. Dieser Befehl hat auf dedizierte Signale keinen Einfluss.

> **WARNHINWEIS**
>
> **Es ist zu beachten, dass dieser Befehl alle Signale, mit Ausnahme der vorgenannten, sogar im Wiedergabemodus, abschaltet.**

SIGNAL *Signalnummer1,..,Signalnummer 6*

Funktion

schaltet bestimmte externe oder interne Ausgangssignale ein oder ab.

Parameter

Signalnummer1,..,Signalnummer6
Nummern der externen Ausgangssignale oder der internen Signale. Die positive Zahl schaltet das Signal ein und die negative schaltet dieses Signal ab. Die Funktion übernimmt bis zu 6 Argumenten, die Argumente kann man weglassen.

Zusatzinformationen

Die Nummer des Signals bestimmt, ob dieses extern oder intern ist.
Zulässige Signalnummern
Externe Ausgangssignale 1 – 8
MODBUS-Signale 9-56
Signale am Arm der Achse JT3 57-58 (Version B des Roboters)
Interne Signale 2001−2016
Die externen Eingangssignale können nicht angegeben werden
Wenn die Signalnummer eine positive Zahl ist, wird das Signal eingeschaltet; wenn diese eine negative Zahl ist, wird das Signal abgeschaltet.

Beispiel

>***SIGNAL -1*** ↲	Abschalten des externen Ausgangsignals 1,
>***SIGNAL -1,2*** ↲	Abschalten des externen Ausgangsignals 1 und Einschalten des externen Ausgangssignals 2
>***SIGNAL 2005*** ↲	Einschalten des internen Signals 2005,
>***SIGNAL -reset*** ↲	Wenn die Variable "Reset" positiven Wert hat, wird das durch diesen Wert bestimmte Ausgangssignal abgeschaltet.

PULSE *Signalnummer, Zeit*

Funktion

schaltet das bestimmte externe Ausgangssignal oder das interne Signal für eine bestimmte Dauer ein.

Parameter
Signalnummer
wählt die Nummern der externen Ausgangs- oder Eingangssignale aus. Die Auswahl eines dedizierten Signals hat einen Fehler zur Folge.

Zulässige Signalnummern	
Externes Ausgangssignal	1 – tatsächliche Anzahl der Signale
Internes Signal	2001-2016

Zeit

stellt die Zeit der Signalexposition (in Sekunden) ein. Wenn sie nicht bestimmt wird, wird sie automatisch auf 0.0 Sekunden eingestellt.

Beispiel

PULSE 3, 0.4 stellt das Signal 4 auf 0,4 Sek. ein

DLYSIG *Signalnummer, Zeit*

Funktion

sendet ein bestimmtes Signal nach Ablauf einer bestimmten Zeit.

Parameter
Signalnummer
wählt die Nummer des externen Ausgangssignals oder des internen Signals aus. Wenn die Signalnummer positiv ist, ist das Signal eingeschaltet; wenn sie negativ ist, ist das Signal abgeschaltet. Die Auswahl des dedizierten Signals hat einen Fehler zur Folge.

Zulässige Signalnummern	
Externes Ausgangssignal	1 – tatsächliche Anzahl der Signale
Internes Signal	2001-2016

Zeit

bestimmt die Verzögerung des Signalausgangs in Sekunden.

Beispiel

DLYSIG 1, 0.8 stellt das Signal 1 nach 0.8 Sek. ab Aufruf des Befehls in hohen Zustand ein.

OPENI

Funktion

erlaubt mit einem Befehl den einen Ausgang in hohen Zustand und den anderen Ausgang in niedrigen Zustand zu versetzen. Die Ausgänge sind durch die Einstellung von Handling Clamp bestimmt.

CLOSEI

Funktion

erlaubt mit einem Befehl den einen Ausgang in niedrigen Zustand und den anderen Ausgang in hohen Zustand zu versetzen. Die Ausgänge werden durch die Einstellung von Handling Clamp bestimmt.

BITS *Nummer des Anfangssignals, Anzahl der Signale = Dezimalwert*

Funktion

verteilt eine Gruppe von externen Ausgangssignalen oder internen Signalen im binären Muster. Die Signalzustände werden als ON/OFF gemäß binärer Entsprechung eines bestimmten Dezimalwertes eingestellt.

Parameter

Nummer des Startsignals
bestimmt das erste Signal zur Einstellung des Signalzustands.

Zulässige Signalnummern	
Externes Ausgangssignal	1 – tatsächliche Anzahl der Signale
Internes Signal	2001-2016

Anzahl der Signale

bestimmt die Anzahl der Signale, die als ON/OFF eingestellt werden sollen. Die maximale zulässige Anzahl ist 16.

Dezimalwert

(der vorgebenen Signalnummer) wird eingestellt und sonstige Bits bestimmen den Dezimalwert, der zur Einstellung der geforderten ON/OFF-Signalzustände dient. Der Dezimalwert wird in die binäre Notation transformiert und jedes Bit des Binärwertes stellt einen Zustand des Signals ein, von am wenigsten bedeutenden Bit beginnend. Wenn der binäre Code dieses Wertes mehr Bits hat als die Anzahl der Signale, so wird der Zustand der jeweiligen Anzahl der Signale, beginnend von…, ignoriert.

Beispiel

BITS 9,16 = 12082 stellt in Signalen den 9- bis 16-Bit-Wert der Zahl 12082 ein

10.5 BEFEHLE ZUR BEDIENUNG DER PROGRAMME UND DER DATEN

DELETE	löscht ein Programm oder einen Punkt aus dem Roboterspeicher
DELETE/P	löscht ein Programm aus dem Roboterspeicher
DELETE/L	löscht einen Punkt aus dem Roboterspeicher

DELETE *Programmbezeichnung/Punktbezeichnung*

DELETE/P *Programmbezeichnung*

DELETE/L *Punktbezeichnung*

Funktion

löscht bestimmte Daten aus dem Roboterspeicher

Parameter

/P – Programmbezeichnung, /L Punktbezeichnung

Beispiel:

> ***DELETE TEST***↲ löscht das Programm Test, wenn das Programm Test nicht gefunden wurde, versucht die Punktvariable mit der Bezeichnung Test zu löschen

> ***DELETE/L P1***↲ löscht den Punkt P1 aus dem Roboterspeicher.

10.6 BEFEHLE ZUM ANZEIGEN VON MITTEILUNGEN

PRINT zeigt Daten im Terminal an

TYPE zeigt Daten des numerischen Typs im Terminal an

PRINT *Daten zum Anzeigen*

Funktion

zeigt am Terminal vorgegebene Daten an.

Parameter

Daten zum Anzeigen
Es ist eine oder mehrere aus nachfolgenden Optionen auszuwählen.
(1) eine Zeichenfolge z. B. ***PRINT $TST***
(2) ein Ausdruck des reellen Typs (der Wert wird zuerst berechnet und demnächst angezeigt, z. B. ***PRINT SIN(30)***

Beispiel

In diesem Beispiel ist der Wert der reellen Variablen "i" gleich 5
>***PRINT i*** ↲
Die Anzeige soll wie unten aussehen:
>***5.00***

>***PRINT „HELLO”***
>***HELLO***

>***$TEMP = „MEIN TEXT”*** ↲
>**PRINT $TEMP**↲
>***MEIN TEXT***

TYPE *Daten zum Anzeigen*

Funktion

zeigt am Terminal Daten des numerischen Typs (Zahlen) an.

Parameter

Daten zum Anzeigen
Es ist eine oder mehrere aus nachfolgenden Optionen auszuwählen.
(1) eine Zeichenfolge z. B. ***TYPE TST***
(2) ein Ausdruck des reellen Typs (der Wert wird zuerst berechnet und demnächst angezeigt, z. B. ***TYPE SIN(30)***

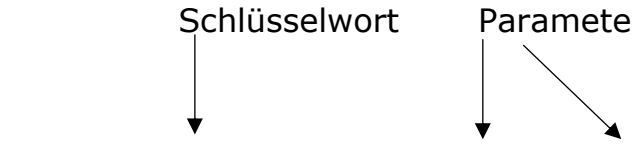
Beispiel

In diesem Beispiel ist der Wert der reellen Variablen "i" gleich 5
>***TYPE i*** ↲
Die Anzeige soll wie unten aussehen:
>***5.00***

11 PROGRAMMKOMMANDOS

Das Programmkommando besteht aus einem Schlüsselwort, das den Befehl ausdrückt und aus einem Parameter (aus Parametern), wie im nachfolgenden Beispiel gezeigt wurde.

Beispiel



LAPPRO *Positionsvariable, Abstand*

Das Schlüsselwort und der Parameter müssen mit einem Leerschritt getrennt sein.

11.1 BEWEGUNGSKOMMANDOS

TOOL	schaltet das Werkzeugsystem ein
JMOVE	Bewegung des Roboters mit axialer Interpolation
LMOVE	Bewegung des Roboters mit linearer Interpolation
XMOVE	Bewegung des Roboters mit linearer Interpolation mit Unterbrechung
DELAY	hält die Bewegung des Roboters für bestimmte Zeit an
BRAKE	hält die Bewegung des Roboters an
JAPPRO	bringt den Roboter an den Bestimmungspunkt in einer Bewegung mit axialer Interpolation
LAPPRO	bringt den Roboter an den Bestimmungspunkt mit linearer Interpolation näher
JDEPART	verlässt die laufende Position mit einer Bewegung mit axialer Interpolation
LDEPART	verlässt die laufende Position in einer Bewegung mit linearer Interpolation
HOME	Bewegung des Roboters zur Hausposition
DRIVE	Bewegung in Richtung einer Einzelachse
DRAW	Übergang um einen bestimmten Abstand in Richtung der Achsen X, Y, Z des globalen Koordinatensystems
TDRAW	Übergang um einen bestimmten Abstand in Richtung der Achsen X, Y, Z des Koordinatensystems des Werkzeugs
C1MOVE	Bewegung mit Kreisinterpolation
C2MOVE	Bewegung mit Kreisinterpolation
ALIGN	richtet die Orientierung des Roboters (Achse **vom** TOOL-System) zur nächsten Achse des BASE-System aus

TOOL *Nummer des Toolsystems*

TOOL Variable des Transformationswertes

Funktion

Dieser Befehl definiert die Transformationskoordinaten für das Werkzeug, welche die Position des Koordinatensystems des Werkzeugs gegenüber dem Null-Koordinatensystem bestimmen.

Parameter

Nummer des Tool-Systems
Eines der 3 gespeicherten Tool-Systeme.
Variable des Transformationswertes
Punktvariable, welche die Daten: **X,Y,Z** und **O,A,T** der Verschiebungen und der Drehung des Koordinatensystems des Werkzeugs enthält

Wert
1
2
3
4 – Werte, die vom Programmniveau bezogen werden

Zusatzinformationen

Wenn der Wert von 1 bis 3 als Parameter angegeben wird, sind die Transformationskoordinaten vom Werkzeugkoordinatensystem auf die im Roboterspeicher gespeicherten Werte eingestellt. Der Wert ist für das vom Programmniveau aufgerufene TOOL-System reserviert und wird als Variable des Typs Punkt bezeichnet. Das Null-Werkzeugkoordinatensystem hat seine Mitte am Montageflansch des Werkzeugs und die Achsen sind zur Achse des letzten kinematischen Paars des Roboters parallel angeordnet. Während der Initialisierung des Systems werden die Transformationskoordinaten des Werkzeugs automatisch auf Nullkoordinaten eingestellt.
Nachdem der Roboter zu einer mithilfe der Transformationskoordinaten definierten Einstellung oder durch manuelle Steuerung im Basissystem oder im Koordinatensystem der Werkzeuge übergeht, berechnet das System automatisch die Position des Roboters, indem es die eingegebenen Koordinaten der Werkzeugtransformation berücksichtigt.

Der Parameter 4 ist für die Situation bestimmt, wenn die Daten des TOOL-Systems vom Programmniveau des Roboters bezogen werden:

POINT T1 = TRANS(0,0,100,0,0,0)
TOOL T1

Beispiel

>**TOOL 1** ↲ verändert die Position des Koordinatensystems des Werkzeugs zu der im Reiter TOOL bestimmten Position.

JMOVE *Positionsvariable*

LMOVE *Positionsvariable*

Funktion

Bewegung des Roboters zur bestimmten Position.
JMOVE: Bewegung mit axialer Interpolation.
LMOVE: Bewegung mit linearer Interpolation.

Parameter

Positionsvariable
bestimmt die Bestimmungsposition des Roboters.

Zusatzinformationen

Der JMOVE-Befehl bedingt die Bewegung des Roboters mit axialer Interpolation. Der Roboter bewegt sich so, dass das Verhältnis des zurückgelegten Abstands zum Gesamtabstand für alle Achsen über die gesamte Dauer der Bewegung von der Anfangsposition bis zur Endposition gleich bleibt. Bei der Ausführung des LMOVE-Befehls bewegt sich der Roboter mit einer Bewegung mit linearer Interpolation. Der Anfangspunkt des Koordinatensystems des Werkzeugs (TCP) verschiebt sich entlang der linearen Bahn.

Beispiel

JMOVE #pick die Bewegung mit axialer Interpolation zur Position, welche mithilfe der Verschiebungswerte der Achse "#pick" bestimmt ist.
LMOVE P1 die Bewegung mit linearer Interpolation zur Position, welche mithilfe der Transformationskoordinaten P1 bestimmt ist.
LMOVE #pick die Bewegung mit linearer Interpolation zur Position, welche mithilfe der Werte der Achsenverschiebungen "#pick" bestimmt ist.

XMOVE *Positionsvariable TILL Signalnummer*

Funktion

Bewegung des Roboters zu einer bestimmten Position, bis das Signal erhalten ist.

Parameter

Positionsvariable
bestimmt die Bestimmungsposition des Roboters.
Signalnummer

Zusatzinformationen

Das Kommando JMOVE bedingt die Bewegung des Roboters mit axialer Interpolation. Der Roboter bewegt sich so, dass das Verhältnis des zurückgelegten Abstands zum Gesamtabstand für alle Achsen über die gesamte Dauer der Bewegung von der Anfangsposition bis zur Endposition gleich bleibt. Bei der Ausführung des LMOVE-Kommandos bewegt sich der Roboter in einer Bewegung mit linearer Interpolation. Der Anfangspunkt des Koordinatensystems des Werkzeugs (TCP) verschiebt sich entlang der linearen Bahn.

Beispiel

JMOVE #pick - eine Bewegung mit axialer Interpolation zu einer Position, welche mithilfe der Verschiebungswerte der Achse "#pick" bestimmt ist.
LMOVE P1 - eine Bewegung mit linearer Interpolation zu einer Position, welche mithilfe der Transformationskoordinaten P1 bestimmt ist.
LMOVE #pick - eine Bewegung mit linearer Interpolation zu einer Position, welche mithilfe der Werte der Achsenverschiebungen "#pick" bestimmt ist.

DELAY *Zeit*

Funktion

hält die Bewegung des Roboters für eine bestimmte Zeit an.

Parameter

Zeit - bestimmt die Zeit des Anhaltens der Roboterbewegung in Sekunden.

Zusatzinformationen

Im AS-System wird der DELAY-Befehl als ein Bewegungskommando, das "die Position nicht verändert", behandelt.
Sogar wenn die Bewegung des Roboters durch das DELAY-Kommando angehalten wird, werden alle Programmschritte, die sich vor dem weiteren Bewegungskommando befinden, vor dem Anhalten ausgeführt.

Beispiel

DELAY 2.5 hält die Bewegung des Roboters 2,5 Sekunden lang an.

JAPPRO *Positionsvariable , Abstand*

LAPPRO *Positionsvariable , Abstand*

Funktion

Die Bewegung des Werkzeugs in Richtung der Z-Achse für den vorgegebenen Abstand von der erlernten Position.
JAPPRO: Bewegung mit axialer Interpolation.
LAPPRO: Bewegung mit linearer Interpolation.

Parameter

Positionsvariable
bestimmt die Endposition (in Transformationswerten oder in Werten der Achsenverschiebungen).
Abstand
bestimmt den Abstand (in Millimetern) zwischen der Endposition und der tatsächlichen Position, die durch den Roboter in Richtung der Z-Achse des Koordinatensystems des Werkezugs erreicht wurde. Wenn der vorgegebene Abstand positiv ist, bewegt sich der Roboter in negativer Richtung der Z-Achse. Wenn der vorgegebene Abstand negativ ist, bewegt sich der Roboter in positiver Richtung der Z-Achse.

Zusatzinformationen

In diesen Befehlen wird das Werkzeug in einer bestimmten Position orientiert und die Position wird in einem bestimmten Abstand von der vorgegebenen Position in Richtung der Z-Achse des Werkzeugkoordinatensystems eingestellt.

Beispiel

JAPPRO place,100
Die Bewegung mit axialer Interpolation zur Position 100 mm von der Position "place", in Richtung der Z-Achse des Werkzeugkoordinatensystems. Die Position "place" ist mithilfe der Transformationskoordinaten angegeben.
LAPPRO place, offset
Die Bewegung mit linearer Interpolation zur Position, welche von der Position "place" entfernt ist, angegeben mithilfe der Transformationskoordinaten um einen durch die Variable "offset" in Richtung der Z-Achse des Werkzeugkoordinaten-systems bestimmten Abstand.

BRAKE

JDEPART *Abstand*

LDEPART *Abstand*

Funktion

Die Bewegung des Roboters zur Position, welche von der aktuellen Position um vorgegebenen bestimmten Abstand entfernt ist, entlang der Z-Achse des Werkzeugkoordinatensystems.
JDEPART: Bewegung mit axialer Interpolation.
LDEPART: Bewegung mit linearer Interpolation.

Parameter

Abstand
bestimmt den Abstand zwischen der aktuellen Position und der Bestimmungsposition in Richtung der Z-Achse des Werkzeugkoordinatensystems in Millimetern. Wenn der angegebene Abstand positiv ist, bewegt sich der Roboter "zurück" oder anders gesagt, in Richtung der negativen Werte der Z-Achse. Wenn der angegebene Abstand negativ ist, bewegt sich der Roboter "nach vorn", oder anders gesagt, in Richtung der positiven Werte der Z-Achse.

Beispiel

JDEPART 80
Die Bewegung des Roboters mit axialer Interpolation zurück um 80 mm in Richtung der Z-Achse des Werkzeugkoordinatensystems.
LDEPART 2*offset
Die Bewegung des Roboters mit linearer Interpolation zurück, Abstand 2∗offset (200 mm wenn offset = 100) in Richtung der Z-Achse des Koordinatensystems des Werkzeugs.

HOME

Funktion

Bewegung mit axialer Interpolation zur Position, die als HOME definiert ist.

Beispiel

HOME
Die Bewegung mit axialer Interpolation zur Home-Position, die mithilfe des Befehls/Kommandos SETHOME oder in der astorino- und astorinoIDE-Software bestimmt ist.

DRIVE *Nummer der Achse, Verschiebung, Geschwindigkeit*

Funktion

Bewegung einer Achse des Roboters.

Parameter

Nummer der Achse
Nummer der Achse, die verschoben werden soll. (Bei Robotern, die mit sechs Achsen ausgestattet sind, sind diese mit den Nummern von 1 bis 6 gekennzeichnet, von der am meisten vom Montageflansch entfernten Achse gerechnet.)
Verschiebung
Die Größe der Achsenverschiebungen wird als positiver oder negativer Wert angegeben.
Dieser Wert wird in denselben Einheiten wiedergegeben, die zur Beschreibung der Position der Achse verwendet werden, d. h. wenn die Achse eine Drehachse ist, wird dieser Wert in Grad (°) angegeben und wenn die Achse eine Schubachse ist, wird dieser Wert in Längenmaßeinheit (mm) angegeben.
Geschwindigkeit
Geschwindigkeit der Bewegung. Ähnlich wie es bei der Standardbewegungsgeschwindigkeit der Fall ist, wird diese in Prozentzahlen der Überwachungsgeschwindigkeit angegeben. Wird dieser Parameter nicht angegeben, wird die Überwachungsgeschwindigkeit mit 100% angenommen.

Zusatzinformationen

Das in Rede stehende Kommando verschiebt nur eine Achse.
Die Überwachungsgeschwindigkeit dieses Kommandos ist eine Kombination der in diesem Kommando angegebenen Geschwindigkeit und der Überwachungsgeschwindigkeit. Die im Programm eingestellte Geschwindigkeit hat auf dieses Kommando keinen Einfluss.

Beispiel

DRIVE 2,-10,75
Die Verschiebung der Achse 2 (JT2) von der laufenden Position um −10°. Diese Geschwindigkeit beträgt 75% der Überwachungsgeschwindigkeit.

DRAW *Verschiebung X, Verschiebung Y, Verschiebung Z, Drehung um X, Drehung um Y, Drehung um Z*

TDRAW *Verschiebung X, Verschiebung Y, Verschiebung Z, Drehung um X, Drehung um Y, Drehung um Z*

Funktion

Roboterbewegung mit linearer Interpolation, mit bestimmter Geschwindigkeit, von der laufenden Position zu dem vorgegebenen Abstand in Richtung der Achse X, Y, Z und Drehung um den vorgegebenen Winkel um jede von diesen Achsen. Das Kommando DRAW verschiebt den Roboter im globalen Koordinatensystem und das Kommando TDRAW verschiebt den Roboter im Werkzeugkoordinatensystem.

Parameter

Verschiebung X
Die Größe der Verschiebung in der X-Achse, in mm angegeben. Wird dieser Parameter nicht angegeben, wird der Wert 0 mm eingepflegt.
Verschiebung Y
Die Größe der Verschiebung in der Y-Achse, in mm angegeben. Wird dieser Parameter nicht angegeben, wird der Wert 0 mm eingepflegt.
Verschiebung Z
Die Größe der Verschiebung in der Z-Achse, in mm angegeben. Wird dieser Parameter nicht angegeben, wird der Wert 0 mm eingepflegt.
Drehung um X
Der Drehwinkel um die X-Achse, in Grad angegeben. Der zulässige Wertbereich ist kleiner als ±180°. Falls dieser Parameter nicht angegeben wird, wird der Wert 0 Grad angenommen.
Drehung um Y
Der Drehwinkel um die Y-Achse, in Grad angegeben. Der zulässige Wertbereich ist kleiner als ±180°. Falls dieser Parameter nicht angegeben wird, wird der Wert 0 Grad angenommen.
Drehung um Z
Der Drehwinkel um die Z-Achse, in Grad angegeben. Der zulässige Wertbereich ist kleiner als ±180°. Falls dieser Parameter nicht angegeben wird, wird der Wert 0 Grad angenommen.

Zusatzinformationen

Der Roboter wird in linearer Bewegung von der aktuellen Position zur vorgegebenen Position verfahren.

Beispiel

DRAW 50,0,-30	Roboterbewegung in linearer Interpolation um 50 mm in Richtung der X-Achse und um –30 mm in Richtung der Achse Z des globalen Koordinatensystems.
TDRAW 0,0,50	Roboterbewegung in linearer Interpolation um 50 mm in Richtung der Achse Z des Werkzeugs.

C1MOVE *Positionsvariable*

C2MOVE *Positionsvariable*

Funktion

Bewegung des Roboters zur bestimmten Position in Kreisinterpolation.

Parameter

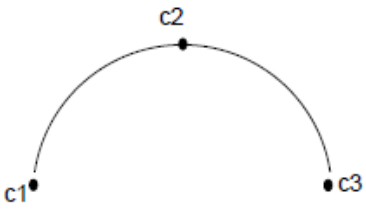
Positionsvariable
legt die Bestimmungsposition der Roboterbewegung fest. (Sie kann mithilfe der Transformationskoordinaten, des Wertes der komplexen Transformation, des Wertes der Achsenverschiebungen oder Funktionen mit der Information über die Position angegeben werden.)

Zusatzinformationen

Das C1MOVE-Kommando verfährt den Roboter zu einem Punkt, der sich in der Mitte des Weges auf der Kreisbahn befindet und das C2MOVE-Kommando verfährt ihn bis zum Ende der Kreisbahn.
Damit man den Roboter in einer Bewegung mit Kreisinterpolation verfahren kann, ist es erforderlich, drei Positionen einzuteachen. Diese drei Positionen unterscheiden sich für die Kommandos C1MOVE und C2MOVE.
1. Position von dem letzten Bewegungskommando oder die aktuelle Position.
2. Position, die als Parameter des C1MOVE-Kommandos verwendet wird.
3. Position des nächsten Bewegungskommandos C2MOVE.

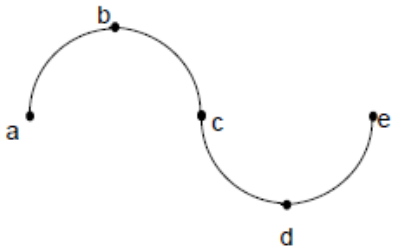
Przykład

```
JMOVE   c1
C1MOVE c2
C2MOVE c3
```



Robot przesuwa się ruchem z interpolacją osiową do c1, a następnie przesuwa się ruchem z interpolacją kołową po łuku utworzonym przez c1, c2 i c3.

```
JMOVE   #a
  C1MOVE #b      } łuk a,b,c
  C2MOVE #c
C1MOVE #d        } łuk c,d,e
  C2MOVE #e
```



Beispiel

Der Roboter bewegt sich mit einer Bewegung in axialer Interpolation zu c1 und demnächst verschiebt er sich mit einer Bewegung in Kreisinterpolation über einem Bogen, den c1, c2 und c3 bilden.

a,b,c-Bogen c,d,e-Bogen

ALIGN

Funktion

verfährt die Z-Achse des Koordinatensystems so, dass sie zu der nächsten Achse der Basiskoordinaten parallel ist.

Zusatzinformationen

Wenn man will, dass die Richtungsbewegung entlang der Z-Richtung des Werkzeugs eingestellt ist, ermöglicht die Funktion **DO ALIGN** einfache Ausrichtung der Werkzeugrichtung zu den Basiskoordinaten vor dem Einteachen der Punkte.

11.2 BEWEGUNGSKOMMANDOS IN KOOPERATION MIT DEM FÖRDERBAND

CVLMOVE	lineare Bewegung in Kooperation mit dem Förderband
CVLAPPRO	verfährt den Roboter näher an den Zielpunkt linear in Kooperation mit dem Förderband
CVLDEPART	verlässt die aktuelle Position linear in Kooperation mit dem Förderband
CVDELAY	hält die Bewegung des Roboters für eine bestimmte Zeit in Kooperation mit dem Förderband an
CVWAIT	hält den Betrieb des Roboters, bis das Förderband seinen vorgegebenen Wert erreicht hat, an
CVRESET	überschreibt die aktuelle Position des Förderbands
CVPOS	liest die aktuelle Position des Förderbands 1 ab
CVPOS2	liest die aktuelle Position des Förderbands 2 ab
CVCOOPJT	schaltet die Kooperation des Roboters mit dem Förderband 1 oder 2 ein

CVLMOVE *Positionsvariable*

Funktion

Bewegung des Roboters zur bestimmten Position in linearer Interpolation, die sich mit dem Förderband synchronisiert.

Parameter

Positionsvariable

legt die Bestimmungsposition der Roboterbewegung fest (sie kann in Transformationswerten, komplexen Transformationswerten oder Werten der Achsenwinkel enthalten sein).

Erklärung

TCP verfolgt die lineare Trajektorie von der Anfangsposition zur Endposition, indem es sich mit dem Förderband synchronisiert.

Beispiel

CVLMOVE #pick	Bewegung in linearer Interpolation, definiert durch die Werte der Achsenwinkel (#pick) während der Synchronisierung mit dem Förderband.
CVLMOVE Place	Bewegung in linearer Interpolation zur Position, die durch die Transformationsvariable „place" während der Synchronisierung mit dem Förderband definiert ist.

CVLAPPRO *Positionsvariable, Abstand*

Funktion

Bewegung in linearer Interpolation für einen bestimmten Abstand von bestimmter Position, indem sie mit dem Förderband synchronisiert wird.

Parameter

Positionsvariable

bestimmt die Bestimmungsposition (in Transformationswerten oder Achsenwinkeln)

Abstand

bestimmt den Abstand in Richtung der Z-Achse des Werkzeugs zwischen der vorgenannten Bestimmungsposition und der Position, welche der Roboter tatsächlich erreicht. (Einheit: mm)

Nach Angabe des positiven Abstandwertes verfährt den Roboter von der Bestimmungsposition (negative Richtung der Z-Achse des Werkzeugs). Die Angabe des negativen Wertes verfährt den Roboter in Richtung Bestimmungsposition (positive Richtung der Z-Achse des Werkzeugs).

Erklärung

Bei diesem Kommando ist die Orientierung des Werkzeugs in der Position, welche der Roboter tatsächlich erreicht, durch die Orientierung der angegebenen Positionsvariablen festgelegt. Die Lage des Werkzeugs wird zu einer Lage, die von der bestimmten Position um einen bestimmten Abstand in positiver oder negativer Richtung der Z-Achse des Werkzeugs entfernt ist.

Beispiel

CVLAPPRO Place,100	Der Roboter synchronisiert sich mit dem Förderband und bewegt sich mit der Bewegung in linearer Interpolation zu der um 100 mm in Richtung der Z-Achse des Werkzeugs entfernten Lage, welche durch die Transformationswerte „Place" definiert ist, hin.

CVLDEPART Abstand

Funktion

Bewegung in linearer Interpolation für einen bestimmten Abstand von der aktuellen Position, indem es mit Förderband synchronisiert wird.

Parameter

Abstand

bestimmt den Abstand in Richtung der Z-Achse des Werkzeugs zwischen der aktuellen Position und der Position, welche der Roboter tatsächlich erreicht. (Einheit: mm)

Die Angabe eines positiven Abstandswertes verfährt den Roboter von der aktuellen Position (negative Richtung der Z-Achse des Werkzeugs). Die Angabe eines negativen Wertes verfährt den Roboter in Richtung der aktuellen Position (positive Richtung der Z-Achse des Werkzeugs).

Erklärung

Bei diesem Kommando wird die Orientierung des Werkzeugs in der Position, welche der Roboter tatsächlich erreicht, durch die Orientierung der aktuellen Position des Roboters bestimmt. Die Lage des Werkzeugs wird zu einer Lage, welche von der derzeitigen Position um einen bestimmten Abstand in positiver oder in negativer Richtung der Z-Achse des Werkzeugs entfernt ist.

Beispiel

CVLDEPART 100	Der Roboter synchronisiert sich mit dem Förderband und bewegt mit einer Bewegung in linearer Interpolation zu der Position, die um 100 mm in Richtung der Z-Achse des Werkzeugs von der aktuellen Lage entfernt ist, hin.

CVDELAY *Zeit*

Funktion

"hält" die Bewegung des Roboters, vom Bezugspunkt des Förderbands gesehen, für eine bestimmte Dauer „an".

Parameter

Zeit

bestimmt die Zeit, in welcher der Roboter von Perspektive des Förderbands gesehen, "unbeweglich" bleiben muss (Einheit: Sekunden)

Erklärung

Das Kommando **CVDELAY** ist ein Bewegungskommando des Roboters. Nachdem dieses Kommando ausgeführt wurde, wird die Bewegung des Roboters so kontrolliert, damit dieselbe Position gegenüber dem sich bewegenden Förderband erhalten bleibt, daher seitens des Förderbands gesehen, scheint der Roboter anzuhalten. (Von außen gesehen bewegt sich der Roboter mit dem Förderer, wodurch dieselbe Position gegenüber dem zu bearbeitenden Gegenstand/Detail auf dem Förderband erhalten bleibt).

Beispiel

CVDELAY 2.5 Der Roboter bleibt 2,5 Sekunden, von der Perspektive des Förderbands schauend, unbeweglich.

CVWAIT *Ausgangsposition*

Funktion

hält die Ausführung des Programms, bis das Förderband die bestimmte Position erreicht hat, an. (Einheit: mm).

Parameter

Ausgangsposition

Der Betrieb des Roboters beginnt wieder, wenn das Förderband diese Position erreicht.

Erklärung

Wenn dieses Kommando im Programm ausgeführt wird, hält der Roboter an und wartet (führt im Programm keinen weiteren Schritt aus), bis das Förderband die bestimmte Position erreicht hat. Der Roboter nimmt den Betrieb wieder auf, wenn das Förderband die bestimmte Position erreicht hat.

Beispiel

CVWAIT 50 Die weitere Ausführung des Programms ist eingestellt, bis die Fahrt des Förderbands von 50 mm erreicht ist.

CVRESET *Position*

Funktion

resettet den Positionswert des aktuell kooperierenden Förderbands.

Parameter

bestimmt den Positionswert, auf welchen das aktuell kooperierende Förderband resettet werden soll.

Erklärung

Das Kommando CVRESET resettet das aktuell kooperierende Förderband (mit dem Kommando CVCOOPJT beschrieben) auf die vorgegebene Position.

Beispiel

CVRESET 200 stellt die aktuelle Förderbandposition auf 200 mm ein

CVPOS

Funktion

schreibt die aktuelle Position des ersten Bandförderers der Variablen zu.

Beispiel

CONV1 = CVPOS schreibt der Variablen CONV1 den aktuellen Positionswert des ersten Förderbands zu

CVPOS2

Funktion

schreibt die aktuelle Position des zweiten Bandförderers der Variablen zu.

Beispiel

CONV2 = CVPOS2 schreibt der Variablen CONV2 den aktuellen Positionswert des zweiten Bandförderers zu

CVCOOPJT *Nummer des Bandförderers*

Funktion

schaltet die Kooperation des Roboters mit dem Förderband ein.

Parameter

Nummer der Förderbands

bestimmt, mit welchem Förderband der Roboter aktuell kooperieren soll.
Mögliche Werte sind 1 und 2.

Zusatzinformationen

Wenn die Funktion nicht angewandt wurde, kooperiert der Roboter mit erstem Förderband.

Beispiel

CVCOOPJT 1 schaltet die Kooperation des Roboters mit erstem Förderband ein

11.3 KOMMANDOS ZUR STEUERUNG DER GESCHWINDIGKEIT UND DER GENAUIGKEIT

SPEED stellt Bewegungsgeschwindigkeit (Programmgeschwindigkeit) ein

ACCEL stellt die Beschleunigung ein

DECEL stellt das Bremsen ein

SPEED *Geschwindigkeit*
SPEED *Geschwindigkeit* ALWAYS

Funktion

bestimmt die Geschwindigkeit der Roboterbewegung.

Parameter

Geschwindigkeit
bestimmt die Geschwindigkeit des Programms. In der Regel wird diese angegeben als ein Prozentwert aus dem Bereich 0,01 do 100 (%). Man kann ebenfalls den absoluten Wert angeben, indem die Einheit: MM/S hinzugefügt wird
ALWAYS
Die vorgegebene Geschwindigkeit wird jedes weitere Bewegungskommando betreffen, bis zum nächsten SPEED-Kommando

Zusatzinformationen

Die tatsächliche Bewegungsgeschwindigkeit des Roboters ist ein Produkt der Überwachungsgeschwindigkeit und der Bewegungsgeschwindigkeit, die mithilfe dieser Anleitung eingestellt wird (Überwachungsgeschwindigkeit x Programmgeschwindigkeit). Nicht desto weniger kann man jedoch nicht garantieren, dass die volle Geschwindigkeit in folgenden Fällen erreicht wird:
1. wenn der Abstand zwischen zwei eingeteachten Positionen zu klein ist,
2. wenn die Bewegung in Kreisinterpolation eingeteacht wurde, die die maximale Winkelgeschwindigkeit der Achse überschreitet.

Die Bewegungsgeschwindigkeit wird anders bestimmt bei der Bewegung mit axialer Interpolation und bei der Bewegung mit linearer Interpolation. Im Falle der axialen Interpolation, wird die Bewegungsgeschwindigkeit als ein Prozentsatz der maximalen Geschwindigkeit jeder von Achsen bestimmt. Bei der Bewegung mit linearer Interpolation, wird die Bewegungsgeschwindigkeit als ein Prozentsatz der maximalen Geschwindigkeit am Anfangspunkt des Werkzeugkoordinatensystems bestimmt. Wenn die Geschwindigkeit mit einer Einheit: Entfernung pro Zeiteinheit angegeben wird, wird die Geschwindigkeit der linearen Bewegung am Anfangspunkt des Werkzeugkoordinatensystems konfiguriert. Bei der Bewegung in axialer Interpolation wird die Geschwindigkeit in Prozent eingestellt. (Sogar wenn die Geschwindigkeit als absoluter Wert oder als Bewegungszeit eingestellt wurde, wird sich der Roboter mit einer so eingestellten Geschwindigkeit bewegen. Die Geschwindigkeit wird hingegen als Prozentzahl der maximalen Geschwindigkeit, die der jeweilige Wert ist, verarbeitet.)
Die absolute Geschwindigkeit wird in mm/s angegeben. Die Reduktion der Überwachungsgeschwindigkeit verursacht eine verhältnismäßige Reduktion dieser Geschwindigkeiten.

WARNHINWEIS

Sogar wenn das Produkt der Programmgeschwindigkeit und der mithilfe des Kommandos SPEED eingestellten Geschwindigkeit (Programmgeschwindigkeit) 100 % überschreitet, überschreitet die tatsächliche Bewegungsgeschwindigkeit keine 100 %.

Beispiel
Wenn die Geschwindigkeit eingestellt ist wie unten angegeben und die Überwachungsgeschwindigkeit 100 % beträgt:
SPEED 50
stellt die Geschwindigkeit der nächsten Bewegung auf 50 % der maximalen Geschwindigkeit ein.
SPEED 100
stellt die Geschwindigkeit der nächsten Bewegung auf 100 % der maximalen Geschwindigkeit ein.
SPEED 200
stellt die Geschwindigkeit der nächsten Bewegung auf 100 % der maximalen Geschwindigkeit ein (Geschwindigkeit, die 100 % überschreitet, wird als 100 % abgelesen).
SPEED 20 MM/S
Die Geschwindigkeit des Anfangspunktes des Werkzeugkoordinatensystems (TCP) ist auf 20 mm/Sek. eingestellt (wenn die Überwachungsgeschwindigkeit 100 % beträgt).
SPEED 100 MM/S ALWAYS
Die Geschwindigkeit des Anfangspunktes des Werkzeugkoordinatensystems (TCP) ist auf 100 mm/Sek. eingestellt (wenn die Überwachungsgeschwindigkeit 100 % beträgt), jede Bewegung nach diesem Kommando wird mit einer Geschwindigkeit von 100 MM/S oder mit entsprechender Geschwindigkeit für die Punkt-zu-Punkt-Bewegungen ausgeführt.

ACCEL *Beschleunigung*
ACCEL *Beschleunigung* ALWAYS
DECEL *Bremsen*
DECEL *Bremsen* ALWAYS

Funktion
stellt die Beschleunigung (oder das Bremsen) der Roboterbewegung ein.
Parameter
Beschleunigung / Bremsen
stellt die Beschleunigung oder das Bremsen der Roboterbewegung in Prozent der maximalen Beschleunigung (des Bremsens) ein. Der zulässige Wertbereich beträgt von 0,01 bis 100. Werte, die über diesen Bereich hinausgehen, werden als 100 gehandhabt und Werte unterhalb des Bereiches werden als 0,01 behandelt.
ALWAYS
Die/Das vorgegebene Beschleunigung/Bremsen wird jedes weitere Bewegungskommando, bis zum weiteren ACCEL/DECEL-Befehl, betreffen.
Zusatzinformationen
Der ACCEL-Befehl stellt die Beschleunigung, wenn die Roboterbewegung beginnt, als einen Prozentsatz der maximalen Beschleunigung ein. Der DECEL-Befehl stellt das Bremsen, wenn die Roboterbewegung endet, als einen Prozentsatz des maximalen Bremsens ein.
Beispiel
ACCEL 80 ALWAYS
Die Beschleunigung ist auf 80 % für alle Bewegungen, die diesem Kommando folgen, eingestellt.
DECEL 50
Das Bremsen für die nächsten Bewegungskommandos wird auf 50 % eingestellt.

11.4 KOMMANDOS ZUR STEUERUNG DES PROGRAMMS

TWAIT	hält die Ausführung des Programms bis zum Ablauf der festgelegten Zeit an
CALL	ruft das Subprogramm (die Subroutine) auf
RETURN	verlässt das Subprogramm (die Subroutine)

TWAIT Zeit

Funktion

hält die Ausführung des Programms bis zum Ablauf der festgelegten Zeit an.

Parameter

Zeit
bestimmt die Zeit in Sekunden, wie lange das Programm angehalten wird.

Zusatzinformationen

Dieses Kommando hält die Ausführung des Programms bis zum Ablauf der festgelegten Zeit an.

Beispiel

TWAIT 0.5
0,5 Sekunden Warten.
TWAIT Delta
Warten, bis die Zeit, durch die Variable "Delta" bestimmt, abläuft.

CALL *Programmbezeichnung*

Funktion

Hält die Ausführung des laufenden Programms an und springt zu einem neuen Programm (Subprogramm) über. Wenn die Ausführung des Subprogramms beendet ist, kehrt die Verarbeitung zum originalen Programm zurück und führt den Schritt nach dem CALL-Kommando aus.

Parameter

Programmbezeichnung

bestimmt den Namen der Subroutine, die aufgerufen werden soll

Erklärung

Dieses Kommando stellt vorläufig die Ausführung des laufenden Programms an und springt zum ersten Schritt der festgelegten Subprozedur über.

WARNHINWEIS

In einem astorino-Roboter ist es möglich, dass das Programm maximal bis zum 5. Grad installiert wird. Das Programm erlaubt nicht, die Elternfunktion aufzurufen. Es ist nicht möglich, das Programm rekurrent aufzurufen.

Beispiel:

CALL INIT

RETURN

Funktion

beendet die Ausführung des Subprogramms und kehrt zum Schritt nach dem Kommando CALL in dem Programm, welches das Subprogramm aufgerufen hat, zurück.

Erklärung

Dieses Kommando beendet die Ausführung des Subprogramms und kehrt zu dem Programm zurück, welches das Subprogramm aufgerufen hat. Wenn die Subprozedur nicht von einem anderen Programm aufgerufen wird (z. B. wenn die Subprozedur durch das Kommando EXECUTE ausgeführt wird), wird die Ausführung des Programms beendet.

Am Ende der Subprozedur kehrt die Ausführung des Programms zum originalen Programm zurück, sogar dann, wenn das RETURN-Kommando nicht vorhanden ist. Das RETURN-Kommando soll als letztes Kommando des Subprogramms (oder am beliebigem Platz, an welchem die Subprozedur beendet werden soll) geschrieben werden.

11.5 KOMMANDOS DER PROGRAMMSTRUKTUR

IF……THEN…ELSE……END

WHILE……DO……END

DO……UNTIL

FOR……END

CASE … OF …. VALUE … ANY …. END

IF logischer Ausdruck THEN

Programmkommandos (1)

ELSE

Programmkommandos (2)

END

Funktion

führt eine der Schrittgruppen des Programms, abhängig vom Wert des logischen Ausdrucks, aus.

Parameter

Logischer Ausdruck
Logischer Ausdruck oder ein Ausdruck mit reellen Werten. Dieser Ausdruck gibt den Wert True (Wert anders als 0) oder False (0) wieder.
Programmkommandos (1)
Ausgeführte Programmkommandos, wenn der logische Ausdruck den logischen Wert TRUE hat.
Programmkommandos (2)
Ausgeführte Programmkommandos, wenn der logische Ausdruck den logischen Wert FALSE hat.

Zusatzinformationen

Das in Rede stehende Kommando führt eine von zwei Gruppen der Kommandos, abhängig vom Wert des logischen Ausdrucks, aus. Im Folgenden wurde die Ausführungsprozedur dargestellt:
1. den Wert des logischen Ausdrucks berechnen und zum Schritt 4 übergehen, wenn dieser Ausdruck den Wert 0 (FALSE) hat.
2. den Wert des logischen Ausdrucks berechnen und zur Ausführung des Programmkommandos (1) übergehen, wenn dieser Ausdruck den Wert 1 (TRUE) hat.
3. zu 5 übergehen.
4. Wenn das Kommando ELSE vorhanden ist, Programmkommandos (2) ausführen.
5. Programmausführung ab dem Schritt fortsetzen, der hinterm Schlüsselwort END platziert ist.

WARNHINWEIS

1. Die Kommandos ELSE und END müssen einzige Schlüsselwörter in den Linien sein, in denen sie sich befinden.

2. Im Kommando IF...THEN muss das Schlüsselwort END am Ende stehen.

Beispiel

Im nachfolgenden Beispielprogramm, wenn n größer ist als 5, wird die Programmgeschwindigkeit auf 10 %, und wenn nicht, dann auf 20 % eingestellt.

```
IF n>5 THEN
        sp=10
ELSE
        sp=20
END
SPEED sp
```

Das im Folgenden dargestellte Programm prüft zuerst den Wert der Variablen "m". Wenn die Variable "m" anders ist als 0, prüft das Programm externes Eingangssignal 1001 und stellt die entsprechende Mitteilung, abhängig vom Status dieses Signals, dar. In diesem Beispiel besitzt die äußere IF-Struktur kein ELSE-Kommando.

```
IF m THEN
        IF SIG(1001) THEN
                PRINT 1.0
         ELSE
                 PRINT 0.0
         END
END
```

WHILE *Bedingung* DO

Programmkommandos

END

Funktion

Wenn die vorgegebene Bedingung den Wert TRUE hat, werden die Programmkommandos ausgeführt. Wenn die Bedingung den Wert FALSE hat, wird das WHILE-Kommando übergangen.

Parameter

Bedingung
Ein logischer Ausdruck oder ein Ausdruck mit reellen Werten. Sie prüft, welchen Wert der Ausdruck hat: TRUE (anders als 0), oder FALSE (0).

Programmkommandos
Eine Gruppe der Kommandos, die ausgeführt werden sollen, wenn die Bedingung den Wert TRUE hat.

Zusatzinformationen

Die in Rede stehende Struktur wiederholt bestimmte Programmschritte ständig, wenn die Bedingung den Wert TRUE hat. Im Folgenden wurde die Ausführungsprozedur dargestellt:
1. den Wert des logischen Ausdrucks berechnen und zum Schritt 4 übergehen, wenn dieser Ausdruck den Wert 0 (FALSE) hat.

2. den logischen Ausdruck berechnen und Programmkommandos ausführen, wenn dieser Ausdruck den Wert 1 (TRUE) hat.
3. zu 1 übergehen.
4. Ausführung des Programms mit dem Schritt, der hinterm Schlüsselwort END platziert ist, fortsetzen.

WARNHINWEIS

Abweichend vom Kommando DO, wenn der Ausdruck den Wert FALSE hat, werden die Schritte innerhalb der Konstruktion WHILE nicht ausgeführt. Bei der Nutzung dieses Kommandos kann die Bedingung von TRUE zu FALSE wechseln.

Beispiel

In dem unten platzierten Beispiel wird das Ausgangssignal 1001 überwacht und die Bewegung des Roboters wird abhängig von seinem Zustand angehalten. Wenn das vom Teilespender stammende Signal seinen Wert auf 0 wechselt (Spender leer), hält der Roboter an und die Ausführung wird vom Schritt nach dem END-Kommando (Schritt 27 in diesem Beispiel) fortgesetzt.
Wenn einer der Spender beim Ingangsetzen der WHILE-Struktur (externes Eingangssignal = 0) leer ist, wird kein Strukturschritt ausgeführt und die Verarbeitung wird vom Schritt nach dem Wort END fortgesetzt.

```
WHILE SIG(1001) DO
        LMOVE P1
        TWAIT 1
        LMOVE part
END
```

DO

Programmkommandos

UNTIL *logischer Ausdruck*

Funktion

erstellt die Iterationsschleife DO

Parameter

Programmkommandos
Diese Kommandos werden die ganze Zeit wiederholt, wenn der logische Ausdruck den Wert FALSE hat.

Logischer Ausdruck
Der logische Ausdruck oder ein Ausdruck mit reellen Werten. Wenn dieser logische Ausdruck den Wert zu TRUE wechselt, werden die Kommandos innerhalb der Schleife nicht mehr ausgeführt.

Zusatzinformationen

Die in Rede stehende Struktur führt eine Gruppe der Programmkommandos aus, wenn die vorgegebene Bedingung (logischer Ausdruck) den Wert FALSE hat.
Im Folgenden wurde die Ausführungsprozedur dargestellt:
1. Programmkommandos ausführen.
2. Wert des logischen Ausdrucks prüfen und wenn dies der Wert FALSE ist, den Schritt 1 wiederholen.
Wenn der logische Wert den Wert TRUE hat, zum Schritt 3 wechseln.
3. Die Ausführung des Programms vom Schritt, der nach dem Kommando UNTIL platziert ist, fortsetzen.
Die Ausführung der Struktur DO ist beendet, wenn der Wert des logischen Ausdrucks von FALSE zu TRUE wechselt.

WARNHINWEIS

Abweichend von der Konstruktion WHILE, werden die Kommandos der Konstruktion DO immer mindestens einmal ausgeführt. Die Programmkommandos, die sich zwischen den Wörtern DO und UNTIL befinden, kann man auslassen. Wenn keine Kommandos eingestellt sind, wird der Wert des logischen Ausdrucks nach dem Wort UNTIL ständig bestimmt. Zum Zeitpunkt, wenn der Wert dieses Ausdrucks zu TRUE wechselt, wird die Schleife verlassen und erfolgt der Übergang zu der Konstruktion DO.

Die Konstruktion DO muss immer mit dem Schlüsselwort UNTIL beendet werden.

Beispiel

Im nachfolgenden Beispiel realisiert die Konstruktion DO die folgende Aufgabe: ein Teil wird angehoben und in den Puffer übertragen. Nachdem der Puffer gefüllt ist, wird das binäre Eingangssignal "buffer" eingeschaltet. Die Einschaltung dieses Signals erzwingt das Anhalten des Roboters und den Übergang zur Umsetzung einer anderen Aufgabe.

```
DO
        JMOVE get
        JMOVE put
UNTIL (SIG(buffer))
```

FOR *Steuerungsvariable* = *Anfangswert* TO *Schlusswert* STEP *Schritt*

Programmkommandos

END

Funktion

Wiederholung der Ausführung von Programmkommandos.

Parameter

Steuerungsvariable
Die reelle Variable oder der reelle Wert. Diese Variable wird ganz am Anfang auf den Anfangswert eingestellt und demnächst wird sie um 1 nach jeder Ausführung einer Schleife gesteigert.
Anfangswert
ein Wert oder ein Ausdruck vom reellen Typ. Dieser Parameter dient zur Konfigurierung des Anfangswertes der Variablen, die die Schleife steuert.
Schlusswert
ein Wert oder ein Ausdruck vom reellen Typ. Dieser Wert wird mit dem aktuellen Wert der Steuerungsvariablen der Schleife verglichen und zum Zeitpunkt des Erreichens von diesem Wert erfolgt das Verlassen der Schleife.

SCHRITT

ein Schritt, um welchen die Steuerungsvariable verändert wird.

Zusatzinformationen

Die in Rede stehende Konstruktion wiederholt die Ausführung des Programmkommandos zwischen den Schlüsselwörtern FOR und END. Die Steuerungsvariable der Schleife wird um einen bestimmten Schrittwert nach jeder Ausführung der Schleife wiederholt.

Im Folgenden wurde die Ausführungsprozedur dargestellt:
1. Anfangswert der Steuerungsvariablen der Schleife zuordnen.
2. Schlusswert und Schrittwert berechnen.
3. Wert der Steuerungsvariablen der Schleife mit dem Endwert vergleichen.
a. wenn der Schrittwert positiv ist und die Steuerungsvariable der Schleife größer ist als der Schlusswert, zum Schritt 7 wechseln.
b. wenn der Schrittwert negativ ist und die Steuerungsvariable der Schleife kleiner ist als der Schlusswert, zum 7 wechseln.
In sonstigen Fällen zum Schritt 4 wechseln.
4. Die Programmkommandos nach dem Schlüsselwort FOR ausführen.
5. Nachdem das Kommando END erreicht wurde, den Schrittwert zur Steuerungsvariablen der Schleife hinzufügen.
6. Zum Schritt 3 zurückkehren.
7. Programmkommandos, die nach dem Kommando END platziert sind, ausführen.
(Wert der Steuerungsvariablen der Schleife zum Zeitpunkt des Vergleiches im Schritt 3 wird nicht geändert.)

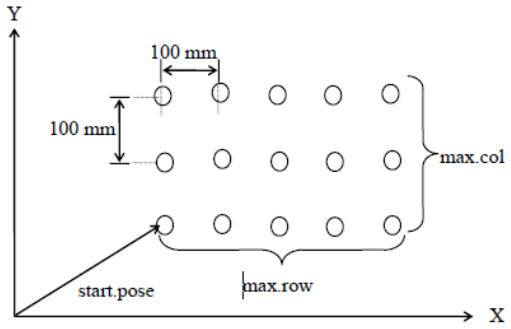
WARNHINWEIS

Jedes Schlüsselwort FOR muss das ihm entsprechende Schlüsselwort END haben. Beachten Sie, dass wenn die Steuerungsvariable der Schleife größer ist als der Schlusswert beim ersten Vergleich der Werte, wird keins von Programmkommandos zwischen den Schlüsselworten FOR und END ausgeführt.

Beispiel

Das Programm "pick" hebt einen Teil an und platziert ihn in der Position "hole". Die Teile werden wie auf der nachfolgenden Zeichnung platziert. (Die Palette ist parallel zur Achse X und Y des globalen Koordinatensystems platziert und die Entfernung zwischen den Teilen beträgt 100 mm).

```
FOR row = 1 TO maxrow
        POINT hole = SHIFT (startpose BY (row-1) * 100, 0, 0)
        FOR col = 1 TO maxcol
                JMOVE pick
                POINT hole = SHIFT ( hole BY 0, 100, 0)
        END
END
```



CASE *Steuerungsvariable* OF
VALUE *Wert 1,…*:
Programmkommandos
VALUE *Wert 2, Wert 3,…*:
Programmkommandos
.

.

VALUE *Wert n, …*:

Programmkommandos

ANY :

Programmkommandos

END

Funktion

führt das Programm gemäß konkretem Wert der Steuerungsvariablen aus

Parameter

Steuerungsvariable
eine Variable oder ein reeller Wert. Ihr Wert entscheidet, welcher Fall VALUE in Gang gesetzt wird.
Erklärung

Diese Struktur erlaubt dem Programm aus mehreren Gruppen wählen und Kommandos ausführen. Es ist ein mächtiges Werkzeug der AS-Sprache, welches eine bequeme Methode bietet, die mehrere Alternativen im Programm zulässt.

Es gibt die folgende Ausführungsprozedur:

1. Wert der Steuerungsvariablen, die in das Kommando **CASE** eingepflegt ist, wird geprüft.

2. Schritte **VALUE** werden geprüft und der erste Schritt, der den gleichen Wert mit dem Wert der Steuerungsvariablen enthält, wird gefunden.

3. Kommandos werden Schritt für Schritt ausgeführt.

4. Nach der Ausführung des Programmkommandos wechselt zum Kommando **END**.

5. Der Parameter **ANY** kann ausgelassen werden.

Wenn kein Wert vorhanden ist, der zur Steuerungsvariablen passt, sind Kommandos ANY zu programmieren. Die Kommandos werden ausgeführt, falls keine Übereinstimmung vorliegt, die durch Kommandos **VALUE** aufgedeckt werden. Wenn kein Kommando **ANY** vorhanden ist und kein Wert aufgedeckt wurde, wird keiner von **CASE**-Schritten ausgeführt.

Beispiel

Das nachfolgende Programm wird nach diesen drei Fällen ausgeführt:

1. wenn der Wert eine gerade Zahl aus dem Bereich von 0 bis 10 ist,
2. wenn der Wert eine ungerade Zahl aus dem Bereich von 1 bis 9 ist,
3. wenn der Wert eine andere Zahl ist als die vorgenannten.

```
CASE x OF
VALUE 0,2,4,6,8,10:
PRINT "The number x is EVEN"
VALUE 1,3,5,7,9:
PRINT "The number x is ODD”
ANY :
PRINT "The number x is larger than 10"
END
```

11.6 KOMMANDOS FÜR BINÄRE SIGNALE

SWAIT lässt das Programm so lange hängen, bis die bestimmte Bedingung des Signalzustands eingestellt ist

SWAIT Signalnummer1,…, Signalnummer6

Funktion

wartet bis zu dem Zeitpunkt, wenn bestimmte externe Eingangssignale oder interne Signale den geforderten Zustand haben.

Parameter

Signalnummer1,..,Signalnummer6
Die Nummer des externen Eingangssignals oder des internen Signals, der überwacht werden soll. Es ist möglich, 6 Signale zugleich zu überwachen.
Eine negative Nummer bedeutet, dass die Bedingung erfüllt ist, wenn das Signal abgeschaltet ist.
Zulässige Signalnummern
Externe Eingangssignale 1001-1008
MODBUS-Signale 1009-1056
Signale am Arm der Achse JT3 1057-1058 (Version B des Roboters)
Interne Signale 2001−2016

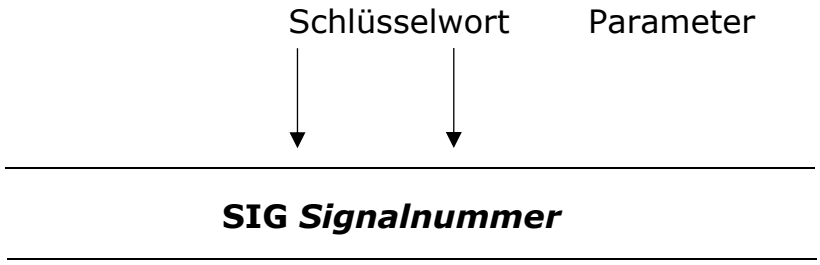
Beispiel

SWAIT 1001,2001	wartet bis das externe Eingangssignal 1001 und das interne Signal 2001 eingeschaltet sind.
SWAIT -2001	wartet bis zur Abschaltung des internen Signals 2001.

12 FUNKTIONEN

In diesem Kapitel werden Funktionen vorgestellt, die im AS-System angewandt werden. Diese Funktionen werden meistens in Verbindung mit Bildschirmbefehlen und mit Programmkommandos angewandt. Im Folgenden wurde ein Funktionsformat dargestellt. Das Schlüsselwort bestimmt die Funktion und die Parameter, die in Klammern eingepflegt werden, bestimmen Werte, an welchen die Funktion operiert.

Beispiel



Das Schlüsselwort und der Parameter müssen mit einem Leerschritt getrennt sein.

12.1 FUNKTIONEN, DIE AUF REELLEN WERTEN OPERIEREN

SIG führt den logischen Vorgang AND an einem bestimmten Signal aus

BITS gibt den Wert wieder, welcher der Bitformel des Signals (bis zu 16 Signalen) entspricht, wieder.

DISTANCE gibt den Abstand zwischen zwei Punkten wieder

DEXT gibt die bestimmte Komponente des Punktes wieder

DX,DY,DZ gibt den Wert des Verfahrens (X,Y,Z) wieder

TRUE gibt die logische Eins (1.0) wieder

FALSE gibt die logische Null (0.0) wieder

VAL gibt den reellen Wert von der String-Variablen (eine Zeichenfolge) wieder

INT gibt den ganzen Teil der vorgegebenen Zahl wieder

EXISTREAL gibt die Information wieder, ob die jeweilige Zahlenvariable existiert

EXISTJOINT gibt die Information wieder, ob die jeweilige Punkt-zu-Punkt-Variable existiert

EXISTTRANS gibt die Information wieder, ob die jeweilige Transformationsvariable existiert

EXISTCOM gibt die Information wieder, ob Daten zum Ablesen von der seriellen Datenübertragung (Serial) vorhanden sind

INRANGE gibt die Information wieder, ob die jeweilige Position in der Betriebsreichweite ist

ROUND gibt den abgerundeten Wert der Zahl wieder

TIMER stellt/gibt den aktuellen Wert des Timers ein/wieder.

WAIT hält die Ausführung des Programms bis zum Zeitpunkt der Erfüllung einer bestimmten Bedingung an.

SIG (Signalnummer)

Funktion

führt die logische Operation AND am vorgegebenen Signal aus.

Parameter

Signalnummer
Die Nummern der externen oder internen Eingangssignale.

Zusatzinformationen

Führt die logische Operation AND am vorgegebenen Signal aus und gibt den Ergebniswert wieder. Wenn das Signal den logischen Wert TRUE hat, wird der logische Wert 1 (True) wiedergegeben. Andernfalls wird der Wert 0 (FALSE) wiedergegeben. Die externen oder internen Eingangssignale werden mithilfe entsprechender Nummern, wie im Folgenden dargestellt wurde, bestimmt.

Zulässige Signalnummern	
Externe Eingangssignale	1001 – 1008
MODBUS-Signale	1009-1056
Signale am Arm der Achse JT3	1057-1058 (Version B des Roboters)
Interne Signale	2001−2016

Es wird angenommen, dass Signale, die mit positiven Zahlen bestimmt werden, den Wert TRUE (WAHRHEIT) annehmen, wenn diese eingeschaltet sind (ON), hingegen nehmen Signale, die mit negativen Zahlen bezeichnet werden, den Wert TRUE (WAHRHEIT) an, wenn sie abgeschaltet sind (OFF). Der Signalnummer "0" entspricht kein Signal, es wird also angenommen, dass er immer den Wert TRUE hat.

Beispiel

Wenn das binäre Eingangssignal 1001=ON, 1004=ON, 2004=OFF, dann:
SIG(1001) == TRUE
SIG(2004) == FALSE
SIG(−1004) == FALSE

BITS (*Nummer des Startsignals, Anzahl der Signale*)

Funktion

liest weitere binäre Signale ab und gibt den Dezimalwert, der den Bitformeln bestimmter binärer Signale entspricht, wieder.

Parameter

Nummer des Startsignals
bestimmt das erste Signal zum Ablesen.
Anzahl der Signale
bestimmt die Anzahl der Signale zum Ablesen. Die maximale akzeptable Zahl ist 16.

Erklärung

Diese Funktion gibt den Dezimalwert, welcher der Bitformel bestimmter Signale entspricht, wieder.
In dem binären Ausdruck des durch diese Funktion wiedergegebenen Wertes entspricht das unbedeutendste Bit der Anfangszahl des Signals.

Zulässige Signalnummern	
Externes Ausgangssignal	1 – reelle Anzahl der Signale
internes Eingangssignal	1001 - reelle Anzahl der Signale
internes Signal	2001-2016

Beispiel
Sind die Zustände der Signale wie folgt, ist 5 das Ergebnis des nachfolgenden Ausdrucks.
x = BITS(1003,4)

Die logischen Werte der 4 Signale, die mit 1003 beginnen (d. h. 1003, 1004, 1005 und 1006) werden als
Bitformel 0101 oder 5 in der Dezimalnotation abgelesen.

Signale:	1008	1007	**1006**	**1005**	**1004**	**1003**	1002	1001
Zustand:	1	1	**0**	**1**	**0**	**1**	0	0

DISTANCE (*Variable des Transformationswertes 1, Variable des Transformationswertes 2*)

Funktion
berechnet den Abstand zwischen zwei Positionen, die in Transformationswerten ausgedrückt sind.
Parameter
Variable des Transformationswertes 1, Variable des Transformationswertes 2
bestimmt Bezeichnungen zweier Variablen des Transformationswertes, für welche der Abstand zwischen ihnen berechnet werden soll.
Erklärung
Sie gibt den Abstand zwischen zwei Positionen in Millimetern wieder. (Beide Positionen kann man in beliebiger Reihenfolge einpflegen)

Beispiel

k=DISTANCE(HERE,part)
berechnet den Abstand zwischen dem aktuellen TCP und der Position "part" und wandelt das Ergebnis zu „k" um.

DEXT (*Positionsvariable, Nummer des Elementes*)

Funktion
gibt ein bestimmtes Element einer bestimmten Position wieder.

Parameter
Positionsvariable
bestimmt den Namen der Positionsvariablen, die durch Transformationswerte oder durch Punkt-zu-Punkt-Werte definiert ist, wieder.
Nummer des Elementes
bestimmt das Element, welches in reellen Zahlen wiedergegeben werden soll, wie auf der Zeichnung unten dargestellt.

Nummer des Elementes	Position	
	Transformationswerte	Punkt-zu-Punkt-Werte
1	X	JT1
2	Y	JT2
3	Z	JT3
4	O	JT4
5	A	JT5
6	T	JT6
7	JT7	JT7

Beispiel
Wenn die Transformationswerte für "aa" (0, 0, 0, -160, 0, 0, 300) betragen, so ist die Anwendung dieser Funktion:

Typ DEXT(aa, 7)

schreibt im 300-Terminal den Wert JT7 dieses Punktes.

DX (*Variable des Transformationswertes*)

DY (*Variable des Transformationswertes*)

DZ (*Variable des Transformationswertes*)

Funktion
gibt die Transformationswerte (X, Y, Z) der definierten Position durch die bestimmte Positionsvariable wieder.

Parameter
Variable des Transformationswertes
bestimmt die Bezeichnung der Variablen des Transformationswertes, dessen Bestandteil X, Y oder Z erforderlich ist.

Erklärung
Jede von diesen drei Funktionen gibt den Bestandteil X, Y oder Z der bestimmten Position wieder.

Jeden Bestandteil des Transformationswertes kann man ebenfalls mithilfe des Kommandos DECOMPOSE erreichen. Die Werte O, A und T werden mithilfe des Kommandos DECOMPOSE erreicht.

Beispiel
Wenn die Position "Start" die Transformationswerte hat:

X	Y	Z	O	A	T
125.00	250.00	-50.0	135.00	50.00	75.00

dann:
x=DX(Start) gibt die Funktion DX **x = 125.00** wieder
y=DY(start) gibt die Funktion DY **y = 250.00** wieder
z=DZ(start) gibt die Funktion **z = -50.00** wieder

TRUE

Funktion
gibt die logische Eins (1.0) wieder

Erklärung
Diese Funktion ist bequem, wenn es erforderlich ist, die logische Bedingung TRUE zu bestimmen.

FALSE

Funktion
gibt die logische Null (0.0) wieder

Erklärung
Diese Funktion ist bequem, wenn es erforderlich ist, die logische Bedingung FALSE zu bestimmen.

VAL (*$Variable einer Zeichenfolge*)

Funktion
gibt den reellen Wert in einer bestimmten Zeichenfolge wieder.

Parameter
Zeichenfolge
bestimmt eine Zeichenfolge, eine Variable der Zeichenfolge oder einen Folgenausdruck.
Beispiel
VAL("123") gibt den reellen Wert 123 wieder.

INT (*Zahlenausdruck*)

Funktion
gibt die ganze Zahl eines bestimmten Zahlenausdrucks wieder.

Parameter
Zahlenausdruck

Erklärung
gibt die ganze Zahl (d. h. die linke Seite des Dezimalkommas) wieder.
Das negative Zeichen bleibt mit der ganzen Zahl, es sei denn, die ganze Zahl ist gleich 0.

Beispiel

INT(0.123)	der Wert 0 wird wiedergegeben.
INT(10.8)	der Wert 10 wird wiedergegeben.
INT(-5.462)	der Wert -5 wird wiedergegeben.

EXISTREAL (*reelle Variable*)

Funktion
prüft, ob die bestimmte Variable existiert.

Parameter
Reelle Variable
bestimmt die reelle Variable in Form einer Zeichenfolge.

Erklärung
Wenn die Bezeichnung der Variablen existiert, gibt sie den Wert 1 wieder. Wenn sie nicht existiert, gibt sie den Wert 0 wieder.

Beispiel

ret=EXISTREAL(pp)	Wenn die reelle Variable pp existiert, ret=-1. Wenn nicht, ret=0.

EXISTJOINT (*Bezeichnung der Variablen eines Punkt-zu-Punkt-Wertes*)

Funktion
prüft, ob eine bestimmte Punkt-zu-Punkt-Variable existiert.

Parameter
Bezeichnung der Variablen des Punkt-zu-Punkt-Wertes
bestimmt die Bezeichnung der Variablen vom Wert des Punkt-zu-Punkt-Verfahrens in Form einer Zeichenfolge. Die Bezeichnung ist mit dem Zeichen # zu beginnen.

Erklärung
Wenn die Variable existiert, gibt den Wert 1 wieder. Wenn sie nicht existiert, gibt sie den Wert 0 wieder.

Beispiel

ret=EXISTJOINT("#pos")	Wenn die Variable #pos existiert, ret= 1. Wenn nicht, ret=0.

EXISTTRANS (*Variable des Transformationswertes*)

Funktion
prüft, ob eine bestimmte Variable durch Transformationswerte definiert ist.

Parameter
Variable des Transformationswertes
bestimmt die Bezeichnung der Variablen vom Transformationswert in Form einer Zeichenfolge.
Erklärung
Wenn die Variable existiert, gibt sie den Wert 1 wieder. Wenn sie nicht existiert, gibt sie den Wert 0 wieder.
Beispiel

ret=EXISTTRANS(pos1)	Wenn die Variable vom Transformationswert pos1 durch die Transformation definiert ist, dann existieren die Werte, ret=1. Wenn nicht, ret=0.

EXISTCOM

Funktion
prüft, ob die Eingangsdaten im Puffer der seriellen Datenübertragung existieren.

Erklärung
Wenn die Eingangsdaten im Puffer der seriellen Datenübertragung existieren, dann wird der Wert TRUE (1.0) wiedergegeben, wenn die Daten nicht existieren, wird der Wert FALSE (0.0) wiedergegeben. Diese Funktion ist nützlich, wenn der Roboter auf zu empfangenden Daten vom externen Gerät warten soll.

Beispiel
```
WHILE (EXISTCOM == FALSE)
        TWAIT 0.1
END
```

Das vorstehende Beispiel bringt das Programm zu einer Schleife, bis die Daten in den Puffer der seriellen Datenübertragung eingehen.

INRANGE (*Positionsvariable*)

Funktion
prüft, ob sich die Position im Bereich der Roboterbewegung befindet und gibt den vom Ergebnis (1.0 lub 0.0) abhängigen Wert wieder.

Parameter
Positionsvariable
bestimmt, welche Position geprüft werden soll.

ROUND (*Zahlenwert*)

Funktion
gibt den auf die erste Dezimalstelle abgerundeten Wert wieder.

Parameter
Zahlenwert
Dieser Wert wird mit einer Genauigkeit auf die erste Stelle nach dem Komma abgerundet.

Erklärung
gibt den auf die erste Stelle nach dem Komma abgerundeten Wert, bestimmt als Parameter, wieder. Wenn der bestimmte Wert ein negativer Wert ist, wird der Wert als Absolutwert abgerundet und demnächst wird negatives Zeichen hinzugefügt. Das Zeichen des Zahlenwertes, bestimmt als Parameter, bleibt unverändert, es sei denn, das Ergebnis ist 0.

Beispiel

ROUND (0.123)	gibt den Wert 0 wieder.
ROUND (10.8)	gibt den Wert 11 wieder.
ROUND (-5.462)	gibt den Wert -5 wieder.
ROUND (-5.662)	gibt den Wert -6 wieder.

TIMER(*Zahlenausdruck*)

TIMER Zahlenausdruck = Wert

Funktion

stellt/gibt den Wert des bestimmten Timers in Sekunden ein/wieder.
Parameter

Zahlenwert
bestimmt, welcher Timer abgelesen/gespeichert werden soll. Zulässige Zahlen sind von 0 bis 10.
Wert
Zeit in Sekunden, beim Timer gespeichert.
Erklärung

Mithilfe der Funktion TIMER kann der Wert des Timers zu einem beliebigen Zeitpunkt abgelesen werden. Sie liest die Zeit (in Sekunden) ab und gibt sie wieder, die ab dem Wert abgelaufen ist, der früher durch das Kommando TIMER eingestellt wurde. Wenn das Kommando TIMER keinen Wert eingestellt hat, gibt der Wert TIMER die Zeit, die ab Einschaltung der Einspeisung abgelaufen ist, an.

Bei Anwendung ohne Klammern () wird der Timer sofort auf eine bestimmte Zeit eingestellt.

Beispiel

TIMER 1 = 0	stellt den Wert Timer 1 auf 0 ein.
LMOVE P1	führt die Bewegung aus.
PRINT TIMER(1)	zeigt die abgelaufene Zeit an.

WAIT *numerischer Wert*

WAIT Bedingung

WAIT Funktion

Funktion

wartet, bis die Bedingung den Wert true wiedergibt.
Parameter

Numerischer Wert
bestimmt die Zeit in Sekunden,
Bedingung
bestimmt die Bedingung, auf welche die Funktion warten soll.
Funktion
bestimmt die Funktion, bei welcher das Kommando WAIT die Wiedergabe des Wertes true erwarten soll
Zusatzinformationen

Diese Funktion stellt die Ausführung des Programms bis zu dem Zeitpunkt ein, wenn die bestimmte Bedingung erfüllt ist. Diese Funktion kann an Zahlenwerten, -bedingungen oder -funktionen operieren.

Beispiel

WAIT 2	hält den Betrieb für 2 Sek. an
WAIT SIG(1001)	hält die Ausführung an, bis der Eingang 1 den Wert HIGH erreicht
WAIT TIMER(3) >2	hält die Ausführung an, bis der TIMER 3 zwei Sek. überschreitet

12.2 MATHEMATISCHE FUNKTIONEN

ABS	gibt den Absolutwert eines numerischen Ausdrucks wieder
SIN	gibt den Wert Sinus des numerischen Ausdrucks wieder
COS	gibt den Wert Cosinus des numerischen Ausdrucks wieder
ATAN2	gibt den Wert Arkustangens des numerischen Ausdrucks wieder
PI	gibt den Wert der PI-Zahl wieder
SQRT	gibt die Quadratwurzel des numerischen Ausdrucks wieder
RANDOM	gibt einen Zufallswert von 0.0 bis 1.0 an

ABS (*reeller Wert*)

Funktion

gibt den Absolutwert des numerischen Ausdrucks wieder.

Beispiel

X = ABS(Y)

SIN (*reeller Wert*)

Funktion

gibt den Sinus-Wert des numerischen Ausdrucks wieder

Beispiel

X = SIN(Y)

COS (*reeller Wert*)

Funktion

gibt den Cosinus-Wert des numerischen Ausdrucks wieder

Beispiel

X = COS(Y)

ATAN2 (*reeller Wert1, reeller Wert2*)

Funktion

gibt den Arkustangens-Wert des numerischen Ausdrucks wieder

Erklärung

$$\mathrm{ATAN2} = \begin{cases} \tan^{-1}(dy/dx) & dx > 0,\ dy > 0 \\ \tan^{-1}(dy/dx)+\pi & dx > 0,\ dy < 0 \\ \tan^{-1}(dy/dx)-\pi & dx < 0,\ dy < 0 \\ +\pi/2 & dx > 0,\ dy = 0 \\ -\pi/2 & dx < 0,\ dy = 0 \\ \text{undefined} & dx = 0,\ dy = 0 \end{cases}$$

Beispiel

X = ATAN2(Wert Y, Wert X)

PI

Funktion

gibt den Wert der Pi-Zahl wieder

Beispiel

D = 2*PI*R

SQRT(*reeller Wert*)

Funktion

gibt die Quadratwurzel des numerischen Ausdrucks wieder

Beispiel

X = SQRT(Y)

RANDOM

Funktion

gibt einen Pseudo-Zufallswert von 0.0 bis 1.0 wieder

Beispiel

X = RANDOM * 10

12.3 FUNKTIONEN DER ZEICHENFOLGE (STRING)

$DECODE sondert Zeichen aus, die mit bestimmten Zeichen getrennt sind.

$ENCODE gibt eine Zeichenfolge, gebildet aus reellem Wert, an.

$DECODE ($*Variable einer Zeichenfolge, $Trennzeichen*)

Funktion

sondert Zeichen aus, die mit bestimmten Zeichen getrennt sind

Parameter

Variable einer Zeichenfolge

bestimmt eine Zeichenfolge (String), aus welcher die Zeichen bezogen werden. Die infolge dieser Funktion ausgesonderten Zeichen werden aus dieser Folge entfernt.

Trennzeichen

bestimmt das Zeichen, welches als Trennzeichen abgelesen werden soll. (als Trennzeichen kann man ein beliebiges Zeichen von der ASCII-Tabelle bestimmen).

Alle Zeichen, die mit dem ersten Zeichen der String-Variablen bis zum Trennzeichen beginnen, werden wiedergegeben. Die wiedergegebene Zeichenfolge wird aus der String-Variablen entfernt. Das Trennzeichen wird aus der Variablen der Zeichenfolge entfernt.

Erklärung

Diese Funktion sucht ein Trennzeichen in einer bestimmten Zeichenfolge aus und sondert die Zeichen vom Beginn der Folge bis zum Trennzeichen aus. Die ausgesonderten Zeichen werden durch die Wirkung der Funktion wiedergegeben und gleichzeitig werden sie aus der originalen Folge entfernt.

Beispiel

Das nachfolgende Beispiel zerschlägt die Variable der Zeichenfolge $TEMP in drei getrennte Variablen ***$VAL1***, ***$VAL2***, ***$VAL3*** so, dass ***$VAL1 =”123”***, ***$VAL2 = „456”***, ***$VAL3 = „789”.*** Demnächst wandelt die Funktion ***VAL*** diese Variablen der Zeichenfolge in den Typ der reellen Werte ***DATAX***, ***DATAY***, ***DATAZ*** um. Die Daten der reellen Werte werden demnächst zur Erstellung des Punktes **TEST** verwendet.

```
$TEMP = “123/456/789/”

$COMMAND = $DECODE($TEMP,"/")

$VAL1 = $DECODE($TEMP,"/")
$VAL2 = $DECODE($TEMP,"/")
$VAL3 = $DECODE($TEMP,"/")
DATAX = VAL($VAL1)
DATAY = VAL($VAL2)
DATAZ = VAL($VAL3)
POINT TEST = TRANS(DATAX,DATAY,DATAZ,180,0,90)
LMOVE TEST
```

$ENCODE (*reeller Wert*)

Funktion

gibt eine auf Grundlage des reellen Wertes gebildete Folge wieder. Die Folge wird genauso gebildet wie im Falle der Anwendung des Kommandos **TYPE/PRINT**

Parameter

Reeller Wert

Reeller Wert (der Wert wird berechnet und angezeigt)

Erklärung

Diese Funktion ermöglicht die Bildung von Zeichenfolgen in Programmen bei Anwendung des reellen Wertes.

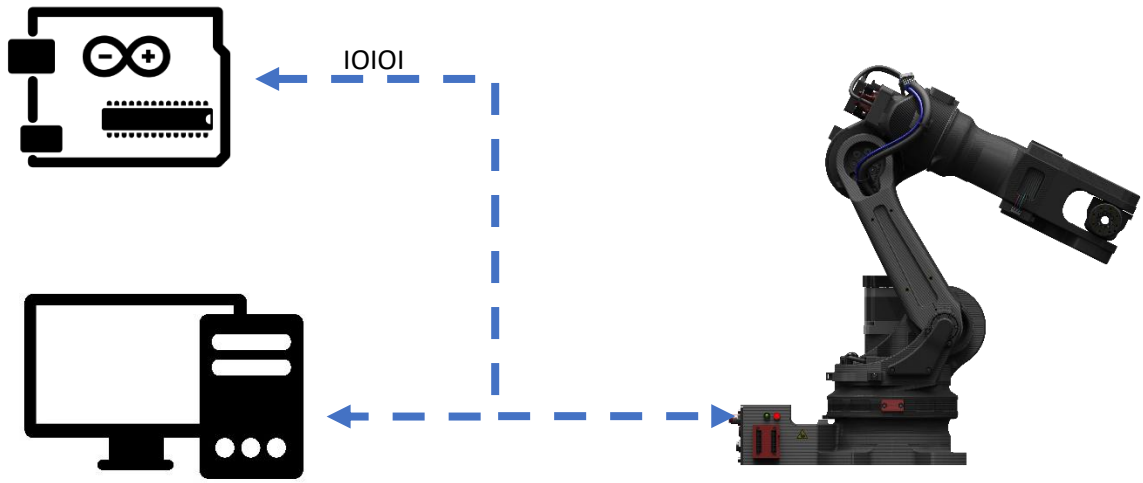
Beispiel

$output = $output + $ENCODE(count)

Der Wert der reellen Variablen "count" wird in eine Zeichenfolge konvertiert und am Ende von "$output" hinzugefügt. Demnächst wird eine verbundene Zeichenfolge wieder in der Variablen "$output" ersetzt.

12.4 SERIELLE DATENÜBERTRAGUNG

Die serielle Datenübertragung dient zur Übersendung von Daten zwischen dem Roboter und dem externen Zubehör wie Mikrokontroller (z. B. Arduino, ESP32) peripherem Zubehör (z. B. OpenMV, Sensoren) oder PC-Computer oder z. B. Raspberry PI.



⚠ WARNHINWEIS

Die serielle Datenübertragung funktioniert mit der Spannung 3,3V, bitte Elektronik anwenden, die mit der Spannung 3,3V kompatibel ist oder Spannungswandler einsetzen.

Die Spannung 5V kann das Hauptsystem CPU beschädigen!

[**VORSICHT**]

Die Parameter der seriellen Datenübertragung sind:

- Baudrate: 115200
- Data size: 8
- Parity: None
- Handshake : OFF

Die serielle Datenübertragung kann auch zum Austausch der Daten zwischen dem Zubehör, der andere Typen der seriellen Datenübertragung verwendet wie RS232 und RS485 oder zwischen dem astorino-Roboter und dem PC, genutzt werden. Die nachfolgende Tabelle stellt das erforderliche, zusätzliche Zubehör zur Umsetzung des Datenaustausches dar.

RS232	**Wandler UART (TTL) 3,3V -> RS232**
RS485	**Wandler UART (TTL) 3,3V -> RS485**
PC	**Wandler UART (TTL) 3,3V -> USB**

12.5 BEDIENFUNKTIONEN DER SERIELLEN DATENÜBERTRAGUNG

SEND sendet die Daten durch die serielle Datenübertragung (Serial/UART)

RECEIVE empfängt die Daten, die von der seriellen Datenübertragung (Serial/UART) erhalten wurden

EXISTCOM gibt die Information wieder, ob Daten zur Ablesung von der seriellen Datenübertragung (Serial/UART) vorhanden sind

SEND $*Variable einer Zeichenfolge*

Funktion
sendet die Daten durch die serielle Datenübertragung.

Erklärung
Wenn die Funktion aufgerufen wird, werden die Daten durch die serielle Datenübertragung (Serial/UART) geschickt

Beispiel
$dane = "Hello"
SEND $dane

[**VORSICHT**]

Die Parameter der seriellen Datenübertragung sind:

- Baudrate: 115200
- Data size: 8
- Parity: None
- Handshake : OFF

RECEIVE

Funktion
empfängt Daten, die von der seriellen Datenübertragung (Serial/UART) erhalten wurden.

Erklärung
Wenn die Funktion aufgerufen wird, werden verfügbare Daten vom Puffer der seriellen Datenübertragung bezogen. Wenn im Datenpuffer keine Daten verfügbar sind, wird ein **Timeout** von **5 Sek.** eingeschaltet, wenn in diesen **5 Sek.** nach Aufruf der Funktion keine Daten im Puffer erscheinen, gibt die Funktion einen Fehler wieder und das Programm wird angehalten.

Beispiel
$dane = RECEIVE

[**VORSICHT**]

Die Parameter der seriellen Datenübertragung sind:

- Baudrate: 115200
- Data size: 8
- Parity: None
- Handshake : OFF

EXISTCOM

Funktion
prüft, ob Eingangsdaten im Puffer der seriellen Datenübertragung vorhanden sind.

Erklärung
Wenn die Eingangsdaten im Puffer der seriellen Datenübertragung vorhanden sind, so wird der Wert TRUE (1.0) wiedergegeben, wenn die Daten nicht vorhanden sind, wird der Wert FALSE (0.0) wiedergegeben. Die Funktion ist nützlich, wenn der Roboter auf Daten warten soll, die von einem externen Gerät eingehen sollen.

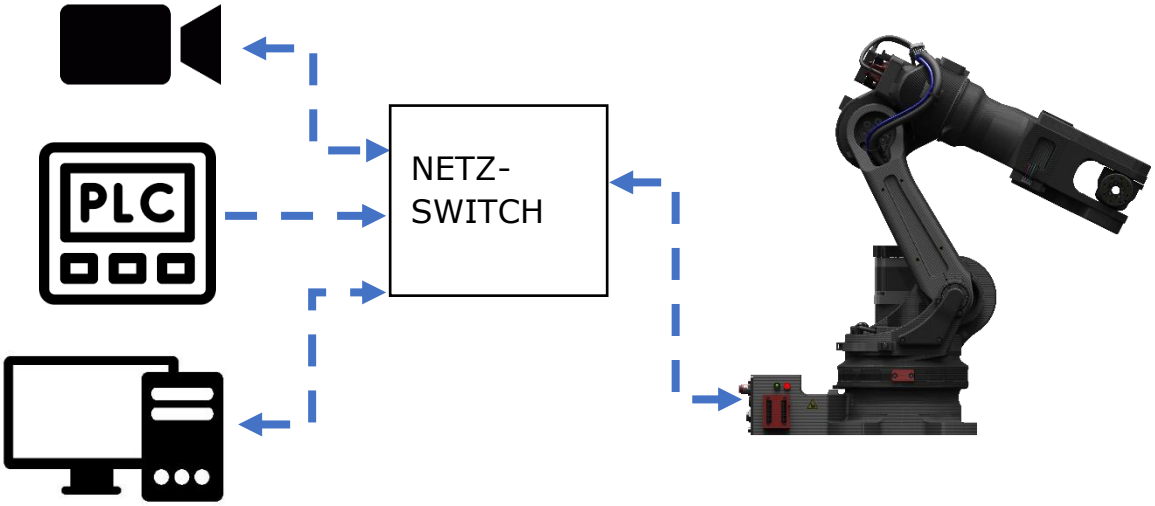
Beispiel

```
WHILE (EXISTCOM == FALSE)
        TWAIT 0.1
END
```

Das vorstehende Beispiel bringt das Programm zu einer Schleife, bis die Daten in den Puffer der seriellen Datenübertragung eingehen.

12.6 TCP/IP- UND UDP-KOMMUNIKATION

Die Kommunikation TCP/IP und UDP dient zur Übersendung der Daten zwischen dem Roboter und dem externen Zubehör wie SPS-Steuerungen, Bedienfronten, industrielles Zubehör wie Sensoren, Visionssysteme und PC



Der astorino-Roboter kann als Server und auch als Client in der Kommunikation funktionieren.

12.7 BEDIENFUNKTIONEN DER TCP/IP-KOMMUNIKATION

TCP_ACCEPT	prüft, ob die Forderung nach einer Verbindung empfangen wurde
TCP_CLOSE	unterbricht die Kommunikation
TCP_CONNECT	bildet ein Socket und sendet die Forderung nach einer Verbindung
TCP_LISTEN	bildet ein Socket und wartet auf die Forderung einer Verbindung
TCP_END_LISTEN	endet das Warten auf die Forderung einer Verbindung
TCP_SEND	sendet eine Datenfolge
TCP_RECEIVE	empfängt eine Datenfolge

TCP_ACCEPT *wiedergegebene Variable, Port-Nummer*

Funktion

Ein Programmkommando, das zur Prüfung der Forderung nach einer Verbindung dient (verwendet durch den Server zum Ingangsetzen des Kommunikations-dienstes).

Sie prüft, ob die Forderung nach einer Verbindung für die Kommunikation mit dem Roboter durch einen bestimmten Port empfangen wurde und wenn ja, legt sie die Verbindung fest. Diese Verbindung wird eingestellt, wenn das Kommando normal endet. Wenn bei der Ausführung ein Kommunikationsfehler auftritt, wird der Fehlercode (-1.0) wiedergegeben.

Es besteht die Möglichkeit, an den astorino- Roboter maximal acht (8) Clients gleichzeitig anzuschließen.

Parameter

Wiedergegebene Variable

stellt die Variable ein, welche die Ergebnisse der Ausführung eines Verbindungsversuches aufbewahrt. Sie bewahrt socketID (0-7) des verbundenen Clients auf. Wenn der Verbindungsversuch misslingt, wird der Wert (-1.0) wiedergegeben.

Der Wert -1.0 wird dann wiedergegeben, wenn ein Kommunikationsfehler auftritt. Die Ausführung des Programms hält jedoch nicht an dem Kommunikationsfehler an.

Portnummer

bestimmt die Portnummer, welche das Kanal vom anderen Ende der Verbindung anzeigt. Der zulässige Bereich ist 8192 – 65535, sonst tritt ein Fehler auf.

Beispiel:

```
WHILE socketID < 0 DO
        TCP_ACCEPT socketID, port
        TWAIT 0.2
END
```

Das vorstehende Beispiel bringt das Programm zu einer Schleife, bis der Client an den Roboter anschließt.

TCP_CLOSE *wiedergegebene Variable, Port-Nummer*

Funktion

Das Programmkommando zur Unterbrechung der Verbindung (dient zur Beendigung des Kommunikationsdienstes), schließt die Verbindung für die TCP/IP-Kommunikation und schließt den Port. Wenn ein Kommunikationsfehler auftritt, wird der Fehlercode (-1.0) wiedergegeben und die Ausführung des Programm wird nicht angehalten.

Parameter

Wiedergegebene Variable

stellt die Variable, die die Ausführungsergebnisse aufbewahrt, ein.

0 wird aufbewahrt, wenn die Ausführung normal erfolgt,

-1, wenn ein Fehler bei der Ausführung der Funktion aufgetreten ist.

Port-Nummer

spezifiziert das socketID, welches durch die Wirkung der Funktion **TCP_ACCEPT** oder **TCP_CONNECT** erlangt wurde.

Beispiel

TCP_CLOSE .ret, socketID

TCP_CONNECT *wiedergegebene Variable, Port-Nummer, IP-Adresse*

Funktion

Ein Programmkommando, welches zur Forderung nach einer Verbindung dient (wird durch den Client zum Ingangsetzen des Kommunikationsdienstes mit dem Server genutzt). Es bildet ein Socket und verbindet sich mit einer bestimmten Port-Nummer. Demnächst wird die Forderung nach einer Verbindung an einen bestimmten Knoten verschickt und die Verbindung wird geknüpft. Der Knoten wird durch die ID-Adresse des Servers bestimmt. Wenn während der Ausführung ein Kommunikationsfehler auftritt, wird der Wert -1.0 wiedergegeben und die Ausführung des Programms wird nicht angehalten.

Parameter

Wiedergegebene Variable

bestimmt eine Variable, welche die Ausführungsergebnisse aufbewahrt. Der Wert 8 wird wiedergegeben, wenn die Ausführung normal erfolgt.

Der Wert -1 wird wiedergegeben, wenn ein Fehler bei einem Verbindungsversuch auftritt.

Port-Nummer

bestimmt die Port-Nummer, welche aufs Kanal vom anderen Ende der Verbindung hinweist. Das Socket ist mit dieser Port-Nummer verbunden.

Der zulässige Bereich ist 8192 – 65535, sonst tritt ein Fehler auf.

IP-Adresse

bestimmt die Array-Variable, welche die IP-Adresse des Servers (32 Bits) aufbewahrt.

Die IP-Adresse wird in 8-Bit-Inkrementen zu jedem Element der Array-Variablen in der Reihenfolge ab Anfang der IP-Adresse aufbewahrt.

Beispiel:

IP[1] = 192

IP[2] = 168

IP[3] = 0

IP[4] = 100

WHILE socketID < 0 DO

TCP_CONNECT socketID,port,IP[1]

TWAIT 0.2

END

Das vorstehende Beispiel bringt das Programm zu einer Schleife, bis sich der Roboter an den Server angeschlossen hat.

TCP_LISTEN *wiedergegebene Variable, Port-Nummer*

Funktion

Ein Programmkommando, welches dazu dient, das Warten auf die Forderung nach der Verbindung zu beginnen (verwendet durch den Server zum Ingangsetzen des Kommunikationsdienstes). Es bildet ein Socket und verbindet sie mit einer bestimmten Port-Nummer und wartet auf die Forderung, mit diesem Socket verbunden zu werden. Wenn bei der Ausführung ein Kommunikationsfehler auftritt, wird der Wert -1.0 wiedergegeben und die Ausführung des Programms wird nicht angehalten.

Parameter

Wiedergegebene Variable

bestimmt eine Variable, welche die Ergebnisse der Ausführung aufbewahrt.

Der Wert 0 wird wiedergegeben, wenn die Ausführung normal erfolgt.

Der Wert -1 wird wiedergegeben, wenn während der Ausführung ein Fehler auftritt.

Port-Nummer

bestimmt die Port-Nummer, welches das Kanal des anderen Endes der Verbindung hinweist. Das Socket ist mit dieser Port-Nummer verbunden.

Der zulässige Bereich ist 8192 – 65535, sonst tritt ein Fehler auf.

Beispiel:

TCP_LISTEN .ret, port

TCP_END_LISTEN *Wiedergegebene Variable, Port-Nummer*

Funktion

Ein Programmkommando, welches zur Unterbrechung der Verbindung dient.

Es beendet das Warten auf die Forderung, die Verbindung bei einem durch TCP_LISTEN bestimmten Socket herzustellen und schließt dieses Socket. Wenn ein Kommunikationsfehler auftritt, wird der Wert -1.0 wiedergegeben und die Ausführung des Programms wird nicht angehalten.

Parameter

Wiedergegebene Variable

bestimmt die Variable, welche die Ausführungsergebnisse aufbewahrt.

Der Wert 0 wird wiedergegeben, wenn die Ausführung normal erfolgt.

Der Wert -1 wird wiedergegeben, wenn bei der Ausführung ein Fehler auftritt.

Port-Nummer

bestimmt das Socket, welches aktuell auf die Forderung nach einer Verbindung wartet (das im Kommando **TCP_LISTEN** bestimmte Socket).

Beispiel:

TCP_END_LISTEN .ret, port

TCP_SEND *Wiedergegebene Variable, SocketID, $Variable einer Zeichenfolge*

Funktion

Ein Programmkommando zur Versendung von Daten. Es sendet die Daten in Anlehnung an das TCP-Protokoll. Daten, die versendet werden sollen, sind als Variablen der Zeichenfolgen bezeichnet.

Wenn ein Kommunikationsfehler auftritt, wird der Wert -1.0 wiedergegeben und die Ausführung des Programms wird nicht angehalten.

Parameter

Wiedergegebene Variable

stellt die Variable, welche die Ergebnisse der Ausführung aufbewahrt, ein.

Der Wert 0 wird wiedergegeben, wenn die Ausführung normal erfolgt.

Der Wert -1.0 wird wiedergegeben, wenn während der Ausführung ein Fehler auftritt.

SocketID

bestimmt die Identifikationsnummer des Sockets, welche nach der Durchführung des Kommandos **TCP_ACCEPT** oder **TCP_CONNECT** erlangt wird.

$Variable einer Zeichenfolge

bestimmt die Variable einer Zeichenfolge, in welcher die zu versendenden Daten aufbewahrt werden.

Die Elemente der Variablen einer Zeichenfolge werden in der Reihenfolge vom ersten bis zum letzten versendet.

Die Zahlenangaben können nach der Umwandlung in ein Format einer Zeichenfolge mithilfe der Funktion **$ENCODE** versendet werden.

Beispiel:

$data = „HELLO”

TCP_SEND .ret, socketID, $data

TCP_RECV *Wiedergegebene Variable, socketID, $Variable der Zeichenfolge*

Funktion

Ein Programmkommando zum Empfang der Daten. Sie empfängt die Daten, welche über das TCP-Protokoll versendet werden und speichert diese in einer bestimmten Variablen der Zeichenfolgen.

Wenn ein Kommunikationsfehler auftritt, wird der Wert -1 wiedergegeben und die Ausführung des Programms wird nicht angehalten.

Parameter

Wiedergegebener Wert

bestimmt eine Variable, welche die Ergebnisse der Ausführung aufbewahrt.

Der Wert 0 wird wiedergegeben, wenn die Ausführung normal erfolgt.

Der Wert -1.0 wird wiedergegeben, wenn während der Ausführung ein Fehler auftritt.

socketID

bestimmt die Identifikationsnummer des Sockets, welche nach der Ausführung des Kommandos **TCP_ACCEPT** oder **TCP_CONNECT** erlangt wird.

$Variable einer Zeichenfolge

bestimmt die Variable einer Zeichenfolge, in welcher die empfangenen Daten aufbewahrt werden.

Die Daten werden empfangen als Zeichendaten mit einer Länge von je 1 Byte. Beim Empfang der Zahlendaten sind diese von einer Variablen der Zeichenfolge in eine Zahlenvariable durch die Funktion **VAL** umzuwandeln.

Beispiel:

```
WHILE .ret < 0 DO
    TCP_RECV .ret, socketID, $data
    TWAIT 0.2
END
PRINT $data
```

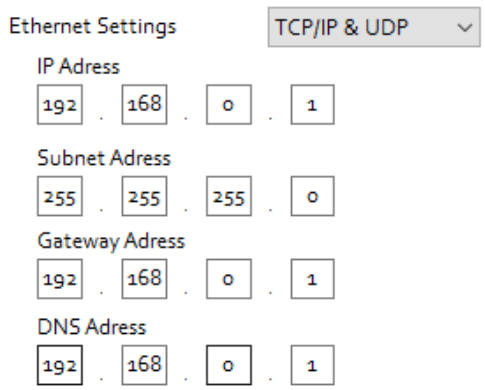
Das vorstehende Beispiel bringt das Programm zu einer Schleife, bis die Daten empfangen worden sind.

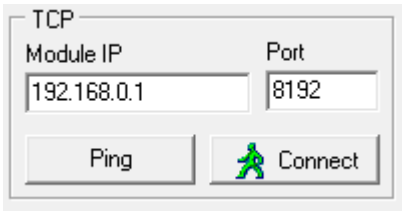
12.8 TCP/IP SERVER-BEISPIEL

Im nachfolgenden Beispiel funktioniert astorino als ein TCP/IP-Kommunikationsserver. Um die Funktion des Programms zu testen, wurde das Programm Hercules der Firma HW-group.com eingesetzt

(https://www.hw-group.com/software/hercules-setup-utility)

Einstellungen des Roboters und des Hercules-Programms





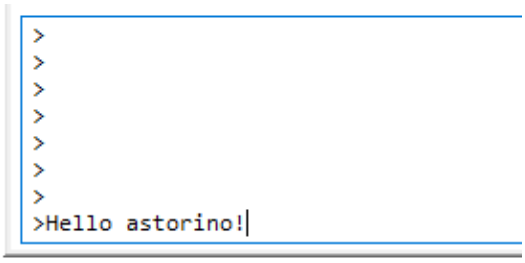
Programm-Code, der durch den Roboter ausgeführt wird:

```
.PROGRAM TCP1
  .ret = -1
  .ret2 = -1
  socketID= -1
  port = 8192
  $data = "Hello PC!"
  $data2 = ""
  TCP_LISTEN .ret, port
  WHILE socketID < 0 DO
    TCP_ACCEPT socketID, port
    TWAIT 0.2
  END
  TCP_SEND .ret, socketID, $data
  WHILE .ret2 < 0 DO
    TCP_RECV .ret2, socketID, $data2
    TWAIT 0.2
  END
  PRINT $data2
  TCP_CLOSE .ret,socketID
  TCP_END_LISTEN .ret, port
.END
```

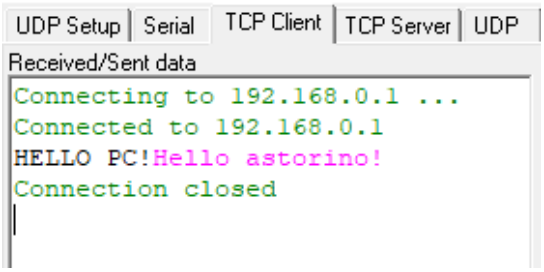
Daten, die von der Hercules-Software versendet werden:

Ergebnisse des vorstehenden Programms:

Ansicht des Terminals astorino-Software



Ansicht eines Fensters Hercules-Programm

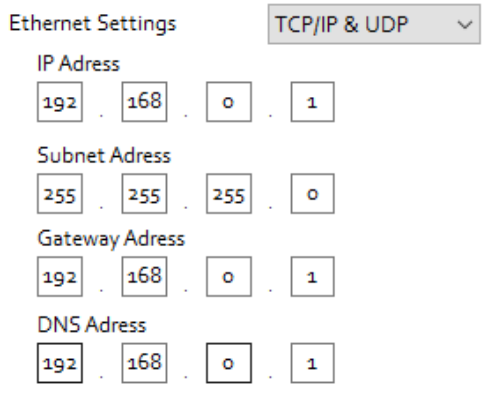


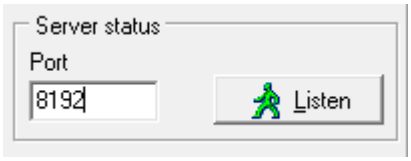
12.9 TCP/IP CLIENT-BEISPIEL

Im nachfolgenden Beispiel funktioniert astorino wie ein TCP/IP- Kommunikations-Client. Zum Testen der Programmfunktion wurde das Programm Hercules der Firma HW-group.com eingesetzt

(https://www.hw-group.com/software/hercules-setup-utility)

Einstellungen des Roboters und des Hercules-Programms





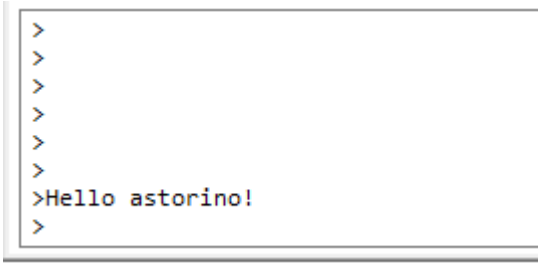
Programm-Code, der durch den Roboter ausgeführt wird:

```
.PROGRAM TCP
  .ret = -1
  .ret2 = -1
  socketID = -1
  port = 8192
  IP[1] = 192
  IP[2] = 168
  IP[3] = 0
  IP[4] = 100
  $data = "Hello PC"
  $data2 = ""
  WHILE socketID < 0 DO
    TCP_CONNECT socketID,port,IP[1]
    TWAIT 0.2
  END
  WHILE .ret< 0 DO
    TCP_SEND .ret, socketID, $data
    TWAIT 0.2
  END
  WHILE .ret2 < 0 DO
    TCP_RECV .ret2, socketID, $data2
    TWAIT 0.2
  END
  PRINT $data2
  TCP_CLOSE .ret,socketID
.END
```

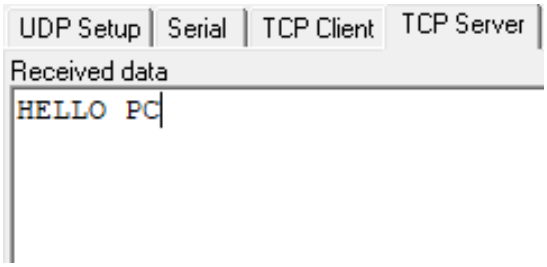
Daten, die vom Hercules-Software versendet werden:

Ergebnisse des vorstehenden Programms:

Ansicht des Terminals astorino-Software



Ansicht des Fensters Hercules-Programm



12.10 FUNKTIONEN DES UDP-KOMMUNIKATIONS-DIENSTES

UDP_SENDTO	sendet bestimmte Daten der Zeichenfolge über das UDP/IP-Protokoll.
UDP_RECVFROM	empfängt und bewahrt die Daten in der bestimmten Variablen der Zeichenfolge über das UDP/IP-Protokoll auf.

UDP_SENDTO *Wiedergegebene Variable, IP-Adresse, Port-Nummer, $Variable der Zeichenfolge*

Funktion

sendet eine Datenfolge über das UDP-Protokoll. Daten, die versendet werden sollen, sind in der Variablen der Zeichenfolge bestimmt. Dieses Kommando bildet das Socket, sendet Daten und schließt das Socket in einer Sequenz.

Wenn ein Kommunikationsfehler auftritt, wird der Wert -1 wiedergegeben und die Ausführung des Programms wird nicht angehalten.

[**VORSICHT**]

Die Funktion **UDP_SENDTO** öffnet das Socket beim Port **8192**

Parameter

Wiedergegebene Variable

bestimmt die Variable, welche die Ergebnisse der Ausführung aufbewahrt.

Der Wert 0 wird wiedergegeben, wenn die Ausführung normal erfolgt.

Der Wert -1.0 wird wiedergegeben, wenn während der Ausführung ein Fehler auftritt.

IP-Adresse

bestimmt die Array-Variable, welche die IP-Adresse des Servers (32 Bits) aufbewahrt.

Die IP-Adresse wird in 8-Bit-Inkrementen zu jedem Element der Array- Variablen in der Reihenfolge ab Anfang der IP-Adresse aufbewahrt.

Port-Nummer

bestimmt die Port-Nummer, welche aufs Kanal vom anderen Ende der Verbindung hinweist. Das Socket ist mit dieser Port-Nummer verbunden.

Der zulässige Bereich ist 8192 - 65535, sonst tritt ein Fehler auf.

$Variable der Zeichenfolge

bestimmt die Variable einer Zeichenfolge, in welcher die zu versendenden Daten aufbewahrt werden. Die Elemente der Variablen einer Zeichenfolge werden in der Reihenfolge vom ersten bis zum letzten versendet. Die Zahlendaten können versendet werden, nachdem sie in eine Zeichenfolge mithilfe der Funktion **$ENCODE** umgewandelt werden.

Beispiel:

UDP_SENDTO .ret, IP[1],port,$data

UPD_RECVFROM Wiedergegebene Variable, IP-Adresse, Port-Nummer, $Variable der Zeichenfolge

Funktion

empfängt eine Folge der Daten, die über das UDP-Protokoll versendet werden. Die empfangenen Daten werden in der Variablen einer Zeichenfolge gespeichert. Dieses Kommando bildet ein Socket, versendet Daten und schließt das Socket in einer Sequenz.

Wenn ein Kommunikationsfehler auftritt, wird der Wert -1 wiedergegeben und die Ausführung des Programms wird angehalten.

Parameter

Wiedergegebene Variable

bestimmt die Variable, welche die Ergebnisse der Ausführung aufbewahrt.

Der Wert 0 wird wiedergegeben, wenn die Ausführung normal erfolgt.

Der Wert -1.0 wird wiedergegeben, wenn während der Ausführung ein Fehler auftritt.

IP-Adresse

bestimmt die Array-Variable, welche die IP-Adresse des Servers (32 Bits) aufbewahrt.

Die IP-Adresse wird in 8-Bit-Inkrementen zu jedem Element der Array-Variablen in der Reihenfolge ab Anfang der IP-Adresse aufbewahrt.

Port-Nummer

bestimmt die Port-Nummer, welche aufs Kanal vom anderen Ende der Verbindung hinweist. Das Socket ist mit dieser Port-Nummer verbunden.

Der zulässige Bereich ist 8192 - 65535, sonst tritt ein Fehler auf.

$Variable einer Zeichenfolge

bestimmt die Variable einer Zeichenfolge, in welcher die zu versendenden Daten aufbewahrt werden. Die Elemente der Variablen einer Zeichenfolge werden in der Reihenfolge vom ersten bis zum letzten versendet. Die Zahlendaten können nach der Umwandlung mithilfe der Funktion **$ENCODE** in ein Format der Zeichenfolge versendet werden.

Beispiel:

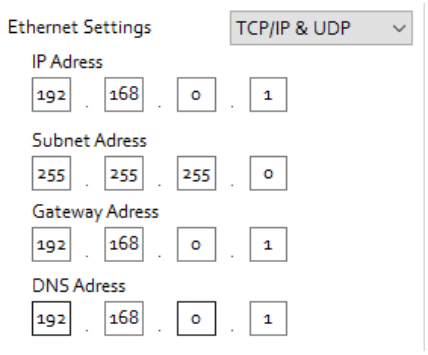
UDP_RECVFROM .ret,port, $data

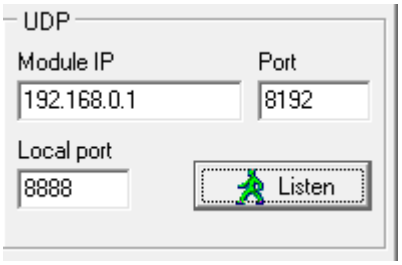
12.11 UDP BEISPIEL DER DATENVERSENDUNG

Im nachfolgenden Beispiel funktioniert astorino als Sender der Daten des UDP-Protokolls. Zum Testen der Programmfunktion wurde das Programm Hercules der Firma HW-group.com eingesetzt.

(https://www.hw-group.com/software/hercules-setup-utility)

Einstellungen des Roboters und des Hercules-Programms



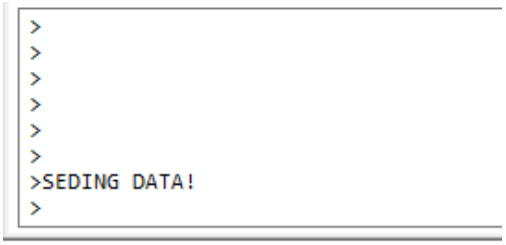


Der durch den Roboter ausgeführte Programmcode:

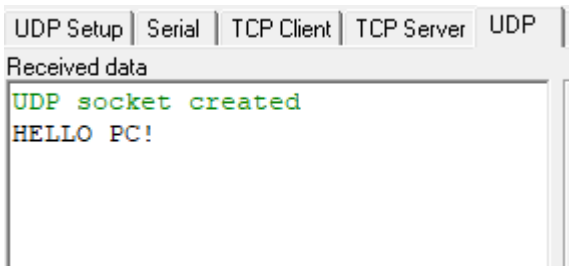
```
.PROGRAM UDP2
  .ret = 0
  $data = "Hello PC!"
  IP[1] = 192 ;computer IP
  IP[2] = 168
  IP[3] = 0
  IP[4] = 100
  port = 8888 ;localport
  PRINT "SEDING DATA!"
  UDP_SENDTO .ret, IP[1],port,$data
.END
```

Ergebnisse des vorgenannten Programms:

Ansicht des Terminals astorino-Software



Ansicht des Fensters Hercules-Programm

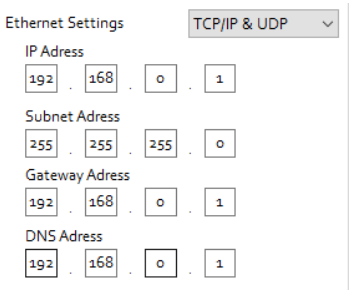


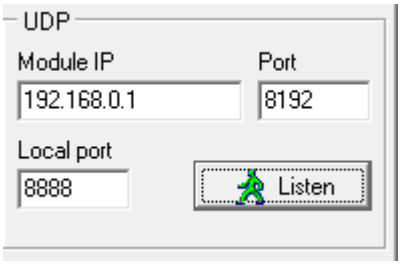
12.12 UDP BEISPIEL DES DATENEMPFANGS

Im nachfolgenden Beispiel funktioniert astorino als Empfänger der Daten des UDP-Protokolls. Zum Testen der Programmfunktion wurde das Programm Hercules der Firma HW-group.com eingesetzt

(https://www.hw-group.com/software/hercules-setup-utility)

Einstellungen des Roboters und des Hercules-Programms



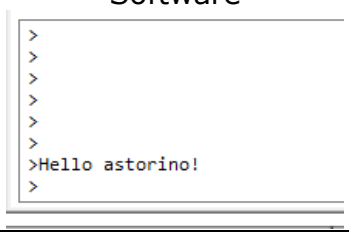
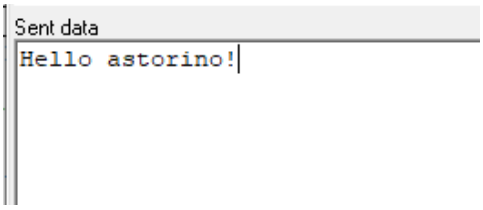


Der durch den Roboter ausgeführte Programmcode:

```
.PROGRAM UDP
  port = 8192
  $data = ""
  .ret = -1
  WHILE .ret < 0 DO
    UDP_RECVFROM .ret,port, $data
    TWAIT 0.1
  END
  PRINT $data
.END
```

Daten, die vom Hercules-Software versendet werden:

Ergebnisse des vorstehenden Programms:

Ansicht des Terminals astorino-Software	Ansicht des Fensters Hercules-Programm
	

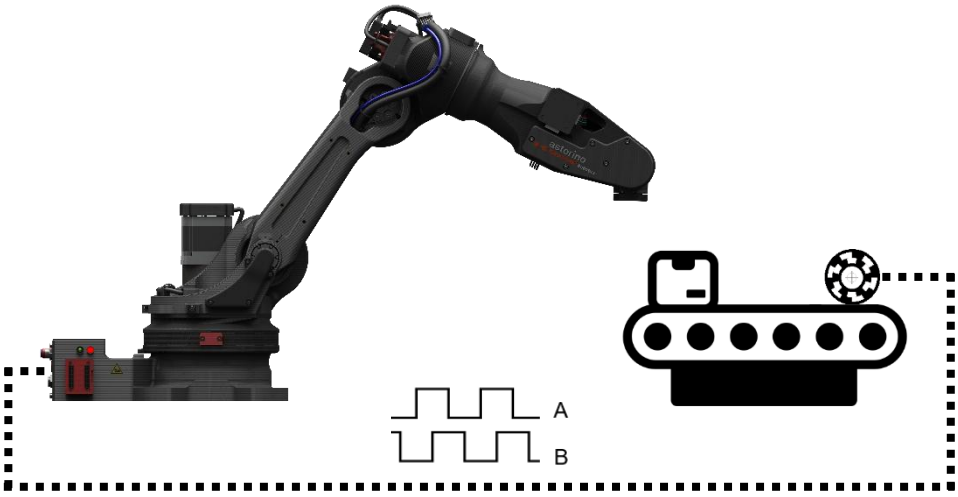
12.13 KOOPERATION MIT EXTERNEM ENKODER

Bei Standardvorgängen der Roboter bleibt der/das zu verarbeitende Gegenstand/Detail während der Arbeit unbeweglich. Die Synchronisierungsfunktion mit dem Förderband ermöglicht Vorgänge an Gegenständen, die sich auf dem Förderband bewegen, auszuführen. Nutzt man die Kooperationsfunktion mit dem externen Enkoder, bewegt sich der Roboter, indem er seine Bewegung mit dem beweglichen Gegenstand auf dem Förderband synchronisiert. Um mit dem beweglichen Detail zu synchronisieren, kann der Roboter maximal zwei externe Inkrementalenkoder einsetzen.
Berücksichtigt man die Reihenfolge der Bewegungen und den Programmfluss, sind:

- Bewegungen, die zur Folge haben, dass der Arbeitsbereich des Roboters verlassen wird,
- die unnötige Einstellung der Arbeit (Anhalten des Roboters während des Vorbeifahrens des Gegenstands am Roboter)

zu vermeiden.

Vor der Nutzung der Synchronfunktion des Förderbands sind die Auflösungsparameter und die Richtungsparameter der Förderbandbewegung einzustellen. Diese Daten sind im Astorino-Software einzustellen.

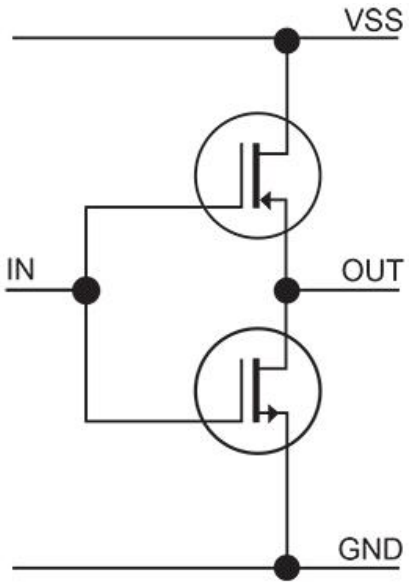


12.14 BEDIENTE ENKODER

Astorino ist imstande bis zu zwei zusätzlichen Inkrementalenkoder gleichzeitig zu bedienen.

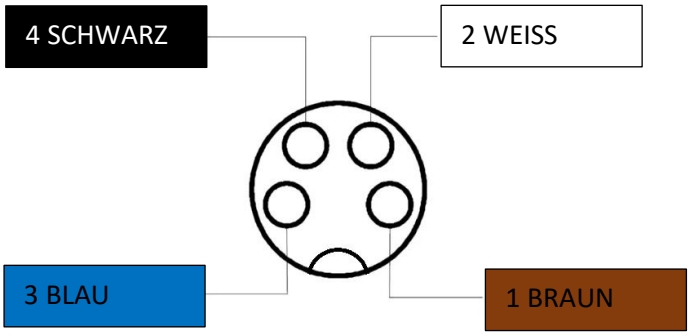
Die Grundparameter der bedienten Enkoder sind:

- Betriebsspannung 24V,
- Signalausgänge A und B,
- empfohlene Auflösung nicht größer als 300 PPR (Puls pro Drehung),
- Ausgänge, die in der Konfiguration PUSH-PULL funktionieren.



Ausgang in der PUSH-PULL-Konfiguration

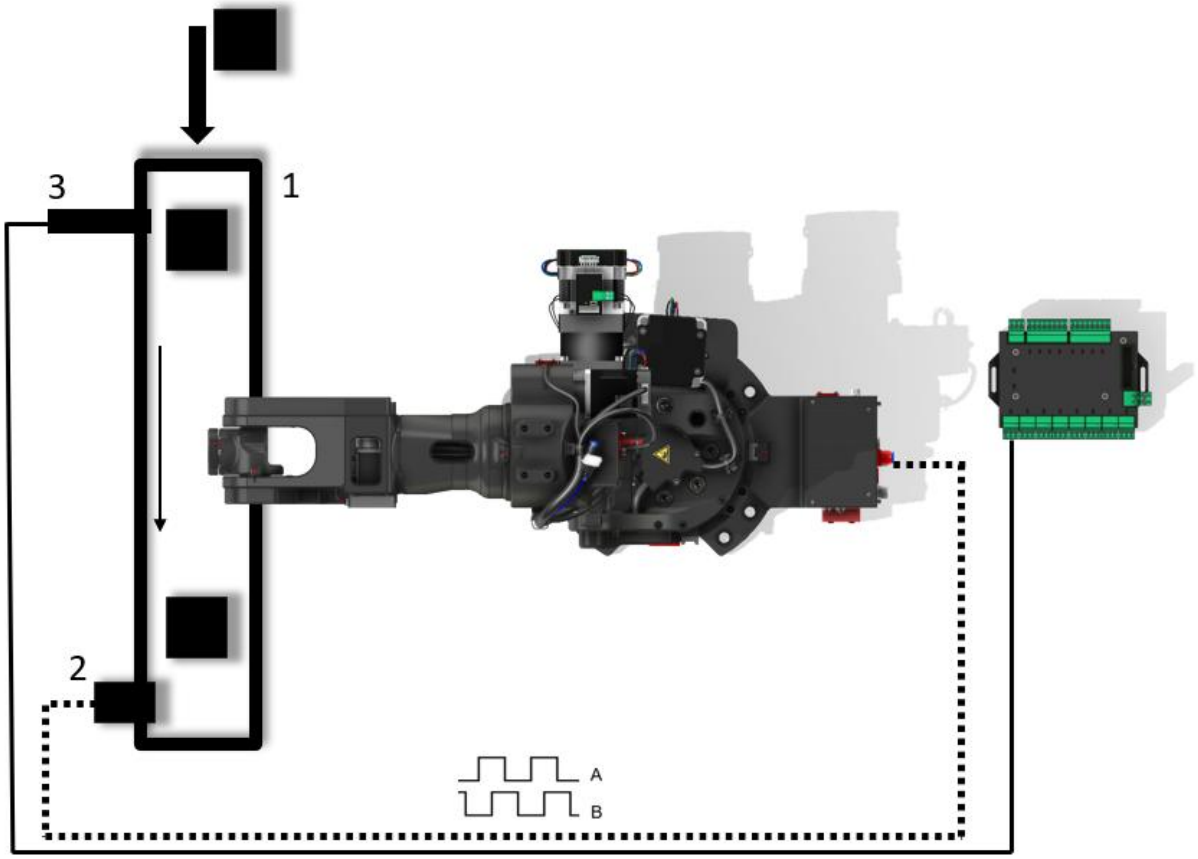
Jeder Enkodereingang im Roboter ist mit einem M8-Stecker mit vier Pins ausgeführt.



Nr. des Eingangs	1 BRAUN	2 WEISS	3 BLAU	4 SCHWARZ
1	A	B	24V	GND
2	A	B	24V	GND

12.15 BEISPIEL EINER ANWENDUNG, DIE DAS FÖRDERBAND NUTZT

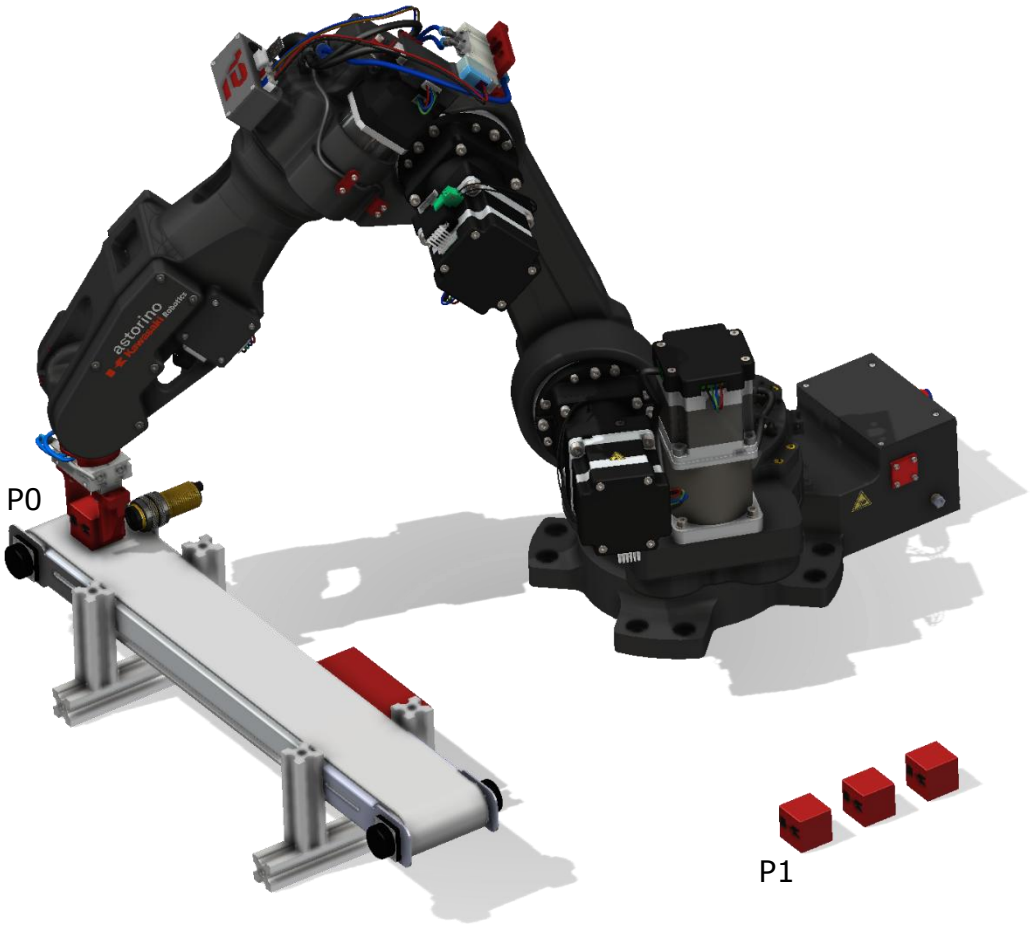
Im nachfolgenden Beispiel wurde der Roboter mit einem Bandförderer (1), einem Impulsenkoder 24V (2) und einem Näherungssensor (3) ausgestattet. Der Enkoder wurde an den ersten Enkoderanschluss und der Näherungssensor an den ersten Eingang im Eingangs-/Ausgangsmodul 24V angeschlossen. Um das Schema übersichtlicher zu gestalten, wurde die Verbindung des IO-Moduls 24V mit dem Roboter nicht gezeigt. Im nachfolgenden Beispiel wird es angenommen, dass das Förderband seine eigene Steuerung hat und dass seine Bewegung mit dem Pfeil auf der Zeichnung übereinstimmt.



In der vorstehenden Anwendung gibt der Nutzer Details (Würfel) am Anfang des Förderbands, zum Zeitpunkt, wenn der Sensorkegel überschritten wird, wird der Punkt gespeichert, an welchen dann der Roboter fährt und das Detail entnimmt, indem er sich zugleich mit dem Förderband synchronisiert. Demnächst legt er die entnommenen Elemente an einem anderen Ort ab.

Was man zuerst machen soll, ist die Einstellung des Förderbands in Robotereinstellungen zu konfigurieren. In diesem Beispiel beträgt die Auflösung 0.1mm/bit, und die Richtung ist auf X- eingestellt.

Vor der Inbetriebnahme sind der Punkt, in welchem sich das Detail zum Zeitpunkt der Erkennung durch den Sensor (3) befindet und der Punkt des Ablegens P1 zu speichern.



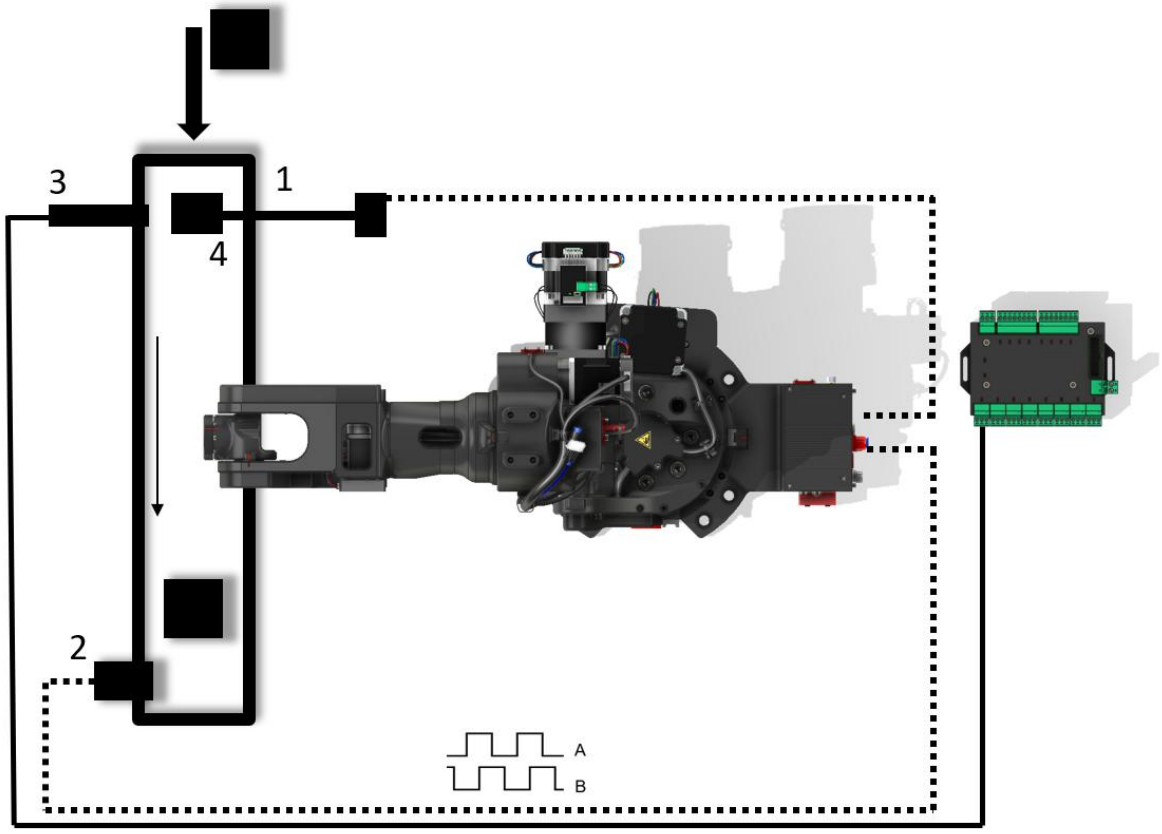
Damit die vorgenannte Anwendung richtig funktioniert, sind die Details in derselben Orientierung und Position auf dem Förderband (hinsichtlich der Breite) zu geben, man kann das machen durchs Nachplanen entsprechender Anschläge, die das Detail auf dem Förderband automatisch mittig positionieren werden.

Ein Beispielprogramm:

```
.PROGRAM CONV
  SPEED 100 MM/S ALWAYS
  TOOL 1
  POINT PICK = P0 ;P0 saved point at sensor
  POINT PLACE = P1 ;P1 saved put away point
  HOME
  CVCOOPJT 8; synch with 1st conv
  CVRESET 8
  WHILE SIG(1002) == TRUE DO
    SWAIT 1001 ;wait conv sensor signal
    ENC = CVPOS
    POINT/8 PICK = ENC ;store current encoder value to PICK
    CVWAIT 50 ; wait till conv moved 50 mm
    CVLAPPRO PICK,50
    SPEED 50 MM/S
    CVLMOVE PICK ;move to PICK
    CVDELAY 0.5 ;wait above conv 0.5s
    SIGNAL 1 ;close gripper
    CVDELAY 1 ;wait above conv 1s
    CVLDEPART 50
    JAPPRO PLACE, 50
    SPEED 20 MM/S
    LMOVE PLACE
    TWAIT 0.5
    SIGNAL -1
    TWAIT 1
    LDEPART 50
    POINT PLACE = SHIFT(PLACE BY 0,-50,0)
    IF CVPOS > 5000 THEN
      CVRESET 8 ; reset encoder if too big
    END
  END
.END
```

12.16 BEISPIEL EINER ANWENDUNG, DIE DAS FÖRDERBAND UND DAS VISIONSSYSTEM NUTZT

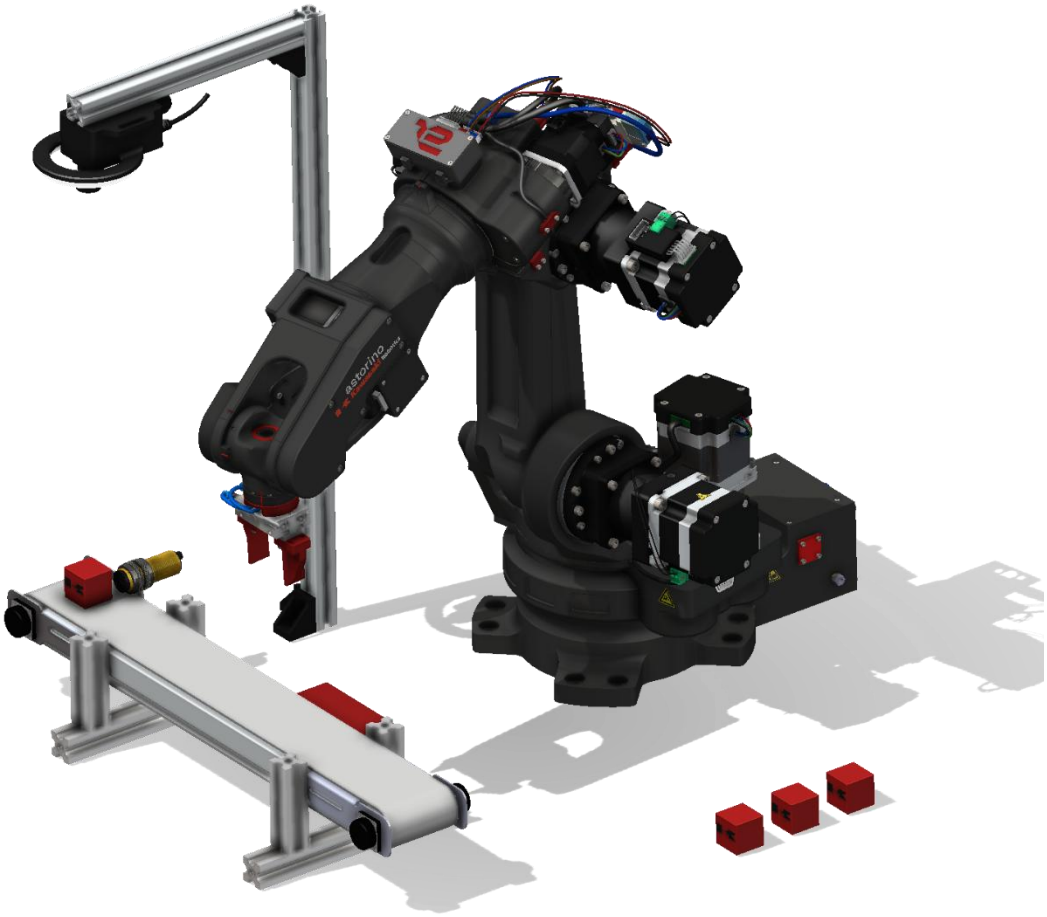
Im nachfolgenden Beispiel wurde der Roboter mit einem Bandförderer (1), einem Impulsenkoder 24V (2), einem Näherungssensor (3) und einem Visionssystem (4) ausgestattet. Der Enkoder wurde an den ersten Enkodereingang, der Näherungssensor an den ersten Eingang im Eingangs-/Ausgangsmodul 24V und das Visionssystem an den Serial-Eingang am Sockel des Roboters angeschlossen. Um das Schema übersichtlicher zu gestalten, wurde die Verbindung des IO-Moduls 24V mit dem Roboter nicht gezeigt. Im nachfolgenden Beispiel wird es angenommen, dass das Förderband seine eigene Steuerung hat und dass seine Bewegung mit dem Pfeil auf der Zeichnung übereinstimmt.



In der vorgenannten Anwendung gibt der Nutzer Details (Würfel) am Anfang des Förderbands, zum Zeitpunkt, wenn der Sensorkegel überschritten wird, wird eine Kamera aktiviert, welche das Objekt auf dem Förderband erkennt und Koordinaten an den Roboter sendet. Der Punkt, an welchen dann der Roboter fährt und das Detail entnimmt, indem er sich zugleich mit dem Förderband synchronisiert, wird gespeichert. Demnächst legt er die entnommenen Elemente an einem anderen Ort ab.

Was man zuerst machen soll, ist die Einstellung des Förderbands in Robotereinstellungen zu konfigurieren. In diesem Beispiel beträgt die Auflösung 0.1mm/bit, und die Richtung ist auf X- eingestellt.

Vor dem Ingangsetzen ist die Kamera gemäß Anweisung des Visionssystems einzustellen und der Ablegepunkt P1 einzuteachen. Es ist auch wie im letzten Beispiel der Punkt P0 an der beliebigen Position des Förderbands anzufahren, um die Koordinate Z der Entnahmeposition abzulesen. Demnächst ist sie im Programm in der Zeile **POINT PICK = TRANS (dataX, dataY, gemessene_Höhe,0,0,0)** einzutragen. In dem Beispielprogramm wurde der Wert 100 mm angegeben.



```
.PROGRAM CONV
  SPEED 100 MM/S ALWAYS
  TOOL 1
  POINT PLACE = P1 ;P1 saved put down point
  HOME
  CVCOOPJT 8; synch with 1st conv
  CVRESET 8
  WHILE SIG(1002) == TRUE DO
    SWAIT 1001 ;wait conv sensor signal
    SEND "T"
    WHILE EXISTCOM == false DO
      TWAIT 0.05
    END
    $temp = RECEIVE
    $temp2 = $DECODE($temp,"/")
    $temp3 = $DECODE($temp, "/")
    $temp4 = $DECODE($temp, "/")
    dataX = VAL($temp2)
    dataY = VAL($temp3)
    dataA = VAL($temp4)
    IF ((dataX <> 0) OR (dataY <> 0)) THEN
      POINT PICK = TRANS(dataX,dataY,100,0,0,0)
      POINT/OAT PICK = P0
      POINT PICK = PICK + RZ(dataA)
      ENC = CVPOS
      POINT/8 = ENC
      CVWAIT 100 ; wait till conv moved 50 mm
      SPEED 100 MM/S ALWAYS
      CVLAPPRO PICK, 40
      SPEED 40 MM/S ALWAYS
      CVLMOVE PICK ;move to PICK
      CVDELAY 0.5 ;wait above conv 0.5s
      SIGNAL 1 ;close gripper
      CVDELAY 1 ;wait above conv 1s
      CVLDEPART 50
      JAPPRO PLACE, 50
      SPEED 20 MM/S
      LMOVE PLACE
      TWAIT 0.5
      SIGNAL -1
      TWAIT 1
      LDEPART 50
      POINT PLACE = SHIFT(PLACE BY 0,-50,0)
      IF CVPOS > 5000 THEN
        CVRESET 8 ; reset encoder if too big
      END
    ELSE
      PRINT "No workpiece"
      CVRESET 8
    END
  END
.END
```

13 BEISPIELPROGRAMME

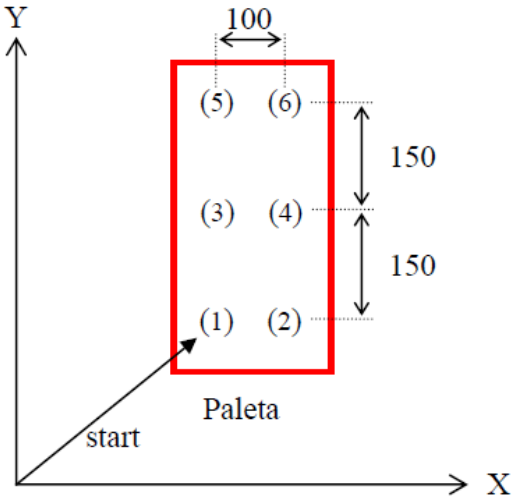
In diesem Kapitel wurden die Beispielprogramme in der AS-Sprache dargestellt.

13.1 ANFANGSKONFIGURIERUNG DER PROGRAMME

Die Ausführung der nachfolgend angegebenen Tätigkeiten vor der Nutzung der jeweiligen Funktionen des Roboters erleichtert das Programmieren.

- der Roboter ist zur Hausposition zu verfahren.
- die erforderlichen Variablen für jede Aufgabe sind zu definieren. (z. B. zur Palettierung ist die Anzahl der Gegenstände auf der Palette einzugeben)
- der Zähler, die Flagge etc. sind zu initialisieren
- das Werkzeugkoordinatensystem für die umzusetzende Aufgabe ist einzustellen.
- das globale Koordinatensystem für die umzusetzende Aufgabe ist einzustellen.

Im Folgenden wurde ein Beispiel des Unterprogramms zur Initialisierung der Einstellungen für die auf der Zeichnung dargestellte Aufgabe Palettierung vorgegeben.

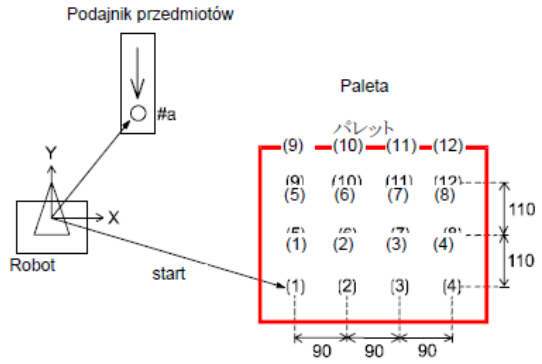


In dem platzierten Beispiel werden die Gegenstände in der Reihenfolge von (1) bis (6) palettiert. Es sind die im Folgenden vorgegebenen Anfangseinstellungen einzupflegen. Die Palette ist parallel zu dem globalen Koordinatensystem des Roboters platziert.

```
TOOL 1 ;transformacja narzędzia (1)
rowmax = 3  ; 3 wiersze
colmax = 2   ; 2 kolumny
xs = 100  ; odległość rozmieszczania w kierunku osi X(ΔX=100mm).
ys = 150  ; odległość rozmieszczenie w kierunku osi Y(ΔY=150mm).
POINT put = start   ; przypisanie pozycji start do pozycji put.
HOME   ; przejście do pozycji domowej
```

13.2 PALETTIERUNG

Im nachfolgenden Beispiel werden die Gegenstände aus dem Spender entnommen und auf Paletten in drei Reihen (im Abstand 110 mm) und in vier Spalten (im Abstand 90 mm) platziert. Um das Beispiel einfacher zu gestalten, sind sowohl die Palette als auch Gegenstände auf der Palette parallel zu der Ebene XY im globalen Koordinatensystem des Roboters angeordnet. Es wurde auch die Prozedur der Synchronisierung des Spenders und des Roboters mithilfe der externen Signale Eingang/Ausgang (Kommandos SWAIT, SIGNAL, etc.) ausgelassen.



- Die Palette ist parallel zur Ebene XY des globalen Koordinatensystems eingestellt.
- Die Position #P0 (Gegenstandsspender) und die Position "P0" (in welcher der erste Gegenstand platziert ist) müssen vor Beginn der Programmausführung definiert werden.

Ein Beispielprogramm

```
.PROGRAM palletize
  ; initial settings (3 rows, 4 columns)
  ; span X=90, Y=110
  rowmax = 3
  colmax = 4
  xs = 90
  ys = 110
  gripper = 1 ; 1st signal - gripper
  SPEED 100 MM/S ALWAYS
  POINT put = P0
  SIGNAL -gripper ;open gripper
  ; Palletization start
  FOR row = 1 TO rowmax
    FOR col = 1 TO colmax
      JAPPRO #P0, 100
      SPEED 30
      LAPPRO #P0, 50
      SPEED 30 MM/S
      LMOVE #P0
      TWAIT 0.2
      SIGNAL gripper
      TWAIT 1
      LDEPART 200
      JAPPRO put, 200
      SPEED 30 MM/S
      LMOVE put
      TWAIT 0.2
      SIGNAL -gripper
      TWAIT 1
      LDEPART 200
      ;next place point calculation - row
      POINT put=SHIFT (put BY xs, 0, 0)
    END
    ;next place point calculation - column
    POINT put = SHIFT (P0 by 0, ys * row, 0)
  END
.END
```

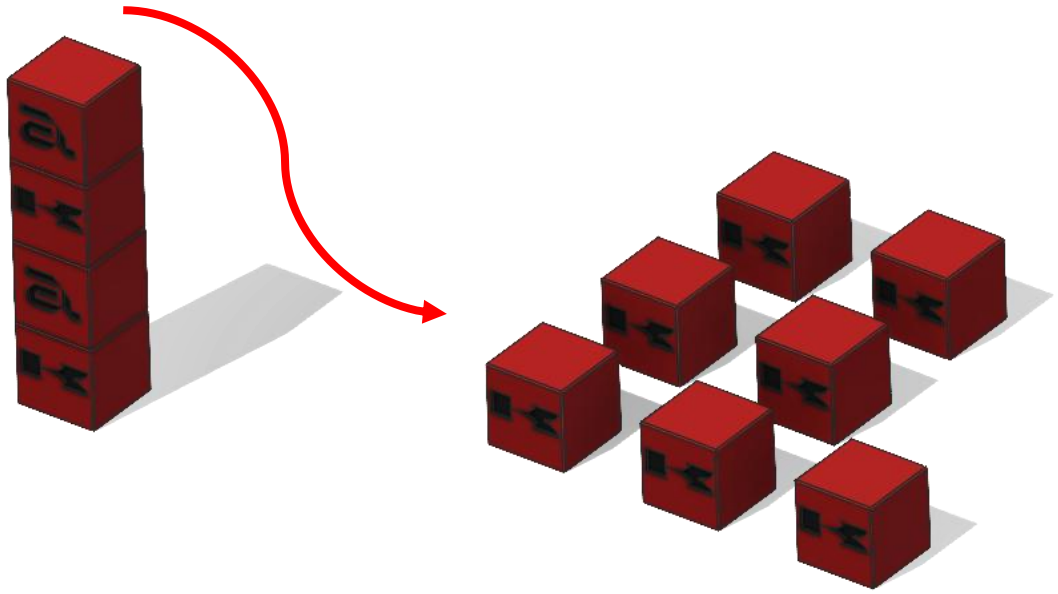
13.3 PICK&PLACE – PALETTIERUNGSBEISPIEL

Dieses Programm entnimmt Würfel aus einem Einzelturm und demnächst platziert es sie nach Anzahl der Reihen, Anzahl der Spalten und Anzahl der Ebenen.

Der Nutzer kann:

- die Größe des Gegenstands (des Würfels)
- den Abstand zwischen den Würfeln,
- die Anzahl der Reihen, Spalten und Ebenen,

anpassen.



```
.PROGRAM PAL1
;-------------- Init ---------
deltaX = 60 ;distance between workpieces X
deltaY = 60 ;distance between workpieces Y
deltaZ = 30 ;layer height
numLev = 2
numRow = 1
numCol = 2
numPcs = numLev*numCol*numRow  ;pieces count
height = 25   ;height of a workpiece (25 mm)
;-------------- variable init ---------
x = 0
y = 0
z = 1
SIGNAL -1
SPEED 100 mm/s always
POINT place = p2
POINT pick = P1
POINT pick = SHIFT(p1 BY 0,0,numPcs*height)
;P1 on the table, pick shifted by number of pieces in Z
HOME
LAPPRO pick, 40
;---------------------------------- Pal-----------------
FOR z = 0 TO (numLev-1)
  FOR y = 0 TO (numRow-1)       ; rows in Y
    FOR x = 0 TO (numCol-1)      ;col in X
      POINT pick = SHIFT(pick BY 0,0,-height);calc new pick pose
      JAPPRO pick,40
      speed 20 mm/s
      LMOVE pick
      TWAIT 0.5
      SIGNAL 1 ;close the gripper
      TWAIT 0.5
      LDEPART 50
      LMOVE P3
      POINT place = p2
      POINT place = SHIFT(p2 BY deltax*x,deltay*y,deltaz*z)
      LAPPRO place,30
      speed 20 mm/s
      LMOVE place
      TWAIT 0.5
      SIGNAL -1 ;open the gripper
      TWAIT 0.5
      LDEPART 30
      LMOVE P3
    END
  END
END
.END
```

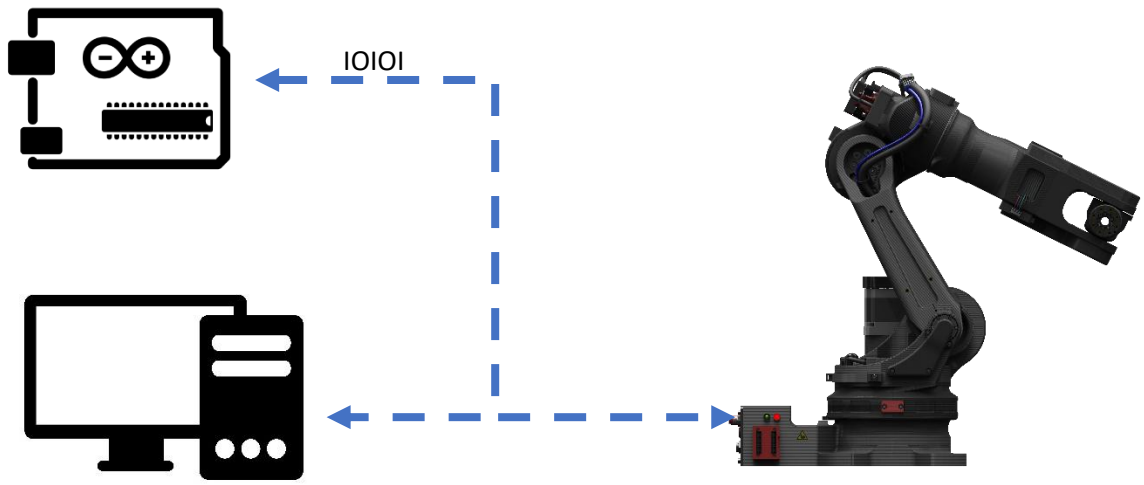
13.4 BEISPIELPROGRAM EINGANG/AUSGANG

Dieses Beispielprogramm zeigt, wie man Signale auf viele Weisen nutzen kann.

```
.PROGRAM IO
; ----- IO example program
; ----- Robot reads and sets IOs
  sensor = 1002 ;sets variable
  SWAIT 2001 ;wait until internal 1 signal is on
  SIGNAL 8 ;sets 8 output HIGH
  IF SIG(sensor) == TRUE THEN
    ;checks if sensor(2 input) is high
    SIGNAL 2002 ; sets 2 internal HIGH
  ELSE
    IF SIG(1001) == FALSE THEN
      SIGNAL -8 ;sets 1 output LOW
    END
  END
  BITS 1,4 = 12
  ;changes 12 to 4bit binäry and sets that on out    puts from 1
  data = BITS(1004,4) ;read binäry data from inputs
  ;4 bit from   4th output and changes that to decimal
  PRINT data
.END
```

13.5 BEISPIELPROGRAMM FÜR DIE SERIELLE DATENÜBERTRAGUNG

In diesem Beispiel wurde gezeigt, wie die serielle Datenübertragung zu nutzen ist. Das Programm kann die Daten zwischen dem astorino-Roboter und dem PC-Computer (z. B. Matlab oder SerialTerminal) oder Mikrokontroller (z. B. Arduino oder ESP32) austauschen.



```
.PROGRAM serial
; ----- Serial communication example program
; ----- Robot command frame form Serial Port
; ----- frames: P/ or L/x/y/z/
; ----- From X,Y,Z point is created
; ----- Sends current location if frame is P/
  SPEED 150 MM/S ALWAYS
  $S_FRAME = "XYZ"
  $S_FRAME2 = "JT"
  WHILE EXISTCOM == FALSE DO
    TWAIT 0.1
  END
  $TEMP = RECEIVE
  $COMMAND = $DECODE($TEMP,"/")
  PRINT $COMMAND
  ;RECEIVE DATA FROM SERIAL AND CREATE A POINT
  IF $COMMAND == "L" THEN
    $VAL1 = $DECODE($TEMP,"/")
    $VAL2 = $DECODE($TEMP,"/")
    $VAL3 = $DECODE($TEMP,"/")
    DATAX = VAL($VAL1)
    DATAY = VAL($VAL2)
    DATAZ = VAL($VAL3)
    POINT TEST = TRANS(DATAX,DATAY,DATAZ,0,0,0)
    POINT/OAT TEST = P0
    LMOVE TEST
    SEND "OK"
  END
  ;SEND CURRENT LOCATION TO SERIAL PORT
  IF $COMMAND == "P" THEN
    HERE TEMP
    HERE #TEMP
    DECOMPOSE TAB[0] = TEMP
    DECOMPOSE TAB2[0] = #TEMP
    FOR I = 0 TO 5
      TAB2[I] = TAB2[I]*180/PI
      $S_FRAME = $S_FRAME + $ENCODE(TAB[I]) + "/"
      $S_FRAME2 = $S_FRAME2 + $ENCODE(TAB2[I]) + "/"
    END
    SEND $S_FRAME
    SEND $S_FRAME2
  END
.END
```

⚠ WARNHINWEIS

Die serielle Datenübertragung funktioniert mit der Spannung 3,3V, bitte kompatible Elektronik mit der Spannung 3,3V oder Spannungswandler verwenden. Die Spannung 5V kann das CPU-Hauptsystem beschädigen!

14 INFORMATIONEN ÜBER DEN HERSTELLER

Kawasaki Robot
ASTORINO ANLEITUNG DER AS-SPRACHE

2024-04: 4. Ausgabe

Veröffentlichung: KAWASAKI Robotics GmbH