Änderungen in dieser Handbuchrevision

07.08.06
von Ichihashi

Titel	SPEL+ Sprachreferenz (RC+ Ver.5.0)	Rev.1a	1/1

Änderungen im Vergleich zur vorherigen Handbuchversion

Programmbeispiele für die Era-, die Erl- und die Ert-Funktion wurden geändert.

Fehler im vorherigen Handbuch wurden korrigiert.

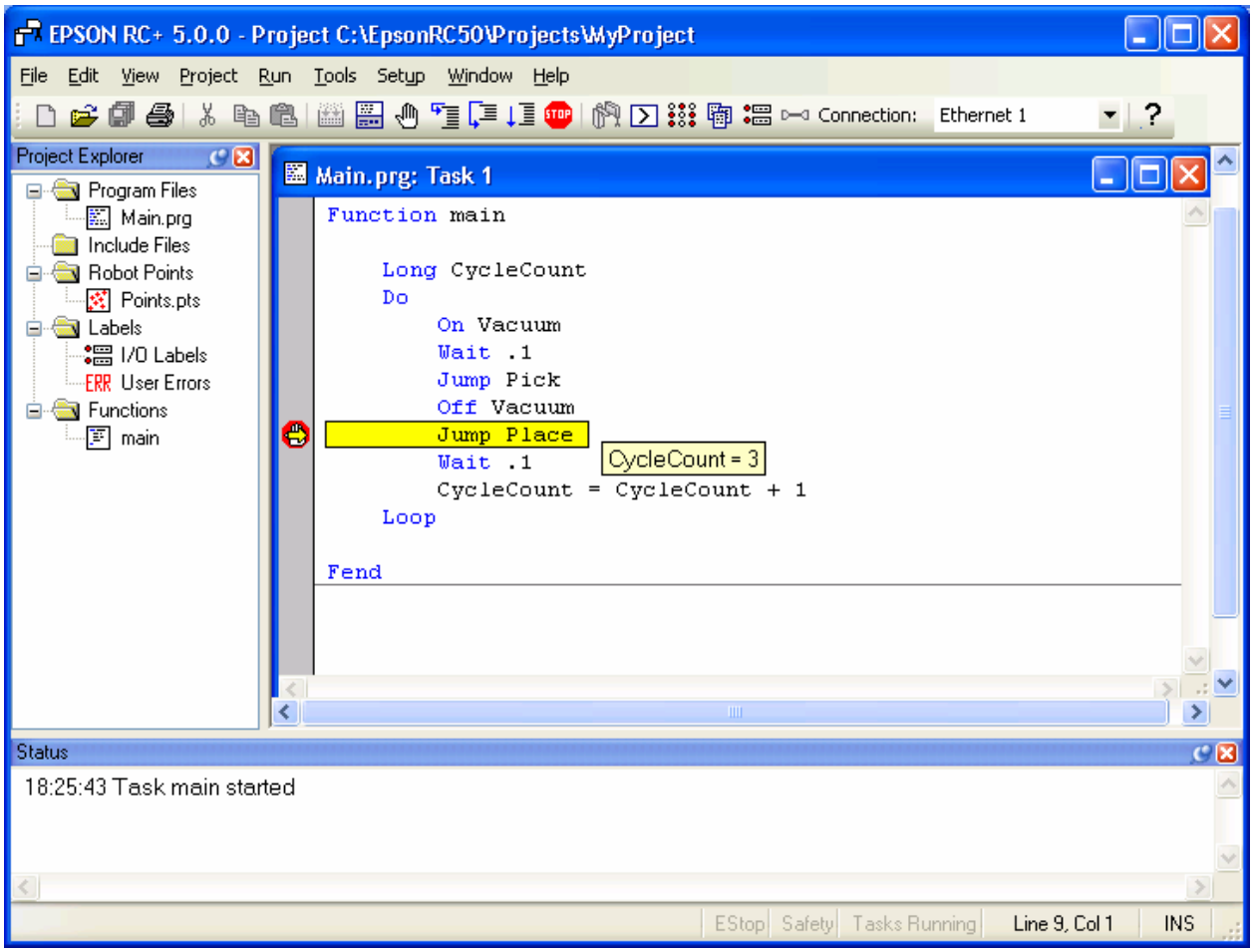
Teil	Seite	Details zu Ergänzungen und Änderungen	
Gesamtes Handbuch	-	Änderung	Ver.5.0 Rev.1 → Ver.5.0 Rev.1a Handbuchnummer: EM066S1327F → EM068S1369F
Era-Funktion	104	Änderung	Programmbeispiel wurde geändert.
Erf$-Funktion	106	Änderung	Programmbeispiel wurde geändert.
Erl-Funktion	107	Änderung	Programmbeispiel wurde geändert.
ErrMsg$-Funktion	109	Änderung	Programmbeispiel wurde geändert.
Ert-Funktion	111	Änderung	Programmbeispiel wurde geändert.
PCTRCLR-Anweisung	(256)	Auslassung	Diese Anweisung wurde gelöscht.
Vorkehrungen in Bezug auf die EPSON RC+ 4.0-Kompatibilität, Übersicht	462	Änderung	Precausion of … → Precaution of …
		Änderung	EPAON → EPSON
SPEL+Fehlermeldung	409-461	Änderung	Fehlermeldungen wurden geändert: 1026, 1031 1032 1046, 1503, 2201, 3251, 3326, 4045, 4046, 4050
		Ergänzung	Fehlermeldungen wurden ergänzt: 0517, 1048, 1049, 1510, 2318, 2319, 2351, 2352, 2420, 2522, 2523, 2904, 3000, 3173, 3174, 3175, 3176, 3177, 3806,7033, 9017
Vorkehrungen in Bezug auf die EPSON RC+ 4.0-Kompatibilität, Allgemeine Unterschiede	463	Änderung	Tabelle „Portnummer in Ethernet” 128 to 47 → 128 to 147

EPSON

EPSON RC+ 5.0

SPEL+ Sprachreferenz

Rev.1a EM068S1369F



EPSON RC+ 5.0 SPEL+ Sprachreferenz Rev.1a

EPSON RC+ 5.0

SPEL+ Sprachreferenz

Rev. 1a

VORWORT

Vielen Dank, dass Sie unsere Roboterprodukte erworben haben.
Dieses Handbuch beinhaltet die nötigen Informationen für die richtige Bedienung des Bedienpults.
Bitte lesen Sie dieses und die anderen relevanten Handbücher sorgfältig, bevor Sie das Robotersystem installieren.
Bewahren Sie dieses Handbuch so auf, dass es jederzeit griffbereit ist.

GARANTIE

Das Robotersystem sowie alle optionalen Teile werden vor der Lieferung strengen Qualitätskontrollen, Tests und Prüfungen unterzogen. So wird sichergestellt, dass das System in einem einwandfreiem Zustand ist und unseren hohen Leistungsanforderungen genügt.

Produktfehler, die trotz normalen Betriebs und normaler Handhabung entstehen, werden innerhalb der normalen Garantiezeit kostenlos repariert. (Bitte informieren Sie sich bei Ihrem Händler über die übliche Garantiezeit.)

Für die Reparatur folgender Schäden muss der Kunde selbst aufkommen (selbst wenn sie innerhalb der Garantiezeit auftreten):

1. Schäden oder Fehlfunktionen durch bestimmungswidrige oder nachlässige Verwendung. Es handelt sich um eine Verwendung die in diesem Handbuch nicht als bestimmungsgemäß beschrieben wird.
2. Fehlfunktionen durch unerlaubte Demontage durch den Kunden.
3. Schäden durch unerlaubte Einstellungen oder Reparaturversuche
4. Schäden durch Naturkatastrophen (wie z. B. Erdbeben, Überschwemmung oder Hochwasser usw.)

Warnhinweise, Nutzung:

1. Werden das Robotersystem oder die mit ihm verbundene Ausrüstung nicht entsprechend den festgelegten Betriebsbedingungen und Produktspezifikationen, die in diesem Handbuch angegeben sind, betrieben, verfällt der Garantieanspruch.
2. Wenn Sie Sicherheitshinweise in diesem Handbuch, die mit WARNUNG und VORSICHT gekennzeichnet sind, nicht befolgen, übernehmen wir keine Haftung für Fehlfunktionen oder Unfälle, die aus dieser Nichtbeachtung resultieren. Dies gilt auch für Unfälle, die zu Verletzungen oder zum Tod führen.
3. Wir können nicht alle möglichen Gefahren und die daraus resultierenden Folgen vorhersehen. Aus diesem Grund kann dieses Handbuch den Nutzer nicht vor allen Gefahren warnen.

WARENZEICHEN

Microsoft, Windows, und das Windows-Logo sind eingetragene Warenzeichen oder Warenzeichen der Microsoft Corporation in den USA und / oder anderen Ländern. Andere Marken und Produktnamen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Inhaber.

WARENZEICHEN IN DIESEM HANDBUCH

Betriebssystem Microsoft® Windows® XP
Betriebssystem Microsoft® Windows® 2000
Windows XP und Windows 2000 beziehen sich in diesem Handbuch auf die o. g. Betriebssysteme. In manchen Fällen bezieht sich Windows allgemein auf Windows XP und Windows 2000.

HINWEIS

Wir behalten uns vor, die in diesem Handbuch enthaltenen Informationen ohne Vorankündigung zu ändern.
Bitte benachrichtigen Sie uns, wenn Sie in diesem Handbuch Fehler finden, oder uns etwas zum Inhalt mitteilen möchten.

KONTAKT

Bei Reparaturen, Inspektionen oder Neueinstellungen wenden Sie sich bitte an das unten angegebene Service-Center.

Sollten an dieser Stelle keine Information bezüglich Ihres Service-Centers angegeben sein, wenden Sie sich bitte an ihren Händler.

Bitte halten Sie folgende Informationen bereit, wenn Sie sich an uns wenden.

- Das Steuerungsmodell und die Seriennummer
- Das Manipulatormodell und die Seriennummer
- Die Softwareversion Ihres Robotersystems
- Die Beschreibung des Problems

SERVICE-CENTER

HERSTELLER UND LIEFERANTEN

Japan und andere

SEIKO EPSON CORPORATION
Suwa Minami Plant
Factory Automation Systems Div.
1010 Fujimi, Fujimi-machi,
Suwa-gun, Nagano, 399-0295
JAPAN
TEL. : +81-(0)266-61-1802
FAX : +81-(0)266-61-1846

LIEFERANTEN

Nord- und Süd-Amerika

EPSON AMERICA, INC.
Factory Automation/Robotics
18300 Central Avenue
Carson, CA 90746
USA
TEL. : +1-562-290-5900
FAX : +1-562-290-5999
E-MAIL:info@robots.epson.com

Europa

EPSON DEUTSCHLAND GmbH
Factory Automation Division
Otto-Hahn-Str.4
D-40670 Meerbusch
Deutschland
TEL. : +49-(0)-2159-538-1391
FAX : +49-(0)-2159-538-3170
E-MAIL:robot.infos@epson.de

SICHERHEITSVORKEHRUNGEN

Die Installation von Robotern und Roboterausrüstung darf nur von qualifiziertem Personal in Übereinstimmung mit nationalen und örtlichen Vorschriften durchgeführt werden. Bitte studieren Sie dieses Handbuch und andere damit in Verbindung stehende Handbücher eingehend, wenn Sie diese Software verwenden.
Halten Sie dieses Handbuch zu jedem Zeitpunkt griffbereit.

 WARNUNG	■ Dieses Symbol weist Sie auf die Gefahr schwerer Verletzungen oder Lebensgefahr hin, die besteht, wenn die zugehörigen Anweisungen nicht befolgt werden.
 VORSICHT	■ Dieses Symbol weist Sie auf mögliche Personen- oder Sachschäden an der Ausrüstung oder am System hin, die entstehen können, wenn die zugehörigen Anweisungen nicht befolgt werden.

INHALTSVERZEICHNIS

Übersicht der SPEL+-Befehle 1

Systemverwaltungsbefehle 1
Roboter-Steuerungsbefehle 1
Eingangs- / Ausgangs-Befehle....3
Befehle zur Punkteverwaltung....3
Befehle zur Änderung von Koordinaten 4
Befehle zur Programmsteuerung 5
Befehle zur Programmausführung 6
Pseudo-Anweisungen 6
Befehle bezüglich numerischer Werte 6
Zeichenketten-Befehle 7
Logische Operatoren....8
Befehle für Variablen....8
Mit VB-Guide verwendete Befehle 8

SPEL+ Sprachreferenz 9

SPEL+-Fehlermeldungen 436

Ereignisse....436
Warnungen....438
Hauptsteuerung....439
Bedienpult....445
Teach-Pendant....446
Interpreter....448
Parser....467
Motorsteuerung 475
Servo 483
Punkte....491
Feldbus....493
Hardware....494
EPSON RC+ 498

Vorkehrungen in Bezug auf die EPSON RC+ 4.0-Kompatibilität 499

Übersicht 499
Allgemeine Unterschiede 500
Kompatibilitätsliste für Befehle 501
Liste neuer Befehle....511

Übersicht der SPEL+-Befehle

Nachfolgend finden Sie eine Übersicht der SPEL+-Befehle.

Systemverwaltungsbefehle

Reset	Setzt die Robotersteuerung zurück.
SysConfig	Zeigt die Systemdaten der Robotersteuerung an.
Date	Stellt das Systemdatum ein.
Time	Stellt die Systemzeit ein.
Date$	Gibt das Systemdatum als Zeichenkette aus.
Time$	Gibt die Systemzeit als Zeichenkette aus.
Hour	Zeigt die Betriebsstunden der Robotersteuerung an bzw. gibt sie aus.
Stat	Gibt den Status der Robotersteuerung aus.
CtrlInfo	Gibt Informationen über die Robotersteuerung aus.
RobotInfo	Gibt Informationen über den Roboter aus.
RobotInfo$	Gibt Textinformationen über den Roboter aus.
TaskInfo	Gibt Taskinformationen aus.
TaskInfo$	Gibt Textinformationen über Tasks aus.

Roboter-Steuerungsbefehle

Power	Definiert den Servo-Power-Modus bzw. gibt ihn aus.
Motor	Definiert den Motorstatus bzw. gibt ihn aus.
SFree	Schaltet die angegebene Servoachse frei.
SLock	Stellt Servo-Power für die angegebene Servoachse wieder her.
Jump	Führt die Bewegung zu einem Punkt mittels einer PTP-Bewegung aus.
Jump3	Führt die Bewegung zu einem Punkt mittels einer dreidimensionalen Gate-Bewegung aus.
Jump3CP	Führt die Bewegung zu einem Punkt mittels einer dreidimensionalen CP-Bewegung aus.
Arch	Definiert die Bogenparameter für die Jump-Bewegung bzw. gibt sie aus.
LimZ	Definiert den oberen Z-Grenzwert für den Jump-Befehl.
Sense	
JS	Gibt den Status einer Sense-Funktion aus.
JT	Gibt den Status des zuletzt ausgegebenen Jump-Befehls für den aktuellen Roboter aus.
Go	Führt die Roboterbewegung zu einem Punkt mittels einer PTP-Bewegung aus.
Pass	Führt vier PTP-Bewegungen gleichzeitig aus und bewegt den Manipulatorarm dabei an festgelegten Punkten vorbei.

	(Verschleifen von Hilfspunkten, ohne diese exakt zu überfahren.)
Pulse	Bewegt den Roboter zu einer in Pulsen definierten Position.
BGo	Führt eine relative PTP-Bewegung im ausgewählten lokalen Koordinatensystem aus.
BMove	Führt im ausgewählten lokalen Koordinatensystem eine relative linearinterpolierte Bewegung aus.
TGo	Führt eine relative PTP-Bewegung im aktuellen Werkzeug-Koordinatensystem aus.
TMove	Führt im ausgewählten Werkzeug-Koordinatensystem eine relative linearinterpolierte Bewegung aus.
Till	Spezifiziert das Bewegungsende, wenn eine Eingabe vorgenommen wird.
!…!	Verarbeitet Prozessanweisungen parallel zur Ausführung von Bewegungsbefehlen.
Speed	Definiert bzw. gibt die Geschwindigkeit für eine PTP-Bewegung aus.
Accel	Definiert bzw. gibt die Einstellungen für Beschleunigung und Verzögerung von PTP-Bewegungen aus.
Inertia	Definiert das Massenträgheitsmoment des Roboterarms bzw. gibt es aus.
Weight	Definiert die Gewichtseinstellungen des Roboterarms bzw. gibt sie aus.
Arc	Führt eine kreisinterpolierte Bewegung aus.
Arc3	Bewegt den Arm kreisinterpoliert in 3 Dimensionen.
Move	Führt eine linearinterpolierte Roboterbewegung aus.
SpeedS	Definiert die Geschwindigkeit für interpolierte Befehle bzw. gibt sie aus.
AccelS	Definiert die Einstellungen für Beschleunigung und Verzögerung für interpolierte Bewegungen bzw. gibt sie aus.
SpeedR	Definiert die Geschwindigkeit für die Werkzeugrotation bzw. gibt sie aus.
AccelR	Definiert die Beschleunigung und Verzögerung für Werkzeugrotation bzw. gibt sie aus.
Home	Bewegt den Roboter in die benutzerdefinierte Home-Position.
HomeClr	Löscht die Definition der Home-Position.
HomeDef	Gibt den Status der Definition der Home-Position aus.
HomeSet	Definiert die benutzerdefinierte Home-Position.
Hordr	Definiert die Reihenfolge der Bewegungen bei Ausführung des Home-Befehls.
InPos	Überprüft, ob sich der Roboter in Position befindet (bewegungslos).
CurPos	Gibt während der Bewegung die aktuelle Position aus.

Pallet	Definiert eine Palette oder gibt einen Palettenpunkt aus.
Fine	Definiert die Grenzwerte für Positionierungsfehler.
QP	Definiert den Quick-Pause-Status bzw. gibt ihn aus.
JRange	Definiert die Arbeitsbereichsgrenzen für eine Achse bzw. gibt sie aus.
Range	Definiert die Grenzwerte für alle Achsen.
XYLim	Definiert die kartesischen Grenzwerte des Roboter-Arbeitsbereichs bzw. gibt sie aus.
PTPBoost	Definiert die Verstärkungswerte für eine kurze PTP-Bewegung bzw. gibt sie aus.
CX	Definiert den Koordinatenwert der X-Achse eines Punktes bzw. gibt ihn aus.
CY	Definiert den Koordinatenwert der Y-Achse eines Punktes bzw. gibt ihn aus.
CZ	Definiert den Koordinatenwert der Z-Achse eines Punktes bzw. gibt ihn aus.
CU	Definiert den Koordinatenwert der U-Achse eines Punktes bzw. gibt ihn aus.
CV	Definiert den Koordinatenwert der V-Achse eines Punktes bzw. gibt ihn aus.
CW	Definiert den Koordinatenwert der W-Achse eines Punktes bzw. gibt ihn aus.
Pls	Gibt den Pulse-Wert einer Achse aus.
Agl	Gibt den Achswinkel der aktuellen Position aus.
PAgl	Gibt den Achswert eines bestimmten Punktes aus.
PPls	Gibt die Pulse-Position einer bestimmten Achse für einen angebenen Punkt aus.

Eingangs- / Ausgangs-Befehle

On	Schaltet einen Ausgang ein.
Off	Schaltet einen Ausgang aus.
Oport	Liest den Status eines Ausgangs.
Sw	Gibt den Status eines Eingangs aus.
In	Liest 8 Eingänge.
InW	Gibt den Status des angegebenen Eingangs-Wortport aus.
InBCD	Liest 8 Eingänge als BCD-Code (Binär-Dezimal-Code).
Out	Setzt 8 Ausgänge bzw. gibt deren Status aus.
OutW	Setzt 16 Ausgänge gleichzeitig.
OpBCD	Definiert 8 Ausgänge gleichzeitig unter Verwednung des BCD-Formats.
MemOn	Schaltet einen Merker ein.
MemOff	Schaltet einen Merker aus.
MemSw	Gibt den Status eines Merkers aus.
MemIn	Liest 8 Merker.
MemOut	Setzt 8 Merker bzw. gibt deren Status aus.
Wait	Wartet auf eine Bedingung oder Zeit.
TMOut	Definiert das Zeitlimit für die Wait-Anweisung.

Tw	Gibt den Status der Wait-Bedingung und des Wait-Timer-Intervalls aus.
Input	Eingabe einer oder mehrerer Variablen über das aktuelle Anzeigefenster.
Print	Zeigt Daten im aktuellen Anzeigefenster an.
Line Input	Dateneingabe einer Zeichenkette im aktuellen Anzeigefenster.
Input #	Eingabe einer oder mehrerer Variablen aus einem Kommunikationsport.
Print #	Datenausgabe an Kommunikationsport.
Line Input #	Eingabe einer Zeichenkette aus einem Kommunikationsport.
Lof	Gibt die Anzahl von Zeilen in einem Kommunikationspuffer aus.

Befehle zur Punkteverwaltung

ClearPoints	Löscht alle Punktedaten aus dem Speicher.
LoadPoints	Lädt alle Punktedaten aus einer Datei im Speicher.
SavePoints	Speichert alle Punktedaten in einer Datei im Speicher.

Befehle zur Änderung von Koordinaten

Arm	Definiert den aktuellen Arm bzw. gibt ihn aus.
ArmSet	Definiert einen Arm.
ArmDef	Gibt den Status einer Armdefinition aus.
ArmClr	Löscht eine Armdefinition.
Tool	Definiert die aktuelle Werkzeugnummer bzw. gibt sie aus.
TLSet	Definiert ein Werkzeug-Koordinatensystem oder zeigt es an.
TLDef	Gibt den Status einer Werkzeugdefinition aus.
TLClr	Löscht eine Werkzeugdefinition.
ECP	Definiert die aktuelle ECP-Nummer bzw. gibt sie aus.
ECPSet	Definiert einen externen Kontrollpunkt oder zeigt ihn an.
ECPDef	Gibt den Status einer ECP-Definition aus.
ECPClr	Löscht eine ECP-Definition.
Base	Definiert das Basiskoordinatensystem und zeigt es an.
Local	Definiert ein lokales Werkzeug-Koordinatensystem.
LocalDef	Gibt den Status einer lokalen Definition aus.
LocalClr	Löscht ein lokales Koordinatensystem (macht Definitionen rückgängig).
Elbow	Stellt die Ellenbogenausrichtung eines Punktes ein bzw. gibt sie aus.
Hand	Stellt die Armausrichtung eines Punktes ein bzw. gibt sie aus.

Wrist	Stellt die Handgelenkausrichtung eines Punktes ein bzw. gibt sie aus.
J4Flag	Definiert die J4Flag-Einstellung eines Punktes bzw. gibt sie aus.
J6Flag	Definiert die J6Flag-Einstellung eines Punktes bzw. gibt sie aus.

Befehle zur Programmsteuerung

Function	Deklariert eine Funktion.
For...Next	Führt eine oder mehrere Anweisungen bis zum angegebenen Zählwert aus.
GoSub	Führt ein Unterprogramm aus.
Return	Rückkehr zum Hauptprogramm nach Ausführung eines Unterprogramms.
GoTo	Verzweigt ohne Bedingung zu einer Zeilennummer oder zu einem Label.
Call	Ruft eine Anwenderfunktion auf.
If..Then..Else..EndIf	Führt eine Anweisung bedingt aus.
Else	Wird mit der *If*-Anweisung verwendet, damit Anweisungen auch dann ausgeführt werden können, wenn die *If*-Anweisung falsch ist. Else ist eine Option für die If/Then-Anweisung.
Select ... Send	Führt eine von mehreren Anweisungsgruppen je nach Ausdruckswert aus.
Do...Loop	Do...Loop-Konstrukt.
Trap	Spezifiziert eine Trap-Behandlungsroutine.
OnErr	Definiert eine Fehlerbehandlungsroutine.
Era	Gibt die Roboterachsnummer für den zuletzt aufgetretenen Fehler aus.
Erf$	Gibt den Funktionsnamen für den zuletzt aufgetretenen Fehler aus.
Erl	Gibt die Zeilennummer des Fehlers aus.
Err	Gibt die Fehlernummer aus.
Ert	Gibt die Tasknummer des Fehlers aus.
ErrMsg$	Gibt die Fehlermeldung aus.
Signal	Sendet ein Signal an die Tasks, die WaitSig ausführen.
SyncLock	Synchronisiert Tasks mithilfe eines Mutex-Locks.
SynUnlock	Gibt eine Sync-ID frei, die vorher unter SyncLock gesperrt wurde.
WaitSig	Wartet auf ein Signal von einem anderen Task.

Befehle zur Programmausführung

Xqt	Führt einen Task aus.
Pause	Unterbricht kurzfristig die Programmausführung bei all den Tasks, bei denen der Pause-Befehl aktiviert ist.
Halt	Unterbricht einen Task.
Quit	Beendet einen Task.
Resume	Setzt einen im Halt-Status befindlichen Task fort.
MyTask	Gibt den aktuellen Task aus.

Pseudo-Anweisungen

#define	Definiert ein Makro.
#ifdef ... #endif	Bedingte Kompilierung.
#ifndef ... #endif	Bedingte Kompilierung.
#include	Bezieht eine Datei ein.

Befehle bezüglich numerischer Werte

Oport	Liest den Status eines Ausgangs.
Sw	Gibt den Status eines Eingangs aus.
In	Liest 8 Eingänge.
InBCD	Liest 8 Eingänge als BCD-Code (Binär-Dezimal-Code).
Ctr	Gibt den Zählerstand eines Zählers aus.
CTReset	Setzt einen Zähler zurück.
Tmr	Gibt den Zählerstand eines Timers aus.
TmReset	Setzt einen Timer auf 0 zurück.
Time	Stellt die Systemzeit ein.
Js	Gibt den Status einer Sense-Funktion aus.
CX	Definiert den Koordinatenwert der X-Achse eines Punktes bzw. gibt ihn aus.
CY	Definiert den Koordinatenwert der Y-Achse eines Punktes bzw. gibt ihn aus.
CZ	Definiert den Koordinatenwert der Z-Achse eines Punktes bzw. gibt ihn aus.
CU	Definiert den Koordinatenwert der U-Achse eines Punktes bzw. gibt ihn aus.
CV	Definiert den Koordinatenwert der V-Achse eines Punktes bzw. gibt ihn aus.
CW	Definiert den Koordinatenwert der W-Achse eines Punktes bzw. gibt ihn aus.
Pls	Gibt den Pulse-Wert einer Achse aus.
Agl	Gibt den Achswinkel der aktuellen Position aus.
Era	Gibt die Roboterachsnummer für den zuletzt aufgetretenen Fehler aus.
Erf$	Gibt den Funktionsnamen für den zuletzt aufgetretenen Fehler aus.
Erl	Gibt die Zeilennummerr für den zuletzt aufgetretenen Fehler aus.
Err	Gibt die Fehlernummer für den zuletzt aufgetretenen Fehler aus.
Ert	Gibt die Tasknummer für den zuletzt aufgetretenen Fehler aus.
TW	Gibt den Status der Wait-Bedingung und des Wait-Timer-Intervalls aus.
MyTask	Gibt den aktuellen Task aus.
Stat	Gibt den Status der Robotersteuerung aus.

Sin	Gibt den Sinuswert eines Winkels aus.
Cos	Gibt den Kosinuswert eines Winkels aus.
Tan	Gibt den Tangenswert eines Winkels aus.
Acos	Gibt den Arkuskosinus aus.
Asin	Gibt den Arkussinus aus.
Atan	Gibt den Arkustangens aus.
Atan2	Gibt den Arkustangens auf der Basis der X,Y –Position aus.
Sqr	Gibt die Quadratwurzel einer Zahl aus.
Abs	Gibt den Absolutwert einer Zahl aus.
Sgn	Gibt das Vorzeichen einer Zahl aus.
Int	Konvertiert eine relle Zahl in ein Integer.
Not	Not-Operator.
LShift	Verschiebt Bits nach links.
RShift	Verschiebt Bits nach rechts.

Zeichenketten-Befehle

String	Deklariert Zeichenkettenvariablen.
Asc	Gibt den ASCII-Wert eines Zeichens aus.
Chr$	Gibt das Zeichen eines numerischen ASCII-Wertes aus.
Left$	Gibt eine Teilkette von der linken Seite einer Zeichenkette aus.
Mid$	Gibt eine Teilkette aus.
Right$	Gibt eine Teilkette von der rechten Seite einer Zeichenkette aus.
Len	Gibt die Länge einer Zeichenkette aus.
LSet$	Gibt eine Zeichenkette mit abschließenden Leerzeichen aus.
RSet$	Gibt eine Zeichenkette mit führenden Leerzeichen aus.
Space$	Gibt eine Zeichenkette aus, die Leerzeichen enthält.
Str$	Konvertiert eine Zahl in eine Zeichenkette.
Val	Konvertiert eine numerische Zeichenkette in eine Zahl.
ErrMsg$	Gibt die Fehlermeldung aus.
LCase$	Konvertiert eine Zeichenkette in Kleinbuchstaben.
UCase$	Konvertiert eine Zeichenkette in Großbuchstaben.
LTrim$	Entfernt Leerzeichen vom Anfang der Zeichenkette.
RTrim$	Entfernt Leerzeichen vom Ende der Zeichenkette.
Trim$	Entfernt Leerzeichen vom Anfang und vom Ende der Zeichenkette.
ParseStr	Analysiert eine Zeichenkette syntaktisch und gibt eine Textelement-Matrix aus.
FmtStr$	Formatiert eine Zahl oder Zeichenkette.

Logische Operatoren

And	Führt einen logischen und bitweisen UND-Vorgang durch.
Or	ODER-Operator.
LShift	Verschiebt Bits nach links.
Mod	Modulo-Operator.
Not	Nicht-Operator.
RShift	Verschiebt Bits nach rechts.
Xor	EXKLUSIV-ODER-Operator.
Mask	Führt einen bitweisen UND-Vorgang in Wait-Anweisungen durch.

Befehle für Variablen

Boolean	Deklariert die Booleschen Variablen.
Byte	Deklariert Byte-Variablen.
Double	Deklariert doppelte Variablen.
Global	Deklariert globale Variablen.
Integer	Deklariert Integer-Variablen.
Long	Deklariert Long Integer Variablen.
Real	Deklariert Real-Variablen.
String	Deklariert Zeichenkettenvariablen.

Mit VB-Guide verwendete Befehle

SPELCom_Event	Löst ein Event im SpelNetLib-Client aus.

SPEL+ Sprachreferenz

Die Beschreibung der SPEL+-Befehle ist wie folgt strukturiert:

Syntax	Die Syntax beschreibt das Format, das für jeden Befehl verwendet wird. Für einige Befehle wird mehr als eine Syntax aufgezeigt, zusammen mit einer Zahl, auf die in der Befehlsbeschreibung Bezug genommen wird. Parameter werden in Kursivschrift dargestellt.
Parameter	Beschreibt jeden in diesem Befehl enthaltenen Parameter.
Rückgabewerte	Beschreibt alle Werte, die auf den Befehl hin ausgegeben werden.
Beschreibung	Beschreibt Details hinsichtlich der Funktionsweise des Befehls.
Hinweise	Gibt zusätzliche Informationen zu diesem Befehl.
Verwandte Befehle	Zeigt andere, mit diesem Befehl zusammenhängende Befehle. Die Seitenangabe dieser verwandten Befehle finden Sie im Inhaltsverzeichnis.
Beispiel	Gibt eines oder mehrere Beispiele für die Verwendung des Befehls.

SYMBOLE

Dieses Handbuch verwendet die folgenden Symbole, um zu zeigen, in welchem Kontext dieser Befehl verwendet werden kann:

>	Kann vom Befehlseingabefenster aus verwendet werden.
S	Kann als Anweisung in einem SPEL+-Programm verwendet werden.
F	Kann als Funktion in einem SPEL+-Programm verwendet werden.

!...! Parallelbearbeitung

Verarbeitet Ein- / Ausgangsanweisungen parallel zur Ausführung von Bewegungsbefehlen.

> S

Syntax

motion cmd **!***statements* **!**

Parameter

motion cmd	Jeder gültige Bewegungsbefehl der folgenden Liste: Arc, Arc3, Go, Jump, Jump3, Jump3CP, Move.
statements	Alle gültigen Parallelbearbeitungs-E/A-Anweisung(en), die während der Bewegung ausgeführt werden kann / können. (Siehe unten stehende Tabelle.)

Beschreibung

Parallelbearbeitungsbefehle sind Bewegungsbefehlen angegliedert, damit E/A-Anweisungen gleichzeitig mit dem Beginn der Verfahrbewegung ausgeführt werden können. Dies bedeutet, dass E/A-Anweisungen ausgeführt werden können, *während* sich der Arm bewegt, anstatt zu warten, bis der Arm eine Stop-Position erreicht hat und E/A-Anweisungen *dann* auszuführen. Außerdem kann definiert werden, zu welchem Zeitpunkt innerhalb der Bewegung die Ausführung von E/A-Anweisungen beginnen sollte. (Siehe auch die in der unten stehenden Tabelle beschriebenen Dn-Parameter.)

Die folgende Tabelle zeigt alle gültigen Parallelbearbeitungsbefehle. Jeder dieser Befehle kann als Einzelbefehl oder zusammen in einer Guppe verwendet werden, damit während einer Bewegungsanweisung mehrere E/A-Anweisungen ausgeführt werden können.

D*n*	Legt den %-Verfahrweg vor Ausführung der nächsten Parallelanweisung fest. *n* ist ein Prozentsatz zwischen 0 und 100, der die Position innerhalb der Bewegung festlegt, an der die parallel zu bearbeitende Anweisung anfangen soll. Anweisungen, die den Dn-Parametern folgen, beginnen mit der Ausführung, wenn n % der Verfahrbewegung zurückgelegt worden sind. Bei den Befehlen Jump, Jump3 und Jump3CP enthält der %-Verfahrweg nicht die Depart- und die Approach-Bewegung. Um Anweisungen auszuführen, nachdem die Depart-Bewegung beendet ist, stellen Sie ein D0 (null) an den Beginn der Anweisung. Dn kann in einer Parallelbearbeitungsanweisung maximal 16 Mal erscheinen.
On / Off *n*	Schaltet Ausgangnummer *n* ein bzw. aus.
MemOn / MemOff *n*	Schaltet Merkernumer *n* ein bzw. aus.
Out *p,d*	Gibt Ausgabedaten *d* an Ausgangsport *p* aus.
MemOut *p, d*	Gibt Ausgabedaten *d* an Merkerport *p* aus.
Signal *s*	Generiert ein Synchronisierungssignal.
Wait *t*	Verzögert die Ausführung der nächsten Parallelbearbeitungsanweisung um *t* Sekunden.
WaitSig *s*	Wartet auf das Signal *s,* um die nächste Anweisung auszuführen.
Wait Sw(*n*) = *j*	Verzögert die Ausführung der nächsten Parallelbearbeitungsanweisung, bis der Eingang n gleich der durch j definierten Bedingung ist. (Ein oder Aus)
Wait MemSw(*n*) = *j*	Verzögert die Ausführung der nächsten Parallelbearbeitungsanweisung, bis der Merker n gleich der durch j definierten Bedingung ist. (Ein oder Aus)
Print / Input	Gibt Daten an das Anzeigegerät aus bzw. von dort ein.

Print # / Input #	Gibt Daten an den Kommunikationsport aus bzw. von dort ein.

Hinweise

Bewegung ist beendet, bevor alle E/A-Befehle ausgeführt sind

Falls die Ausführung der Parallelbearbeitungsanweisungen nach Ausführung der Bewegungsbefehle noch nicht abgeschlossen ist, wird die nachfolgende Programmausführung solange verschoben, bis alle Parallelbearbeitungsanweisungen vollständig ausgeführt wurden. Diese Situation tritt mit hoher Wahrscheinlichkeit bei kurzen Bewegungen auf, bei denen parallel viele E/A-Anweisungen ausgeführt werden.

Was geschieht, wenn die Till-Anweisung verwendet wird, um den Arm anzuhalten, bevor die beabsichtigte Bewegung vollständig ausgeführt wurde?

Wenn Till verwendet wird, um den Arm an einer Zwischenposition des Verfahrweges anzuhalten, wird die weitere Programmausführung solange verzögert, bis alle Parallelbearbeitungsanweisungen vollständig ausgeführt wurden.

Wenn n nahe 100 % angegeben wird, wird die CP-Bewegung möglicherweise verzögert.

Wenn während einer CP-Bewegung ein hoher n-Wert verwendet wird, bringt der Roboter die aktuelle Bewegung möglicherweise verzögert zuende. Das liegt daran, dass die angegebene Position normalerweise während der Verzögerung durchgeführt werden würde, wenn CP nicht verwendet wird. Um eine Verzögerung zu vermeiden, können Sie die Bearbeitungsanweisung hinter den Bewegungsbefehl stellen. Im folgenden Beispiel wurde die On 1-Anweisung von der Parallelbearbeitung während der Bewegung zu P1 hinter die Bewegung verschoben.

```
CP On
Jump P1 !D96; On 1!
Go P2

CP On
Jump P1
On 1
Go P2
```

Jump-Anweisung und Parallelbearbeitung

Sollten Parallelbearbeitungsanweisungen ausgeführt werden, die zusammen mit dem Jump-Befehl verwendet werden, so beginnen diese, nachdem die Aufwärtsbewegung vollständig ausgeführt wurde und enden bei Beginn der Abwärtsbewegung.

Verwandte Befehle

Arc, Arc3, Go, Jump, Jump3, Jump3CP, Move, Pulse

!...! Beispiel einer Parallelbearbeitung

Das folgende Beispiel zeigt unterschiedliche Möglichkeiten, das Parallelbearbeitungsfeature im Zusammenhang mit Bewegungsbefehlen zu verwemden:

Die Parallelbearbeitung zusammen mit dem Jump-Befehl schaltet den Ausgang 1 am Ende der Aufwärtsbewegung der Z-Achse dann ein, wenn sich die erste, zweite und vierte Achse zu bewegen beginnen. Der Ausgang 1 wird wieder ausgeschaltet, wenn 50 % der Jump-Verfahrbewegung ausgeführt worden sind.

```
Function test
   Jump P1 !D0; On 1; D50; Off 1!
Fend
```

Die Parallelbearbeitung zusammen mit dem Move-Befehl schaltet den Ausgang 5 ein, wenn die Achsen 10 % ihrer Bewegung zu Punkt 1 ausgeführt haben. 0,5 Sekunden später wird der Ausgang 5 ausgeschaltet.

```
Function test2
   Move P1 !D10; On 5; Wait 0.5; Off 5!
Fend
```

#define

S

Definiert eine ID-Zeichenkette, die durch eine festgelegte Ersatzzeichenkette ausgetauscht werden soll.

Syntax

#define *identifier* [(*parameter*, [*parameter*])] *string*

Parameter

identifier Vom Benutzer definiertes Schlüsselwort, das eine Abkürzung des *Zeichenketten*-Parameters ist. Die Regeln für Zeichenketten lauten wie folgt:

- Das erste Zeichen muss ein Alphabetzeichen, die übrigen können alphanumerische Zeichen oder Unterstriche (_) sein.
- Leerzeichen oder Tabulatoren sind als Teil der *ID-Zeichenkette* nicht zulässig.

parameter Wird normalerweise verwendet, um eine Variabel (oder mehrere Variablen) anzugeben, die möglicherweise von der Ersatzzeichenkette verwendet werden. Dies sorgt für einen dynamischen define-Mechanismus, der wie ein Makro verwendet werden kann. Für den #define-Befehl dürfen maximal 8 Parameter verwendet werden. Die einzelnen Parameter müssen durch Kommata voneinander getrennt werden und die Parameterliste muss in Klammern stehen.

string Dies ist die Ersatzzeichenkette, die die ID-Zeichenkette ersetzt, wenn das Programm kompiliert wird. Die Regeln für Ersatzzeichenketten lauten wie folgt:

- Leerzeichen oder Tabulatoren sind als Teil einer Ersatzzeichenkette nicht zulässig.
- ID-Zeichenketten, die mit anderen #define-Anweisungen verwendet werden, können nicht als Ersatzzeichenketten genutzt werden.
- Wird das Kommentarsymbol (') ebenfalls verwendet, werden die Zeichen nach dem Kommentarsymbol als Kommentar interpretiert und nicht in die Ersatzzeichenkette einbezogen.
- Die Ersatzzeichenkette ist nicht unbedingt erforderlich. In diesem Falle wird die angegebene Zeichenkette durch „nothing“ oder eine Null-Zeichenkette ersetzt. Dies löscht die ID-Zeichenkette aus dem Programm.

Beschreibung

Der #define-Befehl ersetzt innerhalb eines Programms eine bestimmte ID-Zeichenkette durch eine Ersatzzeichenkette. Jedes Mal, wenn diese ID-Zeichenkette gefunden wird, wird sie vor dem Kompilieren durch die Ersatzzeichenkette ersetzt. Jedoch verbleibt nicht die Ersatzzeichenkette, sondern die ID-Zeichenkette im Quellcode. Dies macht den Code vielfach leichter lesbar, weil an Stelle schwer lesbarer Code-Zeichenketten aussagekräftige Namen für ID-Zeichenketten verwendet werden.

Die bestimmte ID-Zeichenkette kann für eine bedingte Kompilierung verwendet werden, indem sie mit den Befehlen #ifdef oder #ifndef kombiniert wird.

Wenn ein Parameter angegeben ist, kann die neue ID-Zeichenkette als Makro verwendet werden.

Hinweise

Verwenden von #define für die Variablendeklaration oder Labelersetzungen verursache einen Fehler:

Bitte beachten Sie, dass die Verwendung des #define-Befehls für die Variablendeklaration einen Fehler verursacht.

Verwandte Befehle

#ifdef
#ifndef

Beispiel für #define

```
' Kommentarzeichen in der nächsten Zeile für den Debug-Modus entfernen.
' #define DEBUG

Input #1, A$
#ifdef DEBUG
    Print "A$ = ", A$
#endif
Print "The End"

#define SHOWVAL(x) Print "var = ", x

Integer a

a = 25

SHOWVAL(a)
```

#ifdef...#else...#endif

Bedingte Kompilierung. **S**

Syntax

#ifdef *[ID-Zeichenkette]*
... geben Sie den zur bedingten Kompilierung ausgewählten Quellcode hier an.
[**#else**
... geben Sie den ausgewählten Quellcode für die falsche Bedingung hier an.]
#endif

Parameter

identifier Benutzerdefiniertes Schlüsselwort, das, wird es definiert, die Kompilierung des Quellcodes zwischen #ifdef und #else oder #endif ermöglicht. Die ID-Zeichenkette ist somit die Bedingung für die bedingte Kompilierung.

Beschreibung

#ifdef...#else...#endif ermöglicht das bedingte Kompilieren eines ausgewählten Quellcodes. Ob diese Kompilierung durchgeführt wird, hängt von der *ID-Zeichenkette* ab. #ifdef prüft zuerst, ob die angegebene ID-Zeichenkette derzeit von #define definiert wird. Die #else-Anweisung ist nicht unbedingt erforderlich.

Wenn die ID-Zeichenkette definiert wurde und die #else-Anweisung nicht genutzt wird, werden die Anweisungen zwischen #ifdef und #endif kompiliert. Andernfalls, also wenn die #else-Anweisung genutzt wird, werden die Anweisungen zwischen #ifdef und #else kompiliert.

Wenn nicht definiert und wenn die #else-Anweisung nicht genutzt wird, werden die Anweisungen zwischen #ifdef und #endif ohne Kompilierung übersprungen. Andernfalls, also wenn die #else-Anweisung genutzt wird, werden die Anweisungen zwischen #else und #endif kompiliert.

Verwandte Befehle

#define, #ifndef

Beispiel für #ifdef

Ein Teil eines Codes aus einem Beispielprogramm, das #ifdef verwendet, ist unten dargestellt. Im unten stehenden Beispiel erfolgt ein Ausdruck des A$-Variablenwertes, abhängig davon, ob eine Definition des #define DEBUG Pseudo-Befehls vorliegt oder nicht. Wenn der #define DEBUG Pseudo-Befehl in dieser Quelle bereits verwendet wurde, wird die Zeile ‚Print A$' kompiliert und später ausgeführt, wenn das Programm läuft. Der Ausdruck der Zeichenkette "The End" erfolgt trotz des #define DEBUG Pseudo-Befehls.

```
' Kommentarzeichen in der nächsten Zeile für den Debug-Modus entfernen.
' #define DEBUG

Input #1, A$
#ifdef DEBUG
    Print "A$ = ", A$
#endif
Print "The End"
```

#ifndef...#endif

Bedingte Kompilierung.

S

Syntax

#ifndef *[ID-Zeichenkette]*
…geben Sie den zur bedingten Kompilierung ausgewählten Quellcode hier an.
[**#else**
…geben Sie den ausgewählten Quellcode für die wahre Bedingung hier an.]
#endif

Parameter

identifier Benutzerdefiniertes Schlüsselwort, das, wenn es **nicht** definiert ist, die Kompilierung des Quellcodes zwischen #ifndef und #else oder #endif ermöglicht. Die ID-Zeichenkette ist somit die Bedingung für die bedingte Kompilierung.

Beschreibung

Dieser Befehl wird "if not defined" (“falls nicht definiert”) –Befehl genannt. #ifndef...#else...#endif ermöglicht das bedingte Kompilieren eines ausgewählten Quellcodes. Die #else-Anweisung ist nicht unbedingt erforderlich.

Wenn die ID-Zeichenkette definiert ist und wenn die #else-Anweisung nicht genutzt wird, werden die Anweisungen zwischen #ifndef und #endif nicht kompiliert. Andernfalls, also wenn die #else-Anweisung genutzt wird, werden die Anweisungen zwischen #else und #endif kompiliert.

Wenn die ID-Zeichenkette nicht definiert ist und wenn die #else-Anweisung nicht genutzt wird, werden die Anweisungen zwischen #ifndef und #endif kompiliert. Andernfalls, also wenn die #else-Anweisung genutzt wird, werden die Anweisungen zwischen #else und #endif nicht kompiliert.

Hinweise

Der Unterschied zwischen #ifdef und #ifndef:

Der grundlegende Unterschied zwischen #ifdef und #ifndef ist, dass der #ifdef-Befehl den angegebenen Quellcode kompiliert, wenn die ID-Zeichenkette **definiert ist**. Der #ifndef-Befehl kompiliert den angegebenen Quellcode, wenn die ID-Zeichenkette **NICHT definiert ist**.

Verwandte Befehle

#define, #ifdef

Beispiel für #ifndef

Ein Teil eines Codes aus einem Beispielprogramm, das #ifndef verwendet, ist unten dargestellt. Im unten stehenden Beispiel erfolgt ein Ausdruck des A$-Variablenwertes, abhängig davon, ob eine Definition des #define NODELAY Pseudo-Befehls vorliegt oder nicht. Wenn der #define NODELAY Pseudo-Befehl in dieser Quelle bereits verwendet wurde, wird die Zeile 'Wait 1' **NICHT** zusammen mit dem Rest der Quelle für dieses Programm **kompiliert**, wenn dieses kompiliert wird, (d. h. zur Ausführung vorgelegt). Wenn der #define NODELAY Pseudo-Befehl in dieser Quelle noch nicht verwendet wurde (d. h. NODELAY ist nicht definiert), **wird die Zeile ‚Wait 1' kompiliert** und später ausgeführt, wenn das Programm läuft. Der Ausdruck der Zeichenkette "The End" erfolgt trotz des #define NODELAY Pseudo-Befehls.

```
' Kommentiert die nächste Zeile aus, um Verzögerungen zu erzwingen.
#define NODELAY 1

Input #1, A$
#ifndef NODELAY
    Wait 1
#endif
Print "The End"
```

#include

Bezieht die angegebene Datei in die Datei mit ein, die die #include-Anweisung verwendet. **S**

Syntax

#include "*fileName*.**INC"**

Parameter

fileName Der Dateiname muss der Name einer Include-Datei im aktuellen Projekt sein. Alle Include-Dateien haben eine **INC**-Erweiterung. Der Dateiname spezifiziert die Datei, die in die aktuelle Datei einbezogen wird.

Beschreibung

#include fügt den Inhalt der spezifizierten Include-Datei in die aktuelle Datei ein, in der die #include-Anweisung verwendet wird.

Normalerweise enthalten Include-Dateien #define-Anweisungen.

Eine #include-Anweisung muss außerhalb jeglicher Funktionsdefinition verwendet werden.

Eine Include-Datei kann eine sekundäre Include-Datei beinhalten. Beispielsweise kann FILE2 in FILE1 enthalten sein und FILE3 in FILE2. Dieser Vorgang wird als Verschachtelung bezeichnet.

Verwandte Befehle

#define, #ifdef, #ifndef

Beispiel für #include

Include-Datei (Defs.inc)

```
#define DEBUG 1
#define MAX_PART_COUNT 20
```

Programmdatei (main.prg)

```
#include "defs.inc"

Function  main
    Integer i

    Integer Parts(MAX_PART_COUNT)

Fend
```

#undef

Macht die Defintion einer ID-Zeichenkette rückgängig, die vorher durch #define definiert wurde. **S**

Syntax

#undef *identifier*

Parameter

identifier Schlüsselwort aus einer vorherigen #define-Anweisung.

Verwandte Befehle

#define, #ifdef, #ifndef

Abs-Funktion

Gibt den Absolutwert einer Zahl aus.

F

Syntax

Abs(*Zahl*)

Parameter

number Jeglicher gültige numerische Ausdruck.

Rückgabewerte

Der Absolutwert einer Zahl.

Beschreibung

Der Absolutwert einer Zahl ist der Wert ohne Vorzeichen. Beispielsweise ist der Absolutwert von **Abs**(-1) und **Abs**(1) in beiden Fällen 1.

Verwandte Befehle

Atan, Atan2, Cos, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Beispiel einer Abs-Funktion

Die folgenden Befehle werden mittels des Print-Befehls im Befehlseingabefenster eingegeben.

```
> print abs(1)
1
> print abs(-1)
1
> print abs(-3,54)
3.54
>
```

Accel-Anweisung

Stellt Beschleunigungs- und Verzögerungsrampen für die PTP-Bewegungsbefehle Go, Jump und Pulse ein bzw. zeigt sie an.

> S

Syntax

(1) Accel *accel*, *decel* [, *departAccel, departDecel, approAccel, approDecel*]
(2) Accel

Parameter

accel	Integer-Ausdruck zwischen 1-100, der für einen prozentualen Anteil der maximalen Beschleunigungsrampe steht.
decel	Integer-Ausdruck zwischen 1-100, der für einen prozentualen Anteil der maximalen Verzögerungsrampe steht.
departAccel	Optional. Depart-Beschleunigung für Jump. Gültige Einträge sind 1-100.
departDecel	Optional. Depart-Verzögerung für Jump. Gültige Einträge sind 1-100.
approAccel	Optional. Approach-Beschleunigung für Jump. Gültige Einträge sind 1-100.
approDecel	Optional. Approach-Verzögerung für Jump. Gültige Einträge sind 1-100.

Rückgabewerte

Wenn Parameter ausgelassen werden, werden die aktuellen Accel-Parameter angezeigt.

Beschreibung

Accel spezifiziert Beschleunigung und Verzögerung für alle Bewegungen des PTP-Typs. Dies bezieht auch Bewegungen ein, die durch die Roboter-Bewegungsbefehle Go, Jump und Pulse ausgelöst wurden.

Jeder durch den **Accel**-Befehl definierte Beschleunigungs- und Verzögerungsparameter kann ein Integer-Wert von 1-100 sein. Diese Zahl steht für einen Prozentwert der erlaubten maximalen Beschleunigung (oder Verzögerung).

Der Accel-Befehl kann verwendet werden, um neue Beschleunigungs- und Verzögerungswerte einzustellen, oder einfach, um die aktuellen Werte auszudrucken. Wenn der Accel-Befehl verwendet wird, um neue Beschleunigungs- und Verzögerungswerte einzustellen, sind die ersten 2 Parameter (*accel* und *decel*) im **Accel**-Befehl erforderlich.

Die optionalen Parameter *departAccel, departDecel, approAccel* und *approDecel* sind nur für den Jump-Befehl wirksam und spezifizieren Beschleunigungs- und Verzögerungswert für die Depart-Bewegung am Anfang des Jump-Befehls und die Approach-Bewegung am Ende des Jump-Befehls.

Der **Accel**-Wert wird auf die Standardwerte zurückgesetzt (niedrige Beschleunigung), wenn eine der folgenden Aktionen ausgeführt wird:

Controller Power On
Motor On
SFree, SLock
Reset
Stop-Button oder Tasten Strg + C

Hinweise

Ausführen des Accel-Befehls im Low-Power-Modus (Power Low):

Wird **Accel** ausgeführt, wenn sich der Roboter im Low-Power-Modus befindet (Power Low), werden die neuen Werte gespeichert, die aktuellen Werte jedoch auf niedrige Werte begrenzt.

Die aktuellen Beschleunigungswerte sind wirksam, wenn Power auf High eingestellt ist und der Teach-Modus ausgeschaltet ist.

Der Unterschied zwischen den Befehlen Accel und AccelS:

Es ist wichtig zu wissen, dass der **Accel-**Befehl nicht die Beschleunigungs- und Verzögerungswerte für geradlinige Bewegungen und Bogenbewegungen einstellt. Der Befehl **AccelS** wird hingegen verwendet, um Beschleunigungs- und Verzögerungsrampen für geradlinige Bewegungen und Bogenbewegungen einzustellen.

Verwandte Befehle

AccelR, AccelS, Go, Jump, Jump3, Power, Pulse, Speed, TGo

Beispiel einer Accel-Anweisung

Das folgende Beispiel zeigt ein einfaches Bewegungsprogramm, in dem Beschleunigung (**Accel**) und Geschwindigkeit (Speed) unter Zuhilfenahme vordefinierter Variablen eingestellt werden.

```
Function acctest
   Integer slow, accslow, decslow, fast, accfast, decfast

   slow = 20     'setzt die slow speed-Variable
   fast = 100    'setzt die high speed-Variable
   slow = 20     'setzt die slow acceleration-Variable
   slow = 20     'setzt die slow deceleration-Variable
   accfast = 100     'setzt die fast acceleration-Variable
   slow = 100     'setzt die fast deceleration-Variable

   Accel accslow, decslow
   Speed slow
   Jump pick
   On gripper
   Accel accfast, decfast
   Speed fast
   Jump place
   .
   .
   .
Fend
```

Beispiel 2

Wenn der Roboter im Low-Power-Modus (Power Low) ist und der Benutzer versucht, über das Befehlseingabefenster den Accel-Wert auf 100 einzustellen, wird automatisch ein maximaler Beschleunigungswert von 10 eingestellt, weil sich der Roboter im Low-Power-Modus befindet, . (Das System lässt im Low-Power-Modus keinen Beschleunigungswert >10 zu.)

```
>Accel 100,100
>
>Accel
Low-Power-Modus:
    100     100
    100     100
    100     100
```

Beispiel 3

Stellen Sie eine langsame abwärtsgerichtete Verzögerung der Z-Achse ein, um beim Verwenden des JUMP-Befehls ein sanftes Ablegen des Teils zu ermöglichen. Dies bedeutet, dass der *Zdnd*-Parameter niedrig gewählt werden muss, wenn die **Accel**–Werte eingestellt werden.

```
>Accel 100,100,100,100,100,35

>Accel
    100     100
    100     100
    100     35
>
```

Accel-Funktion

> F

Gibt den spezifizierten Beschleunigungswert aus.

Syntax

Accel(*Parameternummer*)

Parameter

paramNumber Integer-Ausdruck, der die folgenden Werte haben kann:

1: Angegebener Wert der Beschleunigung

2: Angegebener Wert der Verzögerung

3: Angegebener Depart-Beschleunigungswert für Jump

4: Angegebener Depart-Verzögerungswert für Jump

5: Angegebener Approach-Beschleunigungswert für Jump

6: Angegebener Approach-Verzögerungswert für Jump

Rückgabewerte

Integer von 1-100 %

Verwandte Befehle

Accel-Anweisung

Beispiel einer Accel-Funktion

In diesem Beispiel wird die **Accel**-Funktion in einem Programm verwendet:

```
Integer currAccel, currDecel

' Holt die aktuellen Werte für accel und decel
currAccel = Accel(1)
currDecel = Accel(2)
Accel 50, 50
Jump pick
' Stellt vorherige Einstellungen wieder her
Accel currAccel, currDecel
```

AccelR-Anweisung

> S

Stellt Beschleunigungs- und Verzögerungswerte für die Werkzeugrotationssteuerung der CP-Bewegung ein bzw. zeigt diese an.

Syntax

(1) AccelR *accel*, [*decel*]
(2) AccelR

Parameter

accel	Reeller Ausdruck in Grad/Sekunde².
decel	Reeller Ausdruck in Grad/Sekunde².

Rückgabewerte

Wenn Parameter ausgelassen werden, werden die aktuellen AccelR-Parameter angezeigt.

Beschreibung

AccelR wird wirksam, wenn die ROT-Bedingung in den Bewegungsbefehlen Move, Arc, Arc3, BMove, TMove und Jump3CP verwendet wird.

Der **AccelR**-Wert wird auf die Standardwerte zurückgesetzt, wenn eine der folgenden Aktionen ausgeführt wird:

Controller Power On
Motor On
SFree, SLock
Reset
Stop-Taste oder Strg + C

Verwandte Befehle

Arc, Arc3, BMove, Jump3CP, Power, SpeedR, TMove

Beispiel einer AccelR-Anweisung

```
AccelR 360, 200
```

AccelR-Funktion

Gibt den spezifizierten Beschleunigungswert der Werkzeugrotation aus.

> F

Syntax

AccelR(*Parameternummer*)

Parameter

paramNumber Integer-Ausdruck, der die folgenden Werte haben kann:

1: Angegebener Wert der Beschleunigung

2: Angegebener Wert der Verzögerung

Rückgabewerte

Reeller Wert in Grad/Sekunde².

Verwandte Befehle

AccelR-Anweisung

Beispiel einer AccelR-Funktion

```
Real currAccelR, currDecelR

' Holt die aktuellen Werte für accel und decel
currAccelR = AccelR(1)
currDecelR = AccelR(2)
```

AccelS-Anweisung

> S

Stellt Beschleunigungs- und Verzögerungsraten für geradlinige Bewegungen und CP-Roboter-Bewegungsbefehle, wie z. B. Move, Arc, Arc3, Jump3, etc. ein.

Syntax

(1) AccelS *accel*, [*decel*], [*departAccel*], [*departDecel*], [*approAccel*], [*approDecel*]
(2) AccelS

Parameter

accel	Reeller Ausdruck (6-Achsroboter: 1-25000, andere Roboter: 1-5000) angegeben in mm/s^2, um Beschleunigungs- und Verzögerungswerte für geradlinige Bewegungen und CP-Roboter-Bewegungsbefehle zu definieren. Wenn *decel* ausgelassen wird, wird *accel* verwendet, um sowohl Beschleunigungs- als auch Verzögerungsrampen zu spezifizieren.
decel	Optional. Reeller Ausdruck (6-Achsroboter: 1-25000, andere Roboter: 1-5000) angegeben in mm/sec^2, um den Verzögerungswert zu definieren.
departAccel	Optional. Reeller Ausdruck (6-Achsroboter: 1-50000, andere Roboter: 1-5000) für Depart-Beschleunigungswerte für Jump3, Jump3CP.
departDecel	Optional. Reeller Ausdruck (6-Achsroboter: 1-50000, andere Roboter: 1-5000) für Depart-Verzögerungswerte für Jump3, Jump3CP.
approAccel	Optional. Reeller Ausdruck (6-Achsroboter: 1-50000, andere Roboter: 1-5000) für Approach-Beschleunigungswerte für Jump3, Jump3CP.
approDecel	Optional. Reeller Ausdruck (6-Achsroboter: 1-50000, andere Roboter: 1-5000) für Approach-Verzögerungswerte für Jump3, Jump3CP.

Rückgabewerte

Zeigt Beschleunigungs- und Verzögerungswerte an, wenn sie ohne Parameter verwendet werden.

Beschreibung

AccelS spezifiziert Beschleunigung und Verzögerung für alle interpolierten Bewegungen, inklusive linearinterpolierter und kreisinterpolierter Bewegungen. Dies schließt die Bewegungen ein, die durch die Bewegungsbefehle Move und Arc initiiert wurden.

Der **AccelS**-Wert wird auf die Standardwerte zurückgesetzt, wenn eine der folgenden Aktionen ausgeführt wird:

Controller Power On
Motor On
SFree, SLock
Reset
Stop-Taste oder Strg + C

Hinweise

Ausführen des AccelS-Befehls im Low-Power-Modus (Power Low):

Wird **AccelS** ausgeführt, wenn sich der Roboter im Low-Power-Modus befindet (Power Low), werden die neuen Werte gespeichert, aber die aktuellen Werte auf niedrige Werte begrenzt.

Die aktuellen Beschleunigungswerte sind wirksam, wenn Power auf High eingestellt ist und der Teach-Modus ausgeschaltet ist.

Der Unterschied zwischen den Befehlen Accel und AccelS:

Es ist wichtig zu wissen, dass der **AccelS-**Befehl nicht die Beschleunigungs- und Verzögerungswerte für PTP-Bewegungen einstellt. (D. h. für Bewegungen, die durch die Befehle Go, Jump, und Pulse initiiert wurden.) Beschleunigungs- und Verzögerungswerte für PTP-Bewegungen werden mit dem Accel-Befehl eingestellt.

Verwandte Befehle

Accel, Arc, Arc3, Jump3, Jump3CP, Power, Move, TMove, SpeedS

Beispiel einer AccelS-Anweisung

Das folgende Beispiel zeigt ein einfaches Bewegungsprogramm, in dem die geradlinige Beschleunigung / CP-Beschleunigung (AccelS) und die geradlinige Geschwindigkeit / CP-Geschwindigkeit (SpeedS) unter Zuhilfenahme vordefinierter Variablen eingestellt werden.

```
Function acctest
   Integer slow, accslow, fast, accfast

   slow = 20     'setzt die slow speed-Variable
   fast = 100       'setzt die high speed-Variable
   slow = 200     'setzt die slow acceleration-Variable
   accfast = 5000     'setzt die fast acceleration-Variable
   AccelS accslow
   SpeedS slow
   Move P1
   On 1
   AccelS accfast
   SpeedS fast
   Jump P2
   .
   .
   .
Fend
```

Beispiel 2

Wenn der Roboter im Low-Power-Modus (Power Low) ist und der Benutzer versucht, über das Befehlseingabefenster den AccelS-Wert auf 1000 einzustellen, wird automatisch ein maximaler Beschleunigungswert von 200 eingestellt, weil sich der Roboter im Low-Power-Modus befindet, . (Das System lässt im Low-Power-Modus keinen Beschleunigungswert >200 zu.)

```
>AccelS 1000

>AccelS
Low-Power-Modus:
     1000.000    1000.000
>
```

AccelS-Funktion

Gibt Beschleunigung- und Verzögerung für CP-Bewegungsbefehle aus.

> F

Syntax

AccelS(*Parameternummer*)

Parameter

paramNumber Integer-Ausdruck, der die folgenden Werte haben kann:

1: Beschleunigungswert
2: Verzögerungswert
3: Depart-Beschleunigungswert für Jump3, Jump3CP
4: Depart-Verzögerungswert für Jump3, Jump3CP
5: Approach-Beschleunigungswert für Jump3, Jump3CP
6: Approach-Verzögerungswert für Jump3, Jump3CP

Rückgabewerte

Reeller Wert von 0 - 5000 mm/s/s

Verwandte Befehle

AccelS-Anweisung, Arc 3, SpeedS, Jump3, Jump3CP

Beispiel einer AccelS-Funktion

```
Real savAccelS
savAccelS = AccelS(1)
```

Acos-Funktion

F

Gibt den Arkuskosinus eines numerischen Ausdrucks aus.

Syntax

Acos(*number***)**

Parameter

number Numerischer Ausdruck, der für den Kosinus eines Winkels steht.

Rückgabewerte

Reeller Wert in Radianten, der für den Arkuskosinus des Parameters ‚*number'* steht.

Beschreibung

Acos gibt den Arkuskosinus des numerischen Ausdrucks aus. Der Wertebereich liegt zwischen -1 und 1. Der von **Acos** ausgegebene Wert liegt zwischen 0 und PI Radianten. Wenn die Zahl < -1 oder > 1 ist, tritt ein Fehler auf.

Verwenden Sie die RadToDeg-Funktion, um Radiantenwerte in Gradzahlen umzuwandeln.

Verwandte Befehle

Abs, Asin, Atan, Atan2, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Beispiel einer Acos-Funktion

```
Function acostest
   Double x

   x = Cos(DegToRad(30))
   Print "Acos of ", x, " is ", Acos(x)
Fend
```

Agl-Funktion

F

Gibt den Achswinkel für die ausgewählte Rotationsachse oder die Position für die ausgewählte Linearachse aus.

Syntax

Agl(*Achsnummer*)

Parameter

jointNumber	Integer-Ausdruck, der für die Achsnummer steht. Die Werte liegen zwischen 1 und der Anzahl von Roboterachsen.

Rückgabewerte

Der Achswinkel für die ausgewählte Rotationsachse oder die Position für die ausgewählten Linearachsen.

Beschreibung

Die **Agl**-Funktion wird verwendet, um den Achswinkel für die ausgewählte Rotationsachse oder die Position für die ausgewählte Linearachse zu erhalten.

Wenn die gewählte Achse eine Rotationsachse ist, gibt **Agl** den gegenwärtigen Winkel aus. Dieser wird in Grad (°) gemessen, von der Nullposition der ausgewählten Achse aus. Der ausgegebene Wert ist eine reelle Zahl.

Wenn die gewählte Achse eine Linearachse ist, gibt **Agl** die aktuelle Position in Millimetern (mm) aus, von der Nullposition der ausgewählten Achse aus gemessen. Der ausgegebene Wert ist eine reelle Zahl.

Wurde über die Arm-Anweisung ein zusätzlicher Arm ausgewählt, gibt **Agl** den Winkel (bzw. die Position) von der Null-Pulse-Position des Standardarms zum ausgewählten Arm aus.

Verwandte Befehle

PAgl, Pls, PPls

Beispiel einer Agl-Funktion

Die folgenden Befehle werden mittels des Print-Befehls im Befehlseingabefenster eingegeben.

```
> print agl(1), agl(2)
 17.234   85.355
```

AglToPls-Funktion

F

Wandelt Roboterwinkel in Pulse um.

Syntax

AglToPls(*j1*, *j2*, *j3*, *j4*, [*j5*], [*j6*]**)**

Parameter

j1 - j6 Reelle Ausdrücke, die für den Achswinkel stehen.

Rückgabewerte

Ein Roboterpunkt, dessen Ort durch die in Pulse umgewandelten Achswinkel bestimmt wird.

Beschreibung

Verwenden Sie AglToPls, um einen Punkt aus Achswinkeln zu erstellen.

Hinweis

Die Zuweisung zu einem Punkt kann dazu führen, dass ein Teil der Achsposition verloren geht.

In bestimmten Fällen, wenn das Ergebnis von **AglToPls** einer Punktedatenvariablen zugewiesen wird, bewegt sich der Arm in eine Achsposition, die von der durch **AglToPls** bestimmten Achsposition abweicht.

Zum Beispiel:

```
P1 = AglToPls(0, 0, 0, 90, 0, 0)
Go P1 ' Verfährt in die AglToPls (0, 0, 0, 0, 0, 90) Achsposition
```

Gleichermaßen wird, wenn eine AglToPls-Funktion als Parameter eines CP-Bewegungsbefehls verwendet wird, der Arm in eine andere Achsposition gebracht als die durch **AglToPls** spezifizierte.

```
Move AglToPls(0, 0, 0, 90, 0, 0) ' Verfährt in die AglToPls(0, 0, 0, 0,
0, 90) Achsposition
```

Wenn die **AglToPls**-Funktion als Parameter in einem PTP-Bewegungsbefehl verwendet wird, tritt dieses Problem nicht auf.

Verwandte Befehle

Agl, JA, Pls

Beispiel einer AglToPls-Funktion

```
Go AglToPls(0, 0, 0, 90, 0, 0)
```

And-Operator

Operator, der verwendet wird, um eine logische oder bitweise **And**-Verknüpfung von zwei Ausdrücken herzustellen.

Syntax

result = *expr1* **And** *expr2*

Parameter

expr1, expr2	Bei logischen And-Verknüpfungen jeglicher gültige Ausdruck, der ein Boolesches Ergebnis ausgibt. Bei bitweisen And-Verknüpfungen ein Integer-Asudruck.
result	Bei logischen And-Verknüpfungen ist das Ergebnis ein Boolean-Wert. Bei bitweisen And-Verknüpfungen ist das Ergebnis ein Integer.

Beschreibung

Eine logische And-Verknüpfung wird verwendet, um die Ergebnisse zweier oder mehrerer Ausdrücke zu einem Booleschen Ergebnis zu kombinieren. Die folgende Tabelle listet mögliche Kombinationen auf.

expr1 (Ausdruck 1)	***expr2 (Ausdruck 2)***	***result (Ergebnis)***
Wahr	Wahr	Wahr
Wahr	Falsch	Falsch
Falsch	Wahr	Falsch
Falsch	Falsch	Falsch

Eine bitweise And-Verknüpfung führt einen bitweisen Vergleich der identisch positionierten Bits in zwei numerischen Ausdrücken durch und stellt das entsprechende Bit gemäß der nachfolgenden Tabelle in *result* ein.

If-Bit in expr1 ist	**And-Bit in expr1 ist**	**Ergebnis**
0	0	0
0	1	0
1	0	0
1	1	1

Verwandte Befehle

LShift, Mask, Not, Or, RShift, Xor

Beispiel eines And-Operators

```
Function LogicalAnd(x As Integer, y As Integer)

  If x = 1 And y = 2 Then
    Print "The values are correct"
  EndIf
Fend

Function BitWiseAnd()

  If (Stat(0) And &H800000) = &H800000 Then
    Print "The enable switch is open"
  EndIf
Fend

>print 15 and 7
7
>
```

Arc- und Arc3-Anweisungen

> S

Arc bewegt den Arm mittels einer kreisinterpolierten Bewegung an den spezifizierten Punkt in der XY-Ebene.
Arc3 bewegt den Arm mittels einer kreisinterpolierten Bewegung in 3 Dimensionen an den spezifizierten Punkt in der XY-Ebene.

Syntax

(1) Arc *midPoint, endPoint* [**ROT**] [**CP**] [*searchExpr*] [*!...!*]
(2) Arc3 *midPoint, endPoint* [**ROT**] [**ECP**] [**CP**] [*searchExpr*] [*!...!*]

Parameter

midPoint	Punktausdruck. Der Mittelpunkt (vorher vom Benutzer geteacht), über den der Arm auf seinem Weg vom aktuellen Punkt zum *endPoint (Endpunkt)* hinweg verfährt.
endPoint)	Punktausdruck. Der Endpunkt (vorher vom Benutzer geteacht), zu dem der Arm während einer Bogenbewegung verfährt. Dies ist die Endposition am Ende der Bogenbewegung.
ROT	Optional. : Bestimmt Geschwindigkeit / Beschleunigung / Verzögerung zugunsten der Werkzeugrotation.
ECP	Optional. Externe Kontrollpunkt-Bewegung. Dieser Parameter ist gültig, wenn die ECP-Option aktiviert ist.
CP	Optional. Spezifiziert die CP-Bewegung.
searchExpr	Optional. Ein Till- oder Find-Ausdruck. **Till** | **Find** **Till Sw**(*expr*) = {**Ein** | **Aus**} **Find Sw**(*expr*) = {**Ein** | **Aus**}
!...!	Parallelbearbeitungsanweisungen dürfen zusammen mit der Arc-Anweisung verwendet werden. Sie sind optional. (Bitte lesen Sie die Beschreibung der Parallelbearbeitungsanweisungen für weitere Informationen.)

Beschreibung

Arc und **Arc3 werden** verwendet, um den Arm in einer Kreisbewegung von seiner gegenwärtigen Position über den *Mittelpunkt* zum *Endpunkt* zu verfahren. Das System berechnet automatisch eine auf drei Punkten basierende Kurve (aktuelle Position, *Endpunkt*, und *Mittelpunkt*) und bewegt den Arm an dieser Kurve entlang, bis der als *Endpunkt* definierte Punkt erreicht wird. Die Koordinaten von *Mittel-* und *Endpunkt* müssen geteacht werden, bevor der Befehl ausgeführt wird. Die Koordinaten können nicht im Befehl selber spezifiziert werden.

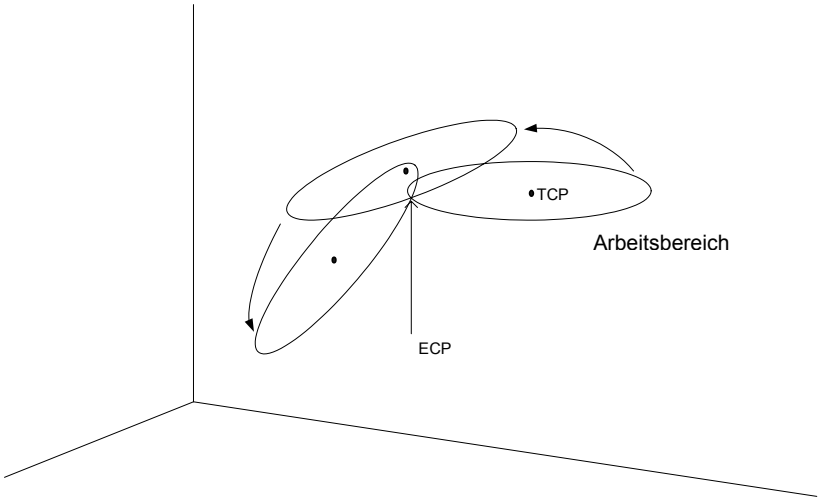
Arc und **Arc3** verwenden den Geschwindigkeitswert aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS. Lesen Sie den Abschnitt *Arc3 mit CP verwenden* für Informationen über die Beziehung Geschwindigkeit-Beschleunigung und Beschleunigung-Verzögerung. Wenn jedoch der ROT-Parameter verwendet wird, verwenden **Arc** und **Arc3** den Geschwindigkeitswert aus SpeedR und die Beschleunigungs- und Verzögerungswerte aus AccelR. In diesem Fall haben die Geschwindigkeitswerte aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS keine Wirkung.

Wenn der Bewegungsabstand bei 0 liegt und nur die Werkzeugausrichtung verändert wird, tritt für gewöhnlich ein Fehler auf. Wenn jedoch der ROT-Parameter verwendet wird und die Beschleunigung und Verzögerung der Werkzeugrotation bevorrechtigt werden, ist eine fehlerfreie Bewegung möglich. Wenn es keine Ausrichtungsänderung durch den ROT-Parameter gibt und die Bewegungsdistanz nicht bei 0 liegt, tritt ein Fehler auf.

Auch wenn die Werkzeugrotation im Vergleich zur Bewegungsdistanz groß ist und die Rotationsgeschwindigkeit die spezifizierte Geschwindigkeit des Manipulators überschreitet, tritt ein

Fehler auf. In diesem Fall reduzieren Sie die Geschwindigkeit oder hängen Sie den ROT-Parameter an, um die Rotationsgeschwindigkeit / -beschleunigung /-verzögerung zu bevorrechtigen.

Wenn ECP verwendet wird (nur bei Arc3), bewegt sich der Trajektoriebereich des externen Kontrollpunktes entsprechend der ECP-Nummer, die durch die ECP-Anweisung festgelegt wurde, kreisförmig in Bezug auf das Werkzeugkoordinatesystem. In diesem Fall folgt der Trajektoriebereich des Werkzeugmittelpunktes keiner kreisförmigen Linie.



Einstellen von Geschwindigkeit und Beschleunigung für die Bogenbewegung

Geschwindigkeit und Beschleunigung für die **Arc**–Anweisung werden mithilfe der Befehle SpeedS und AccelS eingestellt. SpeedS und AccelS erlauben es dem Benutzer, eine Geschwindigkeit in mm/s und eine Beschleunigung in mm/s^2 zu spezifizieren.

Hinweise

Der Arc-Befehl funktioniert ausschließlich auf horizontaler Ebene

Der **Arc**-Pfad ist ein tatsächlicher Bogen in der horizontalen Ebene. Der Pfad ist interpoliert und verwendret die *endPoint*-Werte als Basis für Z und U. Verwenden Sie **Arc3** für dreidimensionale Bögen.

Überprüfen eines Bereichs für den Arc-Befehl

Die Anweisungen **Arc** und **Arc3** können vor der Bogenbewegung keine Überprüfung des Trajektoriebereichs berechnen. Daher gilt, dass der Roboter versuchen kann sogar auf dem Weg zu Zielpositionen, die sich innerhalb eines erlaubten Bereichs befinden, einen Pfad entlang zu fahren, der einen ungültigen Bereich hat. In diesem Fall hält der Manipulator abrupt an, wodurch er erheblich beschädigt werden kann. Um dies zu vermeiden, führen Sie Bereichsüberprüfungen aus, wenn das Programm bei langsamer Geschwindigkeit operiert. Lassen Sie das Programm erst danach bei höherer Geschwindigkeit laufen.

Vorgeschlagene Bewegung um die Arc-Bewegung einzurichten

Da die Bogenbewegung in der aktuellen Position beginnt, kann es notwendig sein, den Go-, den Jump- oder einen anderen verwandten Bewegungsbefehl zu verwenden, um den Roboter in die gewünschte Position zu bringen, bevor der **Arc**- oder der **Arc3**-Befehl ausgeführt wird.

Arc und Arc3 mit CP verwenden

Der CP-Parameter veranlasst den Arm, zum Endpunkt zu fahren ohne langsamer zu werden oder an dem durch *endPoint* definierten Punkt anzuhalten. Dadurch wird es dem Benutzer ermöglicht, eine Serie von Bewegungsbefehlen miteinander zu verketten. Das veranlasst den Arm, sich entlang eines kontinuierlichen Weges zu bewegen und während dieser Bewegung eine bestimmte Bewegung einzuhalten. Die Befehle **Arc** und **Arc3** ohne CP veranlassen den Arm immer dazu, bis zum vollständigen Halt herunterzubremsen, bevor er das Punktziel *endPoint* erreicht hat.

Mögliche Fehler

Änderung der Armattribute

Achten Sie besonders auf die Armattribute der Punkte, die mit dem **Arc**-Befehl verwendet werden. Wenn sich die Ausrichtung des Arms während der kreisinterpolierten Bewegung ändert (von rechtsarmiger zu linksarmiger Ausrichtung oder umgekehrt), tritt ein Fehler auf. Dies bedeutet, dass die Armattributwerte (/L Lefty, oder /R Righty) für die aktuelle Position, den *Mittelpunkt* und den *Endpunkt* identisch sein müssen.

Versuch, den Arm außerhalb des Arbeitsbereichs zu bewegen

Wenn die spezifizierte Kreisbewegung versucht, den Arm außerhalb seines Arbeitsbereichs zu bewegen, tritt ein Fehler auf.

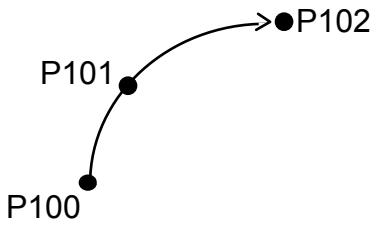
Verwandte Befehle

!Parallel Processing!, AccelS, Move, SpeedS

Beispiel einer Arc-Anweisung

Das folgende Diagramm zeigt eine Bogenbewegung, die ihren Ursprung in Punkt P100 hat, sich dann durch Punkt P101 bewegt und bei Punkt P102 endet. Die folgende Funktion würde einen solchen Bogen generieren:

```
Function ArcTest
   Go P100
   Arc P101, P102
Fend
```



Tipp

Wenn Sie den **Arc**-Befehl zum ersten Mal verwenden, ist es ratsam, einen einfachen Bogen mit Punkten direkt vor dem Roboter und ungefähr in der Mitte seines Arbeitsbereichs zu wählen. Versuchen Sie, sich den zu generierenden Bogen bildlich vorzustellen und stellen Sie sicher, dass Sie keine Punkte teachen, für deren Erreichen der Roboter versuchen müsste, sich außerhalb seines normalen Arbeitsbereichs zu bewegen.

Arch-Anweisung

Definiert die **Arch**-Parameter für die Verwendung mit den Befehlen Jump, Jump3, Jump3CP oder zeigt diese an.

> S

Syntax

(1) Arch *archNumber, departDist, approDist*
(2) Arch *archNumber*
(3) Arch

Parameter

archNumber Integer-Ausdruck, der für die zu definierende **Arch**-Nummer steht. Gültige **Arch-**Nummern sind (0-6), was eine Gesamtanzahl von 7 Einträgen in der **Arch**-Tabelle ergibt. (Siehe auch die Standard-**Arch-**Tabelle weiter unten.)

departDist Die zurückgelegte vertikale Distanz (Hubbewegung, Z) zu Beginn der Jump-Bewegung und vor Anfang der horizontalen Bewegung. (Angegeben in Millimetern)

approDist Die nötige vertikale Distanz (Absenkstrecke, gemessen von der Z-Position des Punktes, auf den sich der Arm zubewegt), um nach vollständigem Abschluss der Horizontalbewegung komplett vertikal zu verfahren. (Angegeben in Millimetern)

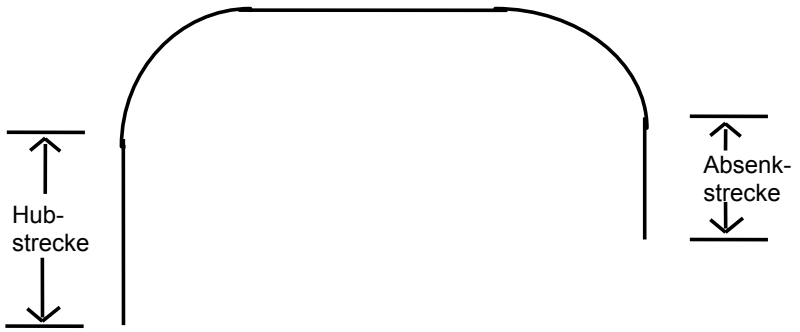
Rückgabewerte

Zeigt die **Arch**-Tabelle an, wenn sie ohne Parameter genutzt wird.
Die **Arch**-Tabelle der spezifizierten **Arch**-Nummer wird nur angezeigt, wenn nur die **Arch**-Nummer spezifiziert wurde.

Beschreibung

Der **Arch**-Befehl dient vor allem dazu, Werte in der **Arch**-Tabelle zu definieren, die für die Nutzung mit dem Jump-Bewegungsbefehl benötigt werden. Die **Arch**-Bewegung wird über die Parameter ausgeführt, die der Arch-Nummer entsprechen, die in der Jump C Bedingung ausgewählt wurde. (Für ein umfassendes Verständnis des **Arch**-Befehls ist zunächst das Verständnis des Jump-Befehls erforderlich.

Die **Arch**-Definitionen ermöglichen es dem Benutzer, bei Verwendung des Jump C-Befehls in Z-Richtung “Ecken abzurunden”. Der Arch-Befehl gibt den Punkt an, auf den die Bewegung gerichtet ist (inklusive der endgültigen Position der Z-Achse). Die Einträge in der **Arch**-Tabelle hingegen spezifizieren, wie lang die Hubstrecke sein muss, bevor die horizontale Bewegung (*riseDist*) einsetzt, und welche Distanz von der endgültigen Position der Z-Achse zurückgelegt werden muss, um die horizontale Bewegung abzuschließen (*fallDist, Absenkstrecke*). (Siehe folgendes Diagramm)



Die **Arch-**Definitionstabelle enthält insgesamt 8 Einträge, von denen 7 (0-6) benutzerdefinierbar sind. Der 8. Eintrag (**Arch** 7) ist der Standard-Arch, der keinen Arch spezifiziert; er wird Gate-Bewegung genannt. (Siehe unten stehendes Gate-Bewegungs-Diagramm). Verwendet mit der Standard-**Arch**-Eingabe (Eingabe 8) veranlasst der Jump-Befehl den Arm zu folgender Bewegung:

1) Beginn der Bewegung mit ausschließlichem Verfahren der Z-Achse, bis sie den durch den LimZ-Befehl vorgegebenen Z-Koordinatenwert erreicht hat. (Oberer Z-Wert)
2) Dann Bewegung in horizontaler Richtung zur Zielpunktposition, bis die endgültigen X-, Y- und U- Positionen erreicht sind.
3) Der Jump-Befehl wird dann abgeschlossen, indem der Arm nur mit einer Bewegung der Z-Achse abwärts bewegt wird, bis die Zielposition der Z-Achse erreicht ist.

Gate-Bewegung

(Jump mit Arch 7)

Arch-Tabellen-Standardwerte:

Arch-Nummer	Depart-Distanz	Approach-Distanz
0	30	30
1	40	40
2	50	50
3	60	60
4	70	70
5	80	80
6	90	90

Hinweise

Der Trajektoriebereich der Jump-Bewegung ändert sich je nach Bewegung und Geschwindigkeit.

Der Trajektoriebereich des Jump-Befehls beinhaltet die vertikale und die horizontale Bewegung. Es handelt sich nicht um einen CP-Trajektoriebereich. Der tatsächliche Jump-Trajektoriebereich der Bogenbewegung wird nicht ausschließlich durch die **Arch**-Parameter bestimmt. Auch die Bewegung und die Geschwindigkeit spielen eine Rolle.

Optimieren Sie den Jump-Trajektoriebereich in Ihren Anwendungen stets mit großer Sorgfalt. Führen Sie Jump mit der gewünschten Bewegung und Geschwindigkeit aus, um den tatsächlichen Trajektoriebereich zu überprüfen.

Wenn die Geschwindigkeit verringert wird, wird auch der Trajektoriebereich verringert. Wenn Jump bei hoher Geschwindigkeit ausgeführt wird, um den Trajektoriebereich eines Bewegungsbogens festzulegen, ist es möglich, dass der Greifer bei niedrigerer Geschwindigkeit mit Hindernissen kollidiert.

In einem Jump-Trajektoriebereich vergrößert sich die Depart-Distanz und verkleinert sich die Approach-Distanz, wenn die Bewegungsgeschwindigkeit auf hoch gestellt ist. Wenn die Absenkstrecke des Trajektoriebereichs kürzer als erwartet ist, verringern Sie die Geschwindigket bzw. die Verzögerung oder verlängern Sie die Absenkstrecke.

Selbst wenn Jump-Befehle mit derselben Distanz und Geschwindigkeit ausgeführt werden, wird der Trajektoriebereich durch die Bewegung der Roboterarme beeinflusst. Bei SCARA-Robotern vergrößert sich z. B. die vertikale Hubstrecke und verkleinert sich die vertikale Absenkstrecke, wenn

die Bewegung des ersten Armes sehr groß ist. Wenn sich die vertikale Absenkstrecke verringert und der Trajektoriebereich kürzer ist als erwartet, verringern Sie die Geschwindigkeit bzw. die Verzögerung oder verlängern Sie die Absenkstrecke.

Eine andere Ursache für die Gate-Bewegung

Wenn der spezifizierte Wert für die Hub- oder Absenkstrecke größer ist als die tatsächliche Z-Achsen-Distanz, über die sich der Roboter bewegen muss, um die Zielposition zu erreichen, tritt eine Gate-Bewegung auf. (D. h. es tritt keine Art von **Arch**-Bewegung auf.)

Die Arch-Werte werden beibehalten

Die Werte der **Arch**-Bewegung werden permanent gespeichert und nicht verändert, bis ein Benutzer sie verändert.

Verwandte Befehle

Jump, Jump3, JumpCP

Beispiel einer Arch-Anweisung

Im Folgenden finden Sie Beispiele für Arch-Einstellungen, die über das Befehlseingabefenster vorgenommen wurden.

```
> arch 0, 15, 15
> arch 1, 25, 50
> jump p1 c1
> arch
  arch0 =    15.000        15.000
  arch1 =    25.000        50.000
  arch2 =    50.000        50.000
  arch3 =    60.000        60.000
  arch4 =    70.000        70.000
  arch5 =    80.000        80.000
  arch6 =    90.000        90.000
>
```

Arch-Funktion

Gibt die Arch-Einstellungen aus.

> F

Syntax

Arch (*archNumber, paramNumber*)

Parameter

archNumber	Integer-Ausdruck, der für die Arch-Einstellungen zur Ausgabe eines Parameters (0 bis 6) steht.
paramNumber	1: Depart-Distanz 2: Approach-Distanz

Rückgabewerte

Eine reelle Zahl, die die Strecke beinhaltet.

Verwandte Befehle

Arch-Anweisung

Beispiel einer Arch-Funktion

```
Real archValues(6, 1)
Integer i

'Speichert die aktuellen Arch-Werte
For i = 0 to 6
    archValues(i, 0) = Arch(i, 1)
    archValues(i, 1) = Arch(i, 2)
Next i
```

Arm-Anweisung

Wählt die zu verwendende Armnummer aus bzw. zeigt sie an. > S

Syntax

(1) Arm *armNumber*
(2) Arm

Parameter

armNumber	Optionaler Integer-Ausdruck. Gültiger Bereich: 0 - 3. Der Benutzer kann bis zu vier verschiedene Arme auswählen. Arm 0 ist der Standard- (Vorgabe-) Roboterarm. Die Arme 1 - 3 sind Zusatzarme, die über die Nutzung des ArmSet-Befehls definiert werden. Werden sie ausgelassen, wird die Nummer des aktuellen Arms angezeigt.

Rückgabewerte

Wenn der **Arm-**Befehl ohne Parameter ausgeführt wird, zeigt das System die aktuelle Armnummer an.

Beschreibung

Die Arm-Anweisung ermöglicht es dem Benutzer zu spezifizieren, welcher Arm für Roboterbefehle zu verwenden ist. Der **Arm-**Befehl ermöglicht es jedem Zusatzarm, normale Positionsdaten zu verwenden. Wenn keine Zusatzarme installiert sind, operiert der Standardarm (Arm Nummer 0). Da zum Auslieferungszeitpunkt werksseitig die Armnummer 0 spezifiziert ist, ist es nicht notwendig, den **Arm**-Befehl zur Auswahl eines Armes zu verwenden. Werden jedoch Zusatzarme genutzt, so müssen sie zuerst mit dem ArmSet-Befehl definiert werden.

Die Möglichkeit, einen Zusatzarm zu konfigurieren, erlaubt es dem Benutzer, die richtigen Roboterparameter für seinen Roboter zu konfigurieren, wenn die Roboterkonfiguration sich leicht von der eines Standardroboters unterscheidet. Wenn der Benutzer beispielsweise eine zweite Ausrichtungsachse an die zweite Roboterachse montiert hat, möchte er wahrscheinlich die richtigen Roboterachsverbindungen für den neu angegliederten Roboterarm definieren. Dies ermöglicht die korrekte Funktionsweise des neu definierten Roboterarms unter den folgenden Bedingungen:

- ein einzelner Datenpunkt soll von zwei oder mehr Armen angefahren werden
- bei Verwendung des Befehls PALLET
- bei Angabe einer CP-Bewegung
- bei Spezifikation einer relativen Position
- bei Verwendung lokaler Koordinaten

Beim Betrieb eines SCARA-Roboters mit drehbaren Achsen oder Robotern mit Zylinderkoordinaten in einem kartesischen Koordinatensystem werden die Berechnungen der Achswinkel auf der Basis der ArmSet-Parameter ausgeführt. Daher ist dieser Befehl kritisch, wenn eine Definition für einen Zusatzarm oder eine -hand benötigt wird.

Hinweise

Arm 0

Arm 0 kann vom Benutzer nicht über den ArmSet-Befehl definiert oder geändert werden. Er ist reserviert, da er dazu verwendet wird, die Standard-Roboterkonfiguration zu definieren. Wenn der Benutzer ‘Arm’ auf 0 setzt, bedeutet dies, dass die Standardparameter des Roboterarms verwendet werden sollen.

Armnummer nicht definiert

Die Anwahl einer zuvor nicht durch den ArmSet-Befehl definierten Zusatzarmnummer führt zu einem Fehler.

Verwandte Befehle

ArmClr, ArmSet, ECPSet, TLSet

Beispiel einer Arm-Anweisung

Die folgenden Beispiele enthalten mögliche Zusatzarm-Definitionen, die die Befehle ArmSet und Arm verwenden. Der ArmSet-Befehl definiert den Zusatzarm und die Arm-Anweisung definiert, welcher Arm als aktueller Arm genutzt wird. (Arm 0 ist der Vorgabe-Roboterarm und kann vom Benutzer nicht eingestellt werden.)

Über das Befehlseingabefenster:

```
> ArmSet 1, 300, -12, -30, 300, 0
> ArmSet
  arm0 250  0  0  300  0
  arm1 300  -12  -30  300  0

> Arm 0
> Jump P1     'Springt unter Verwendung von Standard Arm Config zu P1
> Arm 1
> Jump P1     'Springt unter Verwendung des Zusatzarmes 1 zu P1
```

Arm-Funktion

> F

Gibt die aktuelle Armnummer des aktuellen Roboters aus.

Syntax

Arm

Rückgabewerte

Integer, der die aktuelle Armnummer enthält.

Verwandte Befehle

Arm-Anweisung

Beispiel einer Arm-Funktion

```
Print "The current arm number is: ", Arm
```

ArmClr-Anweisung

> S

Löscht eine Armdefinition.

Syntax

ArmClr *armNumber*

Parameter

armNumber	Integer-Ausdruck, der für den zu löschenden Arm der drei Arme steht. (Arm 0 ist der Vorgabearm und kann nicht gelöscht werden.)

Verwandte Befehle

Arm, ArmSet, ECPSet, Local, LocalClr, Tool, TLSet

ArmClr Example

```
ArmClr 1
```

ArmDef-Funktion

Gibt den Status einer Armdefinition aus.

> F

Syntax

ArmDef (*armNumber*)

Parameter

armNumber Integer-Ausdruck, der für den Arm steht, dessen Status ausgegeben werden soll.

Rückgabewerte

Wahr, wenn der spezifizierte Arm definiert wurde, ansonsten Falsch.

Verwandte Befehle

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLClr, TLSet

Beispiel einer ArmDef-Funktion

```
Function DisplayArmDef(armNum As Integer)

   Integer i

   If ArmDef(armNum) = False Then
      Print "Arm ", ArmNum, "is not defined"
   Else
      Print "Arm ", armNum, " Definition:"
      For i = 1 to 5
        Print ArmSet(armNum, i)
      Next i
   EndIf
Fend
```

Armset-Anweisung

Spezifiziert Zusatzarme und zeigt sie an.

> S

Syntax

(1) ArmSet *armNumber* , *link2Dist*, *joint2Offset, zOffset,* [*link1Dist]*, [*orientAngOffset*]
(2) ArmSet *armNumber*
(3) ArmSet

Parameter

armNumber	Integer-Ausdruck: Gültiger Bereich von 1-3. Der Benutzer kann bis zu 3 verschiedene Zusatzarme definieren.
link2Dist	**(SCARA-Roboter)** Die horizontale Entfernung von der Mittellinie des Ellenbogengelenks zur Mittellinie der neuen Ausrichtungsachse. (D. h. die Position, an der sich die Mittellinie der Ausrichtungsachse des neuen Zusatzarms befindet.)
joint2Offset	**(SCARA-Roboter)** Der Versatz (in Grad) zwischen der Linie, die zwischen der normalen Ellenbogen-Mittellinie und der normalen Mittellinie der Ausrichtungsachse gezogen wird, und der Linie, die zwischen der Ellenbogen-Mittellinie des neuen Zusatzarms und der neuen Mittellinie der Ausrichtungsachse gezogen wird. (Diese zwei Linien sollten sich an der Ellenbogen-Mittellinie überschneiden. Der dadurch gebildete Winkel heißt *joint2Offset*.)
zOffset	**(SCARA-Roboter)** Der Unterschied des Z-Höhenversatzes zwischen der Mitte der neuen Ausrichtungsachse und der Mitte der alten Ausrichtungsachse. (Dabei handelt es sich um eine Entfernung.)
link1Dist	**(SCARA-Roboter)** Die Entfernung der Schulter-Mittellinie zur Ellenbogen-Mittellinie der Ellenbogenausrichtung der neuen Zusatzachse.
orientAngOffset	**(SCARA-Roboter)** Der Winkelversatz (in Grad) für die neue Ausrichtungsachse gegenüber der alten Ausrichtungsachse.

Rückgabewerte

Wenn der **ArmSet**-Befehl ohne Parameter initiiert wird, zeigt das System die aktuellen Definitionsparameter des Zusatzarms an.
Die spezifizierten Armnummern und –parameter werden angezeigt, wenn nur die Armnummer spezifiziert ist.

Beschreibung

Der Befehl ermöglicht es dem Benutzer, Zusatzarm-Parameter zu spezifizieren, die zusätzlich zur Konfiguration des Standardarms verwendet werden können. Dies ist sehr nützlich, wenn ein Zusatzarm oder eine Zusatzhand am Roboter angebracht sind. Wenn ein Zusatzarm verwendet wird, so wird er durch den Arm-Befehl ausgewählt.

Die Parameter *link1Dist* und *orientAngOffset* sind optional. Wenn sie ausgelassen werden, werden die Vorgabewerte als Standard-Armwerte verwendet.

Die Möglichkeit, einen Zusatzarm zu konfigurieren, erlaubt es dem Benutzer, die richtigen Roboterparameter für seinen Roboter zu konfigurieren, wenn die Roboterkonfiguration sich leicht von der eines Standardroboters unterscheidet. Wenn der Benutzer beispielsweise eine zweite Ausrichtungsachse an die zweite Roboterachse montiert hat, möchte er wahrscheinlich die richtigen Roboterachsverbindungen für den neu angegliederten Roboterarm definieren. Dies ermöglicht die korrekte Funktionsweise des neu definierten Roboterarms unter den folgenden Bedingungen:

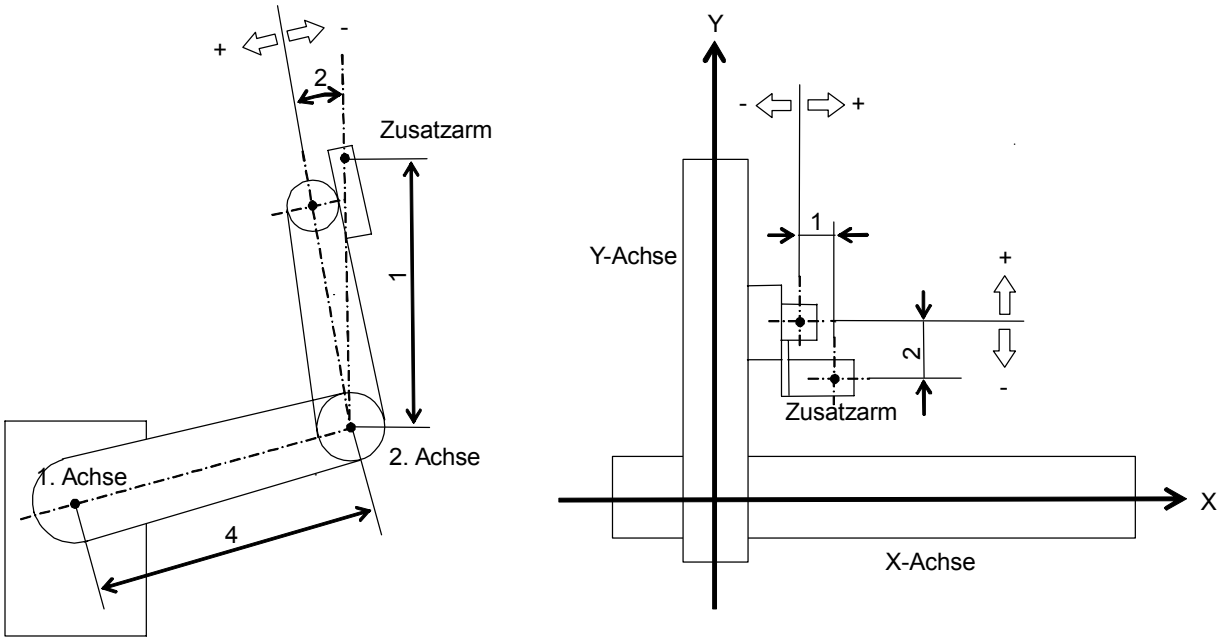
- ein einzelner Datenpunkt soll von zwei oder mehr Armen angefahren werden
- bei Verwendung des Befehls PALLET
- bei Angabe einer CP-Bewegung
- bei Spezifikation einer relativen Position
- bei Verwendung lokaler Koordinaten

Beim Betrieb eines SCARA-Roboters mit drehbaren Achsen in einem kartesischen Koordinatensystem werden die Berechnungen der Achswinkel auf der Basis der ArmSet-Parameter durchgeführt. Daher ist dieser Befehl kritisch, wenn eine Definition für einen Zusatzarm oder eine -hand benötigt wird.

Hinweise

Arm 0

Arm 0 kann durch den Benutzer nicht definiert oder geändert werden. Arm 0 ist reserviert, da er dazu verwendet wird, die Standard-Roboterkonfiguration zu definieren. Wenn der Benutzer ‘Arm’ auf 0 setzt, bedeutet dies, dass die Standardparameter des Roboterarms verwendet werden sollen.



Verwandte Befehle

Arm, ArmClr

Beispiel einer ArmSet-Anweisung

Die folgenden Beispiele enthalten mögliche Zusatzarm-Definitionen, die die Befehle **ArmSet** und Arm verwenden. Der **ArmSet**-Befehl definiert den Zusatzarm. Die Arm-Anweisung definiert, welcher Arm als aktueller Arm genutzt wird. (Arm 0 ist der Vorgabe-Roboterarm und kann vom Benutzer nicht eingestellt werden.)

Über das Befehlseingabefenster:

```
> ArmSet 1, 300, -12, -30, 300, 0
> ArmSet
  arm0 250  0  0  300  0
  arm1 300  -12  -30  300  0

> Arm 0
> Jump P1     'Springt unter Verwendung von Standard Arm Config zu P1
> Arm 1
> Jump P1     'Springt unter Verwendung des Zusatzarmes 1 zu P1
```

ArmSet-Funktion

Gibt einen ArmSet-Parameter aus.

> F

Syntax

ArmSet(*armNumber, paramNumber*)

Parameter

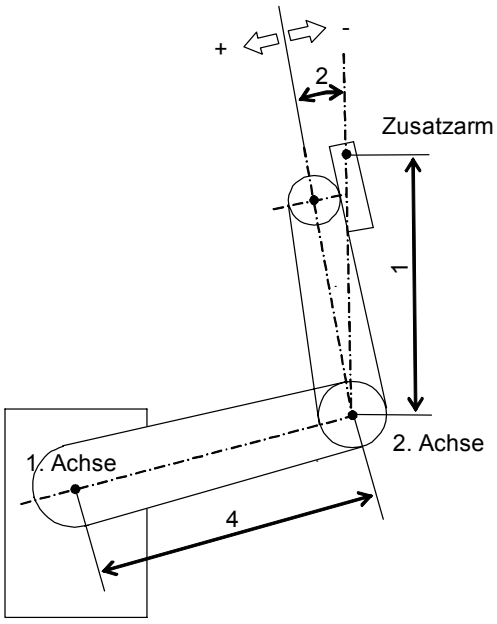
armNumber	Integer-Ausdruck, der für die Armnummer steht, für die Werte ausgegeben werden sollen.
paramNumber	Integer-Ausdruck, der für die auszugebenden Parameter (0 bis 5) steht. Siehe unte.

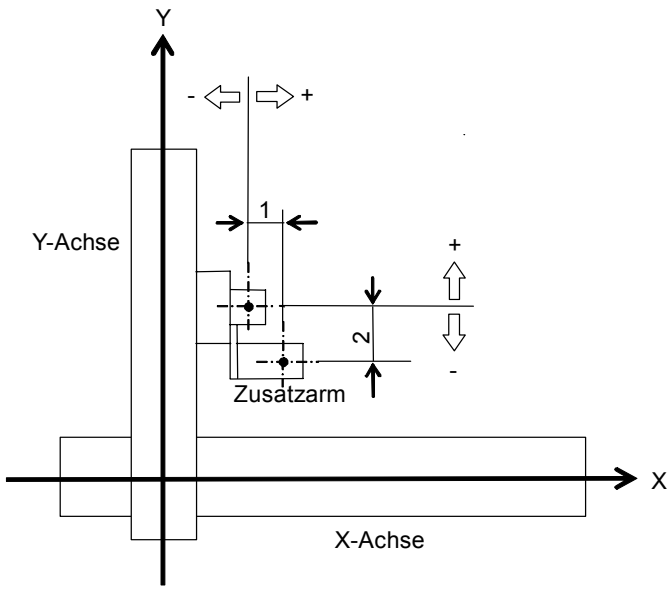
SCARA-Roboter

paramNumber	**Ausgegebener Wert**
1	Horizontale Entfernung von der 2. Achse zur Ausrichtungsmitte (in mm)
2	2. Achse Winkelversatz (in Grad)
3	Höhenversatz (in mm)
4	Horizontale Entfernung der 1. Achse zur 2. Achse (in mm)
5	Winkelversatz der Ausrichtungsachse in Grad°.

Rückgabewerte

Reelle Zahl, die den Wert eines spezifizierten Parameters enthält, wie unten beschrieben.





Verwandte Befehle

ArmClr, ArmSet-Anweisung

Beispiel einer ArmSet-Funktion

```
Real x
x = ArmSet(1, 1)
```

Asc-Funktion

Gibt den ASCII–Code des ersten Zeichens in einer Zeichenkette aus.

F

Syntax

Asc(*string*)

Parameter

string Jeder gültige Zeichenketten-Ausdruck von mindestens einem Zeichen Länge.

Rückgabewerte

Gibt einen Integer aus, der für den ASCII-Code des ersten Zeichens einer Zeichenkette steht, die zur **Asc** -Funktion geschickt wird.

Beschreibung

Die **Asc-**Funktion wird verwendet, um ein Zeichen in seine numerische ASCII-Darstellung umzuwandeln. Bei der zur **Asc**-Funktion gesendeten Zeichenkette kann es sich sowohl um eine konstante, als auch um eine variable Zeichenkette handeln.

Hinweise

Es wird nur der ASCII-Code des ersten Zeichens ausgegeben

Obgleich der **Asc-**Befehl Zeichenketten zulässt, die länger als ein Zeichen sind, wird nur das erste Zeichen auch durch den **Asc-**Befehl genutzt. **Asc** gibt nur den ASCII-Code des ersten Zeichens aus.

Verwandte Befehle

Chr$, InStr, Left$, Len, Mid$, Right$, Space$, Str$, Val

Beispiel einer Asc-Funktion

In diesem Beispiel wird der **Asc**-Befehl in einem Programm und über das Befehlseingabefenster verwendet:

```
Function asctest
    Integer a, b, c
    a = Asc("a")
    b = Asc("b")
    c = Asc("c")
    Print "The ASCII value of a is ", a
    Print "The ASCII value of b is ", b
    Print "The ASCII value of c is ", c
Fend
```

Über das Befehlseingabefenster:

```
>print asc("a")
97
>print asc("b")
98
>
```

Asin-Funktion

F

Gibt den Arkussinus eines numerischen Ausdrucks aus.

Syntax

Asin(*number***)**

Parameter

number Numerischer Ausdruck, der für den Sinus eines Winkels steht.

Rückgabewerte

Reeller Wert, in Radianten, der für den Arkussinus des Parameters ‚*number*' steht.

Beschreibung

Asin gibt den Arkussinus des numerischen Ausdrucks aus. Der Wertebereich liegt zwischen -1 und 1. Der von **Asin** ausgegebene Wert liegt zwischen –PI / 2 und PI / 2 Radianten. Wenn die Zahl < -1 oder > 1 ist, tritt ein Fehler auf.

Verwenden Sie die RadToDeg-Funktion, um Radiantenwerte in Gradzahlen umzuwandeln.

Verwandte Befehle

Abs, Acos, Atan, Atan2, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Beispiel einer Asin-Funktion:

```
Function asintest
   Double x

   x = Sin(DegToRad(45))
   Print "Asin of ", x, " is ", Asin(x)
Fend
```

Atan-Funktion

F

Gibt den Arkustangens eines numerischen Ausdrucks aus.

Syntax

Atan(*number***)**

Parameter

number Numerischer Ausdruck, der für den Tangens eines Winkelwertes steht.

Rückgabewerte

Reeller Wert in Radianten, der für den Arkustangens des Parameters ‚*number'* steht.

Beschreibung

Atan gibt den Arkustangens des numerischen Ausdrucks aus. Der numerische Ausdruck (*number)* kann ein beliebiger numerischer Wert sein. Der von **Atan** ausgegebene Wert liegt zwischen -PI und PI Radianten.

Verwenden Sie die RadToDeg-Funktion, um Radiantenwerte in Gradwerte umzuwandeln.

Verwandte Befehle

Abs, Acos, Asin, Atan2, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Beispiel einer Atan-Funktion

```
Function atantest
   Real x, y
   x = 0
   y = 1
   Print "Atan of ", x, " is ", Atan(x)
   Print "Atan of ", y, " is ", Atan(y)
Fend
```

Atan2-Funktion

F

Gibt den Winkel der imaginären Linie aus (in Radiantenwerten), die die Punkte (0,0) und (X, Y) verbindet.

Syntax

Atan2*(X, Y)*

Parameter

X	Numerischer Ausdruck, der für die X-Koordinate steht.
Y	Numerischer Ausdruck, der für die Y-Koordinate steht.

Rückgabewerte

Numerischer Wert in Radianten (-PI bis +PI).

Beschreibung

Atan2(*X*, *Y*) gibt den Winkel der Linie aus, die die Punkte (0, 0) und (*X*, *Y*) verbindet. Diese trigonometrische Funktion gibt einen Arkustangenswinkel in allen vier Quadranten aus.

Verwandte Befehle

Abs, Acos, Asin, Atan, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Beispiel einer Atan2-Funktion

```
Function at2test
   Real x, y
   Print "Please enter a number for the X Coordinate:"
   Input x
   Print "Please enter a number for the Y Coordinate:"
   Input y
   Print "Atan2 of ", x, ", ", y, " is ", Atan2(x, y)
Fend
```

ATCLR-Anweisung

> S

Löscht und initialisiert das mittlere Drehmoment für eine oder mehrere Achsen.

Syntax

ATCLR [*j1*], [*j2*], [*j3*], [*j4*], [*j5*], [*j6*]

Parameter

j1 - j6	Optional. Integer-Ausdruck, der für die Achsnummer steht. Wenn keine Parameter vorhanden sind, werden die Werte des mittleren Drehmoments für alle Achsen gelöscht.

Beschreibung

ATCLR löscht die Werte des mittleren Drehmoments für die spezifizierten Achsen.

Bevor Sie ATRQ ausführen, müssen Sie ATCLR ausführen.

Verwandte Befehle

ATRQ, PTRQ

Beispiel einer ATCLR-Anweisung

```
> atclr
> go p1
> atrq 1
      0.028
> atrq
      0.028    0.008
      0.029    0.009
      0.000    0.000
>
```

ATRQ-Anweisung

Zeigt das mittlere Drehmoment für die spezifizierte Achse an.

> S

Syntax

ATRQ [*jointNumber*]

Parameter

jointNumber Optional. Integer-Ausdruck, der für die Achsnummer steht.

Rückgabewerte

Zeigt die Werte des mittleren Drehmoments für alle Achsen an.

Beschreibung

ATQR zeigt das mittlere QMW-(quadratischer Mittelwert)-Drehmoment der spezifizierten Achse an. Der Ladezustand des Motors kann durch diesen Befehl erhalten werden. Das Ergebnis ist ein reeller Wert zwischen 0 und 1. 1 ist das maximale mittlere Drehmoment.

Bevor Sie diesen Befehl ausführen, müssen Sie ATCLR ausführen.

Dieser Befehl ist zeitlich beschränkt. Nach der Ausführung von ATCLR haben Sie 60 Sekunden Zeit, um ATRQ auszuführen. Wird die Zeit überschritten, tritt der Fehler 4030 auf.

Verwandte Befehle

ATCLR, ATRQ-Funktion, PTRQ

Beispiel einer ATRQ-Anweisung

```
> atclr
> go p1
> atrq 1
      0.028
> atrq
      0.028    0.008
      0.029    0.009
      0.000    0.000
>
```

ATRQ-Funktion

F

Gibt das mittlere Drehmoment für die spezifizierte Achse aus.

Syntax

ATRQ (*jointNumber*)

Parameter

jointNumber Integer-Ausdruck, der für die Achsnummer steht.

Rückgabewerte

Reeller Wert zwischen 0 und 1.

Beschreibung

ATQR gibt das mittlere QMW-(quadratischer Mittelwert)-Drehmoment der spezifizierten Achse aus. Der Ladezustand des Motors kann durch diesen Befehl erhalten werden. Das Ergebnis ist ein reeller Wert zwischen 0 und 1. 1 ist das maximale mittlere Drehmoment.

Bevor Sie diese Funktion ausführen, müssen Sie ATCLR ausführen.

Dieser Befehl ist zeitlich beschränkt. Nach der Ausführung von ATCLR haben Sie 60 Sekunden Zeit, um ATRQ auszuführen. Wird die Zeit überschritten, tritt der Fehler 4030 auf.

Verwandte Befehle

ATRQ-Befehl, PTCLR, PTRQ-Befehl

Beispiel einer ATRQ-Funktion

In diesem Beispiel wird die **ATRQ**-Funktion in einem Programm verwendet:

```
Function CheckAvgTorque
  Integer i

  Go P1
  ATCLR
  Go P2
  Print "Average torques:"
  For i = 1 To 4
    Print "Joint ", i, " = ", ATRQ(i)
  Next i
Fend
```

Base-Anweisung

Definiert das Basiskoordinatensystem und zeigt es an.

> S

Syntax

(1) Base *pOrigin*
(2) Base *pOrigin, pXaxis, pYaxis*, [{ **X** | **Y** }]

Parameter

pOrigin	Ursprungspunkt des Basiskoordinatensystems.
pXaxis	Ein Punkt auf der X-Achse des Koordinatensystems, wenn die X-Ausrichtung spezifiziert ist.
pYaxis	Ein Punkt auf der Y-Achse des Koordinatensystems, wenn die Y-Ausrichtung spezifiziert ist.
X | **Y**	Optional. Wenn die X-Ausrichtung spezifiziert ist, liegt *pXaxis* auf der X-Achse des neuen Koordinatensystems und nur die Z-Koordinate von *pYaxis* wird verwendet. Wenn die Y-Ausrichtung spezifiziert ist, liegt *pXaxis* auf der Y-Achse des neuen Koordinatensystems und nur die Z-Koordinate von *pXaxis* wird verwendet. Wird die Ausrichtung weggelassen, wird von einer X-Ausrichtung ausgegangen.

Beschreibung

Definiert das Roboterbasis-Koordinatensystem, indem dessen Ursprung und der Rotationswinkel im Verhältnis zum absoluten Roboterkoordinatensystem angegeben werden.

Um das Basiskoordinatensystem auf seine Vorgabewerte zurückzusetzen, führen Sie die folgende Anweisung aus. Hierdurch wird das Basiskoordinatensystem an das absolute Roboterkoordinatensystem angeglichen.

```
Base XY(0, 0, 0, 0)
```

Hinweise

Änderungen des Basiskoordinatensystems beeinträchtigt alle lokalen Definitionen

Wenn die Basiskoordinaten geändert werden, müssen alle lokalen Koordinatesysteme erneut definiert werden.

Verwandte Befehle

Local

Beispiel einer Base-Anweisung

Definiert den Ursprung des Basiskoordinatensystems bei 100 mm auf der X-Achse und bei 100 mm auf der Y-Achse.

```
> Base XY(100, 100, 0, 0)
```

BClr-Funktion

F

Löscht ein Bit in einer Zahl und gibt den neuen Wert aus.

Syntax

BClr (*number, bitNum*)

Parameter

number Spezifiziert den numerischen Wert, der benötigt wird, um das Bit durch einen Ausdruck oder einen numerischen Wert zu löschen.

bitNum Spezifiziert das Bit (integer von 0 bis 31), das durch einen Ausdruck oder einen numerischen Wert gelöscht werden soll.

Rückgabewerte

Gibt den neuen Wert des spezifizierten numerischen Wertes (Integer) aus.

Verwandte Befehle

BSet, BTst

Beispiel einer BClr-Funktion

```
flags = BClr(flags, 1)
```

BGo-Anweisung

Führt eine relative PTP-Bewegung im ausgewählten lokalen Koordinatensystem aus. > S

Syntax

BGo *destination* [**CP**] [*searchExpr*] [*!...!*]

Parameter

destination	Der Zielort einer Bewegung, die einen Punktausdruck verwendet.
CP	Optional. Spezifiziert die CP-Bewegung.
searchExpr	Optional. Ein Till- oder Find-Ausdruck. **Till** | **Find** **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können hinzugefügt werden, um E/A-Befehle und andere Befehle während der Bewegung auszuführen.

Beschreibung

Führt eine PTP-Bewegung im ausgewählten lokalen Koordinatensystem aus, das unter dem Punktausdruck *destination* spezifiziert wurde.

Wenn ein lokales Koordinatensystem spezifiziert wurde, findet die relative Bewegung im Local 0 statt (Basiskoordinatensystem).

Armausrichtungsattribute, die unter dem Punktausdruck *destination* festgelegt wurden, werden ignoriert. Der Manipulator behält die aktuellen Armausrichtungspunkte. Bei einem 6-Achsmanipulator werden die Armausrichtungsattribute jedoch automatisch so geändert, dass die Achsenlaufweite so klein wie möglich ist.

Die Till-Bedingung wird verwendet, um BGo durch Verzögerung und Stoppen des Roboters an einer Zwischenposition des Verfahrwegs abzuschließen, wenn die aktuelle Till-Bedingung erfüllt ist.

Der Find-Befehl wird verwendet, um einen Punkt in FindPos zu speichern, wenn die Find-Bedingung während der Bewegung erfüllt wird.

Wenn Till verwendet wird und die Till-Bedingung erfüllt wurde, hält der Manipulator sofort an und der Bewegungsbefehl ist beendet. Wenn die Till-Bedingung nicht erfüllt wird, bewegt sich der Manipulator zu seiner Zielposition.

Wenn Find verwendet wird und die Find-Bedingung erfüllt wurde, wird die aktuelle Position gespeichert. Bitte lesen Sie den Abschnitt *Find* für weitere Informationen.

Wenn Parallelbearbeitung verwendet wird, können andere Prozesse parallel zu dem Bewegungsbefehl ausgeführt werden.

Verwandte Befehle

Accel, BMove, Find, !....! Parallelbearbeitung, Punkt-Zuweisung, Speed, Till, TGo, TMove, Tool

Beispiel einer BGo-Anweisung

```
> BGo XY(100, 0, 0, 0) 'Verfährt 100 mm in X-Richtung
                       '(im Basiskoordinatensystem)

Function BGoTest

  Speed  50
  Accel  50,  50
  Power  High

  P10 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L
  Local 1, XY(0, 0, 0, 45)

  GoP1
  Print Here
  BGo XY(0, 50, 0, 0)
  Print Here

  Go P2
  Print Here
  BGo XY(0, 50, 0, 0)
  Print Here

  BGo XY(0, 50, 0, 0) /1
  Print Here

Fend

[Output]
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  264,645 Y:  385,355 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

BMove-Anweisung

> S

Führt im gewählten lokalen Koordinatensystem eine relative linearinterpolierte Bewegung aus.

Syntax

BMove *destination* [**ROT**] [**CP**] [*searchExpr*] [*!...!*]

Parameter

destination	Der Zielort einer Bewegung, die einen Punktausdruck verwendet.
ROT	Optional. Bestimmt Geschwindigkeit / Beschleunigung / Verzögerung zugunsten der Werkzeugrotation.
CP	Optional. Spezifiziert die CP-Bewegung.
searchExpr	Optional. Ein Till- oder Find-Ausdruck. **Till** | **Find** **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können hinzugefügt werden, um E/A-Befehle und andere Befehle während der Bewegung auszuführen.

Beschreibung

Führt relative linearinterpolierte Bewegung im ausgewählten lokalen Koordinatensystem aus, das unter dem Punktausdruck *destination* spezifiziert wurde.

Wenn ein lokales Koordinatensystem spezifiziert wurde, findet die relative Bewegung im Local 0 statt (Basiskoordinatensystem).

Armausrichtungsattribute, die unter dem Punktausdruck *destination* festgelegt wurden, werden ignoriert. Der Manipulator behält die aktuellen Armausrichtungspunkte. Bei einem 6-Achsmanipulator werden die Armausrichtungsattribute jedoch automatisch so geändert, dass die Achsenlaufweite so klein wie möglich ist.

BMove verwendet den Geschwindigkeitswert aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS. Lesen Sie den Abschnitt *BMove mit CP verwenden* für Informationen über die Beziehung Geschwindigkeit-Beschleunigung und Beschleunigung-Verzögerung. Wenn jedoch der ROT-Parameter verwendet wird, verwendet **BMove** den Geschwindigkeitswert aus SpeedR und die Beschleunigungs- und Verzögerungswerte aus AccelR. In diesem Fall haben die Geschwindigkeitswerte aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS keine Wirkung.

Wenn der Bewegungsabstand bei 0 liegt und nur die Werkzeugausrichtung verändert wird, tritt für gewöhnlich ein Fehler auf. Wenn jedoch der ROT-Parameter verwendet wird und die Beschleunigung und Verzögerung der Werkzeugrotation bevorrechtigt werden, ist eine fehlerfreie Bewegung möglich. Wenn keine Ausrichtungsänderung durch den ROT-Parameter vorliegt und die Bewegungsdistanz nicht bei 0 liegt, tritt ein Fehler auf.

Auch wenn die Werkzeugrotation im Vergleich zur Bewegungsdistanz groß ist und wenn die Rotationsgeschwindigkeit die spezifizierte Geschwindigkeit des Manipulators überschreitet, tritt ein Fehler auf. In diesem Fall reduzieren Sie die Geschwindigkeit oder hängen Sie den ROT-Parameter an, um die Rotationsgeschwindigkeit / -beschleunigung /-verzögerung zu bevorrechtigen.

Die Till-Bedingung wird verwendet, um BMove durch Verzögerung und Stoppen des Roboters an einer Zwischenposition des Verfahrweges abzuschließen, wenn die aktuelle Till-Bedingung erfüllt ist.

Der Find-Befehl wird verwendet, um einen Punkt in FindPos zu speichern, wenn die Find-Bedingung während der Bewegung erfüllt wird.

Wenn Till verwendet wird und die Till-Bedingung erfüllt wurde, hält der Manipulator sofort an und der Bewegungsbefehl ist beendet. Wenn die Till-Bedingung nicht erfüllt wird, bewegt sich der Manipulator zu seiner Zielposition.

Wenn Find verwendet wird und die Find-Bedingung erfüllt wurde, wird die aktuelle Position gespeichert. Bitte lesen Sie den Abschnitt *Find* für weitere Informationen.

Wenn Parallelbearbeitung verwendet wird, können andere Prozesse parallel zu dem Bewegungsbefehl ausgeführt werden.

Hinweise

BMove mit CP verwenden

Der CP-Parameter veranlasst den Arm, zur *destination* zu fahren, ohne langsamer zu werden oder an dem durch *destination* definierten Punkt anzuhalten. Dadurch wird es dem Benutzer ermöglicht, eine Serie von Bewegungsbefehlen miteinander zu verketten, was den Arm veranlasst, sich entlang eines kontinuierlichen Weges zu bewegen und während dieser Bewegung eine bestimmte Bewegung einzuhalten. Der **Move**-Befehl ohne CP veranlasst den Arm immer dazu, bis zum vollständigen Halt herunterzubremsen, bevor er das Punktziel *destination* erreicht hat.

Verwandte Befehle

AccelS, BGo, Find, !....! Parallelbearbeitung, Punkt-Zuweisung, SpeedS, TGo, Till, TMove, Tool

Beispiel einer BMove-Anweisung

```
> BMove XY(100, 0, 0, 0) 'Verfährt 100 mm in X-Richtung
                           (im lokalen Koordinatensystem)

Function BMoveTest

  Speed  50
  Accel  50,  50
  SpeedS 100
  AccelS 1000, 1000
  Power  High

  P10 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L
  Local 1, XY(0, 0, 0, 45)

  Go P1
  Print Here
  BMove XY(0, 50, 0, 0)
  Print Here

  Go P2
  Print Here
  BMove XY(0, 50, 0, 0)
  Print Here

  BMove XY(0, 50, 0, 0) /1
  Print Here

Fend
```

```
[Output]
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  350.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  264,645 Y:  385,355 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

Boolean-Anweisung

Deklariert Boolesche Variablen (1 Byte Integer). **S**

Syntax

Boolean *varName* [**(***subscripts***)**], [*varName* [**(***subscripts***)]...**]

Parameter

varName	Variablenname, den der Benutzer als **Boolean** deklarieren möchte.
subscripts	*Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus:* *(ubound1, [ubound2], [ubound3])* *Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension.* *Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert.* *Die Gesamtzahl der Elemente der Matrixelemente für lokale und Global Preserve-Variablen ist 1000.* *Die Gesamtzahl der Matrixelemente für globale und Modulvariablen ist 10000.* *Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)*

Beschreibung

Boolean wird verwendet, um Variablen als **Boolean** zu deklarieren. **Boolean-**Variablen können einen von zwei Werten beinhalten, Falsch und Wahr. Lokale Variablen sollten am Anfang einer Funktion deklariert sein. Globale und Modulvariabeln müssen außerhalb von Funktionen deklariert werden.

Verwandte Befehle

Byte, Double, Global, Integer, Long, Real, String

Beispiel einer Boolean Anweisung

```
Boolean partOK
Boolean A(10)         'Eindimensionale Matrix aus Boolean-Werten
Boolean B(10, 10)     'Zweidimensionale Matrix aus Boolean-Werten
Boolean C(10, 10, 10) 'Dreidimensionale Matrix aus Boolean-Werten

partOK = CheckPart()
If Not partOK Then
   Print "Part check failed"
EndIf
```

Brake-Anweisung

Schaltet die Bremse der spezifizierten Achse des aktuellen Roboters ein oder aus. >

Syntax

Brake *status, jointNumber*

Parameter

status Das Schlüsselwort **ON** wird verwendet, um die Bremse einzuschalten. Das Schlüsselwort **OFF** wird verwendet, um die Bremse auszuschalten.

jointNumber Achsennummern 1 bis 6.

Beschreibung

Der Brake-Befehl wird verwendet, um Bremsen für eine Achse ein- oder auszuschalten. Dieser Befehl kann nur als Command-Befehl ausgeführt werden. Dieser Befehl darf nur von Wartungspersonal verwendet werden.

 WARNUNG	■ Seien Sie beim Abschalten einer Bremse extrem vorsichtig. Stellen Sie sicher, dass die Achse richtig gestützt wird. Ansonsten kann die Achse herunterfallen und Schäden am Roboter verursachen oder das Personal verletzen.

Bevor Sie die Bremse lösen, stellen Sie sicher, dass Sie den Not-Aus-Schalter im Falle eines Notfalls sofort drücken können. Wenn sich die Steuerung im Not-Aus-Status befindet, sind die Motorbremsen verriegelt. Beachten Sie, dass der Roboterarm durch sein eigenes Gewicht herunterfallen kann, wenn die Bremse durch den Brake-Befehl ausgeschaltet wurde.

Verwandte Befehle

Motor, Power, Reset, SFree, SLock

Beispiel einer Brake-Anweisung

```
> brake on, 1
> brake off, 1
```

Brake-Funktion

Gibt den Bremsenstatus für eine angegebene Achse aus.

Syntax

Brake (*jointNumber)*

Parameter

jointNumber Integer-Ausdruck, der für die Achsnummer steht. Die Werte liegen zwischen 1 und der Anzahl von Roboterachsen.

Rückgabewerte

0 = Bremse aus, 1 = Bremse ein.

Verwandte Befehle

Brake-Anweisung

Beispiel einer Brake-Funktion

```
If Brake(1) = Off Then
  Print "Joint 1 brake is off"
EndIf
```

BSet-Funktion

F

Stellt ein Bit in einer Zahl ein und gibt den neuen Wert aus.

Syntax

BSet (*number*, *bitNum*)

Parameter

number	Spezifiziert den Wert, dessen Bit durch einen Ausdruck oder numerischen Wert eingestellt wird.
bitNum	Spezifiziert das Bit (integer von 0 bis 31), das durch einen Ausdruck oder einen numerischen Wert eingestellt werden soll.

Rückgabewerte

Gibt den eingestellten Bit-Wert des spezifizierten numerischen Wertes (Integer) aus.

Verwandte Befehle

BClr, BTst

Beispiel einer BSet-Funktion

```
flags = BSet(flags, 1)
```

BTst-Funktion

F

Gibt den Status eines Bits in einer Zahl aus.

Syntax

BTst (*number*, *bitNum*)

Parameter

number Spezifiziert die Nummer für den Bit-Test mit einem Ausdruck oder numerischem Wert.

bitNum Spezifiziert das zu prüfende Bit (Integer von 0 bis 31).

Rückgabewerte

Gibt die Resultate des Bit-Tests (Integer 1 oder 0) des spezifizierten numerischen Wertes aus.

Verwandte Befehle

BClr, Bset

Beispiel einer BTst-Funktion

```
If BTst(flags, 1) Then
    Print "Bit 1 is set"
EndIf
```

Byte-Anweisung

Deklariert Byte-Variablen (1 Byte Integer). **S**

Syntax

Byte *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameter

varName	Variablenname, den der Benutzer als **Byte** deklarieren möchte.
subscripts	Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus: (ubound1, [ubound2], [ubound3]) Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension. Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert. Die Gesamtzahl der Elemente der Matrixelemente für lokale und Global Preserve-Variablen ist 1000. Die Gesamtzahl der Matrixelemente für globale und Modulvariablen ist 10000. Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)

Beschreibung

Byte wird verwendet, um Variablen als **Byte** zu deklarieren. **Byte**-Variablen können ganze Zahlen im Wertebereich von -128 bis +127 enthalten. Lokale Variablen sollten in einer Funktion ganz oben deklariert sein. Globale und Modulvariabeln müssen außerhalb von Funktionen deklariert werden.

Verwandte Befehle

Boolean, Double, Global, Integer, Long, Real, String

Beispiel einer Byte-Anweisung

Das folgende Beispiel deklariert eine **Byte**-Variable und ordnet ihr dann einen Wert zu. Dann wird eine bitweise And-Verknüpfung durchgeführt, um zu sehen, ob das hohe Bit des Wertes in der Variablen ‚Test_ok' Ein-(1) oder Aus-(0) geschaltet ist. Das Ergebnis wird auf Bildschirm ausgegeben. (Natürlich ist in diesem Beispiel das hohe Bit der Variablen 'test_ok' immer eingestellt, da wir der Variablen den Wert 15 zugeordnet haben.)

```
Function Test
    Byte A(10)          'Eindimensionale Matrix aus Bytes
    Byte B(10, 10)      'Zweidimensionale Matrix aus Bytes
    Byte C(10, 10, 10) 'Dreidimensionale Matrix aus Bytes
    Byte test_ok
    test_ok = 15
    Print "Initial Value of test_ok = ", test_ok
    test_ok = (test_ok And 8)
    If test_ok <> 8 Then
       Print "test_ok high bit is ON"
    Else
       Print "test_ok high bit is OFF"
   EndIf
Fend
```

Call-Anweisung

S

Ruft eine Anwenderfunktion auf.

Syntax

Call *funcName* [(*argList*)]

Parameter

funcName	Name einer aufgerufenen Funktion.
argList	Optional. Liste von Argumenten, die in der Funktionsdeklaration spezifiziert wurden.

Beschreibung

Der **Call-**Befehl überträgt die Programmsteuerung an eine Funktion (definiert in Function...Fend). Das bedeutet, dass der **Call-**Befehl die Programmausführung veranlasst, die aktuelle Funktion zu verlassen und zu der vom **Call-**Befehl spezifizierten Funktion zu wechseln. Die Programmausführung fährt dann in dieser Funktion fort, bis eine Exit-Funktion oder ein Fend-Befehl erreicht wird. Bei der nächsten Anweisung nach dem **Call**-Befehl wird die Steuerung dann wieder an die ursprüngliche Aufruf-Funktion zurückgegeben.

Sie können die Klammern für Call-Schlüsselwort und Argument weglassen. Als Beispiel sehen Sie hier eine Call-Anweisung, die mit oder ohne Call-Schlüsselwort verwendet wird:

```
Call MyFunc(1, 2)
MyFunc 1, 2
```

Verwenden Sie GoSub...Return, um ein Unterprogramm innerhalb einer Funktion auszuführen.

Verwandte Befehle

Function, GoSub

Beispiel einer Call-Anweisung

<File1: MAIN.PRG>

```
Function  main
    Call InitRobot
Fend
```

<File2: INIT.PRG>

```
Function InitRobot

    If Motor = Off Then
        Motor On
    EndIf
    Power  High
    Speed  50
    Accel  75,  75
Fend
```

ChkCom-Funktion

F

Gibt die Anzahl von Zeichen im Empfangspuffer eines Kommunikationsports aus.

Syntax

ChkCom (*portNumber)*

Parameter

portNumber Integer-Ausdruck für die zu überprüfende Portnummer.

Rückgabewerte

Anzahl der empfangenen Zeichen (Integer).

Wenn der Port keine Zeichen empfangen kann, werden die folgenden negativen Zeichen ausgegeben, um den Portstatus auszugeben:

-2 Der Port wird von einem anderen Task verwendet
-3 Der Port ist nicht offen

Verwandte Befehle

CloseCom, OpenCom, Read, Write

Beispiel einer ChkCom-Funktion

```
Integer numChars

numChars = ChkCom(1)
```

ChkNet-Funktion

F

Gibt die Anzahl von Zeichen im Empfangspuffer eines Netzwerkports aus.

Syntax

ChkNet (*portNumber)*

Parameter

portNumber Integer-Ausdruck für die zu überprüfende Portnummer.

Rückgabewerte

Anzahl der empfangenen Zeichen (Integer).

Wenn der Port keine Zeichen empfangen kann, werden die folgenden, negativen Zeichen ausgegeben, um den Portstatus auszugeben:

-1 Der Port ist offen, es wurde jedoch keine Kommunikation hergestellt
-2 Der Port wird von einem anderen Task verwendet
-3 Der Port ist nicht offen

Verwandte Befehle

CloseNet, OpenNet, Read, Write

Beispiel einer ChkNet-Funktion

```
Integer numChars

numChars = ChkNet(201)
```

Chr$-Funktion

F

Gibt das Zeichen eines numerischen ASCII-Wertes aus.

Syntax

Chr$*(number)*

Parameter

number Integer-Ausdruck zwischen 1 und 255.

Rückgabewerte

Gibt ein Zeichen aus, das dem spezifizierten ASCII-Code entspricht, spezifiziert durch den *number*-Wert.

Beschreibung

Chr$ gibt eine Zeichenkette aus (1 Zeichen), die den ASCII-Wert des Parameters *number* hat. Wenn die spezifizierte *number* (Nummer) außerhalb des Bereichs von 1 bis 255 liegt, tritt ein Fehler auf.

Verwandte Befehle

Asc, Instr, Left$, Len, Mid$, Right$, Space$, Str$, Val

Beispiel einer Chr$-Funktion

Das folgende Beispiel deklariert eine Zeichenkettenvariable („String“) und ordnet ihr dann die Zeichenkette „ABC“ zu. Der Chr$-Befehl wird verwendet, um die numerischen ASCII –Werte in die Zeichen „A", „B" und „C" zu konvertieren. Das &H bedeutet, dass die folgende Nummer hexadezimal dargestellt wird (&H41 bedeutet Hex 41).

```
Function Test
    String temp$
    temp$ = Chr$(&H41) + Chr$(&H42) + Chr$(&H43)
    Print "The value of temp = ", temp$
Fend
```

ClearPoints

Löscht den Datenspeicher der Roboterposition.

S

Syntax

ClearPoints

Beschreibung

ClearPoints initialisiert den Roboterpositionsdatenbereich. Dieser Befehl wird zum Löschen von im Speicher befindlichen Punktdefinitionen verwendet, bevor neue Punkte geteacht werden.

Verwandte Befehle

Plist, LoadPoints, SavePoints

Beispiel einer ClearPoints-Anweisung

Das folgende Beispiel zeigt einfache Beispiele der Verwendung des **ClearPoints**-Befehls (vom Befehlseingabefenster). Beachten Sie, dass beim Initiieren des Plist-Befehls keine Teach-Punkte angezeigt werden, sobald der **ClearPoints**-Befehl ausgeführt wurde.

```
>P1=100,200,-20,0/R
>P2=0,300,0,20/L
>plist
P1=100,200,-20,0/R
P2=0,300,0,20/L
>clearpoints
>plist
>
```

CloseCom-Anweisung

S

Schließt den zuvor mit dem Befehl OpenCom geöffneten RS-232C-Port.

Syntax

CloseCom #*portNum* | **All**

Parameter

portNum Integer-Ausdruck für die zu schließende Portnummer.
Der Task schließt alle offenen RS-232C-Ports wenn „All“ angegeben wird.

Verwandte Befehle

ChkCom, OpenCom

Beispiel einer CloseCom-Anweisung

```
CloseCom #1
```

CloseNet-Anweisung

S

Schließt den zuvor mit dem Befehl OpenNet geöffneten TCP / IP-Port.

Syntax

CloseNet #*portNumber*

Parameter

portNumber Integer-Ausdruck für die zu schließende Portnummer. Der Bereich liegt zwischen 201 und 208.

Der Task schließt alle offenen TCP / IP-Ports wenn „All“ angegeben wird.

Verwandte Befehle

ChkNet, OpenNet

Beispiel einer CloseNet-Anweisung

```
CloseNet #201
```

Cls-Anweisung

S

Löscht den EPSON RC+-Textbereich des Run-Fensters, des Benutzerfensters und des Befehlseingabefensters.

Syntax

Cls

Beschreibung

Cls löscht entweder den EPSON RC+-Textbereich des Run-Fensters oder den des Benutzerfesnters, je nachdem von wo das Programm gestartet wurde.

Wenn **Cls** von einem Programm ausgeführt wird, das im Befehlseingabefenster gestartet wurde, wird der Textbereich im Befehlseingabefenster gelöscht.

Beispiel einer Cls-Anweisung

Wenn diese Anweisung vom Run- oder vom Benutzerfenster aus gestartet wird, wird der Textbereich des Fensters gelöscht, wenn **Cls** ausgeführt wird.

```
Function  main
    Integer i

    Do
        For i = 1 To 10
            Print i
        Next i
        Wait 3
        Cls
    Loop
Fend
```

Cos-Funktion

F

Gibt den Kosinus eines numerischen Ausdrucks aus.

Syntax

Cos(*number***)**

Parameter

number Numerischer Ausdruck in Radianten.

Rückgabewerte

Numerischer Wert in Radianten, der für den Kosinus des numerischen Ausdrucks *number* steht.

Beschreibung

Cos gibt den Kosinus des numerischen Ausdrucks aus. Der numerische Ausdruck (*number)* muss in Radianteneinheiten angegeben werden. Der von der **Cos**-Funktion ausgegebenene Wert liegt zwischen -1 und 1.

Verwenden Sie die DegToRad-Funktion, um Gradwerte in Radiantenwerte umzuwandeln.

Verwandte Befehle

Abs, Atan, Atan2, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Beispiel einer Cos-Funktion

Das folgende Beispiel zeigt ein einfaches Programm, das den **Cos**-Befehl nutzt.

```
Function costest
   Real x
   Print "Please enter a value in radians"
   Input x
   Print "COS of ", x, " is ", Cos(x)
Fend
```

Die folgenden Beispiele verwenden die **Cos**-Funktion vom Benutzerfenster aus.

Display the cosine of 0.55:

```
>print cos(0.55)
 0.852524522059506
>

Display cosine of 30 degrees:

>print cos(DegToRad(30))
 0.866025403784439
>
```

CP-Anweisung

S

Stellt den CP-Bewegungsmodus ein.

Syntax

CP { **On | Off** }

Parameter

On | Off Das Schlüsselwort On (Ein) wird verwendet, um den CP-Modus zu aktivieren. Das Schlüsselwort Off (Aus) wird verwendet, um den CP-Modus zu deaktivieren.

Beschreibung

Der CP-Bewegungsmodus (Continuous Path) kann für die Roboter-Bewegungsbefehle Arc, Arc3, Go, Jump, Jump3, Jump3CP und Move verwendet werden. Wenn CP aktiviert ist, ermöglichen diese Befehle die Ausführung der nächsten Anweisung bei Beginn der Verzögerung.

Verwandte Befehle

CP-Funktion, Arc, Move, Go

Beispiel einer CP-Anweisung

```
CP On
Move P1
Move P2
CP Off
```

CP-Funktion

F

Gibt den Status einer CP-Bewegung aus.

Syntax

CP

Rückgabewerte

0 = CP-Bewegung aus, 1 = CP-Bewegung ein.

Verwandte Befehle

CP-Anweisung

Beispiel einer CP-Funktion

```
If CP = Off Then
    Print "CP is off"
EndIf
```

Ctr-Funktion

F

Gibt den Zählerwert des angegebenen Zählereingangs aus.

Syntax

Ctr(*bitNumber*)

Parameter

bitNumber	Nummer des Hardwareeingangs, als Zähler eingestellt. Es können nur 16 Zähler gleichzeitig aktiv sein.

Rückgabewerte

Der aktuelle Zählstand des angegebenen Zählereingangs. (Integer-Ausdruck zwischen 0 und 65535)

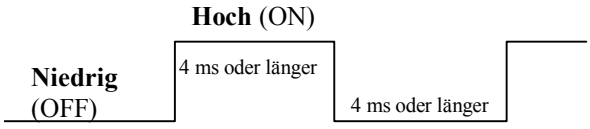
Beschreibung

Ctr arbeitet mit der CTReset-Anweisung, damit Hardwareeingänge als Zähler funktionieren können.

Jedes Mal, wenn ein Hardwareeingang, der als Zähler definiert ist, vom Off- (Aus) zum On- (Ein) Status geschaltet wird, erhöht dieser Eingang den Zählerwert um 1.

Die **Ctr**-Funktion kann jederzeit verwendet werden, um den aktuellen Zählerwert für jegliche Zählereingänge zu erhalten. Jeder Hardwareeingang kann als Zähler verwendet werden. Es können jedoch nur 16 Zähler gleichzeitig aktiv sein.

Zähler-Pulse Eingabe-Zeitkurve



Verwandte Befehle

CTReset

Beispiel einer Ctr-Funktion

Das folgende Beispiel zeigt ein Code-Muster, das verwendet werden könnte, um den Wert eines Zählereingangs zu erhalten.

```
CTReset 3 'Setzt den Zähler für Eingang 3 auf 0 zurück
On 0      'Schaltet einen Ausgangsschalter ein

Do While Ctr(3) < 5
Loop
Off 0     'Wenn 5 Eingabezyklen für Eingang 3 gezählt wurden,
          'wird ausgeschaltet (Ausgang 0 aus)
```

CTReset-Statement

Setzt den Zählerwert des spezifizierten Eingangszählers zurück und macht den Eingang damit zu einem Zählereingang.

> S

Syntax

CTReset(*bitNumber*)

Parameter

bitNumber	Nummer des Eingangs, der als Zähler eingestellt wurde. Die Nummer muss ein Integer-Ausdruck sein, der für einen gültigen Eingang steht. Es können gleichzeitig nur 16 Zähler aktiv sein.

Beschreibung

CTReset arbeitet mit der CTR-Funktion, damit Eingänge als Zähler verwendet werden können. CTReset stellt den angegebenen Eingang als Zähler ein und startet dann den Zähler. Wird der angegebenen Eingang bereits als Zähler genutzt, wird er zurückgesetzt und neu gestartet.

Hinweise

Ausschalten des Stroms und die Auswirkungen auf die Zähler:

Das Abschalten der Versorgungsspannung gibt alle Zähler frei.

Verwendung der Ctr-Funktion:

Die Ctr-Funktion wird für den Erhalt der aktuellen Werte des Zählereingangs verwendet.

Verwandte Befehle

Ctr

Beispiel einer CTReset-Anweisung

Das folgende Beispiel zeigt ein Code-Muster, das verwendet werden könnte, um den Wert eines Zählereingangs zu erhalten.

```
CTReset 3 'Setzt den Zähler 3 auf 0 zurück
On 0      'Schaltet einen Ausgangsschalter ein
Do While Ctr(3) < 5
Loop
Off 0     'Wenn 5 Eingabezyklen für Eingang 3 gezählt wurden,
          'wird ausgeschaltet (Ausgang 0 aus)
```

CtrlDev-Funktion

F

Gibt die Nummer des aktuellen Steuergerätes aus.

Syntax

CtrlDev

Rückgabewerte

21	**RC+**
22	**Remote**
23	**OP**

Verwandte Befehle

CtrlInfo-Funktion

Beispiel einer CtrlDev-Funktion

```
Print "The current control device is: ", CtrlDev
```

CtrlInfo-Funktion

F

Gibt Informationen über die Robotersteuerung aus.

Syntax

CtrlInfo (*index*)

Parameter

index Integer-Ausdruck, der für den Index der auszugebenden Daten steht.

Beschreibung

In der folgenden Tabelle befinden sich Information, die über die **CtrlInfo**-Funktion verfügbar sind.

Index	Bit	Wert	Beschreibung
0	N/A		Steuerungsversion in numerischem Format
1	0	&H1	Bereitschaftsstatus
	1	&H2	Startstatus
	2	&H4	Pausestatus
	3-7		Nicht definiert
	8	&H100	Not-Aus-Status
	9	&H200	Sicherheitsabschrankung offen
	10	&H400	Fehlerstatus
	11	&H800	Kritischer Fehler-Status
	12	&H1000	Warnung
	13-31		Nicht definiert
2	0	&H1	Freigabeschalter ist eingeschaltet
	1-31		Nicht definiert
3	0	&H1	Problem im Teachmodus-Kreis entdeckt
	1	&H2	Problem im Türsicherheitskreis entdeckt
	2	&H4	Problem im Not-Aus-Kreis entdeckt
	3-31		Nicht definiert
4	N/A		0- Normaler Modus 1- Dry-Run-Modus
5	N/A		Steuergerät: 21 - RC+ 22 - Remote 23 - OP
6	N/A		Nicht definiert
7	N/A		Betriebsart: 0- Programmiermodus 1- Automatikmodus
8	N/A		1- Motoren aus 0- Hold (die Motoreneinstellung ist ein, wird jedoch zurückgehalten)

Rückgabewerte

Langer Wert der gewünschten Daten

Verwandte Befehle

CtrlInfo$, RobotInfo, TaskInfo

Beispiel einer CtrlInfo-Funktion

```
Print "The controller version: ", CtrlInfo(0)
```

CurPos-Funktion

F

Gibt die aktuelle Zielposition des spezifizierten Roboters aus.

Syntax

CurPos [(*robotNumber*)]

Parameter

robotNumber	Optional. Definiert, für welchen Roboter Positionsdaten ausgegeben werden sollen. Wird *robotNumber* weggelassen, wird die aktuelle Position des aktuellen Roboters ausgegeben.

Rückgabewerte

Ein Roboterpunkt, der die aktuelle Zielposition des angegebenen Roboters spezifiziert.

Verwandte Befehle

InPos, FindPos, RealPos

Beispiel einer CurPos Funktion

```
Function  main

   Xqt showPosition
   Do
      Jump P0
      Jump P1
   Loop
Fend

Function showPosition

   Do
      P99 = CurPos
      Print CX(P99), CY(P99)
   Loop
Fend
```

Curve-Anweisung

S

Definiert die Daten und Punkte, die notwendig sind, um einen Arm einen geschwungenen Pfad entlang zu bewegen. Viele Datenpunkte können im Pfad definiert werden, um die Präzision des Pfades zu steigern.

Syntax

Curve *fileName, closure, mode, numAxes, pointList*

Parameter

fileName Ein Zeichenkettenausdruck für den Namen der Datei, in der die Punktedaten gespeichert sind. Dem angegebenen *fileName* wird die Dateinamenerweiterung .crv angehängt, so dass keine Dateinamenerweiterung vom Benutzer spezifiziert werden muss. Wenn der **Curve**-Befehl ausgeführt wird, wird *file* erzeugt.

closure Spezifiziert, ob die definierte Kurve am Ende der Kurvenbewegung geschlossen wird, oder offen bleibt. Dieser Parameter muss auf einen von zwei möglichen Werten eingestellt werden, wie im Folgenden angezeigt.

C – Geschlossene Kurve (Closed Curve)

O – Offene Kurve (Open Curve)

Wenn die offene Kurve spezifiziert wird, erzeugt der **Curve**-Befehl die notwendigen Daten, um den Arm am letzten Punkt der spezifizierten Punktserie zu stoppen. Wird die geschlossene Kurve spezifiziert, erzeugt der **Curve**-Befehl die notwendigen Daten, um die Bewegung durch den spezifizierten Endpunkt hindurch fortzusetzen und die Bewegung nach Rückkehr des Arms zur Startposition der angegebenen Punktserie für den **Curve**-Befehl zu stoppen.

mode Spezifiziert, ob der Arm automatisch in Tangentenrichtung der U-Achse interpoliert wird. Kann außerdem die ECP-Nummer in den oberen vier Bits spezifizieren.

Modus-Einstellung		**Tangential-korrektur**	**ECP-Nummer**
Hexadezimal	**Dezimal**		
&H00	0	**Nein (No)**	0
&H10	16		1
&H20	32		2
…	…		…
&HA0	160		10
&HB0	176		11
&HC0	192		12
&HD0	208		13
&HE0	224		14
&HF0	240		15
&H02	2	**Ja (Yes)**	0
&H12	18		1
&H22	34		2
…	…		…
&HA2	162		10
&HB2	178		11
&HC2	194		12
&HD2	210		13
&HE2	226		14
&HF2	242		15

Wenn eine Tangentialkorrektur spezifiziert wird, verwendet der **Curve**-Befehl nur die U-Achsen-Koordinate des Startpunktes in der Punktserie. Die Tangentialkorrektur behält fortlaufend die Ausrichtung der Werkzeugtangente der Kurve in der XY-Ebene bei. Sie wird spezifiziert, wenn Werkzeuge wie z. B. Abschneidevorrichtungen installiert werden, die einer fortlaufenden Tangentenausrichtung bedürfen. Wird (mit dem Parameter *closure*) eine geschlossene Kurve spezifiziert, mit automatischer Interpolation in der Tangentenrichtung der U-Achse, dreht sich die U-Achse von ihrem Startpunkt aus um 360 °. Es ist daher notwendig, vor Ausführung des CVMove-Befehls mit dem Range-Befehl den Bewegungsbereich der U-Achse einzustellen, damit die 360°-Drehung der U-Achse keinen Fehler hervorruft.

Wenn ECP verwendet wird, muss ECP in den oberen vier Bits spezifiziert werden.

numAxes Integer-Nummer zwischen 2, 3, 4 oder sechs, die die Anzahl der in der Kurvenbewegung gesteuerten Achsen wie folgt spezifiziert:

- 2 - Generiert eine Kurve in der XY-Ebene ohne Bewegung der Z-Achse oder Rotation der U-Achse.
- 3 - Generiert eine Kurve in der XYZ-Ebene ohne Rotation der U-Achse.
- 4 - Generiert eine Kurve in der XYZ-Ebene mit Rotation der U-Achse.
- 6- Generiert eine Kurve in der XYZ-Ebene mit Rotation der U-, V- und W-Achse (nur bei 6-Achsrobotern).

Die Achsen, die nicht zur Steuerung während des **Curve**-Befehls angewählt wurden, halten ihre vorherigen Codiererpulslagen bei und bewegen sich während der **Curve** Bewegung nicht.

pointList { Punktausdruck | P(*start*:*finish*) } [, *output command*] ...
Dieser Parameter ist eine Serie von Punktnummern und optionalen Ausgangsanweisungen, die entweder durch Kommata voneinander getrennt sind oder ein aufsteigender Bereich von Punkten, der durch Doppelpunkte getrennt ist. Normalerweise sind die Punktserien durch Kommata voneinander getrennt, wie im folgenden gezeigt:

```
Curve "MyFile", O, 0, 4, P1, P2, P3, P4
```

Manchmal definiert der Benutzer eine Punktserie unter Verwendung eines aufsteigenden Bereichs von Punkten, wie unten dargestellt.

```
Curve "MyFile", O, 0, 4, P(1:4)
```

Im oben dargestellten Beispiel hat der Benutzer eine Kurve mit den Punkten P1, P2, P3, und P4 definiert. Der *output*-Befehl ist optional und wird verwendet, um den Ausgabebetrieb während der Kurvenbewegung zu steuern. Für digitale Ausgänge und Merker können die Befehle On oder Off verwendet werden. Die Eingabe eines Ausgabebefehls nach einer Punktnummer in der Punktserie veranlasst die Ausführung des *output*-Befehls, wenn der Arm den Punkt unmittelbar vor dem *output*-Befehl erreicht. Eine **Curve**-Anweisung darf maximal 16 Ausgangsbefehle enthalten. Im folgenden Beispiel wird der "On 2"–Befehl ausgeführt, sobald der Arm den Punkt P2 erreicht, dann fährt der Arm zu allen Punkten zwischen P3 und P10, P3 und P10 eingeschlossen.

```
Curve "MyFile", C, 0, 4, P1, P2, ON 2, P(3:10)
```

Beschreibung

Der **Curve**-Befehl erzeugt Daten, die den Manipulator die durch die Punkstserie *pointList* definierte Kurve entlang bewegen und speichert die Daten in einer Datei auf der Steuerung. Der CVMove-Befehl verwendet die Daten in der von **Curve** erstellten Datei, um den Manipulator in einer Art CP-Bewegung zu bewegen.

Curve berechnet unabhängige X-, Y-, Z-, U-, V-, und W-Koordinatenwerte für jeden Punkt mithilfe einer kubischen Spline-Funktion, um den Trajektoriebereich zu erstellen. Wenn Punkte weit voneinander entfernt sind oder die Ausrichtung des Roboters plötzlich von Punkt zu Punkt geändert wird, kann daher die gewünschte Bewegungsbahn nicht eingehalten werden.

Es ist nicht notwendig, vor der Ausführung des **Curve**-Befehls Geschwindigkeiten oder Beschleunigungen festzulegen. Die Parameter für Armgeschwindigkeit und -beschleunigung können vor Ausführen von CVMove jederzeit mittels der Befehle SpeedS oder AccelS geändert werden.

Punkte, die in einem lokalen Koordinatensystem definiert wurden, können in Serie verwendet werden, um die Kurve an der gewünschten Position zu lokalisieren. Die Punkte können, wenn sie zum lokalen Koordinatensystem gehören, durch einen Local-Befehl nach dem **Curve**-Befehl geändert werden. Definieren Sie dazu alle spezifizierten Punkte in der Punktserie für den **Curve**-Befehl als Punkte mit lokalen Attributen.

Hinweis

Verwenden Sie nach Möglichkeit die Tangentialkorrektur

Es wird empfohlen, nach Möglichkeit die Tangentialkorrektur zu verwenden, insbesondere, wenn CVMove in einer kontinuierlichen Schleife durch dieselben Punkte verwendet wird. Wenn Sie keine Tangentialkorrektur verwenden, ist es möglich, dass der Roboter bei höheren Geschwindigkeiten nicht dem korrekten Pfad folgt.

Minimale und maximale Anzahl erlaubter Punkte in der offenen Kurve (Open Curve)

Offene Kurven können unter Verwendung von 3 bis 200 Punkten definiert werden.

Minimale und maximale Anzahl erlaubter Punkte in der geschlossenen Kurve (Closed Curve)

Geschlossene Kurven können unter Verwendung von 3 bis 50 Punkten definiert werden.

Mögliche Fehler

Versuch, den Arm außerhalb des Arbeitsbereichs zu bewegen

Der **Curve-**Befehl kann den Bewegungsbereich für den definierten Kurvenpfad nicht prüfen. Dies bedeutet, dass ein vom Benutzer definierter Pfad den Roboterarm dazu veranlassen kann, sich außerhalb des normalen Arbeitsbereichs zu bewegen. In diesem Fall tritt ein “out of range”-Fehler („außerhalb des Arbeitsbereichs“) auf.

Verwandte Befehle

AccelS-Funktion, Arc, CVMove, ECP, Move, SpeedS

Beispiel einer Curve-Anweisung

Im folgenden Beispiel wird der Kurve der noch nicht verwendete Dateiname MYCURVE.CVT zugewiesen, eine Kurve erzeugt, die sich über die Punkte P1 bis P7 erstreckt, bei P2 der Ausgangsport 2 eingeschaltet und der Arm bei P7 verzögert.

Erstellen der Kurve

```
> curve "mycurve", 0, 0, 4, P1, P2, On 2, P(3:7)
```

Bewegen des Armes zu P1 in einer geraden Linie

```
> jump P1
```

Bewegen des Armes nach der “Mycurve” genannten Kurvendefinition.

```
> cvmove "mycurve"
```

CVMove-Anweisung

Führt die kontinuierliche Spline-Pfad-Bewegung aus, die durch den **Curve**-Befehl definiert ist. **S**

Syntax

CVMove *fileName* [**CP**] [*searchExpr*]

Parameter

fileName Zeichenkettenausdruck für Pfad und Dateinamen, die für die CP-Bewegungsdaten zu verwenden sind. Diese Datei muss vorher bereits durch den Curve-Befehl erzeugt worden und auf einer PC-Festplatte gespeichert sein.

CP Optional. Spezifiziert die CP-Bewegung nach dem letzten Punkt.

searchExpr Optional. Ein Till- oder Find-Ausdruck.
Till | **Find**
Till Sw(*expr*) = {**On** | **Off**}
Find Sw(*expr*) = {**On** | **Off**}

Beschreibung

CVMove führt die kontinuierliche Spline-Pfad-Bewegung aus, definiert durch die Daten in der Datei *fileName*, die sich im Steuerungsspeicher befinden. Diese Datei muss vorher bereits durch den Curve-Befehl erzeugt worden sein.

Im System können gleichzeitig multiple Dateien vorhanden sein. Ist keine Dateinamenerweiterung angegeben, wird CVT als Endung zugewiesen.

Der Benutzer kann mit den Befehlen SpeedS und AccelS Geschwindigkeit und Beschleunigung für die CP-Bewegung für **CVMove** ändern.

Wenn der Curve-Befehl im Vorangegangenen bereits unter Verwendung von Punkten mit Local-Definitionen ausgeführt wurde, können Sie die Arbeitsposition mithilfe des Local-Befehls ändern.

Wenn CVMove ausgeführt wird, stellen Sie sicher, dass Roboter nicht mit Peripherigeräten kollidiert. Wenn Sie die Armausrichtung des 6-Achsroboters kurzfristig zwischen zwei nebeneinanderliegenden Punkten ändern möchten, ist es möglich, dass der Roboter seine Ausrichtung der vorherigen und folgenden Punkte ändert und in einem unerwarteten Trajektoriebereich verfährt. Dies liegt an der Art der kubischen Spline-Funktion. Vor der Ausführung von CVMove müssen Sie den Trajektoriebereich sorgfältig überprüfen. Stellen Sie dabei sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. Spezifizieren Sie die Punkte nahe beieinander und in gleichmäßigen Abständen. Ändern Sie die Armausrichtung zwischen nebeneinanderliegenden Punkten nicht kurzfristig.

Verwandte Befehle

AccelS-Funktion, Arc, Curve, Move, SpeedS, Till, TillOn

Beispiel einer CVMove-Anweisung

Im folgenden Beispiel wird der Kurve der noch nicht verwendete Dateiname MYCURVE.CVT zugewiesen, eine Kurve erzeugt, die sich über die Punkte P1 bis P7 erstreckt, bei P2 der Ausgangsport 2 eingeschaltet und der Arm bei P7 verzögert.

Erstellen der Kurve

```
> curve "mycurve", O, 0, 4, P1, P2, On 2, P(3:7)
```

Bewegen des Arms zu P1 in einer geraden Linie

```
> jump P1
```

Bewegen des Arms nach der “Mycurve” benannten Kurvendefinition.

```
> cvmove "mycurve"
```

CX-, CY-, CZ-, CU-, CV-, CW-Anweisungen

F

Stellt die Koordinate eines Punktes ein.

Syntax

CX(*point***)** = *value*
CY(*point***)** = *value*
CZ(*point***)** = *value*
CU(*point***)** = *value*
CV(*point***)** = *value*
CW(*point***)** = *value*

Parameter

point	**P***number* oder **P**(*expr*) oder Punktlabel.
value	Reeller Ausdruck, der für den neuen Koordinatenwert in Millimetern steht.

Verwandte Befehle

CX-, CY-, CZ-, CU-, CV-, CW-Funktionen

Beispiel einer CX-, CY-, CZ-, CU-, CV-, CW-Anweisung

```
CX(pick) = 25.34
```

CX-, CY-, CZ-, CU-, CV-, CW-Funktionen

F

Gibt einen Koordinatenwert von einem Punkt aus.

Syntax

CX(*point***)**
CY(*point***)**
CZ(*point***)**
CU(*point***)**
CV(*point***)**
CW(*point***)**

Parameter

point Punktausdruck.

Rückgabewerte

Gibt den spezifizierten Koordinatenwert aus. Die Rückgabewerte für CX, CY und CZ sind Millimeterangaben in reellen Zahlen. Die Rückgabewerte für CU, CV und CW sind Gradangaben in reellen Zahlen.

Beschreibung

Wird verwendet, um einen einzelnen Koordinatenwert eines Punktes auszugeben.

Um die Koordinate der aktuellen Roboterposition zu erhalten, verwenden Sie **Here** für den Punkteparameter.

Verwandte Befehle

Punktausdruck
CX-, CY-, CZ-, CU-, CV-, CW-Anweisungen

Beispiel für CX-, CY-, CZ-, CU-, CV-, CW-Funktionen

Im folgenden Beispiel wird der X-Achsen-Koordinatenwert von Punkt „pick“ extrahiert und der Koordinatenwert der Variable *x* zugewiesen.

```
Function cxtest
   Real x
   x = CX(pick)
   Print "The X Axis Coordinate of point 'pick' is", x
Fend
```

Date-Anweisung

Spezifiziert das aktuelle Datum in der Steuerung und zeigt es an.

Syntax

Date *yyyy, mm, dd*
Date

Parameter

yyyy	Integer-Ausdruck für eine Jahreszahl.
mm	Integer-Ausdruck für einen Monat.
dd	Integer-Ausdruck für einen Tag.

Rückgabewerte

Wenn der **Date**-Befehl ohne jegliche Parameter eingegeben wird, wird das aktuelle Datum angezeigt.

Beschreibung

Spezifiziert das aktuelle **Datum** für die Steuerung. Dieses Datum wird für die Dateien in der Steuerung verwendet. Alle in der Steuerung befindlichen Daten sind mit einem Datum versehen. Mit der Date-Anweisung wird automatisch der Wochentag für die Date-Anzeige eingestellt.

Verwandte Befehle

Time, Date$

Beispiel einer Date-Anweisung

Die folgenden Beispiele werden vom Befehlseingabefenster aus ausgeführt.

```
> Date
2006/09/27

> Date 2006,10,1

> Date
2006/10/01
```

Date$-Funktion

F

Gibt das Systemdatum aus.

Syntax

Date$

Rückgabewerte

Eine Zeichenkette, die das Datum im folgenden Format beihaltet: *yyyy/mm/dd*.

Beschreibung

Date$ wird verwendet, um das Datum des Steuerungssystems in einer Programmanweisung zu erhalten. Sie müssen die Date-Anweisung verwenden, um das Systemdatum einzustellen.

Verwandte Befehle

Date, Time, Time$

Beispiel einer Date$-Funktion

```
Print "Today's date: ", Date$
```

DegToRad-Funktion

Konvertiert Gradwerte in Radianten.

> F

Syntax

DegToRad(*degrees***)**

Parameter

degrees Reeller Ausdruck, der für die Gradzahlen steht, die in Radianten konvertiert werden sollen.

Rückgabewerte

Ein Double-Wert, der die Zahl des Radianten enthält.

Verwandte Befehle

ATan, ATan2, RadToDeg-Funktion

Beispiel einer DegToRad-Funktion

```
s = Cos(DegToRad(x))
```

DispDev-Anweisung

S

Stellt das aktuelle Anzeigegerät ein.

Syntax

DispDev (*deviceID*)

Parameter

deviceID Die Geräte-ID für das gewünschte Anzeigegerät.

21 RC+
23 OP
24 TP

Verwandte Befehle

DispDev-Funktion

Beispiel einer DispDev-Anweisung

```
DispDev 23
```

DispDev-Funktion

F

Gibt das aktuelle Anzeigegerät aus.

Syntax

DispDev

Rückgabewerte

Integer-Wert, der die Geräte-ID enthält.
21 RC+
23 OP
24 TP

Verwandte Befehle

DispDev-Anweisung

Beispiel einer DispDev-Funktion

```
Print "The current display device is ", DispDev
```

Dist-Funktion

F

Gibt die Entfernung zwischen zwei Roboterpunkten aus.

Syntax

Dist (*point1*, *point2*)

Parameter

point1, point2 Spezifiziert zwei Roboterpunktausdrücke.

Rückgabewerte

Gibt die Entfernung zwischen beiden Punkten aus (reeller Wert in Millimetern).

Verwandte Befehle

CU, CV, CW, CX, CY, CZ

Beispiel einer Dist-Funktion

```
Real distance
distance = Dist(P1, P2)
```

Do...Loop-Anweisung

S

Wiederholt einen Block von Anweisungen, wenn eine Bedingung Wahr ist, oder bis eine Bedingung Wahr wird.

Syntax

Do [{ **While** | **Until** } *condition*]
    [*statements*]
[**Exit Do**]
    [*statements*]
Loop

Oder Sie können diese Syntax verwenden:

Do
    [*statements*]
[**Exit Do**]
    [*statements*]
Loop [{ **While** | **Until** } *condition*]

Die Do…Loop-Anweisungssyntax setzt sich folgendermaßen zusammen:

Teil	Beschreibung
condition	Optional. Numerischer Ausdruck oder Zeichenkettenausdruck, der Wahr oder Falsch ist. Wenn die *condition* (*Bedingung)* Null ist, wird sie als Falsch angesehen.
statements	Eine oder mehrere Anweisungen werden wiederholt, während oder bis die Bedingung Wahr ist.

Beschreibung

Eine beliebige Anzahl von **Exit Do**-Anweisungen kann an einem beliebigen Ort innerhalb von **Do...Loop** platziert werden. So kann ein **Do...Loop**-Anweisung auch verlassen werden. **Exit Do** wird oft nach der Bewertung einer Bedingung verwendet, z. B., **If...Then**. In diesem Fall überträgt die **Exit Do**-Anweisung die Steuerung auf Anweisung direkt hinter der Loop-Anweisung.

Wenn **Exit Do** verschachtelt in **Do...Loop**-Anweisungen verwendet wird, überträgt **Exit Do** die Steuerung an den Loop, der sich eine Verschachtelungsebene über dem Loop befindet, in dem **Exit Do** auftritt. Die Verschachtelung der **Do…Loop**-Anweisungen wird für bis zu 256 Ebenen unterstützt, einschließlich anderer Anweisungen (**If…Then…Else…EndIf**, **Select…Send**).

Verwandte Befehle

For...Next, Select...Send

Beispiel einer Do-Anweisung

```
Do While Not Lof(1)
    Line Input #1, tLine$
    Print tLine$
Loop
```

Double-Anweisung

Deklariert Double-Variablen. (8 Bytes, doppelte Genauigkeit). **S**

Syntax

Double *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameter

varName	Variablenname, den der Benutzer als **Double** deklarieren möchte.
subscripts	Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus: (ubound1, [ubound2], [ubound3]) Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension. Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert. Die Gesamtzahl der Elemente der Matrixelemente für lokale und Global Preserve-Variablen ist 1000. Die Gesamtzahl der Matrixelemente für globale und Modulvariablen ist 10000. Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)

Beschreibung

Double wird verwendet, um Variablen als **Double** zu deklarieren. Lokale Variablen sollten am Anfang einer Funktion deklariert sein. Globale und Modulvariabeln müssen außerhalb von Funktionen deklariert werden.
Die maximale Anzahl von Zahlzeichen für die **Double**-Anweisung ist 14.

Verwandte Befehle

Boolean, Byte, Global, Integer, Long, Real, String

Beispiel einer Double-Anweisung

Das folgende Beispiel zeigt ein einfaches Programm, das unter Verwendung von **Double** einige Variablen deklariert.

```
Function doubletest
  Double var1
  Double A(10)         'Eindimensionale Matrix aus Doubles
  Double B(10, 10)     'Zweidimensionale Matrix aus Doubles
  Double C(10, 10, 10) 'Dreidimensionale Matrix aus Doubles
  Double arrayvar(10)
  Integer i
  Print "Please enter a Number:"
  Input var1
  Print "The variable var1 = ", var1
  For I = 1 To 5
    Print "Please enter a Number:"
    Input arrayvar(i)
    Print "Value Entered was ", arrayvar(i)
  Next I
Fend
```

ECP-Anweisung

Wählt den aktuellen ECP (externen Kontrollpunkt) aus oder zeigt ihn an.

> S

Syntax

(1) ECP *ECPNumber*
(2) ECP

Parameter

ECPNumber	Optional. Integer-Ausdruck von 0 bis 3, der darstellt, welche der 16 ECP-Definitionen mit den anstehenden Bewegungsbefehlen verwendet werden soll. ECP 0 ist eine ungültige ECP-Auswahl.

Rückgabewerte

Zeigt die aktuellen **ECP**-Werte an, wenn sie ohne Parameter verwendet werden.

Beschreibung

Die **ECP**-Anweisung wählt den externen Kontrollpunkt aus, der durch die ECP-Nummer (*ECPNumber*) spezifiziert wurde.

Hinweis

Dieser Befehl ist nur dann anwendbar, wenn die Option External Control Point installiert ist.

Ausschalten des Stroms und die Auswirkungen auf die ECP-Auswahl:

Das Ausschalten des Stroms löscht die Auswahl des ECP.

Verwandte Befehle

ECPSet

Beispiel einer ECP-Anweisung

```
>ecpset 1, 100, 200, 0, 0
>ecp 1
```

ECP-Funktion

Gibt die Nummer des aktuellen ECP aus.

Syntax

ECP

Rückgabewerte

Integer, der die aktuelle ECP-Nummer enthält.

Hinweis

Dieser Befehl ist nur dann anwendbar, wenn die Option External Control Point installiert ist.

Verwandte Befehle

ECP-Anweisung

Beispiel einer ECP-Funktion

```
Integer savECP

savECP = ECP
ECP 2
Call Dispense
ECP savECP
```

ECPClr-Anweisung

Löscht einen externen Kontrollpunkt.

> S

Syntax

ECPClr *ECPNumber*

Parameter

ECPNumber	Integer-Ausdruck, der angibt, welcher der 15 externen Kontrollpunkte zu löschen ist. (ECP0 ist Vorgabeeinstellung und kann nicht gelöscht werden.)

Hinweis

Dieser Befehl ist nur dann anwendbar, wenn die Option External Control Point installiert ist.

Verwandte Befehle

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLSet

Beispiel einer ECPClr-Anweisung

```
ECPClr 1
```

ECPDef-Funktion

Gibt den Status einer ECP-Definition aus.

> F

Syntax

ECPDef (*ECPNumber*)

Parameter

ECPNumber	Integer-Ausdruck, der angibt, für welchen ECP der Status ausgegeben werden soll.

Rückgabewerte

Wahr, wenn der spezifizierte ECP definiert wurde, sonst Falsch.

Verwandte Befehle

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLClr, TLSet

Beispiel einer ECPDef-Funktion

```
Function DisplayECPDef(ecpNum As Integer)

   If ECPDef(ecpNum) = False Then
      Print "ECP ", ecpNum, "is not defined"
   Else
      Print "ECP ", ecpNum, ": ",
      Print ECPSet(ecpNum)
   EndIf
Fend
```

ECPSet-Anweisung

> S

Definiert einen externen Kontrollpunkt oder zeigt ihn an.

Syntax

(1) ECPSet *ECPNum*, *ECPPoint*
(2) ECPSet ***ECPNum***
(3) ECPSet

Parameter

ECPNum	Integer-Nummer von 1 bis 15, die darstellt, welcher der 15 externen Kontrollpunkte definiert werden soll.
ECPPoint	**P***number* oder **P**(*expr*) oder Punktlabel oder Punktausdruck.

Rückgabewerte

Werden die Parameter weggelassen, werden die aktuellen **ECPSet**-Definitionen angezeigt. Wenn nur die **ECP**-Nummer spezifiziert wird, werden die spezifizierten **ECP-Set**-Definitionen angezeigt.

Beschreibung

Definiert einen externen Kontrollpunkt.

Hinweis

Dieser Befehl ist nur dann anwendbar, wenn die Option External Control Point installiert ist.

Beispiel einer ECPSet-Anweisung

```
ECPSet 1, P1
ECPSet 2, 100, 200, 0, 0
```

ECPSet-Funktion

F

Gibt einen Punkt aus, der die ECP-Definition für den spezifizierten ECP enthält.

Syntax

ECPSet(*ECPNumber*)

Parameter

ECPNumber Integer-Ausdruck, der die Nummer des auszugebenden ECPs darstellt.

Rückgabewerte

Ein Punkt, der die ECP-Definition beinhaltet.

Hinweis

Dieser Befehl ist nur dann anwendbar, wenn die Option External Control Point installiert ist.

Verwandte Befehle

ECPSet-Anweisung

Beispiel einer ECPSet-Funktion

```
P1 = ECPSet(1)
```

Elbow-Anweisung

> S

Stellt die Ellenbogenausrichtung eines Punktes ein.

Syntax

(1) Elbow *point*, [*value*]
(2) Elbow

Parameter

point	**P***number* oder **P**(*expr*) oder Punktlabel.
value	Integer-Ausdruck. 1 = Höher (/A) 2 = Niedriger (/B)

Rückgabewerte

Wenn beide Parameter ausgelassen werden, wird die Ellenbogenausrichtung für die aktuelle Roboterposition angezeigt.
Wenn *value* ausgelassen wird, wird die Ellenbogenausrichtung für den spezifizierten Punkt angezeigt.

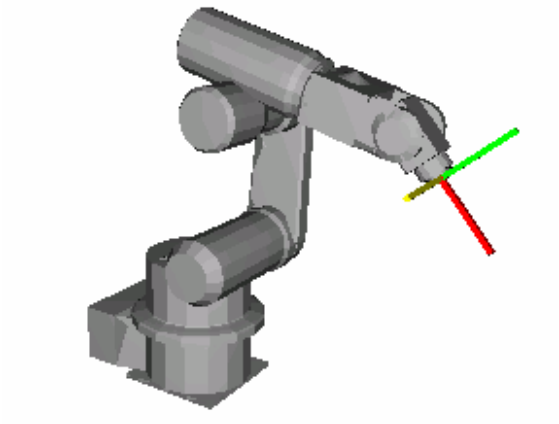
Verwandte Befehle

Elbow-Funktion, Hand, J4Flag, J6Flag, Wrist

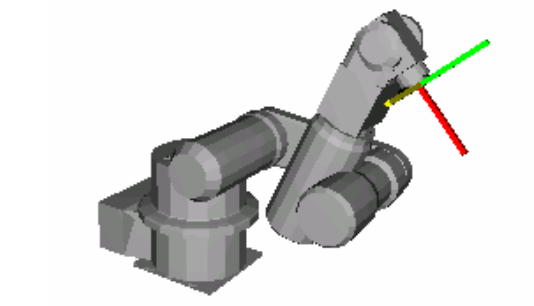
Beispiel einer Elbow-Anweisung

```
Elbow P0, Below
Elbow pick, Above
Elbow P(myPoint), myElbow

P1 = 0.000,  490.000,  515.000,   90.000,  -40.000,  180.000
```



```
Elbow P1, Above
Go P1
```



```
Elbow P1, Below
Go P1
```

Elbow-Funktion

F

Gibt die Ellenbogenausrichtung eines Punktes aus.

Syntax

Elbow [(*point*)]

Parameter

point Optional. Punktausdruck. Wird *point* weggelassen, wird die Ellenbogenausrichtung der aktuellen Roboterposition ausgegeben.

Rückgabewerte

1 Above (/A)
2 Below (/B)

Verwandte Befehle

Elbow-Anweisung, Hand, Wrist, J4Flag, J6Flag

Beispiel einer Elbow-Funktion

```
Print Elbow(pick)
Print Elbow(P1)
Print Elbow
Print Elbow(P1 + P2)
```

Era-Funktion

F

Gibt die Achsnummer aus, bei der ein Fehler aufgetreten ist.

Syntax

Era(*taskNum*)

Parameter

taskNum	Integer-Ausdruck, der für eine Tasknummer von 0 bis 16 steht. Das Weglassen der Tasknummer oder 0 gibt den aktuellen Task an.

Rückgabewerte

Die Achsnummer, die den Fehler im Bereich 0-6 verursacht hat, wie im Folgenden beschrieben:

0 - Der aktuelle Fehler wurde nicht durch eine Servoachse verursacht.
1 - Der aktuelle Fehler wurde durch Achsnummer 1 verursacht.
2 - Der aktuelle Fehler wurde durch Achsnummer 2 verursacht.
3 - Der aktuelle Fehler wurde durch Achsnummer 3 verursacht.
4 - Der aktuelle Fehler wurde durch Achsnummer 4 verursacht.
5 - Der aktuelle Fehler wurde durch Achsnummer 5 verursacht.
6 - Der aktuelle Fehler wurde durch Achsnummer 6 verursacht.

Beschreibung

Era wird verwendet, um beim Auftreten eines Fehlers festzustellen, ob der Fehler von einer der Roboterachsen verursacht wurde und, wenn das der Fall ist, die Nummer der Achse auszugeben, die den Fehler verursacht hat. Wenn der aktuelle Fehler nicht von einer Achse verursacht wurde, gibt die **Era**-Funktion eine 0 (Null) aus.

Verwandte Befehle

Erl, Err, ErrMsg$, Ert, OnErr, Trap

Beispiel einer Era Funktion

```
Function  main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
 If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
   Print "The line where the error occurred is Line ", Erl(errTask)
   If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

EResume-Anweisung

Setzt die Ausführung nach Beendigung einer Fehlerbehandlungsroutine fort.

S

Syntax

EResume [{ *label* | **Next** }]

Beschreibung

EResume

Wenn der Fehler in derselben Prozedur aufgetreten ist wie die Fehlerbehandlungsroutine, wird die Ausführung mit der Anweisung, die den Fehler verursacht hat, fortgesetzt. Wenn der Fehler in einer aufgerufenen Prozedur aufgetreten ist, wird die Ausführung mit der Call-Anweisung aus der Prozedur, die eine Fehlerbearbeitungsroutine enthält, fortgesetzt.

EResume Next

Wenn der Fehler in derselben Prozedur aufgetreten ist wie die Fehlerbehandlungsroutine, wird die Ausführung mit der Anweisung fortgesetzt, die direkt auf diejenige folgt, die den Fehler verursacht hat. Wenn der Fehler in einer aufgerufenen Prozedur aufgetreten ist, wird die Ausführung sofort mit der Anweisung fortgesetzt, die der Call-Anweisung, die zuletzt durch die Prozedur mit der Fehlerbearbeitungsroutine aufgerufen wurde, folgt.

EResume { *label* }

Wenn der Fehler in derselben Prozedur aufgetreten ist wie die Fehlerbehandlungsroutine, wird die Ausführung mit der Anweisung, die das Label beinhaltet, fortgesetzt.

Verwandte Befehle

OnErr

Beispiel einer EResume-Anweisung

```
Function  main
  Integer retry

  OnErr GoTo eHandler
  Do
    RunCycle
  Loop
  Exit Function

eHandler:
  Select Err
    Case MyError
      retry = retry + 1
      If retry < 3 Then
        EResume ' try again
      Else
        Print "MyError has occurred ", retry, " times
      EndIf
  Send
Fend
```

Erf$-Funktion

F

Zeigt den Namen der Funktion an, in der der Fehler aufgetreten ist.

Syntax

Erf$[(*taskNumber*)]

Parameter

taskNumber	Integer-Ausdruck, der für eine Tasknummer von 0 bis 16 steht. Das Weglassen der Tasknummer oder 0 gibt den aktuellen Task an.

Rückgabewerte

Name der Funktion, in der der letzte Fehler aufgetreten ist.

Beschreibung

Erf$** wird zusammen mit OnErr verwendet. Erf$ zeigt den Namen der Funktion an, in der der Fehler aufgetreten ist. Wird **Erf$ in Kombination mit Err, Ert, Erl und Era verwendet, erhält der Benutzer wesentlich mehr Informationen über den aufgetretenen Fehler.

Verwandte Befehle

Era, Erl, Err, ErrMsg$, Ert, OnErr

Beispiel einer Erf$-Funktion

Das folgende Beispiel zeigt ein einfaches Programm, das die Ert-Funktion verwendet, um zu bestimmen, in welchem Task der Fehler aufgetreten ist; Erf$: Der Name der Funktion, in der der Fehler aufgetreten ist; Erl: Zeilennummer, in der der Fehler aufgetreten ist; Era: Wenn eine Achse den Fehler verursacht hat…

```
Function  main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
 If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "Function at which error occurred is ", Erf$(errTask)
   Print "The line where the error occurred is Line ", Erl(errTask)
   If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

Erl-Funktion

F

Gibt die Zeilennummer aus, in der der Fehler aufgetreten ist.

Syntax

Erl[(*taskNumber*)]

Parameter

taskNumber	Integer-Ausdruck, der für eine Tasknummer von 0 bis 16 steht. Das Weglassen der Tasknummer oder 0 gibt den aktuellen Task an.

Rückgabewerte

Zeilennummer, in der der letzte Fehler aufgetreten ist.

Beschreibung

Erl wir zusammen mit OnErr verwendet. Erl gibt die Zeilennummer aus, in der der Fehler aufgetreten ist. Wird **Erl** in Kombination mit Err, Ert und Era verwendet, erhält der Benutzer wesentlich mehr Informationen über den aufgetretenen Fehler.

Verwandte Befehle

Era, Erf$, Err, ErrMsg$, Ert, OnErr

Beispiel einer Erl-Funktion

Das folgende Beispiel zeigt ein einfaches Programm, das die Ert-Funktion verwendet, um zu bestimmen, in welchem Task der Fehler aufgetreten ist; Erl: Wo der Fehler aufgetreten ist; Era: Wenn eine Achse den Fehler verursacht hat…

```
Function  main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
 If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
   Print "The line where the error occurred is Line ", Erl(errTask)
   If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

Err-Funktion

F

Gibt den zuletzt aufgetretenen Fehlerstatus aus.

Syntax

Err [(*taskNumber*)]

Parameter

taskNumber	***Optional.*** Integer-Ausdruck, der für eine Tasknummer von 0 bis 16 steht. 0 gibt den aktuellen Task an.

Rückgabewerte

Gibt einen numerischen Fehlercode in Integer-Form aus.

Beschreibung

Err ermöglicht es dem Benutzer, den aktuellen Fehlercode zu lesen. Zusammen mit den Möglichkeiten der Fehlerbehebung von SPEL+ gibt dies dem Benutzer die Möglichkeit festzustellen, welcher Fehler aufgetreten ist und entsprechend zu reagieren. **Err** wir zusammen mit OnErr verwendet.

Verwandte Befehle

Era, Erf$, Erl, ErrMsg$, EResume, Ert, OnErr, Return

Beispiel einer Err-Funktion

Das folgende Beispiel zeigt ein einfaches Dienstprogramm, das prüft, ob die Punkte P0-P399 existieren. Wenn der Punkt nicht existiert, erscheint eine Meldung für den Benutzer. Das Programm verwendet den CX-Befehl, um für jeden einzelnen Punkt zu testen, ob er definiert wurde oder nicht. Wenn ein Punkt nicht definiert ist, wird die Steuerung an die Fehlerbehandlungsroutine übergeben und auf dem Bildschirm erscheint eine Meldung, die dem Benutzer mitteilt, welcher Punkt nicht definiert war.

```
Function errtest
   Integer i, errnum
   Real x

   OnErr GoTo eHandle
   For i = 0 To 399
     x = CX(P(i))
   Next i
   Exit Function
'
'
'*******************************************
'* Error Handler                           *
'*******************************************
eHandle:
   errnum = Err
   'Überprüft, ob ein undefinierter Punkt verwendet wird
   If errnum = 78 Then
      Print "Point number P", i, " is undefined!"
   Else
      Print "ERROR: Error number ", errnum, " Occurred."
   EndIf
   EResume Next
Fend
```

ErrMsg$-Funktion

F

Gibt die Fehlermeldung aus, die der angegebenen Fehlernummer entspricht.

Syntax

ErrMsg$(*errNumber, langID*)

Parameter

errNumber Integer-Ausdruck, der die Fehlernummer enthält, für die die Nachricht ausgegeben wird.

langID Optional. Integer-Ausdruck, der die Sprach-ID enthält:

0 - Englisch

1 - Japanisch

2 - Deutsch

3 - Französisch

Wird diese Angabe weggelassen, wird Englisch verwendet.

Rückgabewerte

Gibt die Fehlermeldung aus, die in der Fehlercode-Tabelle beschrieben ist.

Verwandte Befehle

Era, Erl, Err, Ert, OnErr, Trap

ErrMsg$ Beispiel

Das folgende Beispiel zeigt ein einfaches Programm, das die Ert-Funktion verwendet, um zu bestimmen, in welchem Task der Fehler aufgetreten ist; Erl: Wo der Fehler aufgetreten ist; Era: Wenn eine Achse den Fehler verursacht hat…

```
Function  main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
 If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
   Print "The line where the error occurred is Line ", Erl(errTask)
   If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

Error-Anweisung

Generiert einen eigendefinierten Fehler. **S**

Syntax

(1) Error *task Number, errorNumber*
(2) Error *errorNumber*

Parameter

taskNumber	*Optional. Integer-Ausdruck, der für eine Tasknummer steht. Bereich von 0 bis 16.* 0 gibt den aktuellen Task an.
errorNumber	Integer-Ausdruck, der für eine gültige Fehlernummer steht. Die Nummern für Eigendefinierte Fehler-Meldungen sind von 8000 bis 8999 festgelegt.

Beschreibung

Verwenden Sie die Error-Anweisung, um system- oder benutzerdefinierte Fehler zu generieren. Sie können Eigendefinierte Fehler-Labels und Beschreibungen mithilfe des Eigendefinierte Fehler-Editors in der Entwicklungsumgebung von EPSON RC+ 5.0 definieren.

Verwandte Befehle

Era, Erl, Err, OnErr

Beispiel einer Error-Anweisung

```
#define ER_VAC 8000

If Sw(vacuum) = Off Then
    Error ER_VAC
EndIf
```

Ert-Funktion

Gibt die Tasknummer aus, in der ein Fehler aufgetreten ist.

> F

Syntax

Ert

Rückgabewerte

Die Tasknummer, in der der Fehler aufgetreten ist.

Beschreibung

Wenn ein Fehler auftritt, wird **Ert** verwendet um festzustellen, in welchem Task der Fehler aufgetreten ist. Die ausgegebene Zahl liegt zwischen 1 und 16.

Verwandte Befehle

Era, Erl, Err, ErrMsg$, OnErr, Trap

Beispiel einer Ert Funktion

Das folgende Beispiel zeigt ein einfaches Programm, das die Ert-Funktion verwendet, um zu bestimmen, in welchem Task der Fehler aufgetreten ist; Erl: Wo der Fehler aufgetreten ist; Err: Welche Art von Fehler aufgetreten ist; Era: Wenn eine Achse den Fehler verursacht hat…

```
Function  main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
 If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
   Print "The line where the error occurred is Line ", Erl(errTask)
   If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
    EndIf
  EndIf
Fend
```

EStopOn-Funktion

F

Gibt den aktuellen Not-Aus-Status aus.
Diese Funktion wird aus Kompatibilitätsgründen freigehalten.
Im Not-Aus-Status werden alle Tasks abgebrochen.

Syntax

EstopOn

Rückgabewerte

Wahr, wenn der Not-Aus-Kreis unterbrochen wurde (ON), sonst Falsch.

Exit-Anweisung

Ein Schleifenkonstrukt oder eine Funktion wird verlassen.

S

Syntax

Exit { **Do | For | Function** }

Beschreibung

Die Syntax der Exit-Anweisung hat die folgende Form:

Anweisung	Beschreibung
Exit Do	Ist eine Möglichkeit, eine Do...Loop Anweisung zu verlassen. Es kann nur innerhalb einer Do...Loop-Anweisung verwendet werden. **Exit Do** überträgt die Steuerung an die Anweisung, die auf die Loop-Anweisung folgt. Wenn **Exit Do** innerhalb verschachtelter **Do...Loop**-Anweisungen verwendet wird, überträgt **Exit Do** die Steuerung an den Loop der sich eine Verschachtelungsebene über dem Loop befindet, in dem **Exit Do** auftritt.
Exit For	Ist eine Möglichkeit, eine For-Schleife zu verlassen. Diese Anweisung kann nur innerhalb einer For…Next-Schleife verwendet werden. **Exit For** überträgt die Steuerung an die Anweisung, die auf die Next-Anweisung folgt. Wenn **Exit For** innerhalb verschachtelter **For**-Schleifen verwendet wird, überträgt **Exit For** die Steuerung an die Schleife, die sich eine Verschachtelungsebene über der Schleife befindet, in der **Exit For** auftritt.
Exit Function	Verlässt sofort die Funktionsprozedur, in der diese Anweisung verwendet wird. Die Ausführung fährt mit der Anweisung fort, die auf diejenige folgt, die die Funktion aufgerufen hat.

Verwandte Befehle

Do...Loop, For...Next, Function...Fend

Beipiel einer Exit-Anweisung

```
For i = 1 To 10
    If Sw(1) = On Then
        Exit For
    EndIf
    Jump P(i)
Next i
```

Find-Anweisung

S

Gibt die Bedingung an, unter der Koordinaten während einer Bewegung gespeichert werden oder zeigt sie an.

Syntax

Find [*condition*]

Parameter

condition Konditionaler Ausdruck der veranlasst, dass die Koordinaten gespeichert werden.

Verwandte Befehle

FindPos, Go, Jump, PosFound

Beispiel einer Find-Anweisung

```
Find Sw(5) = On
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Cannot find the sensor signal."
EndIf
```

FindPos-Funktion

F

Gibt einen Roboterpunkt aus, der während einer Bewegung durch die Find-Anweisung gespeichert wurde.

Syntax

FindPos

Rückgabewerte

Ein Roboterpunkt, der während einer Bewegung durch die Find-Anweisung gespeichert wurde.

Verwandte Befehle

Find, Go, Jump, PosFound, CurPos, InPos

Beispiel einer FindPos-Funktion

```
Find Sw(5) = On
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Cannot find the sensor signal."
EndIf
```

Fine-Anweisung

Definiert die Positionierungsgenauigkeit für Zielpunkte und zeigt sie an.

> S

Syntax

(1) Fine *axis1, axis2, axis3, axis4,* [*axis5*], [*axis6*]
(2) Fine

Parameter

axis1	Integer-Ausdruck im Bereich von 0-65535, der für den zulässigen Positionierungsfehler für die erste Achse steht.
axis2	Integer-Ausdruck im Bereich von 0-65535, der für den zulässigen Positionierungsfehler für die zweite Achse steht.
axis3	Integer-Ausdruck im Bereich von 0-65535, der für den zulässigen Positionierungsfehler für die dritte Achse steht.
axis4	Integer-Ausdruck im Bereich von 0-65535, der für den zulässigen Positionierungsfehler für die vierte Achse steht.
axis5	Optional. Integer-Ausdruck im Bereich von 0-65535, der für den zulässigen Positionierungsfehler für die fünfte Achse steht.
axis6	Optional. Integer-Ausdruck im Bereich von 0-65535, der für den zulässigen Positionierungsfehler für die sechste Achse steht.

Rückgabewerte

Wenn **Fine** ohne Parameter verwendet wird, werden die aktuellen Fine-Werte für jede der vier oder sechs Achsen angezeigt.

Beschreibung

Fine definiert für jede Achse den zulässigen Positionierungsfehler, mit dem der Abschluss beliebiger Bewegungen registriert wird.

Die Prüfung hinsichtlich der Bewegungsvollendung beginnt, nachdem die CPU den Zielpositions-Pulsewert vollständig an das Servo-System gesandt hat. Aufgrund der Servoverzögerung hat der Roboter die Zielposition noch nicht erreicht. Diese Prüfung wird im Abstand von wenigen Millisekunden wiederholt, bis jede Achse in der angegebenen Bereichseinstellung angekommen ist. Die Positionierung wird als abgeschlossen betrachtet, wenn alle Achsen innerhalb dieser angegebenen Bereiche angekommen sind. Sobald die Positionierung abgeschlossen ist, geht die Programmsteuerung zur nächsten Anweisung über. Das Servosystem behält jedoch die Steuerung der Roboterzielposition inne.

Wenn mit dem Fine-Befehl vergleichsweise große Bereiche verwendet werden, wird die Positionierung relativ früh in der Bewegung bestätigt und die nächste Anweisung wird ausgeführt.

Die Standardeinstellungen der **Fine**-Anweisung hängen vom Robotertyp ab. Lesen Sie das Roboterhandbuch für weitere Information.

Hinweise

Zykluszeiten und der Fine-Befehl

Der **Fine**-Wert hat keine Auswirkungen auf die Steuerung der Beschleunigung oder Verzögerung des Manipulators. Kleinere **Fine**-Werte können das System jedoch veranlassen, langsamer zu laufen, da es das Servosystem zusätzliche Zeit kosten kann (einige Millisekunden), um in einen zulässigen Positionsbereich zu gelangen. Sobald der Arm den zulässigen Positionsbereich erreicht hat (definiert durch den **Fine**-Befehl), führt die CPU den nächsten Anwenderbefehl aus. (Beachten Sie, dass alle aktivierten Achsen in Position sein müssen, bevor die CPU den nächsten Anwenderbefehl ausführen kann.)

Initialisierung von Fine (durch die Befehle Motor On, SLock und SFree)

Bei der Verwendung der Folgenden Befehle wird der Fine-Wert auf die Standardwerte gesetzt: SLock, SFree, und Motor.
Stellen Sie sicher, dass Sie die Fine-Werte neu einstellen, nachdem einer der o.g. Befehle ausgeführt wurde.

Mögliche Fehler

Wenn die **Fine**-Positionierung nicht innerhalb von 2 Sekunden abgeschlossen ist, tritt der Fehler 4024 auf. Dieser Fehler bedeutet normalerweise, dass die Balance des Servosystems eingestellt werden muss. **(Bitte kontaktieren Sie in diesem Fall Ihren EPSON-Händler.)**

Verwandte Befehle

Accel, AccelR, AccelS, Arc, Go, Jump, Move, Speed, SpeedR, SpeedS, Pulse

Beispiel einer Fine-Anweisung

Das folgende Beispiel zeigt eine einfache PTP-Bewegung vom Punkt P0 zum Punkt P1 mit einer geradlinigen Rückbewegung zum Punkt P0. Später bewegt sich der Arm geradlinig zum Punkt P2, bis der 2. Eingang eingeschaltet wird. Wenn der 2. Eingang während der Bewegung einschaltet, verzögert sich der Arm bis zum vollständigen Stopp, bevor er den Punkt P2 erreicht und der nächste Programmbefehl ausgeführt wird.

```
Function finetest
    Fine 5, 5, 5, 5           'Reduziert die Präzision auf +/- 1 Pulse
    Go P1
    Go P2
Fend

> Fine 50, 50, 50, 50
>
> Fine
10, 10, 10, 10
```

Fine-Funktion

F

Gibt die **Fine**-Einstellung für eine angegebene Achse aus.

Syntax

Fine(*joint*)

Parameter

joint Integer-Ausdruck, der für die Achsnummer steht, für die die Fine-Einstellungen ausgegeben werden sollen.

Rückgabewerte

Reeller Wert

Verwandte Befehle

Accel, AccelS, Arc, Go, Jump, Move, Speed, SpeedS, Pulse

Beispiel einer Fine-Funktion

In diesem Beispiel wird die **Fine**-Funktion in einem Programm verwendet.

```
Function finetst
    Integer a
    a = Fine(1)
Fend
```

Fix-Funktion

F

Gibt den Integeranteil einer reellen Zahl aus.

Syntax

Fix(*number*)

Parameter

number Reeller Ausdruck, der die zu fixierende Zahl enthält.

Rückgabewerte

Ein Integer-Wert, der den Integer-Anteil einer reellen Zahl enthält.

Verwandte Befehle

Int

Beispiel einer Fix-Funktion

```
>print Fix(1.123)
 1
>
```

FmtStr$-Funktion

F

Formatiert einen numerischen Ausdruck.

Syntax

FmtStr$ (*numeric expression, strFormat*)

Parameter

numeric expression Zu formatierender numerischer Ausdruck.
strFormat Formatspezifizierungs-Zeichenkette.

Rückgabewerte

Eine Zeichenkette, die den formatierten Ausdruck enthält.

Beschreibung

FmtStr$ wird verwendet, um einen numerischen Ausdruck in eine Zeichenkette umzuwandeln.

Spezifikationssymbole für das numerische Format

Formatiert einen numerischen Ausdruck.

Zeichen	Beschreibung
Keine	Anzeige der Zahl ohne Formatierung.
(0)	Zahlzeichen-Platzhalter. Anzeige eines Zahlzeichens oder einer Null. Wenn an der Stelle des Ausdrucks ein Zahlzeichen steht, an der in der Formatzeichenkette eine Null (0) erscheint, wird es angezeigt. Sonst wird an dieser Stelle eine Null angezeigt. Wenn eine Zahl weniger Zahlzeichen hat (egal auf welcher Seite der Dezimalen), als der Formatausdruck Nullen aufweist, werden die vorangehenden oder die anhängenden Nullen angezeigt. Wenn die Zahl mehr Zahlzeichen zur Rechten der Dezimaltrennung hat, als Nullen zur rechten Seite der Dezimaltrennung im Formatausdruck vorhanden sind, wird diese Zahl auf so viele Dezimalstellen gerundet, wie Nullen vorhanden sind. Wenn die Zahl mehr Zahlzeichen zur Linken der Dezimaltrennung hat, als Nullen zur linken Seite der Dezimaltrennung im Formatausdruck vorhanden sind, werden die überzähligen Zahlzeichen ohne Änderung angezeigt.
(#)	Zahlzeichen-Platzhalter. Zeigt ein Zahlzeichen oder gar nichts an. Wenn an der Stelle des Ausdrucks ein Zahlzeichen steht, an der in der Formatzeichenkette ein #-Zeichen erscheint, wird es angezeigt. Sonst wird an dieser Stelle nichts angezeigt. Dieses Symbol funktioniert wie der 0 Zahlzeichen-Platzhalter, mit der Ausnahme, dass die vorangehenden und die anhängenden Nullen nicht angezeigt werden, wenn die Zahl ebenso viele oder weniger Zahlzeichen als #-Zeichen auf beiden Seiten der Dezimaltrennung im Formatausdruck enthält.
(.)	Dezimaler Platzhalter. In einigen Darstellungen wird ein Komma zur Tennung von Dezimalstellen verwendet. Der dezimale Platzhalter bestimmt, wie viele Zahlzeichen zur Linken und Rechten der Dezimalstellen-Trennung angezeigt werden. Wenn der Formatausdruck ausschließlich Nummernzeichen zur Linken dieses Symbols beinhaltet, beginnen Zahlen kleiner als 1 mit einer Dezimalstellen-Trennung. Um eine vorangehende Null mit Nachkommastellen anzuzeigen, verwenden Sie 0 als ersten Zahlzeichen-Platzhalter zur Linken der Dezimalstellen-Trennung. Das Zeichen, das tatsächlich als dezimaler Platzhalter in den formatierten Ausgabedaten verwendet wird, hängt von dem Zahlen-Format ab, das Ihr System erkennt.
(,)	Tausendertrennzeichen. In einigen Darstellungen wird ein Punkt als Tausendertrennzeichen verwendet. Das Tausendertrennzeichen trennt Tausender von Hundertern innerhalb einer Zahl, die vier oder mehr Stellen zur Linken der Dezimalstellen-Trennung besitzt. Die standardmäßige Verwendung des Tausendertrennzeichens wird definiert, wenn das Format ein Tausendertrennzeichen enthält, das von Zahlzeichen-Platzhaltern umgeben ist (0 oder #). Zwei nebeneinander stehende Tausendertrennzeichen oder ein Tausendertrennzeichen direkt zur Linken der

Dezimalstellen-Trennung (je nachdem, ob eine Dezimalstelle definiert ist, oder nicht) bedeutet eine "Skalierung der Zahl, indem man sie durch 1000 teilt und rundet, falls nötig." Beispielsweise können Sie die Format-Zeichenkette “##0,,” verwenden, um 100 Millionen als 100 darzustellen. Zahlen, die kleiner als eine Million sind, werden als 0 angezeigt. Zwei nebeneinanderstehende Tausendertrennzeichen in einer anderen Position als direkt zur Linken der Dezimallstellen-Trennung stellen eine Spezifizierung der Nutzung eines Tausendertrennzeichens dar. Das Zeichen, das tatsächlich als Tausendertrennzeichen in den formatierten Ausgabedaten verwendet wird, hängt von dem Zahlen-Format ab, das Ihr System erkennt.

Verwandte Befehle

Left$, Right$, Str$

Beispiel einer FmtStr$-Funktion

```
Function SendDateCode

    String d$, f$

    f$ = FmtStr$(10, "000.00")
    OpenCom #1
    Print #1, f$
    CloseCom #1
Fend
```

For...Next

S

Die Befehle For...Next werden zusammen verwendet, um eine Schleife herzustellen, in der Befehle, die sich zwischen den Befehlen For und Next befinden, mehrfach ausgeführt werden, wie vom Benutzer angegeben.

Syntax

For *var* **=** *initValue* **To** *finalValue* **[Step** *increment* **]**
statements
Next [*var*]

Parameter

var	Die Zählvariable, die mit der For...Next^-Schleife verwendet wird. Diese Variable wird normalerweise als Integer definiert, kann jedoch auch als Realvariable definiert werden.
initValue	Der Anfangswert für den Zähler *var*.
finalValue	Der Endwert des Zählers *var*. Sobald dieser Wert erreicht ist, ist die For...Next Schleife vollständig und die Ausführung wird mit der Anweisung, die auf den Next-Befehl folgt, fortgesetzt.
increment	Ein optionaler Parameter, der das Zähl-Inkrement für jede Ausführung der Next-Anweisung innerhalb der For...Next Schleife definiert. Diese Variable kann positiv oder negativ sein. Wenn der Wert negativ ist, muss der Erstwert der Variable größer sein als ihr Endwert. Wird der Inkrement-Wert weggelassen, inkrementiert (erhöht) das System automatisch um 1.
statements	Jede gültige SPEL+-Anweisung kann in die For...Next-Schleife eingefügt werden.

Beschreibung

For...Next führt einen Satz von Anweisungen innerhalb einer Schleife mit einer definierten Häufigkeit aus. Die **For**-Anweisung stellt den Anfang der Schleife dar. Die **Next**-Anweisung ist das Ende der Schleife. Wie oft die Anweisungen innerhalb der Schleife ausgeführt werden, wird mithilfe einer Variablen gezählt.

Der erste numerische Ausdruck (*initValue)* ist der Erstwert des Zählers. Dieser Wert kann positiv oder negativ sein, solange die Variable *finalValue* und die Step-Inkrementierung einander korrekt entsprechen.

Der zweite numerische Ausdruck (*finalValue)* ist der Endwert des Zählers. Dies ist der Wert der, sobald er erreicht ist, die Beendigung der For...Next-Schleife auslöst. Die Steuerung des Programms wird an den nächsten auf den Next-Befehl folgenden Befehl weitergegeben.

Programmanweisungen, die auf die **For-**Anweisung folgen, werden ausgeführt, bis ein Next-Befehl erreicht wird. Die Zählervariable (*var*) wird dann durch den Step-Wert inkrementiert, der durch den Parameter *increment* definiert ist. Wird die Step-Option nicht genutzt, wird der Zähler um 1 (eins) inkrementiert.

Die Zählervariable (*var*) wird dann mit dem Endwert verglichen. Wenn der Zählerstand kleiner oder gleich dem Endwert ist, werden die Anweisungen, die dem **For**-Befehl folgen, erneut ausgeführt. Wenn die Zählervariable größer als der Endwert ist, wird die Ausführung außerhalb der For...Next-Schleife verzweigt und mit dem Befehl fortgefahren, der direkt auf den **Next**-Befehl folgt.

Hinweise

Negative Step-Werte:

Wenn der Wert der Step-Inkrementierung (*increment*) negativ ist, wird die Zählervariable (*var*) bei jedem Durchlauf durch die Schleife dekrementiert (verringert) und der Erstwert muss größer sein als der Endwert, damit die Schleife funktioniert.

Eine Variable, die auf Next folgt, wird nicht benötigt:

Der Variablenname, der auf den Next-Befehl folgt, kann weggelassen werden. Für Programme, die verschachtelte **For...Next**-Schleifen beinhalten, wird jedoch empfohlen, den Variablennamen, der auf den Next-Befehl folgt, mit einzubeziehen, damit die Schleifen schnell identifiziert werden können.

Verwandte Befehle

Do...Loop

Beispiel für For...Next

```
Function fornext
    Integer counter
    For counter = 1 to 10
        Go Pctr
    Next counter

    For counter = 10 to 1 Step -1
        Go Pctr
    Next counter
Fend
```

Function...Fend

S

Eine Funktion ist eine Gruppe von Programmanweisungen, die eine **Function**-Anweisung als erste und eine Fend-Anweisung als letzte Anweisung beinhaltet.

Syntax

Function *funcName* [(*argList*)] [**As** *type*]
statements
Fend

Parameter

funcName	Name, der einer spezifischen Gruppe von Anweisungen gegeben wird, die zwischen die Befehle **Function** und **Fend** eingebunden werden. Der Funktionsname muss alphanumerische Zeichen enthalten und darf bis zu 32 Zeichen lang sein. Unterstriche sind ebenfalls zulässig.
argList	Optional. Liste von Variablen, die für Argumente stehen, die an die Funktionsprozedur weitergegeben werden, wenn diese aufgerufen wird. Multiple Variablen werden durch Kommata getrennt.

Das arglist-Argument hat die folgende Syntax:

[{**ByRef | ByVal**}] *varName* [( )] **As** *type*

ByRef	Optional. Spezifizieren Sie ByRef, um Veränderungen im Variablenwert von der Calling-Funktion sehen zu können.
ByVal	Optional. Spezifizieren Sie ByVal, damit Veränderungen im Variablenwert von der Calling-Funktion nicht gesehen werden können. Dies ist die Vorgabeeinstellung.
varName	Erforderlich. Name der Variablen, die für das Argument steht; folgt den Standardkonventionen für die Variablenbenennung.
As *type*	Erforderlich. Der Argumenttyp muss definiert werden.

Rückgabewerte

Wert, dessen Datentyp mit dem **As**-Ausdruck am Ende der Funktionsdeklaration definiert wird.

Beschreibung

Die **Function-**Anweisung zeigt den Beginn einer Gruppe von SPEL+-Anweisungen an. Um anzuzeigen, wo eine Funktion endet, wird die **Fend**-Anweisung verwendet. Alle Anweisungen zwischen **Function** und **Fend** werden als Teile der Funktion betrachtet.

Die durch **Function...Fend** definierte Kombination von Anweisungen kann als ‚Behältnis' betrachtet werden, in dem alle Anweisungen, die sich zwischen den Anweisungen **Function** und **Fend** befinden, zu der Funktion gehören. Eine Programmdatei kann multiple Funktionen beinhalten.

Verwandte Befehle

Call, Fend, Halt, Quit, Return, Xqt

Beispiel für Function...Fend

Das folgende Beispiel zeigt drei Funktionen innerhalb einer Datei. Die Funktionen mit den Namen *task2* und *task3* werden als Hintergrundtasks (Background Tasks) ausgeführt, während der Haupttask mit Namen *Main* im Vordergrund ausgeführt wird.

```
Function main
  Xqt 2, task2 'Führt task2 im Hintergrund aus
  Xqt 3, task3 'Führt task3 im Hintergrund aus
  '... weitere Anweisungen
Fend

Function task2
  Do
    On 1
    On 2
    Off 1
    Off 2
  Loop
Fend

Function task2
  Do
    On 10
    Wait 1
    Off 10
  Loop
Fend
```

Global-Anweisung

Deklariert Variablen mit globalem Gültigkeitsbereich. Globale Variablen sind frei zugänglich. **S**

Syntax

Global [**Preserve**] *dataType varName* [**(***subscripts***)**] [**,** *varName* [**(***subscripts***)**] **,** ...]

Parameter

Preserve	Optional. Wenn Perserve definiert ist, werden die Werte der Variablen beibehalten. Die Werte werden durch Projektänderungen gelöscht.
dataType	Datentypen wie z. B. Boolean, Integer, Long, Real, Double, Byte, oder String.
varName	Variablenname. Namen dürfen bis zu 32 Zeichen enthalten.
subscripts	*Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus:* *(ubound1, [ubound2], [ubound3])* *Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension.* *Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert.* *Die Gesamtzahl der Matrixelemente für Global Preserve-Variablen beträgt 100 bei Zeichenketten und 1000 bei allen anderen Typen.* *Die Gesamtzahl der Matrixelemente für globale Variablen beträgt 1000 bei Zeichenketten und 10000 bei allen anderen Typen.* *Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)*

Beschreibung

Globale Variablen sind Variablen, die in mehr als einer Datei desselben Projektes verwendet werden können. Sie werden gelöscht, wann immer eine Funktion vom Run-Fenster oder Benutzerfenster aus gestartet wird, es sei denn, sie sind mit der Preserve-Option deklariert.

Die Variable behält nach dem Ausschalten der Steuerung ihren Wert bei, wenn dies in der Preserve-Option eingestellt wurde.

Global Preserve-Variablen können ebenfalls mit der VB-Guide-Option verwendet werden.

Es wird empfohlen, den Namen globaler Variablen ein "g_"-Präfix voranzustellen, um globale Variablen im Programm leichter erkennbar zu machen. Zum Beispiel:

```
Global Long g_PartsCount
```

Verwandte Befehle

Boolean, Byte, Double, Integer, Long, Real, String

Beispiel einer Global-Anweisung

Das folgende Beispiel zeigt zwei verschiedene Programmdateien. Die erste Programmdatei definiert einige globale Variablen und initialisiert sie. Die zweite Datei verwendet eben diese globalen Variablen.

FILE1 (MAIN.PRG)

```
Global Integer status1
Global Real numsts

Function Main
   Integer I

   status1 = 10
```

Das folgende Beispiel zeigt zwei verschiedene Programmdateien. Die erste Programmdatei definiert einige globale Variablen und initialisiert sie. Die zweite Datei verwendet auch diese globalen Variablen.

FILE1 (MAIN.PRG)

```
Global Integer g_Status
Global Real g_MaxValue

Function Main

    g_Status = 10
    g_MaxValue = 1.1
    .
    .
Fend
```

FILE2 (TEST.PRG)

```
Function Test

    Print "status1 = , g_Status
    Print "MaxValue = , g_MaxValue
    .
    .
Fend
```

Go-Anweisung

> S

Bewegt den Arm mit einer PTP-Bewegung von seiner aktuellen Position zum spezifizierten Punkt oder in die X-, Y-, Z-, U-, V-, W-Position.
Der Go-Befehl kann eine beliebige Kombination von bis zu sechs Achsen gleichzeitig bewegen.

Syntax

Go destination [**CP**] [searchExpr] [!...!]

Parameter

destination	Der Zielort einer Bewegung, die einen Punktausdruck verwendet.
CP	Optional. Spezifiziert die CP-Bewegung.
searchExpr	Optional. Ein Till- oder Find-Ausdruck. **Till** | **Find** **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können hinzugefügt werden, um E/A-Befehle oder andere Befehler während der Bewegung auszuführen.

Beschreibung

Go bewegt alle Achsen des Roboteramrs mithilfe der PTP-Bewegung gleichzeitig. Das Ziel des Go-Befehls kann auf unterschiedliche Weise definiert werden:

- Verwenden eines spezifischen Punktes, zu dem hin die Bewegung ausgeführt wird. Zum Beispiel: **Go P1**.
- Verwenden einer expliziten Koordinatenposition, zu der hin die Bewegung ausgeführt wird. Zum Beispiel: **Go XY(50, 400, 0, 0)**.
- Verwenden eines Punktes mit Koordinatenversatz. Zum Beispiel: **Go P1 +X(50)**.
- Verwenden eines Punktes mit einem anderen Koordinatenwert. Zum Beispiel: **Go P1 :X(50)**.

Der Speed Befehl bestimmt die Armgeschwindigkeit für die Bewegung, die durch den Go-Befehl initiiert wurde. Der Accel-Befehl definiert die Beschleunigung.

Der **Go-**Befehl ist nicht in der Lage zu überprüfen, ob es einen Bereich gibt, in den sich der Arm physikalisch nicht bewegen kann, bevor die Bewegung selbst begonnen wird. Es ist dem Steuergerät daher möglich, eine verbotene Position auf dem Weg zu einem Zielpunkt zu finden. In diesem Fall kann der Arm abrupt zum Stehen kommen, was einen Stoß und den Servo-Aus-Zustand des Arms hervorrufen kann. Obschon sich die Zielposition innerhalb des Armbereichs befindet, gibt es einige Bewegungen, die nicht ausgeführt werden können, da der Arm einige der Zwischenpositionen nicht erreichen kann, die während der Bewegung benötigt werden.

Der **Go**-Befehl veranlasst den Arm immer dazu, bis zum Halt herunterzubremsen, bevor das Bewegungsziel erreicht wurde.

Hinweise

Der Unterschied zwischen Go und Move

Sowohl der Move- als auch der **Go**-Befehl veranlassen den Roboterarm zu einer Bewegung. Der primäre Unterschied zwischen den beiden Befehlen ist jedoch, dass der **Go**-Befehl eine PTP-Bewegung verursacht, während der Move-Befehl den Arm geradlinig bewegt. Der **Go**-Befehl wird verwendet, wenn es dem Benutzer primär um die Ausrichtung des Arms bei Erreichen des Punktes geht. Der **Move**-Befehl wird verwendet, wenn es wichtig ist, den Pfad des Roboterarms zu steuern, während er sich bewegt.

Der Unterschied zwischen Go und Jump

Sowohl der Jump- als auch der **Go**-Befehl veranlassen den Roboterarm zu einer PTP-Bewegung. Der Jump-Befehl hat jedoch ein zusätzliches Feature. Jump veranlasst den Endeffektor des Roboters dazu, zunächst bis zum LimZ-Wert zu steigen, sich dann in horizontaler Richtung zu bewegen, bis er sich über dem Zielpunkt befindet und schließlich abwärts zum Zielpunkt. Dies ermöglicht es, den Jump-Befehl zu verwenden, um das Umfahren von Hinderniss zu garantieren und, was wichtiger ist, um die Zykluszeiten für Bestückungsbewegungen zu verbessern.

Korrekte Geschwindigkeits- und Beschleunigungs-Befehle mit Go

Die Befehle Speed und Accel werden verwendet, um Geschwindigkeit und Beschleunigung des Manipulators während der **Go**-Bewegung zu definieren. Beachten Sie, dass sich die Befehle Speed und Accel auf PTP-Bewegungen beziehen (wie der für den **Go** Befehl), während die Befehle SpeedS und AccelS für linearinterpolierte und kreisinterpolierte Bewegungen verwendet werden.

Verwenden von Go mit der optionalen Till-Bedingung

Die optionale Till-Bedingung ermöglicht es dem Benutzer, eine Bedingung zu definieren, damit der Roboter an einer Zwischenposition bis zum Halt abbremst, noch bevor er die durch den **Go**-Befehl verursachte Bewegung beendet. Wenn die Till-Bedingung nicht erfüllt wird, bewegt sich der Roboter zu seiner Zielposition. Der Go-Befehl zusammen mit der Till-Bedingung kann auf zwei Arten verwendet werden, wie im Folgenden beschrieben:

(1) Go-Befehl mit Till-Bedingung

Überprüft, ob die aktuelle Till-Bedingung erfüllt wird. Wenn sie erfüllt wird, wird dieser Befehl durch Verzögerung und Anhalten des Roboters an einer Zwischenposition vervollständigt, bevor die Bewegung, die durch den **Go**-Befehl verursacht wurde, beendet wird.

(2) Go-Befehl mit Till-Bedingung, Sw(Eingangsnummer)-Bedingung, und Eingangsbedingung

Diese Version des Go-Befehls mit Till-Bedingung ermöglicht es dem Benutzer, die Till-Bedingung in derselben Zeile wie den Go-Befehl zu spezifizieren, anstatt die aktuelle Definition zu verwenden, die vorher für Till definiert wurde. Die spezifizierte Bedingung ist einfach eine Gegenprüfung zu einem der Eingänge. Dies wird durch Verwenden des Sw-Befehls erreicht. Der Benutzer kann überprüfen, ob die Eingänge aus- oder eingeschaltet sind und den Arm dazu veranlassen, anzuhalten, je nachdem, welche Bedingung spezifiziert ist. Dieses Feature funktioniert ähnlich wie eine Unterbrechung (Interrupt), bei der eine Bewegung unterbrochen (gestoppt) wird, sobald die Eingangs-Bedingung erfüllt ist. Wenn die Bedingung während der Roboterbewegung nie erfüllt wird, erreicht der Arm erfolgreich den durch *destination* definierten Punkt.

Verwenden von Go mit der optionalen Find-Bedingung

Die optionale Find-Bedingung ermöglicht es dem Benutzer, eine Bedingung zu definieren, die den Roboter veranlasst, während der Bewegung durch den **Go**-Befehl eine Position aufzuzeichnen. Der Go-Befehl mit der Find-Bedingung kann auf zwei Arten verwendet werden, wie im Folgenden beschrieben:

(1) **Go**-Befehl mit Find-Bedingung:
Überprüft, ob die aktuelle Find-Bedingung erfüllt wird. Wird sie erfüllt, so wird die aktuelle Position im Sonderpunkt FindPos gespeichert.

(2) Go-Befehl mit Find-Bedingung, Sw(Eingangsnummer)-Bedingung, und Eingangsbedingung:
Diese Version des Go-Befehls mit Find-Bedingung ermöglicht es dem Benutzer, die Find-Bedingung in derselben Zeile wie den Go-Befehl zu spezifizieren, anstatt die aktuelle Definition zu verwenden, die vorher für Find definiert wurde. Die spezifizierte Bedingung ist einfach eine Gegenprüfung zu einem der Eingänge. Dies wird durch Verwenden des Sw-Befehls erreicht. Der Benutzer kann überprüfen, ob die Eingänge aus- oder eingeschaltet sind und veranlassen, dass die aktuelle Position unter dem Sonderpunkt **FindPos** gespeichert wird.

Der Go-Befehl verursacht immer eine Verzögerung bis zum Stillstand

Der **Go**-Befehl veranlasst den Arm immer dazu, bis zum Halt herunterzubremsen, bevor das Bewegungsziel erreicht wird.

Mögliche Fehler

Versuch, sich außerhalb des Roboter-Arbeitsbereichs zu bewegen
Wenn mit dem **Go**-Befehl explizite Koordinaten verwendet werden, müssen Sie sicherstellen, dass sich die definierten Koordinaten innerhalb des Roboter-Arbeitsbereichs befinden. Jeglicher Versuch, den Roboter außerhalb dieses gültigen Arbeitsbereichs zu bewegen, ruft eine Fehlermeldung hervor.

Verwandte Befehle

!...! Parallelbearbeitung, Accel, Find, Jump, Move, Pass, Pn= (Point Assignment / Punktzuordnung), Pulse, Speed, Sw, Till

Beispiel einer Go-Anweisung

Das folgende Beispiel zeigt eine einfache PTP-Bewegung vom Punkt P0 zum Punkt P1 mit einer geradlinigen Rückbewegung zum Punkt P0. Später bewegt sich der Arm geradlinig zum Punkt P2, bis der 2. Eingang eingeschaltet wird. Wenn der 2. Eingang während der Bewegung einschaltet, verzögert sich der Arm bis zum vollständigen Stopp, bevor er den Punkt P2 erreicht und der nächste Programmbefehl ausgeführt wird.

Funktionsbeispiel

```
  Integer i

  Home
  Go P0
  Go P1
  For i = 1 to 10
    Go P(i)
  Next i
  Go P2 Till Sw(2) = On
  If Sw(2) = On Then
    Print "Input #2 came on during the move and"
    Print "the robot stopped prior to arriving on"
    Print "point P2."
  Else
    Print "The move to P2 completed successfully."
    Print "Input #2 never came on during the move."
  EndIf
Fend

Einige Syntax-Beispiele im Befehlseingabefenster sehen aus wie folgt:

>Go Here +X(50)      ' Bewegt den Arm nur in die X-Richtung, 50 mm aus
                       der
                     ' aktuellen Position
>Go P1               ' Einfaches Beispiel, um den Arm nach Punkt P1 zu
                       bewegen
>Go P1 :U(30)        ' Bewegt den Arm nach Punkt 1, verwendet aber +30
                       als Position,
                       zu der sich die U-Achse bewegen soll
>Go P1 /L            ' Bewegt den Arm zu P1, stellt aber sicher, dass
                       der Arm in der linken Position anhält.
>Go XY(50, 450, 0, 30' Bewegt den Arm in die Position X=50, Y=450, Z=0,
                       U=30
```

<Ein weiteres Code-Beispiel>

```
Till Sw(1)=0 And Sw(2) = On  ' Legt die Till-Bedingungen für die
                             ' Eingänge 1 und 2 fest
Go P1 Till                   ' Stoppt den Arm, wenn die aktuelle Till-
                               Bedingung,
                             ' die in der vorangegangenen Zeile
                               definiert wurde, erfüllt ist.
Go P2 Till Sw(2) = ON        ' Stoppt, wenn der 2. Eingang eingeschaltet
                               ist
Go P3 Till                   ' Stoppt den Arm, wenn die aktuelle Till-
                               Bedingung,
                             ' die in der vorangegangenen Zeile
                               definiert wurde, erfüllt ist.
```

GoSub...Return

S

GoSub überträgt die Programmsteuerung an ein Unterprogramm. Sobald das Unterprogramm vollständig ist, kehrt die Programmsteuerung wieder in die Zeile zurück, die auf den **GoSub**-Befehl folgt, der das Unterprogramm initiiert hatte.

Syntax

GoSub { *label* }

{ *label***:**}
statements
Return

Parameter

label	Wenn der Benutzer ein Label spezifiziert, springt die Programmausführung zu der Zeile, in der sich das Label befindet. Das Label kann bis zu 32 Zeichen beinhalten. Das erste Zeichen muss jedoch ein Buchstabe sein (keine Zahl).

Beschreibung

Der **GoSub-**Befehl bewirkt, dass die Programmsteuerung zu dem durch den Benutzer definierten Anweisungslabel verzweigt wird. Das Programm führt dann die Anweisung in dieser Zeile aus und fährt mit der Ausführung in den darauf folgenden Zeilennummern fort, bis es auf einen Return-Befehl trifft. Der Return-Befehl veranlasst die Programmsteuerung, zurück zu der Zeile zu wechseln, die direkt auf diejenige folgt, die **GoSub** ursprünglich initiiert hatte. (D. h., der **GoSub**-Befehl veranlasst die Ausführung eines Unterprogramms, wobei die Ausführung anschließend zu der Anweisung zurückkehrt, die auf den **GoSub**-Befehl folgt.) Stellen Sie sicher, dass jedes Unterprogramm mit Return beendet wird. Diese Vorgehensweise veranlasst die Programmausführung, zu der Zeile zurückzukehren, die auf den **GoSub**-Befehl folgt.

Mögliche Fehler

Verzweigen zu einer nicht existierenden Anweisung

Wenn der **GoSub**-Befehl versucht, die Steuerung zu einem nicht existenten Label zu verzweigen, wird der Fehler Nr. 3108 ausgegeben.

Return-Befehl ohne GoSub-Befehl gefunden

Ein Return-Befehl wird verwendet, um aus einem Unterprogramm in das Ursprungsprogramm „zurückzukehren", das den **GoSub**-Befehl ausgegeben hat. Wenn ein Return-Befehl gefunden wird, ohne dass dem ein **GoSub**-Befehl voran gegangen ist, wird der Fehler Nr. 2383 ausgegeben. Ein alleinstehender Return-Befehl ist ohne Funktion, da das System nicht weiß, wohin es zurückkehren soll.

Verwandte Befehle

GoTo, OnErr, Return

Beispiel einer GoSub-Anweisung

Das folgende Beispiel zeigt eine einfache Funktion, die einen GoSub-Befehl verwendet, um zu einem Label zu verzweigen, einige E/A-Befehle auszuführen und dann zurückzukehren.

```
Function  main
    Integer var1, var2

    GoSub checkio  'GoSub unter Verwendung von Label
    On 1
    On 2
    Exit Function

checkio:    'Hier startet das Unterprogramm
    var1 = In(0)
    var2 = In(1)
    If var1 = 1 And var2 = 1 Then
        On 1
    Else
        Off 1
    EndIf
    Return  'Hier endet das Unterprogramm
Fend
```

GoTo-Anweisung

Der **GoTo-**Befehl bewirkt, dass die Programmsteuerung ohne Bedingung zu einem angegebenen Label verzweigt wird.

S

Syntax

GoTo { *label* }

Parameter

label	Die Programmausführung springt in die Zeile, in der sich das Label befindet. Das Label kann bis zu 32 Zeichen beinhalten. Das erste Zeichen muss jedoch ein Buchstabe sein (keine Zahl).

Beschreibung

Der **GoTo-**Befehl bewirkt, dass die Programmsteuerung zu dem durch den Benutzer definierten Anweisungslabel verzweigt wird. Das Programm führt die Anweisung in der betreffenden Zeile aus und fährt von dort mit der Ausführung fort. **GoTo** wird am häufigsten für den Sprung zu einem Exit-Label auf Grund einer Fehlermeldung verwendet.

Hinweise

Verwendung zu vieler GoTo-Anweisungen

Sie sollten nicht zuviele GoTo-Anweisungen verwenden, da dies in einem Programm dazu führen kann, dass dies nur schwer verständlich ist. Generell sollten so wenig **GoTos** wie möglich verwendet werden. Einige **GoTo**s sind fast immer notwendig. Es kann jedoch leicht zu Problemen führen, wenn man sich mit zu vielen **GoTo**-Anweisungen innerhalb des Quellcodes bewegt.

Verwandte Befehle

GoSub, OnErr

Beispiel einer GoTo-Anweisung

Das folgende Beispiel zeigt eine einfache Funktion, die einen GoTo-Befehl verwendet, um zu einem Zeilenlabel zu verzweigen.

```
Function  main

    If Sw(1) = Off Then
        GoTo mainAbort
    EndIf
    Print "Input 1 was On, continuing cycle"
    .
    .
    Exit Function

mainAbort:
    Print "Input 1 was OFF, cycle aborted!"
Fend
```

Halt-Anweisung

Unterbricht vorübergehend die Ausführung eines spezifizierten Tasks.

S

Syntax

Halt *taskIdentifier*

Parameter

taskIdentifier	Taskname oder Integer-Ausdruck, der für die Tasknummer steht. Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, liegt der Bereich für normale Tasks zwischen 1 und 16 und für Traptasks zwischen 257 und 261.

Beschreibung

Halt unterbricht vorübergehend die Ausführung des aktuellen Tasks, wie durch den Tasknamen oder die -nummer definiert.

Verwenden Sie Resume, um mit der Ausführung des Tasks an der Stelle fortzufahren, wo er unterbrochen wurde. Verwenden Sie Quit, um die Ausführung eines Tasks vollständig abzubrechen. Um den Taskstatus auszuführen, klicken Sie auf das *Task Manager*-Icon in der EPSON RC+-Werkzeugleiste.

Verwandte Befehle

Quit, Resume, Xqt

Beispiel einer Halt-Anweisung

Das folgende Beispiel zeigt eine Funktion mit Namen "flicker", die von *Xqt* gestartet wird, temporär von *Halt* gestoppt und anschließend von Resume wieder aufgenommen wird.

```
Function  main
    Xqt 2, flicker   'Führt die Flicker-Funktion aus

  Do
    Wait 3           'Führt den Task Flicker 3 Sekunden lang aus
    Halt flicker

    Wait 3           'Hält den Task Flicker 3 Sekunden lang an
    Resume flicker

  Loop
Fend
    '

Function flicker
  Do
    On 1
    Wait 0.2
    Off 1
    Wait 0.2
  Loop
Fend
```

Hand-Anweisung

Stellt die Armausrichtung eines Punktes ein.

> S

Syntax

(1) Hand *point*, [*Lefty* | *Righty*]
(2) Hand

Parameter

point	*Pnumber or P(expr) or point label.*
Lefty | *Righty*	**Ausrichtung des Arms**

Rückgabewerte

Wenn beide Parameter ausgelassen werden, wird die Armausrichtung für die aktuelle Roboterposition angezeigt.
Wenn *Lefty* | *Righty* weggelassen wird, wird die Armausrichtung für den spezifizierten Punkt angezeigt.

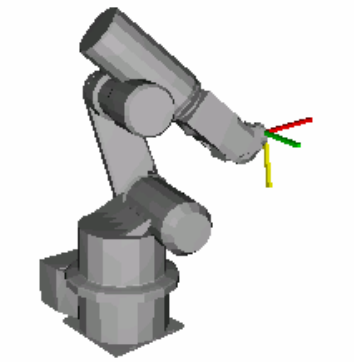
Verwandte Befehle

Elbow, Hand-Funktion, J4Flag, J6Flag, Wrist

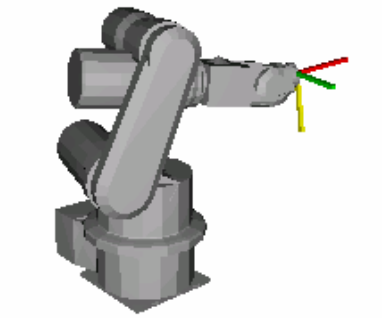
Beispiel einer Hand-Anweisung

```
Hand P0, Lefty
Hand pick, Righty
Hand P(myPoint), myHand

P1 = -364.474, 120.952, 469.384, 72.414, 1.125, -79.991
```



```
Hand P1, Righty
Go P1
```



```
Hand P1, Lefty
Go P1
```

Hand-Funktion

F

Gibt die Armausrichtung eines Punktes aus.

Syntax

Hand [(*point*)]

Parameter

point Optional. Punktausdruck. Wird *point* weggelassen, wird die Armausrichtung der aktuellen Roboterposition ausgegeben.

Rückgabewerte

1 Righty (/R)
2 Lefty (/L)

Verwandte Befehle

Elbow, Wrist, J4Flag, J6Flag

Beispiel einer Hand-Funktion

```
Print Hand(pick)
Print Hand(P1)
Print Hand
Print Hand(P1 + P2)
```

Here-Anweisung

S

Teacht einen Roboterpunkt an der aktuellen Position.

Syntax

Here *point*

Parameter

point **P***number* oder **P**(*expr*) oder Punktlabel.

Verwandte Befehle

Here-Funktion

Beispiel einer Here-Anweisung

```
Here P1
Here pick
```

Here-Funktion

F

Gibt die aktuelle Roboterposition als Punkt aus.

Syntax

Here

Rückgabewerte

Ein Punkt, der für die aktuelle Roboterposition steht.

Beschreibung

Here wird verwendet, um die aktuelle Position des aktuellen Manipulators auszugeben.

Verwandte Befehle

Here-Anweisung

Beispiel einer Here-Funktion

```
P1 = Here
```

Hex$-Funktion

Gibt eine Zeichenkette aus, die die angegebene Zahl im Hexadezimalformat darstellt. > F

Syntax

Hex$*(number)*

Parameter

number Integer-Ausdruck.

Rückgabewerte

Gibt eine Zeichenkette aus, die die ASCII-Darstellung der Zahl im Hexadezimalformat enthält.

Beschreibung

Hex$** gibt eine Zeichenkette aus, die die angegebene Zahl im Hexadezimalformat darstellt. Jedes Zeichen liegt zwischen 0 und 9 oder A und F. **Hex$ wird insbesondere verwendet, um die Ergebnisse der Stat-Funktion zu prüfen.

Verwandte Befehle

Str$, Stat, Val

Beispiel einer Hex$-Funktion

```
> print hex$(stat(0))
A00000
> print hex$(255)
FF
```

Home-Anweisung

Bewegt den Roboterarm in die durch den Benutzer definierte Home-Position. > S

Syntax

Home

Beschreibung

Führt bei niedriger Geschwindigkeit eine PTP-Bewegung zur Home-Position (Standby) durch, die durch den HomeSet-Befehl definiert ist; in der Home-Reihenfolge, die durch *Hordr* definiert ist.

Bei SCARA-Robotern kehrt üblicherweise die Z-Achse (J3) als erstes in die HomeSet-Position zurück, dann kehren die Achsen J1, J2 und J4 gleichzeitig in ihre entsprechenden HomeSet-Koordinatenpositionen zurück. Der Hordr-Befehl kann die Reihenfolge der Rückkehr der Achsen in ihre Home-Positionen verändern.

Hinweis

Ausgang des Home-Status:

Wenn sich der Roboter in seiner Home-Position befindet, ist der Home-Ausgang der Steuerung eingeschaltet.

Mögliche Fehler

Homingversuch ohne definierte HomeSet-Werte

Bei dem Versuch, den **Home**-Befehl auszuführen, ohne die HomeSet-Werte zu setzen, wird der Fehler Nr. 143 ausgegeben.

Verwandte Befehle

HomeClr, HomeDef, Hordr

Beispiel einer Home-Anweisung

Der Home-Befehl kann in einem Programm wie diesem verwendet werden:

```
Function InitRobot
   Reset
   If Motor = Off Then
      Motor On
   EndIf
   Home
Fend
```

```
Oder er kann folgendermaßen vom Befehlseingabefenster aus ausgegeben
werden:

> home
>
```

HomeClr-Funktion

F

Löscht die Definition der Home-Position.

Syntax

HomeClr

Verwandte Befehle

HomeDef, HomeSet

Beispiel einer HomeClr-Funktion

In diesem Beispiel wird die **HomeClr**-Funktion in einem Programm verwendet:

```
Function ClearHome

   If HomeDef = True Then
      HomeClr
   EndIf
Fend
```

HomeDef-Funktion

F

Gibt Informationen darüber aus, ob die Home-Position definiert wurde.

Syntax

HomeDef

Rückgabewerte

Wahr, wenn die Home-Position definiert wurde, sonst Falsch.

Verwandte Befehle

HomeClr, HomeSet

Beispiel einer HomeDef-Funktion

In diesem Beispiel wird die **HomeDef**-Funktion in einem Programm verwendet:

```
Function DisplayHomeSet

   Integer i


If HomeDef = False Then      Print "Home is not defined"
   Else
      Print "Home values:"
      For i = 1 To 4
         Print "J", i, " = ", HomeSet(i)
      Next i
   EndIf
Fend
```

HomeSet-Anweisung

Definiert die Home-Position und zeigt sie an.

> S

Syntax

(1) HomeSet *j1Pulses*, *j2Pulses*, *j3Pulses*, *j4Pulses*, [*j5Pulses*], [*j6Pulses*]
(2) HomeSet

Parameter

j1Pulses	Der Encoder-Pulse-Wert der Home-Position für die 1. Achse.
j2Pulses	Der Encoder-Pulse-Wert der Home-Position für die 2. Achse.
j3Pulses	Der Encoder-Pulse-Wert der Home-Position für die 3. Achse.
j4Pulses	Der Encoder-Pulse-Wert der Home-Position für die 4. Achse.
j5Pulses	Optional bei 6-Achsrobotern. Der Encoder-Pulse-Wert der Home-Position für die 5. Achse.
j6Pulses	Optional bei 6-Achsrobotern. Der Encoder-Pulse-Wert der Home-Position für die 6. Achse.

Rückgabewerte

Zeigt die Pulse-Werte an, die für die aktuelle Home-Position definiert sind, wenn die Parameter weggelassen werden.

Beschreibung

Ermöglicht es dem Benutzer, eine neue Home-Position (Standby) zu definieren, indem er die Pulse-Werte für jede der Roboterachsen eingibt.

Mögliche Fehler

Homingversuch ohne definierte HomeSet-Werte:

Bei dem Versuch, den **HomeSet**-Befehl auszuführen, ohne die HomeSet-Werte zu setzen, wird der Fehler Nr. 2228 ausgegeben.

Der Versuch, HomeSet-Werte anzuzeigen, ohne dass HomeSet-Werte definiert sind:

Bei dem Versuch, Pulse-Werte für eine Home-Position anzuzeigen, ohne dass HomeSet-Werte definiert sind, wird der Fehler Nr. 2228 ausgegeben.

Verwandte Befehle

Home, HomeClr, HomeDef, Hordr, Pls

Beispiel einer HomeSet-Anweisung

In den folgenden Beispielen wird die Anweisung vom Befehlseingabefenster aus ausgeführt:

```
> homeset 0,0,0,0  'Setzt die Home-Position auf 0,0,0,0
> homeset
  0  0
  0  0

> home  'Der Roboter kehrt zur Home-Position 0,0,0,0 zurück
```

Die Pls-Funktion wird verwendet, um die aktuelle Position des Arms als Home-Position zu definieren.

```
> homeset Pls(1), Pls(2), Pls(3), Pls(4)
```

HomeSet-Funktion

F

Gibt die Pulse-Werte der Home-Position für die angegebene Achse aus.

Syntax

HomeSet(*jointNumber)*

Parameter

jointNumber Integer-Ausdruck, der für die Achsnummer steht, für die der HomeSet-Wert ausgegeben werden soll.

Rückgabewerte

Gibt den Pulse-Wert der Home-Position der Achse aus. Wenn die *jointNumber* 0 beträgt, wird 1 ausgegeben, wenn HomeSet ausgeführt wurde und 0, wenn HomeSet nicht ausgeführt wurde.

Verwandte Befehle

HomeSet-Anweisung

Beispiel einer HomeSet-Funktion

In diesem Beispiel wird die **HomeSet**-Funktion in einem Programm verwendet:

```
Function DisplayHomeSet

   Integer i

   If HomeSet(0) = 0 Then
      Print "HomeSet is not defined"
   Else
      Print "HomeSet values:"
      For i = 1 To 4
         Print "J", i, " = ", HomeSet(i)
      Next i
   EndIf
Fend
```

Hordr-Anweisung

Definiert die Reihenfolge, in der die Achsen in ihre Home-Position zurückkehren oder zeigt sie an.

> S

Syntax

(1) Hordr *step1, step2, step3, step4*, [*step5*] ,[*step6*]
(2) Hordr

Parameter

step1	Bitmuster, das festlegt, welche Achsen während des 1. Schrittes des Homing-Prozesses in ihre Home-Position zurückgeführt werden sollen.
step2	Bitmuster, das festlegt, welche Achsen während des 2. Schrittes des Homing-Prozesses in ihre Home-Position zurückgeführt werden sollen.
step3	Bitmuster, das festlegt, welche Achsen während des 3. Schrittes des Homing-Prozesses in ihre Home-Position zurückgeführt werden sollen.
step4	Bitmuster, das festlegt, welche Achsen während des 4. Schrittes des Homing-Prozesses in ihre Home-Position zurückgeführt werden sollen.
step5	Bei 6-Achsrobotern. Bitmuster, das festlegt, welche Achsen während des 5. Schrittes des Homing-Prozesses in ihre Home-Position zurückgeführt werden sollen.
step6	Bei 6-Achsrobotern. Bitmuster, das festlegt, welche Achsen während des 6. Schrittes des Homing-Prozesses in ihre Home-Position zurückgeführt werden sollen.

Rückgabewerte

Zeigt die aktuellen Einstellungen der *Home-Reihenfolge* an, wenn die Parameter weggelassen werden.

Beschreibung

Hordr definiert die Bewegungsreihenfolge für den Home-Befehl. (Die Anweisung definiert also, welche Achse zuerst in ihre Home-Position zurückgeführt werden soll, welche als zweite, dritte, usw.)

Der Zweck des **Hordr-**Befehls ist es, dem Benutzer zu ermöglichen, die Reihenfolge zu ändern, in der die Achsen in ihre Home-Position zurückgeführt werden sollen. Die Home-Reihenfolge ist je nach Robotertyp in 4 bzw. 6 Schritte unterteilt. Der Benutzer verwendet dann **Hordr**, um die spezifischen Achsen zu definieren, die sich während jedes Schrittes in die Home-Position bewegen. Beachten Sie, dass mehr als eine Achse definiert werden kann, um sich während eines einzigen Schrittes in die Home-Position zu bewegen. Dies bedeutet, dass alle Achsen potentiell zur selben Zeit in ihre Home-Position zurückgeführt werden können. Bei 4-Achsrobotern wird jedoch empfohlen, im Normalfall zunächst die Z-Achse in ihre Home-Position zu bewegen (in Schritt 1) und die anderen Achsen in den darauf folgenden Schritten folgen zu lassen. (Siehe Hinweis unten)

Der **Hordr**-Befehl setzt voraus, dass ein Bitmuster für jeden der Schritte definiert wurde. Jeder Achse wird ein spezifisches Bit zugeordnet. Wenn das Bit für einen bestimmten Schritt auf 1 gesetzt wird, bewegt sich die entsprechende Achse in ihre Home-Position. Wenn das Bit nicht gesetzt ist (0), bewegt sich die entsprechende Achse während dieses Schrittes nicht in ihre Home-Position. Die Achsen-Bitmuster werden wie folgt zugeordnet:

Achse:	**1**	**2**	**3**	**4**	**5**	**6**
Bitnummer:	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
Binärcode:	&B0001	&B0010	&B0100	&B1000	&B10000	&B100000

Verwandte Befehle

Home, HomeSet

Beispiel einer Hordr-Anweisung

Im Folgenden finden Sie einige Befehlseingabefensterbeispiele für einen 4-Achsroboter:

In diesen Beispielen wird die Home-Reihenfolge folgendermaßen bestimmt: J3 im ersten Schritt, J1 im zweiten Schritt, J2 im dritten Schritt und J4 im vierten Schritt. Die Reihenfolge wird mit Binärwerten spezifiziert.

```
>hordr  &B0100, &B0001, &B0010, &B1000
```

In diesen Beispielen wird die Home-Reihenfolge folgendermaßen bestimmt: J3 im ersten Schritt, J1, J2 und J4 gleichzeitig im zweiten Schritt. Die Reihenfolge wird mit Dezimalwerten spezifiziert.

```
>hordr  4, 11, 0, 0
```

Dieses Beispiel zeigt die aktuelle Home-Reihenfolge in Dezimalzahlen an.

```
>hordr
4, 11, 0, 0
>
```

Hordr-Funktion

> F

Gibt den Hordr-Wert für einen spezifizierten Schritt aus.

Syntax

Hordr(*stepNumber*)

Parameter

stepNumber Integer-Ausdruck, der für den auszugebenden Hordr-Schritt steht.

Rückgabewerte

Integer, der den Hordr-Wert für den angegebenen Schritt enthält.

Verwandte Befehle

Home, HomeSet

Beispiel einer Hordr-Funktion

```
Integer a
a = Hordr(1)
```

Hour-Anweisung

Zeigt die Betriebszeit der Steuerung an.

Syntax

Hour

Beschreibung

Zeigt die Zeit an, die die Steuerung bereits eingeschaltet ist und SPEL ausführt (Kumulierte Betriebszeit). Die Zeit wird immer in ganzen Stunden angezeigt.

Verwandte Befehle

Time

Beispiel einer Hour-Anweisung

Das folgende Beispiel wird vom Befehlseingabefenster aus ausgeführt:

```
> hour
2560
>
```

Hour-Funktion

F

Gibt die Betriebszeit der Steuerung aus.

Syntax

Hour

Rückgabewerte

Gibt die kumulierte Betriebszeit des Steuergerätes aus (reelle Zahl, in Stunden).

Verwandte Befehle

Time

Beispiel einer Hour-Funktion

```
Print "Number of controller operating hours: ", Hour
```

If…Then…Else…EndIf-Anweisung

Führt Befehle auf der Basis einer angegebenen Bedingung aus.

S

Syntax

(1) **If** *condition* **Then**
stmtT1
.
.
[ElseIf *condition* **Then]**
stmtT1
.
.
[Else]
stmtF1
.
.
EndIf

(2) **If** *condition* **Then** *stmtT1* **[;** *stmtT2***...]** **[Else** *stmtF1* **[;** *stmtF2***...]]**

Parameter

condition	Jede gültige Bedingung, die ein Wahres (beliebige Zahl außer 0) oder Falsches Resultat ausgibt (als 0 ausgegeben). (Beispielbedingungen s.u.)
stmtT1	Wird ausgeführt, wenn die Bedingung Wahr ist. (Hier können Mehrfachanweisungen in geblocktem **If...Then...Else** -Stil geschrieben werden.)
stmtF1	Wird ausgeführt, wenn die Bedingung Falsch ist. (Hier können Mehrfachanweisungen in geblocktem **If...Then...Else** -Stil geschrieben werden.)

Beschreibung

(1) **If...Then...Else** führt stmtT1 aus, usw., wenn die bedingte Anweisung Wahr ist. Ist die Bedingung Falsch, wird stmtF1, usw., ausgeführt. Das **Else-**Element des **If...Then...Else**-Befehls ist optional. Wird die **Else**-Anweisung weggelassen und die bedingte Anweisung ist Falsch, wird die Anweisung ausgeführt, die auf die EndIf-Anweisung folgt. Für geblockte **If...Then...Else**-Anweisungen muss die EndIf-Anweisung den Block schließen, unabhängig davon, ob ein **Else** genutzt wird oder nicht.

(2) **If...Then...Else** kann auch nicht-geblockt verwendet werden. Dies ermöglicht, Anweisungen für **If...Then...Else** in dieselbe Zeile zu schreiben. Bitte beachten Sie, dass die EndIf–Anweisung nicht benötigt wird, wenn **If...Then...Else** in nicht geblockter Form verwendet wird. Wenn die in dieser Zeile spezifizierte Bedingung erfüllt wird (Wahr), werden die Anweisungen zwischen Then und Else ausgeführt. Wenn die Bedingung nicht erfüllt wird (Falsch), werden die Anweisungen ausgeführt, die auf **Else** folgen. Das **Else-**Element von **If...Then...Else** wird nicht benötigt. Existiert kein **Else**-Schlüsselwort, wird die Steuerung an die nächste Anweisung im Programm weitergegeben, wenn die **If**-Bedingung Falsch ist.

Der logische Wert der bedingten Anweisung ist jede Zahl außer 1, wenn die Anweisung Wahr ist und 0, wenn die Anweisung Falsch ist.

Hinweise

Beispielbedingungen:

a = b	:a ist gleich b
a < b	:b ist größer a
a >= b	:a ist größer / gleich b
a <> b	:a ist ungleich b
a > b	:b ist kleiner a
a <= b	:a ist kleiner / gleich b

Logische Vorgänge wie And, Or und Xor können ebenfalls verwendet werden.

Verwandte Befehle

Else, Select...Case, Do...Loop

Beispiel einer If/Then/Else-Anweisung

<Einzeilige If...Then...Else>

Das folgende Beispiel zeigt eine einfache Funktion, die einen Eingang überprüft, um festzustellen, ob ein bestimmter Ausgang ein- oder ausgeschaltet werden soll. Dieser Task könnte ein Background-E/A-Task sein, der fortwährend läuft.

```
Function  main
    Do
       If Sw(0) = 1 Then On 1 Else Off 1
    Loop
Fend
```

<Geblockte If...Then...Else>

Das folgende Beispiel zeigt eine einfaché Funktion, die einige Eingänge überprüft und den Status dieser Eingänge druckt.

```
If Sw(0) = 1 Then Print "Input0 ON" Else Print "Input0 OFF"
'
If Sw(1) = 1 Then
    If Sw(2) = 1 Then
       Print "Input1 On and Input2 ON"
    Else
       Print "Input1 On and Input2 OFF"
    EndIf
Else
    If Sw(2) = 1 Then
       Print "Input1 Off and Input2 ON"
    Else
       Print "Input1 Off and Input2 OFF"
    EndIf
EndIf
```

<Beispiele mit anderer Syntax>

```
If x = 10 And y = 3 Then GoTo 50
If test <= 10 Then Print "Test Failed"
If Sw(0) = 1 Or Sw(1) = 1 Then Print "Everything OK"
```

In-Funktion

Gibt den Status des spezifizierten Eingangsports aus. Jeder Port umfasst 8 Eingangskanäle. **F**

Syntax

In(*portNumber*)

Parameter

portNumber Integer-Zahl, die für ein Achtel eines Bitports (ein Byte) steht.

Rückgabewerte

Gibt einen Integer-Wert zwischen 0 und 255 aus. Der Rückgabewert beträgt 8 Bits. Jedes dieser Bits entspricht einem Eingangskanal.

Beschreibung

In gibt Ihnen die Möglichkeit, die Werte von 8 Eingangskanälen gleichzeitig zu betrachten. Der In-Befehl kann verwendet werden, um den Status der 8 E/A-Kanäle in einer Variable zu speichern. Alternativ kann er auch mit dem Wait-Befehl verwendet werden, um zu warten, bis eine spezifische Bedingung erfüllt ist, die mehr als einen E/A-Kanal einschließt.

Da 8 Kanäle zeitgleich überprüft werden, liegen die Rückgabewerte im Bereich von 0 bis 255. Bitte beachten Sie die Tabelle unten, um zu sehen, wie die Rückgabewerte den einzelnen Eingangskanälen entsprechen.

Eingangskanal-Ergebnis (unter Verwendung von Port 0)

Rückgabewerte	7	6	5	4	3	2	1	0
1	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
5	Aus	Aus	Aus	Aus	Aus	Ein	Aus	Ein
15	Aus	Aus	Aus	Aus	Ein	Ein	Ein	Ein
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Eingangskanal-Ergebnis (unter Verwendung von Port 3)

Rückgabewerte	31	30	29	28	27	26	25	24
3	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
7	Aus	Aus	Aus	Aus	Aus	Ein	Ein	Ein
32	Aus	Aus	Ein	Aus	Aus	Aus	Aus	Aus
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Verwandte Befehle

InBCD, MemIn, MemOff, MemOn, MemSw, Off, On, OpBCD, Oport, Out, Sw, Wait

Beispiel einer In-Funktion

Im unten stehenden Beispiel wird davon ausgegangen, dass die Eingangskanäle 28, 29, 30 und 31 mit sensorischen Geräten verbunden sind, so dass die Anwendung nicht startet, bevor jedes dieser Geräte ein ON-Signal ausgibt und damit Startbereitschaft signalisiert. Das Programmbeispiel erhält den aktuellen Wert der letzten 8 Eingänge und stellt vor dem Fortfahren sicher, dass die Kanäle 28, 29, 30 und 31 eingeschaltet sind. Wenn sie nicht eingeschaltet sind (d. h. sie geben einen Wert von 1 aus) wird eine Fehlermeldung an den Benutzer ausgegeben und der Task wird angehalten.

Im Programm wird die Variable "var1" mit der Zahl 239 verglichen, da das Ergebnis von In(3) 240 oder größer sein muss, damit die Eingänge 28, 29, 30 und 31 alle eingeschaltet sein können. (Die Eingänge 24, 25, 26 und 27 werden in diesem Fall vernachlässigt, so dass jeder Wert zwischen 240 und 255 dem Programm gestattet, fortzufahren.)

```
Function  main
    Integer var1
    var1 = In(3)  'Holt die letzten 8 Eingänge
    If var1 > 239 Then
        Go P1
        Go P2
        'Hier werden anderen Bewegungs-Anweisungen ausgeführt
        '.
        '.
    Else
        Print "Error in initialization!"
        Print "Sensory Inputs not ready for cycle start"
        Print "Please check inputs 28,29,30 and 31 for"
        Print "proper state for cycle start and then"
        Print "start program again"
    EndIf
Fend
```

Es ist nicht möglich, Eingänge vom Befehlseingabefenster aus zu setzen, aber man kann sie überprüfen. Für die folgenden Beispiele wird davon ausgegangen, dass die Eingabekanäle 1, 5, 15 und 30 eingeschaltet sind.. Alle anderen Eingänge sind ausgeschaltet.

```
> print In(0)
34
> print In(1)
128
> print In(2)
0
> print In(3)
64
```

InBCD-Funktion

Gibt den Eingangsstatus von 8 Eingängen im BCD-Format aus. (Binär codierte Dezimazahl) **F**

Syntax

InBCD(*portNumber***)**

Parameter

portNumber Integer-Zahl, die für ein Achtel eines Bitports (ein Byte) steht.

Rückgabewerte

Gibt den Eingangsstatus des Eingansports (0-99) als binär codierte Dezimalzahl (0-9) aus.

Beschreibung

InBCD liest gleichzeitig 8 Eingangskanäle unter Verwendung des BCD-Formats. Der *portNumber*-Parameter für den InBCD-Befehl definiert, welche 8er-Eingangsgruppe gelesen werden soll, wobei *portNumber* = 0 für die Eingänge 0-7 steht , *portNumber* = 1 für die Eingänge 8-15 steht, etc.

Der sich ergebende Wert der 8 Eingänge wird im BCD-Format ausgegeben. Der Rückgabewert kann eine oder zwei Stellen zwischen 0 und 99 haben. Die erste Stelle (oder die Zehnerstelle) entspricht den oberen 4 Ausgängen in der Gruppe von 8 Ausgängen, ausgewählt durch *portNumber*. Die zweite Stelle (oder die Einerstelle) entspricht den unteren 4 Ausgängen in der Gruppe von 8 Ausgängen, ausgewählt durch *portNumber*.

Da gültige Ausgänge im BCD-Format im Bereich von 0 bis 9 für jede Stelle liegen, kann nicht jede E/A-Kombination ausgeben werden. Die Tabelle unten zeigt einige der möglichen E/A-Kombinationen und ihre entsprechenden *outnum*-Werte, davon ausgehend, dass die *portNumber* 0 ist.

Eingangs-Einstellungen (Eingangsnummer)

Rückgabewerte	7	6	5	4	3	2	1	0
01	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
02	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Aus
03	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
08	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Aus
09	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Ein
10	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Aus
11	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Ein
99	Ein	Aus	Aus	Ein	Ein	Aus	Aus	Ein

Bitte beachten Sie, dass das BCD-Format nur die Spezifizierung von Dezimalwerten zulässt. Dies bedeutet, dass es durch die Verwendung des BCD-Formats nicht möglich ist, einen gültigen Wert auszugeben, wenn alle Eingänge für einen bestimmten Port gleichzeitig eingeschaltet sind und der **InBCD**-Befehl verwendet wird. Der größtmögliche Wert, der von **InBCD** ausgegeben werden kann, ist 99. Für den Fall, dass der Rückgabewert 99 beträgt, sind die Eingänge 0, 3, 4 und 7 eingeschaltet und alle anderen ausgeschaltet.

Hinweise

Der Unterschied zwischen InBCD und In

Die Befehle **InBCD** und In sind in der SPEL+-Sprache sehr ähnlich. Es gibt jedoch einen Hauptunterschied zwischen diesen Befehlen. Dieser Unterschied wird im Folgenden aufgezeigt:

- Der **InBCD**-Befehl verwendet das BCD-Format, um das Format des Rückgabewertes für die 8 Eingänge zu definieren. Da das BCD-Format den Gebrauch der Werte &HA, &HB, &HC, &HD, &HE oder &HF ausschließt, können nicht alle Kombinationen für die 8 Eingänge verwendet werden.
- Der In-Befehl ist dem InBCD-Befehl sehr ähnlich, mit dem Unterschied, dass In ermöglicht, den Rückgabewert für alle 8 Eingänge zu verwenden (z. B. 0-255 vs. 0-99 bei InBCD). Dadurch können alle möglichen Kombinationen für die 8-Bit-Eingangsgruppen gelesen werden.

Verwandte Befehle

In, MemOff, MemOn, MemOut, MemSw, Off, On, OpBCD, Oport, Out, Sw, Wait

Beispiel einer InBCD-Funktion

Einige einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus:

Es wird davon ausgegangen, dass an den Eingängen 0, 4, 10, 16, 17, und 18 Spannung anliegt. (An den restlichen Eingängen liegt keine Spannung an.)

```
> Print InBCD(0)
11
> Print InBCD(1)
04
> Print InBCD(2)
07
>
```

Inertia-Anweisung

Spezifiziert die Massenträgheit und Exzentrizität für den aktuellen Roboter.

Syntax

Inertia [*loadInertia*], [*eccentricity*]
Inertia

Parameter

loadInertia Optional. Reeller Ausdruck, der das totale Trägheitsmoment in kgm^2 um das Zentrum der Greiferachse, inklusive Greifer und Objekt, definiert.

eccentricity Optional. Reeller Ausdruck, der die Exzentritität in kgm^2 um das Zentrum der Greiferachse, inklusive Greifer und Objekt, definiert.

Rückgabewerte

Wenn die Parameter weggelassen werden, werden die aktuellen Trägheitsparameter angezeigt.

Beschreibung

Verwenden Sie die **Inertia-**Anweisung, um das vollständige Trägheitsmoment für die Last an der Greiferachse zu definieren. Dadurch kann das System Beschleunigung, Verzögerung und Servoverstärkung für die Greiferachse besser kompensieren. Sie können auch die Entfernung vom Zentrum der Greiferachse zum Schwerpunkt von Greifer und Objekt definieren, indem Sie den Parameter *eccentricity* verwenden.

Verwandte Befehle

Inertia-Funktion

Beispiel einer Inertia-Anweisung

```
Inertia 0.02, 1
```

Inertia-Funktion

F

Gibt den Wert des Trägheitsparameters aus.

Syntax

Inertia(*paramNumber***)**

Parameter

paramNumber Integer-Ausdruck, der die folgenden Werte haben kann:

0: Veranlasst die Funktion, 1 auszugeben, wenn der Roboter Trägheitsparameter unterstützt, oder 0, wenn dies nicht der Fall ist.

1: Veranlasst die Funktion, die Lastträgheit in kgm² auszugeben.

2: Veranlasst die Funktion, die Exzentrizität in mm auszugeben.

Rückgabewerte

Reeller Wert der spezifizierten Einstellung.

Verwandte Befehle

Inertia-Anweisung

Beispiel einer Inertia-Funktion

```
Real loadInertia, eccentricity

loadInertia = Inertia(1)
eccentricity = Inertia(2)
```

InPos-Funktion

F

Gibt den aktuellen Positionsstatus des spezifizierten Roboters aus.

Syntax

InPos

Rückgabewerte

Wahr, wenn die Position erfolgreich abgeschlossen wurde, sonst Falsch.

Verwandte Befehle

CurPos, FindPos, WaitPos

Beispiel einer InPos-Funktion

```
Function  main

  P0 = XY(0, -100, 0, 0)
  P1 = XY(0, 100, 0, 0)

  Xqt MonitorPosition
  Do
    Jump P0
    Wait .5
    Jump P1
    Wait .5
  Loop

Fend

Function MonitorPosition

  Boolean oldInPos, pos

  Do
    Pos = InPos
    If pos <> oldInPos Then
      Print "InPos = ", pos
    EndIf
    oldInPos = pos
  Loop

Fend
```

Input-Anweisung

> S

Ermöglicht es, numerische Daten von der Tastatur zu empfangen und in (einer) Variablen zu speichern.

Syntax

Input *varName* [*, varName, varName,...*]

Parameter

varName	Variablenname. Mit dem **Input**-Befehl können mehrfache Variablen verwendet werden, solange sie durch Kommata voneinader getrennt werden.

Beschreibung

Input empfängt numerische Daten vom Anzeigegerät und ordnet die Daten der/den Variablen zu, die mit dem **Input**-Befehl verwendet werden.

Wenn der Input-Befehl ausgeführt wird, erscheint ein (?)-Hinweis am Anzeigegerät. Drücken Sie nach der Eingabe der Daten die Enter-Taste auf der Tastatur.

Hinweise

Regeln für numerische Eingaben

Wenn bei der Eingabe der numerischen Werte nicht-numerische Daten außer dem Begrenzungszeichen (Komma) gefunden werden, entsorgt der **Input**-Befehl die nicht-numerischen Daten und alle folgenden nicht-numerischen Daten.

Regeln für Zeichenketten-Eingaben

Wenn Zeichenketten eingegeben werden, sind numerische Zeichen und Buchstabenzeichen als Daten zugelassen.

Weitere Regeln für den Input-Befehl

- Wenn im Befehl mehr als eine Variable angegeben ist, müssen die Eingaben numerischer Daten für jede Variable durch Kommazeichen („,") voneinander getrennt werden.
- Numerische Variablennamen und Zeichenketten-Variablennamen sind zulässig. Der Eingabe-Datentyp muss jedoch dem Variablentyp entsprechen.

Mögliche Fehler

Die Anzahl der Variablen und der Eingabedaten ist unterschiedlich

Für Mehrfachvariablen muss die Anzahl der Eingabedaten mit der Anzahl der Eingabe-Variablennamen übereinstimmen. Wenn sich die Anzahl der Variablen, wie sie im Befehl definiert ist, von der Anzahl der von der Tastatur empfangenen numerischen Daten unterscheidet, tritt der Fehler Nr. 2505 auf.

Verwandte Befehle

Input #, Line Input, Line Input #, Print, String

Beispiel einer Input-Anweisung

Diese Funktion zeigt einige einfache Beispiele einer Input-Anweisung auf.

```
Function InputNumbers
    Integer A, B, C

    Print "Please enter 1 number"
    Input A
    Print "Please enter 2 numbers separated by a comma"
    Input B, C
    Print "A = ", A
    Print "B = ", B, "C = ", C
Fend

Eine Mustersitzung des oben ablaufenden Programms wird im Folgenden
dargestellt:
(Verwenden Sie das Run-Menü oder die Taste F5, um das Programm zu
starten.)

Please enter 1 number
?-10000
Please enter 2 numbers separated by a comma
?25.1, -99
-10000
25.1 -99
B = 25.1 C = -99
>
```

Input #-Anweisung

> S

Ermöglicht es, Zeichenketten- oder numerische Daten von einer Datei oder einem Kommunikationsport zu empfangen und in (einer) Variablen zu speichern.

Syntax

Input ***PortNum****, varName* [*, varName, varName,...*]

Parameter

PortNum	Kommunikationsport oder Geräte-ID. Kommunikationsports können in OpenCom- (RS232) und OpenNet- (TCP/IP) Anweisungen definiert werden. *Die Integerwerte der Geräte-IDs lauten wie folgt:* *21 RC+* *23 OP* *24 TP*
varName	Variablenname, um Daten zu empfangen.

Beschreibung

Der **Input #**-Befehl empfängt numerische- oder Zeichenkettendaten von dem Gerät, das durch **PortNum** spezifiziert ist, und ordnet die Daten der / den Variable(n) zu.

Hinweise

Regeln für numerische Eingaben

Wenn bei der Eingabe der numerischen Werte nicht-numerische Daten außer dem Begrenzungszeichen (Komma) gefunden werden, entsorgt der **Input**-Befehl die nicht-numerischen Daten und alle folgenden nicht-numerischen Daten.

Regeln für Zeichenketten-Eingaben

Wenn Zeichenketten eingegeben werden, sind numerische Zeichen und Buchstabenzeichen als Daten zugelassen.

Weitere Regeln für den Input-Befehl

- Wenn im Befehl mehr als eine Variable angegeben ist, müssen die Eingaben numerischer Daten für jede Variable durch Kommazeichen („,") voneinander getrennt werden.
- Numerische Variablennamen und Zeichenketten-Variablennamen sind zulässig. Der Eingabe-Datentyp muss jedoch dem Variablentyp entsprechen.

Mögliche Fehler

Die Anzahl der Variablen und der Eingabedaten sind unterschiedlich

Wenn sich die Anzahl der Variablen, wie sie im Befehl definiert ist, von der Anzahl der vom Gerät empfangenen numerischen Daten unterscheidet, tritt der Fehler Nr. 2505 auf.

Verwandte Befehle

Input, Line Input, Line Input #

Beispiel einer Input #-Anweisung

Diese Funktion zeigt einige einfache Beispiele einer Input#-Anweisung.

```
Function GetData
    Integer A
    String B$

    OpenCom #1
    Print #1, "Send"
    Input #1, A   'Nimmt numerische Daten vom 1. Port an
    Input #1, A   'Nimmt Zeichenketten-Daten vom 1. Port an
    CloseCom #1
Fend
```

InStr-Funktion

F

Gibt die Position einer Zeichenkette innerhalb einer anderen Zeichenkette aus.

Syntax

InStr(*string, searchString*)

Parameter

string	Zu suchender Zeichenkettenausdruck.
searchString	Zeichenkettenausdruck, nach dem innerhalb von *string* gesucht werden soll.

Rückgabewerte

Gibt die Position der Suchkette aus, wenn der Ort gefunden wird, sonst wird -1 ausgegeben.

Verwandte Befehle

Mid$

Beispiel einer Instr-Funktion

```
Integer pos
pos = InStr("abc", "b")
```

Int-Funktion

F

Wandelt eine reelle Zahl in eine Integer-Zahl um. Gibt den größten Integer aus, der kleiner oder gleich dem spezifizierten Wert ist.

Syntax

Int*(number)*

Parameter

number Ein Ausdruck mit einer reellen Zahl.

Rückgabewerte

Gibt einen Integer-Wert der in *number* verwendeten reellen Zahl aus.

Beschreibung

Int(*number*) gibt anhand des Wertes von *number* den größten Integer aus, der kleiner oder gleich *number* ist.

Hinweis

Für Werte kleiner als 1 (negative Zahlen)

Wenn der Parameter *number* einen Wert von weniger als 1 hat, dann hat der Rückgabewert einen größeren Absolutwert als *number*. (Wenn die Zahl z. B. -1,35 lautet, wird -2 ausgegeben.)

Verwandte Befehle

Abs, Atan, Atan2, Cos, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Beispiel einer Int-Funktion

Einige einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus:

```
> Print Int(5.1)
5
> Print Int(0.2)
0
> Print Int(-5.1)
-6
>
```

Integer-Anweisung

S

Deklariert Integer-Variablen (2 Byte Integer).

Syntax

Integer *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameter

varName	Variablenname, den der Benutzer als **Integer** deklarieren möchte.
subscripts	Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus: (ubound1, [ubound2], [ubound3]) Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension. Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert. Die Gesamtzahl der Elemente der Matrixelemente für lokale und Global Preserve-Variablen ist 1000. Die Gesamtzahl der Matrixelemente für globale und Modulvariablen ist 10000. Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)

Beschreibung

Integer wird verwendet, um Variablen als **Integer** zu deklarieren. Integer-Variablen können ganze Zahlen im Wertebereich von -32768 bis 32767 enthalten. Alle lokalen Variablen sollten in einer Funktion ganz oben deklariert sein. Globale und Modulvariabeln müssen außerhalb von Funktionen deklariert werden.

Verwandte Befehle

Boolean, Byte, Double, Global, Long, Real, String

Beispiel einer Integer Anweisung

Das folgende Beispiel zeigt ein einfaches Programm, das unter Verwendung von **Integer** einige Variablen deklariert.

```
Function inttest
    Integer A(10)         'Eindimensionale Matrix aus Integern
    Integer B(10, 10)     'Zweidimensionale Matrix aus Integern
    Integer C(10, 10, 10) 'Dreidimensionale Matrix aus Integern
    Integer var1, arrayvar(10)
    Integer i
    Print "Please enter an Integer Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For I = 1 To 5
        Print "Please enter an Integer Number"
        Input arrayvar(i)
        Print "Value Entered was ", arrayvar(i)
    Next I
Fend
```

InW-Funktion

F

Gibt den Status des spezifizierten Eingangs-Wortports aus. Jeder Wortport enthält 16 Eingangsbits.

Syntax

InW(*WordPortNum*)

Parameter

WordPortNum Integer-Ausdruck, der für das E/A-Eingangswort steht.

Rückgabewerte

Gibt den aktuellen Eingangs-Status aus (lange Integer von 0 bis 65535).

Verwandte Befehle

In, Out, OutW

Beispiel einer InW-Funktion

```
Long word0
word0 = InW(0)
```

IOLabel$-Funktion

F

Gibt das E/A-Label für ein spezifiziertes Eingangs- oder Ausgangsbit, -byte oder -wort aus.

Syntax

IOLabel$(*IOType, IOWidth, portNumber*)

Parameter

IOType Integer-Ausdruck, der für den E/A-Typ steht.
0 - Eingang
1 - Ausgang
2 - Merker

IOWidth Integer-Ausdruck, der für die Breite des Ports steht: 1 (Bit), 8 (Byte), or 16 (Wort).

portNumber Integer-Ausdruck, der für die Bit-, Byte- oder Wort-Portnummer steht, für die ein Label ausgegeben werden soll.

Rückgabewerte

Zeichenkette, die das Label enthält.

Verwandte Befehle

PLabel$, IONumber

Beispiel einer IOLabel$-Funktion

```
Integer i

For i = 0 To 15
  Print "Input ", i, ": ", IOLabel$(0, 1, i)
Next i
```

IONumber-Funktion

F

Gibt die E/A-Nummer eines spezifizierten E/A-Labels aus.

Syntax

IONumber(*IOlabel*)

Parameter

IOlabel Zeichenkettenausdruck, der Standard-E/A- oder Merker-Labels spezifiziert.

Rückgabewerte

Gibt die E/A-Portnummer (Bit, Byte, Wort) eines spezifizierten E/A-Labels aus. Wenn ein solches E/A-Label nicht vorhanden ist, tritt ein Fehler auf.

Verwandte Befehle

IOLabel$

Beispiel einer IONumber-Funktion

```
Integer IObit

IObit = IONumber("myIO")

IObit = IONUmber("Station" + Str$(station) + "InCycle")
```

J4Flag-Anweisung

Stellt das J4Flag-Attribut für einen Punkt ein.

> S

Syntax

(1) J4Flag *point*, [*value*]
(2) J4Flag

Parameter

point	**P***number* oder **P**(*expr*) oder Punktlabel.
value	Optional. Integer-Ausdruck. Der Bereich 0 (/J4F0) J4 liegt zwischen -180 und +180 Grad. Der Bereich 1 (/J4F1) J4 liegt zwischen -360 und -180 oder +180 und +360 Grad.

Rückgabewerte

Das J4Flag-Attribut spezifiziert den Wertebereich für die 4. Achse für einen Punkt. Wenn *value* weggelassen wird, wird der J4Flag-Wert für den spezifizierten Punkt angezeigt. Wenn beide Parameter ausgelassen werden, wird der J4Flag-Wert für die aktuelle Roboterposition angezeigt.

Verwandte Befehle

Elbow, Hand, J4Flag-Funktion, J6Flag, Wrist

Beispiel einer J4Flag-Anweisung

```
J4Flag P0, 1
J4Flag P(mypoint), 0
```

J4Flag-Funktion

F

Gibt das J4Flag-Attribut für einen Punkt aus.

Syntax

J4Flag [(*point*)]

Parameter

point Optional. Punktausdruck. Wird *point* weggelassen, wird die J4Flag-Einstellung der aktuellen Roboterposition ausgegeben.

Rückgabewerte

0 /J4F0
1 /J4F1

Verwandte Befehle

Elbow, Hand, Wrist, J4Flag-Anweisung, J6Flag

Beispiel einer J4Flag-Funktion

```
Print J4Flag(pick)
Print J4Flag(P1)
Print J4Flag
Print J4Flag(Pallet(1, 1))
```

J6Flag-Anweisung

Stellt das J6Flag-Attribut für einen Punkt ein.

> S

Syntax

(1) J6Flag *point*, [*value*]
(2) J6Flag

Parameter

point	**P***number* oder **P**(*expr*) oder Punktlabel.
value	Integer-Ausdruck. Der Bereich liegt zwischen 0 und 127 (J6F0 - /J6F127). Der J6-Bereich für den spezifizierten Punkt sieht folgendermaßen aus: (-180 * (*value*+1) < J6 <= 180 * *value*) und (180 * *value* < J6 <= 180 * (*value*+1))

Rückgabewerte

Das J6Flag-Attribut spezifiziert den Wertebereich für die 6. Achse für einen Punkt. Wenn *value* weggelassen wird, wird der J6Flag-Wert für den spezifizierten Punkt angezeigt. Wenn beide Parameter ausgelassen werden, wird der J6Flag-Wert für die aktuelle Roboterposition angezeigt.

Verwandte Befehle

Elbow, Hand, J4Flag, J6Flag-Funktion, Wrist

Beispiel einer J6Flag-Anweisung

```
J6Flag P0, 1
J6Flag P(mypoint), 0
```

J6Flag-Funktion

F

Gibt das J6Flag-Attribut für einen Punkt aus.

Syntax

J6Flag [(*point*)]

Parameter

point Optional. Punktausdruck. Wird *point* weggelassen, wird die J6Flag-Einstellung der aktuellen Roboterposition ausgegeben.

Rückgabewerte

0 - 127 /J6F0 - /J6F127

Verwandte Befehle

Elbow, Hand, Wrist, J4Flag, J6Flag-Anweisung

Beispiel einer J6Flag-Funktion

```
Print J6Flag(pick)
Print J6Flag(P1)
Print J6Flag
Print J6Flag(P1 + P2)
```

JA-Funktion

F

Gibt einen in Achswinkeln angegebenen Roboterpunkt aus.

Syntax

JA(*j1*, *j2*, *j3*, *j4*, [*j5*], [*j6*]**)**

Parameter

j1 - j6 Reelle Ausdrücke, die für Achswinkel stehen.

Rückgabewerte

Ein Roboterpunkt, dessen Ort durch die dargestellten Achswinkel bestimmt wird.

Beschreibung

Verwenden Sie JA, um einen Roboterpunkt mithilfe der Achswinkel zu spezifizieren.

Wenn die von der JA-Funktion ausgegebenen Punkte eine Singularität des Roboters spezifizieren, stimmen die Achswinkel des Roboters nicht immer mit den Achswinkeln überein, die der JA-Funktion als Argumente während der Ausführung eines Bewegungsbefehls für die Punkte zugeführt wurden. Um den Roboter mithilfe der für die JA-Funktion spezifizierten Achswinkel zu bedienen, vermeiden Sie eine Singularität des Roboters.

Zum Beispiel:

```
> go ja(0,0,0,90,0,-90)
> where
WORLD:  X:    0.000 mm  Y:  655.000 mm  Z:  675.000 mm  U:    0.000 deg V:  -90.000 deg W:  -90.000 deg
JOINT:  1:    0.000 deg 2:    0.000 deg 3:    0.000 deg 4:    0.000 deg 5:    0.000 deg 6:    0.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:        0 pls 5:        0 pls 6:        0 pls

> go ja(0,0,0,90,0.001,-90)
> where
WORLD:  X:   -0,004 mm  Y:  655.000 mm  Z:  675.000 mm  U:    0.000 deg V:  -90.000 deg W:  -89,999 deg
JOINT:  1:    0.000 deg 2:    0.000 deg 3:    0.000 deg 4:   90.000 deg 5:    0,001 deg 6:  -90.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:  2621440 pls 5:       29 pls 6: -1638400 pls
```

Verwandte Befehle

AglToPls, XY

Beispiel einer JA-Funktion

```
P10 = JA(60, 30, -50, 45)
Go JA(135, 90, -50, 90)
P3 = JA(0, 0, 0, 0, 0, 0)
```

Joint-Anweisung

Zeigt die aktuelle Position für den Roboter in Achskoordinaten an.

Syntax

Joint

Verwandte Befehle

Pulse, Where

Beispiel einer Joint-Anweisung

```
>joint
JOINT:  1:   -6,905 deg 2:   23,437 deg 3:   -1.999 mm  4:  -16,529 deg
>
```

JRange-Anweisung

Definiert den erlaubten Arbeitsbereich der spezifizierten Achse in Pulsen.

> S

Syntax

JRange *jointNumber, lowerLimit, upperLmit*

Parameter

jointNumber	Integer-Ausdruck zwischen 1 und 6, der für die Achse steht, für die JRange spezifiziert wird.
lowerLmit	Long Integer-Ausdruck, der für den Zählstand der Encoder-Pulse-Position für den Bereich des unteren Grenzwerts der angegebenen Achse steht.
upperLmit	Long Integer-Ausdruck, der für den Zählstand der Encoder-Pulse-Position für den Bereich des oberen Grenzwerts der angegebenen Achse steht.

Beschreibung

Definiert den erlaubten Arbeitsbereich für die definierte Achse mit unterem und oberem Grenzpunkt in Encoder-Pulsen. **JRange** ist dem Range-Befehl ähnlich. Jedoch müssen für den Range-Befehl alle Grenzwerte der Achsen angegeben werden, während der **JRange**-Befehl verwendet werden kann, um den Grenzwert jeder Achse individuell einzustellen, was wiederum die Anzahl der benötigten Parameter reduziert. Um den definierten Arbeitsbereich zu überprüfen, verwenden Sie bitte den Range-Befehl.

Hinweise

Die unteren Grenzwerte dürfen nicht über den oberen Grenzwerten liegen:

Der untere Grenzwert, der im JRange-Befehl definiert ist, darf nicht über dem oberen Grenzpunkt liegen. Wenn der untere Grenzwert über dem oberen Grenzwert liegt, ruft das einen Fehler hervor, was es unmöglich macht, einen Bewegungsbefehl auszuführen.

Faktoren, die JRange ändern können:

Sobald **JRange**-Werte eingestellt sind, bleiben diese bestehen, bis der Benutzer diese Werte entweder durch den Range- oder den **JRange**-Befehl ändert. Das Ausschalten der Steuerung ändert die Werte der **JRange**-Achsgrenzwerte nicht.

Maximaler und minimaler Arbeitsbereich:

Lesen Sie bitte die Spezifikationen im Roboterhandbuch bezüglich der maximalen Arbeitsbereiche für jedes Robotermodell, da diese Werte von Modell zu Modell unterschiedlich sind.

Verwandte Befehle

Range, JRange Funktion

Beispiel einer JRange-Anweisung

Die folgenden Beispiele werden vom Befehlseingabefenster aus ausgeführt:

```
> JRange 2, -6000, 7000       'Definiert den zweiten Achsbereich
> JRange 1, 0, 7000           'Definiert den ersten Achsbereich
```

JRange-Funktion

F

Gibt den erlaubten Arbeitsbereich der angegebenen Achse in Pulsen aus.

Syntax

JRange(*jointNumber, paramNumber*)

Parameter

jointNumber	Spezifiziert die Referenz-Achsnummer (Integer von 1 bis 6) durch einen Ausdruck oder einen numerischen Wert.
paramNumber	Integer-Ausdruck, der einen von zwei Werten enthält: 1: Spezifizierter unterer Grenzwert 2: Spezifizierter oberer Grenzwert

Rückgabewerte

Bereichseinstellung (Integer-Wert, Pulse) der angegebenen Achse.

Verwandte Befehle

Range, Jrange-Anweisung

Beispiel einer JRange-Funktion

```
Long i, oldRanges(3, 1)

For i = 0 To 3
    oldRanges(i, 0) = JRange(i + 1, 1)
    oldRanges(i, 1) = JRange(i + 1, 2)
Next i
```

JS-Funktion

F

Jump Sense ermittelt, ob der Arm angehalten hat, bevor ein Jump-, Jump3- oder Jump3CP-Befehl, der einen Sense-Eingang verwendet hat, vollständig ausgeführt wurde, oder ob der Arm die Jump-Bewegung ganz ausgeführt hat.

Syntax

JS

Rückgabewerte

Gibt Wahr oder Falsch aus.

Wahr : Wenn der Arm gestoppt wurde, bevor er seinen Zielpunkt erreicht hat, weil eine Sense-Input-Bedingung erfüllt wurde, gibt **JS** ein Wahr aus.

Falsch: Wenn der Arm die normale Bewegung abschließt und den durch den Jump-Befehl definierten Zielort erreicht, gibt **JS** ein Falsch aus.

Beschreibung

JS wird zusammen mit den Befehlen Jump und Sense verwendet. Der Zweck des **JS-**Befehls ist es**,** ein Statusergebnis zur Verfügung zu stellen, das aussagt, ob ein eine Eingabebedingung (die durch den Sense-Befehl definiert wurde) während einer Bewegung (die durch den Jump-Befehl ausgelöst wurde) erfüllt wird, oder nicht. Wenn die Eingabebedingung erfüllt wird, gibt **JS** ein Wahr aus. Wenn die Eingabebedingung nicht erfüllt wird und der Arm die Zielposition erreicht, gibt **JS** ein Falsch aus.

JS ist ein Befehl zur Status-Überprüfung, der keinerlei Bewegung verursacht oder spezifiziert, welcher Eingang während der Bewegung überprüft werden soll. Der Jump-Befehl wird verwendet, um Bewegungen zu initiieren und der Sense-Befehl wird verwendet, um zu spezifizieren, welcher Eingang während einer durch den Jump-Befehl ausgelösten Bewegung überprüft werden soll (wenn ein Eingang überprüft werden soll).

Hinweis

JS funktioniert nur mit dem zuletzt verwendeten Jump-, Jump3- oder Jump3CP-Befehl:

JS kann nur verwendet werden, um die Eingangsüberprüfung des zuletzt verwendeten Jump-Befehls zu überprüfen (der durch den Sense-Befehl initiiert wurde). Sobald ein zweiter Jump-Befehl initiiert wird, kann **JS** nur den Status für den zweiten Jump-Befehl ausgeben. Der **JS-**Status für den ersten Jump-Befehl ist nicht mehr verfügbar. Stellen Sie sicher, dass Sie die **JS**-Statusüberprüfungen für Jump-Befehle immer direkt nach einem ausgeführten, zu prüfenden Jump-Befehl durchführen.

Verwandte Befehle

JT, Jump, Jump3, Jump3CP, Sense

Beispiel einer JS-Funktion

```
Function SearchSensor As Boolean
    Sense Sw(5) = On

    Jump P0
    Jump P1 Sense
    If JS = TRUE Then
        Print "Sensor was found"
        SearchSensor = TRUE
    EndIf
Fend
```

JT-Funktion

F

Gibt den Status des zuletzt ausgegebenen Jump-, Jump3- oder Jump3CP-Befehls für den aktuellen Roboter aus.

Syntax

JT

Rückgabewerte

JT gibt einen Long-Wert mit den folgenden eingestellten oder unbesetzten Bits aus:

Bit 0	Auf 1 gesetzt, wenn die Hubbewegung begonnen hat oder die Hubdistanz gleich 0 ist.
Bit 1	Auf 1 gesetzt, wenn die Horizontalbewegung begonnen hat oder die Horizontaldistanz gleich 0 ist.
Bit 2	Auf 1 gesetzt, wenn die Absenkbewegung begonnen hat oder die Absenkdistanz gleich 0 ist.
Bit 16	Auf 1 gesetzt, wenn die Hubbewegung abgeschlossen ist oder die Hubdistanz gleich 0 ist.
Bit 17	Auf 1 gesetzt, wenn die Horizontalbewegung abgeschlossen ist oder die Horizontaldistanz gleich 0 ist.
Bit 18	Auf 1 gesetzt, wenn die Absenkbewegung abgeschlossen ist oder die Absenkdistanz gleich 0 ist.

Beschreibung

JT wird verwendet, um den Status des zuletzt ausgeführten Jump-Befehls festzustellen, der vor seiner Vollendung durch einen der Befehle Sense, Till, Abort, etc. gestoppt wurde.

Verwandte Befehle

JS, Jump, Jump3, Jump3CP, Sense, Till

Beispiel einer JT-Funktion

```
Function SearchTill As Boolean

    Till Sw(5) = On

    Jump P0
    Jump P1 Till
    If JT And 4 Then
        Print "Motion stopped during descent"
        SearchTill = TRUE
    EndIf
Fend
```

JTran-Anweisung

S

Führt eine relative Bewegung einer Achse aus.

Syntax

JTran *jointNumber, distance*

Parameter

jointNumber	Integer-Ausdruck, der für die zu bewegende Achse steht.
distance	Reeller Ausdruck, der für die Distanz steht, die zurückgelegt werden muss, für Rotationsachsen in Grad und für Linearachsen in Millimetern.

Beschreibung

JTran wird verwendet, um eine Achse über eine definierte Distanz aus ihrer aktuellen Position wegzubewegen.

Verwandte Befehle

Go, Jump, Move, Ptran

Beispiel einer JTran-Anweisung

```
JTran 1, 20
```

Jump-Anweisung

> S

Bewegt den Arm mit einer PTP-Bewegung aus seiner aktuellen Position zum angegebenen Zielpunkt, indem sich der Arm zunächst vertikal nach oben, dann horizontal und zum schließlich senkrecht abwärts bewegt, um an den Zielpunkt der Bewegung zu gelangen.

Syntax

Jump *destination* [**C***archNumber*] [**LimZ** *zLimit*] [*searchExpr*] [*!...!*]

Parameter

destination	Der Zielort einer Bewegung, die einen Punktausdruck verwendet.
archNumber	Optional. Die Arch-Nummer (*archNumber*) definiert, welcher Eintrag der Arch-Tabelle für die Arch-Bewegung verwendet werden soll, die durch den Jump-Befehl hervorgerufen wird. Der *archNumber* muss immer der Buchstabe C vorangestellt werden. (Gültige Einträge sind C0-C7.)
zLimit	Optional. Dies ist ein Z-Grenzwert, der für die Maximalposition steht, zu der sich die Z-Achse während einer **Jump**-Bewegung bewegen lässt. Dieser Wert kann als Obergrenze in Z-Richtung für den **Jump**-Befehl betrachtet werden. Jeder gültige Z-Achsen-Koordinatenwert ist zulässig.
searchExpr	Optional. Ein Sense-, Till- oder Find-Ausdruck. **Sense** | **Till** | **Find** **Sense Sw**(*expr*) = {**On** | **Off**} **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können zum Jump-Befehl hinzugefügt werden, um E/A-Befehle und andere Befehle während der Bewegung auszuführen.

Beschreibung

Jump bewegt den Arm aus seiner aktuellen Position zur *destination* unter Verwendung der so genannten Arch-Bewegung (Bogen-Bewegung). Man kann Jump als drei Bewegungen in einer betrachten. Wenn beispielsweise der durch *archNumber* definierte Arch-Tabelleneintrag 7 ist, werden die folgenden drei Bewegungen ausgeführt:

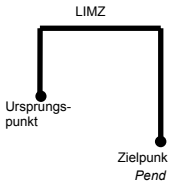
1) Die Bewegung beginnt nur mit der Z-Achse, bis diese die Höhe erreicht hat, die durch die Arch-Nummer errechnet wurde, die für den Jump-Befehl verwendet wird.
2) Als nächstes bewegt der Arm sich horizontal (während er in der Z-Achse noch immer die Hubbewegung durchführt) in Richtung der Zielpunkt-Position, bis der obere Grenzpunkt der Z-Achse (definiert durch LimZ) erreicht ist. Dann beginnt der Arm, sich in Z-Richtung abwärts zu bewegen (während er mit den X-, Y- und U-Achsen-Bewegungen fortfährt), bis die endgültigen X-, Y- und U-Achsen-Positionen erreicht sind.
3) Der **Jump**-Befehl wird dann abgeschlossen, indem der Arm nur mit einer Bewegung der Z-Achse abwärts bewegt wird, bis die Zielposition der Z-Achse erreicht ist.

Die *destination*-Koordinaten (Zielposition für die Bewegung) müssen vor dem Ausführen des **Jump**-Befehls geteacht werden. Die Koordinaten können nicht im **Jump**–Befehl selber spezifiziert werden. Beschleunigung und Verzögerung für die **Jump**-Anweisung werden durch den Accel-Befehl gesteuert. Die Geschwindigkeit für die Bewegung wird durch den Speed-Befehl gesteuert.

archNumber Details

Der Bogen für die **Jump**-Anweisung kann, auf der Basis des *archNumber*-Wertes, der optional zusammen mit dem Jump-Befehl definiert wird, verändert werden. Dies ermöglicht es dem Benutzer, zu definieren, welcher Teil der Z-Bewegung zurückgelegt werden soll, bevor mit der Bewegung der X-, Y-, und U-Achsen begonnen wird. (Der Benutzer kann so den Arm aufwärts bewegen und ein Kollision mit Teilen, Aufgabevorrichtungen und anderen Gegenständen verhindern, bevor mit der Horizontalbewegung begonnen wird.) Gültige *archNumber*-Einträge für den **Jump-**Befehl liegen

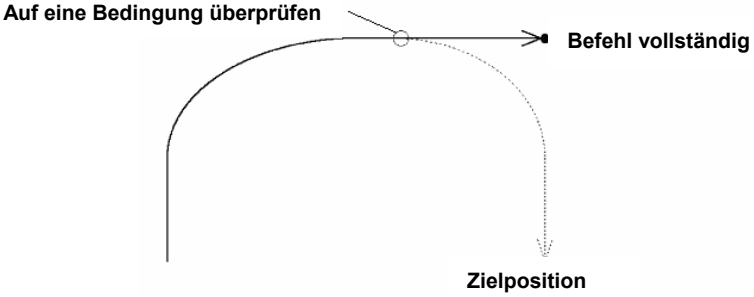
zwischen C0 und C7. Die Arch-Tabellen-Einträge für C0-C6 können über den Arch-Befehl vom Benutzer definiert werden. C7 ist jedoch ein besonderer Arch-Eintrag, der immer die so genannte Gate-Bewegung definiert. Gate-Bewegung bedeutet, dass der Roboter die Z-Achse zuerst vollständig zu der durch LimZ definierten Koordinate bewegt, bevor die X-, Y-, oder U-Achsen-Bewegungen begonnen werden. Sobald der LimZ-Grenzwert erreicht ist, beginnt die Bewegung der Achsen X, Y und U. Nachdem die Achsen X, Y, und U jeweils ihre Zielposition erreicht haben, kann die Z-Achse mit ihrer Abwärtsbewegung zur letzten Z-Achsen-Koordinatenposition beginnen, die durch *destination* (den Zielpunkt) definiert wurde. Die Gate-Bewegung sieht aus wie folgt:



LimZ Details
LimZ *zLimit* spezifiziert den oberen Z-Koordinatenwert für die Horizontalbewegungsebene im aktuellen lokalen Koordinatensystem. Die spezifizierten Arch-Einstellungen können die Achsen X, Y, und U dazu veranlassen, mit ihrer Bewegung zu beginnen, bevor LimZ erreicht ist, aber LimZ ist immer die maximale Z-Höhe für die Bewegung. Wenn der optionale LimZ-Parameter weggelassen wird, wird der davor durch den LimZ-Befehl spezifizierte Wert für die Definition der Horizontalbewegungsebene verwendet.

Es ist wichtig zu beachten, dass die Spezifikation der LimZ *zLimit* -Höhenbegrenzung den Z-Wert für das lokale Roboterkoordinatensystem darstellt. Es ist nicht der Z-Wert für Arm oder Werkzeug. Treffen Sie daher die notwendigen Schutzvorkehrungen für den Gebrauch von Werkzeugen oder Armen mit unterschiedlicher Betriebshöhe.

Sense Details
Der optionale Sense-Parameter ermöglicht es dem Benutzer, eine Eingangsbedingung oder Merkerbedingung zu überprüfen, bevor die endgültige Abwärtsbewegung der Z-Achse beginnt. Ist sie erfüllt, wird dieser Befehl beendet und der Roboter stoppt über der Zielposition, von wo nur die Z-Achse bewegt werden muss, um die Zielposition zu erreichen. Es ist wichtig zu beachten, dass der Roboterarm bei Wahrnehmen der Sense-Eingangsbedingung nicht sofort anhält.



Die Befehle JS oder Stat können dann verwendet werden, um zu beurteilen, ob die Sense-Bedingung erfüllt wurde und der Roboter vor seiner Zielposition gestoppt hat oder ob die Sense-Bedingung nicht erfüllt wurde und der Roboter seinen Weg bis zur Zielposition fortgesetzt hat.

Till Details

Die optionale Till-Bedingung ermöglicht es dem Benutzer, eine Bedingung zu definieren, die den Roboter dazu veranlasst, bis zum vollständigen Stopp zu verzögern, bevor der **Jump**-Befehl vollständig ausgeführt wurde. Die spezifizierte Bedingung ist lediglich eine Gegenprüfung zu einem der E/A-Eingänge oder einem der Merker. Dies wird durch Verwenden der Funktionen Sw oder MemSw erreicht. Der Benutzer kann überprüfen, ob die Eingänge aus- oder eingeschaltet sind und den Arm dazu veranlassen, zu verzögern und anzuhalten, je nachdem, welche Bedingung spezifiziert ist.

Die Stat-Funktion kann verwendet werden, um zu verifizieren, ob die Till-Bedingung erfüllt wurde und dieser Befehl vollständig ausgeführt wurde, oder ob die Till-Bedingung nicht erfüllt wurde und der Roboter an der Zielposition angehalten hat.

Hinweise

Bei 6-Achsrobotern kann Jump nicht ausgeführt werden.

Verwenden Sie bei 6-Achsrobotern Jump3 oder Jump3CP.

Der Trajektoriebereich der Jump-Bewegung ändert sich je nach Bewegung und Geschwindigkeit.

Der Trajektoriebereich des Jump-Befehls beinhaltet die vertikale und die horizontale Bewegung. Es handelt sich nicht um einen CP-Trajektoriebereich. Der tatsächliche Jump-Trajektoriebereich der Bogenbewegung wird nicht ausschließlich durch die **Arch**-Parameter bestimmt. Auch die Bewegung und die Geschwindigkeit spielen eine Rolle.

Optimieren Sie den Jump-Trajektoriebereich in Ihren Anwendungen stets mit großer Sorgfalt. Führen Sie Jump mit der gewünschten Bewegung und Geschwindigkeit aus, um den tatsächlichen Trajektoriebereich zu überprüfen.

Wenn die Geschwindigkeit verringert wird, wird auch der Trajektoriebereich verringert. Wenn Jump bei hoher Geschwindigkeit ausgeführt wird, um den Trajektoriebereich eines Bewegungsbogens festzulegen, ist es möglich, dass der Greifer bei niedrigerer Geschwindigkeit mit Hindernissen kollidiert.

In einem Jump-Trajektoriebereich vergrößert sich die Depart-Distanz und verkleinert sich die Approach-Distanz, wenn die Bewegungsgeschwindigkeit auf hoch gestellt ist. Wenn die Absenkstrecke des Trajektoriebereichs kürzer ist als erwartet, verringern Sie die Geschwindigket und / oder die Verzögerung oder verlängern Sie die Absenkstrecke.

Selbst wenn Jump-Befehle mit derselben Distanz und Geschwindigkeit ausgeführt werden, wird der Trajektoriebereich durch die Bewegung der Roboterarme beeinflusst. Bei SCARA-Robotern vergrößert sich z. B. die vertikale Hubstrecke und verkleinert sich die vertikale Absenkstrecke, wenn die Bewegung des ersten Armes sehr groß ist. Wenn sich die vertikale Absenkstrecke verringert und der Trajektoriebereich kürzer als erwartet ist, verringern Sie die Geschwindigket bzw. die Verzögerung oder verlängern Sie die Absenkstrecke.

Auslassen des *archNumber*-Parameters

Wenn der optionale *archnum*-Parameter weggelassen wird, ist der Vorgabe-Arch-Eintrag für die Verwendung mit dem Jump-Befehl C7. Dies verursacht die Gate-Bewegung, wie oben beschrieben.

Unterschied zwischen Jump und Jump3, Jump3CP

Die Befehle Jump3 und Jump3CP können bei 6-Achsrobotern verwendet werden. Der Jump-Befehl wiederum kann nicht bei 6-Achsrobotern verwendet werden. Bei SCARA-Robotern verkürzt die Verwendung des Jump-Befehls die Achsbewegungszeit bei der Depart- und Approach-Bewegung. Die Depart- und Approach-Bewegungen in Jump3 können entlang der Z-Achse und in andere Richtungen ausgeführt werden.

Unterschied zwischen Jump und Go

Der Go-Befehl ist **Jump** in sofern ähnlich, als beide PTP-Bewegungen verursachen. Dabei gibt es jedoch viele Unterschiede. Der wichtigste Unterschied ist, dass der Go-Befehl PTP-Bewegungen verursacht, bei denen alle Achsen zur selben Zeit starten und anhalten (sie sind synchronisiert). Der Jump-Befehl führt jedoch die vertikale Z-Bewegung zu Beginn und Ende der Bewegung aus. Der Jump-Befehl ist ideal für Bestückungs-Anwendungen.

Verlangsamung bis zum vollständigen Halt mit dem Jump-Befehl

Der **Jump**-Befehl veranlasst den Arm immer dazu, bis zum vollständigen Halt herunterzubremsen bevor dieser den Zielpunkt erreicht hat.

Korrekte Speed- und Acceleration-Befehle (Geschwindigkeit und Beschleunigung) mit Jump:

Die Befehle Speed und Accel werden verwendet, um Geschwindigkeit und Beschleunigung des Roboters während der **Jump-**Bewegung zu definieren. Beachten Sie, dass die Befehle Speed und Accel sich auf PTP-Bewegungen beziehen (Go, Jump, etc), während sich auf linear- und kreisinterpolierte Bewegungen (wie Move oder Arc) die Befehle SpeedS und AccelS beziehen. Für den Jump-Befehl ist es möglich, Geschwindigkeit und Beschleunigung für die folgenden Bewegungen separat zu definieren: Hubbewegung der Z-Achse, horizontale Verfahrbewegung inklusive U-Achsen-Rotation und die Abwärtsbewegung der Z-Achse.

Mögliche Fehler

Der LimZ-Wert ist nicht hoch genug

Wenn die aktuelle Armposition der Z-Achse höher ist als der Wert, der für LimZ eingestellt wurde und der versucht wird, einen Jump-Befehl auszuführen, tritt der Fehler Nr. 4005 auf.

Verwandte Befehle

Accel, Arc, Arch, Go, JS, JT, LimZ, Punktausdruck, Pulse, Sense, Speed, Stat, Till

Beispiel einer Jump-Anweisung

Das folgende Beispiel zeigt eine einfache PTP-Bewegung zwischen den Punkten P0 und P1 und die darauf folgende Rückkehr zum Punkt P0 unter Verwendung des Jump-Befehls. Später im Programm bewegt sich der Arm mittels des Jump-Befehls, bis am 4. Eingang Spannung anliegt. Wenn am 4. Eingang Spannung anliegt, stoppt der Arm die Bewegung. Wenn am 4. Eingang nie Spannung anliegt, führt der Arm den Jump-Befehl vollständig aus und erreicht den Punkt P1.

```
Function jumptest
    Home
    Go  P0
    Go P1
    Sense Sw(4) = 1
    Jump P0 LimZ -10
    Jump P1 LimZ-10 Sense   'Überprüft den 4. Eingang
    If Js(0) = 1 Then
        Print "Input #4 came on during the move and"
        Print "the robot stopped prior to arriving on"
        Print "point P1."
    Else
        Print "The move to P1 completed successfully."
        Print "Input #4 never came on during the move."
    EndIf
Fend

> Jump P10+X50 C0 LimZ-20 Sense !D50;On 0;D80;On 1!
```

Jump3- und Jump3CP-Anweisung

> S

3D-Gate-Bewegung. Jump3 ist eine Kombination aus zwei CP-Bewegungen und einer PTP-Bewegung.
Jump3CP ist eine Kombination aus drei CP-Bewegungen.

Syntax

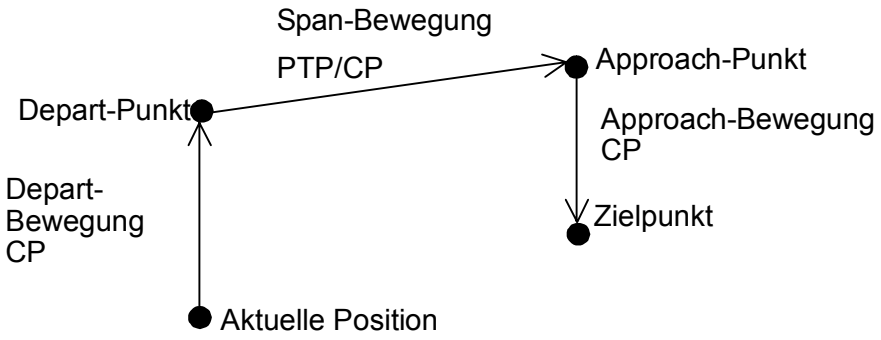
(1) Jump3 *depart, approach, destination* [**C***archNumber*] [**CP**] [*searchExpr*] [*!...!*]
(2) Jump3CP *depart, approach, destination* [**ROT**] [**C***archNumber*] [**CP**] [*searchExpr*] [*!...!*]

Parameter

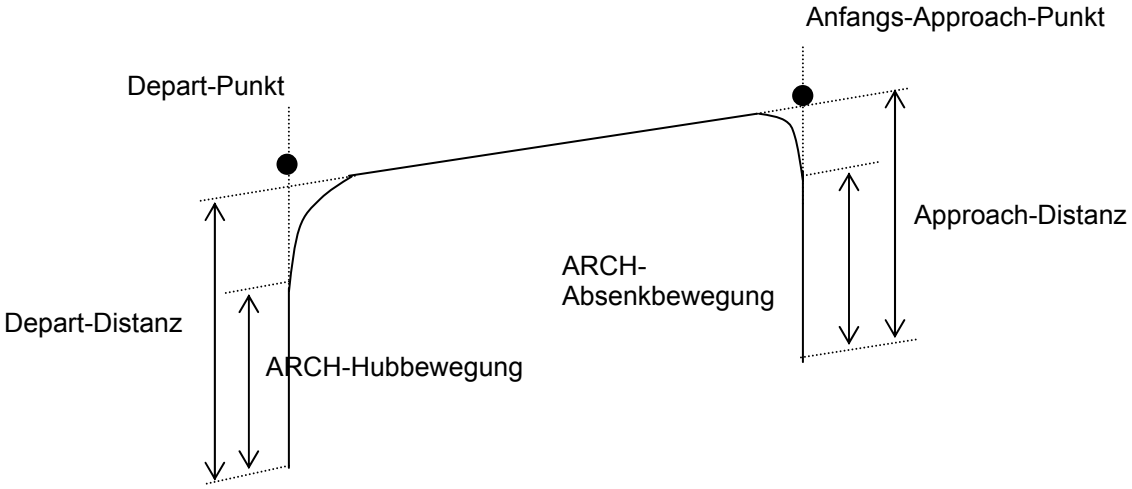
depart	Der Ausgangspunkt über der aktuellen Position unter Verwendung eines Punktausdrucks.
approach	Der Approach-Punkt über der Zielposition, ein Punktausdruck.
destination	Der Zielort einer Bewegung, ein Punktausdruck.
ROT	Optional. :Bestimmt Geschwindigkeit / Beschleunigung / Verzögerung zugunsten der Werkzeugrotation.
archNumber	Optional. Die Arch-Nummer (*archNumber*) definiert, welcher Eintrag der Arch-Tabelle für die Arch-Bewegung verwendet werden soll, die durch den Jump-Befehl hervorgerufen wird. Der *archNumber* muss immer der Buchstabe C vorangestellt werden. (Gültige Einträge sind C0-C7.)
CP	Optional. Spezifiziert die CP-Bewegung.
searchExpr	Optional. Ein Sense-, Till- oder Find-Ausdruck. **Sense** | **Till** | **Find** **Sense Sw**(*expr*) = {**On** | **Off**} **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können zum Jump-Befehl hinzugefügt werden, um E/A-Befehle und andere Befehle während der Bewegung auszuführen.

Beschreibung

Bewegt den Arm in einer 3D-Gate-Bewegung von der aktuellen Position zum Zielpunkt. Die 3D-Bewegung besteht aus einer Depart-Bewegung, einer Span-Bewegung und einer Approach-Bewegung. Die Depart-Bewegung von der aktuellen Position zum Depart-Punkt ist immer eine CP-Bewegung. Die Span-Bewegung vom Depart-Punkt zum Anfangs-Approachpunkt ist eine PTP-Bewegung in **Jump3** und eine CP-Bewegung in **Jump3CP.**
Die Approach-Bewegung vom Anfangs-Approachpunkt zum Zielpunkt ist immer eine CP-Bewegung.



Die Arch-Bewegung wird durch das Spezifizieren der Arch-Nummer erreicht. Die Arch-Bewegung für Jump3 und Jump3CP wird in der Abbildung dargestellt. Für die Arch-Bewegung muss die Depart-Distanz größer sein als die Arch-Hubdistanz und die Approach-Distanz muss größer sein als die Arch-Absenkdistanz.



Jump3CP verwendet den Geschwindigkeitswert aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS. Lesen Sie den Abschnitt *Jump3CP mit CP verwenden* für Informationen über die Beziehung Geschwindigkeit-Beschleunigung und Beschleunigung-Verzögerung. Wenn jedoch der ROT-Parameter verwendet wird, verwendet **Jump3CP** den Geschwindigkeitswert aus SpeedR und die Beschleunigungs- und Verzögerungswerte aus AccelR. In diesem Fall haben die Geschwindigkeitswerte aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS keine Wirkung.

Wenn der Bewegungsabstand bei 0 liegt und nur die Werkzeugausrichtung verändert wird, tritt für gewöhnlich ein Fehler auf. Wenn jedoch der ROT-Parameter verwendet wird und die Beschleunigung und Verzögerung der Werkzeugrotation bevorrechtigt werden, ist eine fehlerfreie Bewegung möglich. Wenn keine Ausrichtungsänderung durch den ROT-Parameter vorliegt und die Bewegungsdistanz nicht bei 0 liegt, tritt ein Fehler auf.

Auch wenn die Werkzeugrotation im Vergleich zur Bewegungsdistanz groß ist und wenn die Rotationsgeschwindigkeit die spezifizierte Geschwindigkeit des Manipulators überschreitet, tritt ein Fehler auf. In diesem Fall reduzieren Sie die Geschwindigkeit oder hängen Sie den ROT-Parameter an, um die Rotationsgeschwindigkeit / -beschleunigung /-verzögerung zu bevorrechtigen.

Hinweise

Die Jump3-Span-Bewegung ist eine PTP-Bewegung.

Es ist schwierig, den exakten Trajektoriebereich der Jump3-Span-Bewegung vorherzusagen. Stellen Sie daher sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert und die Roboterarme nicht mit dem Roboter selbst kollidieren.

Der Trajektoriebereich der Jump3-Bewegung ändert sich je nach Bewegung und Geschwindigkeit.

Der Trajektoriebereich der Jump3-Bewegung setzt sich aus Depart-, Span- und Approach-Bewegungen zusammen. Es handelt sich nicht um einen CP-Trajektoriebereich. Der tatsächliche Jump3-Trajektoriebereich der Bogenbewegung wird nicht ausschließlich durch die **Arch**-Parameter bestimmt. Auch die Bewegung und die Geschwindigkeit spielen eine Rolle.

Optimieren Sie den Jump3-Trajektoriebereich in Ihren Anwendungen mit großer Sorgfalt. Führen Sie Jump3 mit der gewünschten Bewegung und Geschwindigkeit aus, um den tatsächlichen Trajektoriebereich zu überprüfen.

Wenn die Geschwindigkeit verringert wird, wird auch der Trajektoriebereich verringert. Wenn Jump3

bei hoher Geschwindigkeit ausgeführt wird, um den Trajektoriebereich eines Bewegungsbogens festzulegen, ist es möglich, dass der Greifer bei niedrigerer Geschwindigkeit mit Hindernissen kollidiert.

In einem Jump3-Trajektoriebereich vergrößert sich die Depart-Distanz und verkleinert sich die Approach-Distanz, wenn die Bewegungsgeschwindigkeit auf hoch gestellt ist. Wenn die Approach-Distanz des Trajektoriebereichs kürzer ist als erwartet, verringern Sie die Geschwindigket bzw. die Verzögerung oder verlängern Sie die Absenkstrecke.
Selbst wenn Jump-Befehle mit derselben Distanz und Geschwindigkeit ausgeführt werden, wird der Trajektoriebereich durch die Bewegung der Roboterarme beeinflusst. Bei SCARA-Robotern vergrößert sich z. B. die Depart-Distanz und verkleinert sich die Approach-Distanz, wenn die Bewegung des ersten Armes sehr groß ist. Wenn die Approach-Distanz kleiner wird und der Trajektoriebereich kürzer ist als erwartet, verringern Sie die Geschwindigkeit bzw. die Verzögerung oder verlängern Sie die Approach-Distanz.

LimZ hat keinen Einfluss auf Jump3 und Jump3CP

LimZ hat keinen Einfluss auf Jump3 oder Jump3CP, da die Span-Bewegung nicht notwendigerweise senkrecht zur Z-Achse des Koordinatensystems verläuft.

Mögliche Beschleunigungsfehler

Ein Beschleunigungsfehler kann während der Ausführung einer Arch-Bewegung durch den Befehl Jump3 oder Jump3CP auftreten. Dieser Fehler wird häufig ausgegeben, wenn der Großteil der Bewegung während Depart oder Approach dieselbe Achse wie die Span-Bewegung verwendet. Um diesen Fehler zu vermeiden, reduzieren Sie die Beschleunigung- / Verzögerungs-Geschwindigkeit der Span-Bewegung mithilfe des Accel-Befehls für Jump3 oder mithilfe des AccelS-Befehls für Jump3CP. Je nach Bewegung und Ausrichtung des Roboter kann es außerdem hilfreich sein, die Beschleunigung und Verzögerung der Depart-Bewegung (Approach-Bewegung) mithilfe des AccelS-Befehls zu verringern.

Jump3 und Jump3CP mit CP verwenden

Der CP-Parameter veranlasst den Arm, zur *destination* zu fahren, ohne langsamer zu werden oder an dem durch *destination* definierten Punkt anzuhalten. Dadurch wird es dem Benutzer ermöglicht, eine Serie von Bewegungsbefehlen miteinander zu verketten, was den Arm veranlasst, sich entlang eines kontinuierlichen Weges zu bewegen und während dieser Bewegung eine bestimmte Bewegung einzuhalten. Die Befehle **Jump3** und **Jump3CP** ohne CP veranlassen den Arm immer dazu, bis zum vollständigen Halt herunterzubremsen, bevor er den Punkt *destination* erreicht hat.

Verwandte Befehle

Accel, Arc, Arch, Go, JS, JT, Punktausdruck, Pulse, Sense, Speed, Stat, Till

Beispiel einer Jump3-Anweisung

```
' 6-Achsroboterbewegung, die wie Jump bei einem SCARA-Roboter
funktioniert
Jump3  Here :Z(100), P3 :Z(100), P3

' Depart und Approach verwenden Z-Werkzeugkoordinaten
Jump3  Here -TLZ(100), P3 -TLZ(100), P3

' Depart verwendet die Basis Z; Approach verwendet das Werkzeug Z
Jump3  Here +Z(100), P3 -TLZ(100), P3

Beispiel für die Depart-Bewegung von P1 zu Werkzeug 1 und die Approach-
Bewegung von P3 zu Werkzeug 2

Arch 0,20,20
Tool 1
Go P1

P2 = P1 -TLZ(100)
Tool 2
Jump3 P2, P3-TLZ(100), P3 C0
```

LCase$-Funktion

F

Gibt eine Zeichenkette aus, die in Kleinbuchstaben umgewandelt wurde.

Syntax

LCase$(*string*)

Parameter

string Ein gültiger Zeichenkettenausdruck.

Rückgabewerte

Die konvertierte Kleinbuchstaben-Zeichenkette.

Verwandte Befehle

LTrim$, Trim$, RTrim$, UCase$

Beispiel einer LCase$-Funktion

```
str$ = "Data"
str$ = LCase$(str$)  ' str$ = "data"
```

Left$-Funktion

F

Gibt eine Teilkette von der linken Seite eines Zeichenkettenausdrucks aus.

Syntax

Left$*(string, count)*

Parameter

string	Zeichenkettenausdruck, von dem die ganz links stehenden Zeichen kopiert werden.
count	Anzahl der zu kopierenden Zeichen aus *string,* beginnend mit dem Zeichen, das ganz links steht.

Rückgabewerte

Gibt eine Zeichenkette der ganz links stehenden *number*-Zeichen der durch den Benutzer definierten Buchstabenzeichenkette aus.

Beschreibung

Left$** gibt die ganz links stehenden *number*-Zeichen einer durch den Benutzer definierten Zeichenkette aus. **Left$ kann so viele Zeichen ausgeben, wie in der Zeichenkette vorhanden sind.

Verwandte Befehle

Asc, Chr$, InStr, Len, Mid$, Right$, Space$, Str$, Val

Beispiel einer Left$-Funktion

Das Beispiel unten zeigt ein Programm, das Daten eines Teils als Zeichenkette übergeben bekommt und diese in Teilnummer, Teilnamen und Teileanzahl zerlegt.

```
Function ParsePartData(DataIn$ As String, ByRef PartNum$ As String,
ByRef PartName$ As String, ByRef PartCount As Integer)

    Integer pos
    String temp$

    pos = Instr(DataIn$, ",")
    PartNum$ = Left$(DataIn$, pos - 1)

    DataIn$ = Right$(datain$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")

    PartName$ = Left$(DataIn$, pos - 1)

    PartCount = Val(Right$(datain$, Len(DataIn$) - pos))

Fend
```

Weitere Beispiele des Left$-Befehls vom Befehlseingabefenster.

```
> Print Left$("ABCDEFG", 2)
 AB

> Print Left$("ABC", 3)
 ABC
```

Len-Funktion

F

Gibt die Anzahl von Zeichen in einer Zeichenkette aus.

Syntax

Len*(string)*

Parameter

string	Zeichenkettenausdruck.

Rückgabewerte

Gibt einen Integer aus, der die Anzahl der Zeichen in der *string*-Zeichenkette aufzeigt, die als Argument an den **Len**-Befehl gegeben wurde.

Beschreibung

Len gibt einen Integer aus, der für die Anzahl von Zeichen in einer Zeichenkette steht, die durch den Benutzer definiert wurde. **Len** gibt Werte zwischen 0 und 255 aus (da eine Zeichenkette zwischen 0 und 255 Zeichen enthalten kann).

Verwandte Befehle

Asc, Chr$, InStr, Left$, Mid$, Right$, Space$, Str$, Val

Beispiel einer Len-Funktion

Das Beispiel unten zeigt ein Programm, das Daten eines Teils als Zeichenkette übergeben bekommt und diese in Teilnummer, Teilnamen und Teileanzahl zerlegt.

```
Function ParsePartData(DataIn$ As String, ByRef PartNum$ As String,
ByRef PartName$ As String, ByRef PartCount As Integer)

    Integer pos
    String temp$

    pos = Instr(DataIn$, ",")
    PartNum$ = Left$(DataIn$, pos - 1)

    DataIn$ = Right$(datain$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")

    PartName$ = Left$(DataIn$, pos - 1)

    PartCount = Val(Right$(datain$, Len(DataIn$) - pos))

Fend
```

Weitere Beispiele des **Len**-Befehls vom Befehlseingabefenster.

```
> ? len("ABCDEFG")
7

> ? len("ABC")
3

> ? len("")
0
>
```

LimZ-Anweisung

Bestimmt den Vorgabewert für die Höhe der Z-Achse bei Jump-Befehlen.

> S

Syntax

(1) LimZ *zLimit*
(2) LimZ

Parameter

zLimit Ein Koordinatenwert innerhalb des beweglichen Bereichs der Z-Achse.

Rückgabewerte

Zeigt den aktuellen LimZ-Wert an, wenn der Parameter weggelassen wird.

Beschreibung

LimZ bestimmt die maximale Höhe der Z-Achse, die erreicht wird, wenn der Jump-Befehl verwendet wird, wobei der Roboterarm die Z-Achse hebt, sich in der X-Y-Ebene bewegt, und die Z-Achse wieder senkt. **LimZ** ist der Vorgabewert für die Z-Achse, der verwendet wird, um die am höchsten gelegene Position zu definieren, die bei einer durch den Jump-Befehl ausgelösten Bewegung verwendet wird. Wenn ein spezifischer **LimZ-**Wert im Jump-Befehl nicht definiert ist, wird die letzte **LimZ**-Einstellung für den Jump-Befehl verwendet.

Hinweis

Zurücksetzen des LimZ-Wertes auf 0

Ein Neustart der Steuerung und die Ausführung der Befehle SFree, SLock und Motor On setzen den LimZ-Wert auf 0 zurück.

Der LimZ-Wert ist nicht gültig für Arm-, Werkzeug- oder lokale Koordinaten:

Die Höhenbegrenzung für die Z-Achse entspricht dem Z-Achsen-Wert für das Roboter-Koordinatensystem. Es ist nicht der Z-Achsen-Wert für Arm-, Werkzeug- oder lokale Koordinaten. Treffen Sie daher die notwendigen Schutzvorkehrungen für den Gebrauch von Werkzeugen oder Greifern mit unterschiedlicher Betriebshöhe.

LimZ hat keinen Einfluss auf Jump3 und Jump3CP

LimZ hat keinen Einfluss auf Jump3 oder Jump3CP, da die Span-Bewegung nicht notwendigerweise senkrecht zur Z-Achse des Koordinatensystems verläuft.

Verwandte Befehle

Jump

Beispiel einer LimZ-Anweisung

Das Beipiel unten zeigt die Verwendung von LimZ in Jump-Vorgängen.

```
Function  main
    LimZ -10            'Setzt den LimZ-Standardwert
    Jump P1             'Bewegt die Achse für den Jump-Befehl aufwärts
                         in die Position Z=-10
    Jump P2 LimZ -20    'Bewegt die Achse für den Jump-Befehl aufwärts
                         in die Position Z=-20
    Jump P3          'Bewegt die Achse für den Jump-Befehl aufwärts in
                         die Position Z=-10
Fend
```

LimZ-Funktion

F

Gibt die aktuelle LimZ-Einstellung aus.

Syntax

LimZ

Rückgabewerte

Reelle Zahl, die die aktuelle LimZ-Einstellung enthält.

Verwandte Befehle

LimZ-Anweisung

Beispiel einer LimZ-Funktion

```
Real savLimz

savLimz = LimZ
LimZ -25
Go pick
LimZ savLimZ
```

Line Input-Anweisung

> S

Liest eine Zeile der Eingangsdaten und ordnet die Daten einer Zeichenkettenvariablen zu.

Syntax

Line Input *stringVar$*

Parameter

stringVar$	Name einer Zeichenkettenvariablen. (Beachten Sie, dass die Zeichenkettenvariable mit dem $-Zeichen enden muss.)

Beschreibung

Line Input liest eine Zeile der Eingangsdaten des Anzeigegeräts und ordnet die Daten der Zeichenkettenvariablen zu, die in der Line Input Anweisung verwendet wird. Wenn die Line Input-Anweisung bereit ist, Daten vom Benutzer zu empfangen, wird ein "?" -Hinweis auf dem Anzeigegerät angezeigt. Die Eingangsdatenzeile nach dem Prompt wird dann als Wert für die Zeichenkettenvariable aufgenommen. Nach Eingabe der Datenzeile drücken Sie bitte die [ENTER]-Taste.

Verwandte Befehle

Input, Input #, Line Input#, ParseStr

Beispiel einer Line Input-Anweisung

Das folgende Beispiel zeigt die Verwendung von **Line Input**.

```
Function Main
    String A$
    Line Input A$  ' Liest eine Zeile der Eingangsdaten in A$ ein
    Print A$
Fend
```

```
Starten Sie das obige Programm mit der F5-Taste oder aus dem Run-Menü
des EPSON RC+-Hauptfensters. Eine resultierende Run-Session kann wie
folgt aussehen:
```

```
?A, B, C
A, B, C
```

Line Input #-Anweisung

Liest die Daten einer Zeile vom angegebenen Kommunikationsportoder dem Gerät. > S

Syntax

Line Input #*ha*ndle, *stringVar$*

Parameter

handle	Kommunikationsport oder Geräte-ID. Kommunikationsports können in OpenCom- (RS232) und OpenNet- (TCP/IP) Anweisungen definiert werden. Die Integerwerte der Geräte-IDs lauten wie folgt: 21 RC+ 23 OP 24 TP
stringVar$	Eine Zeichenkettenvariable. (Beachten Sie, dass Zeichenkettenvariablen mit dem $-Zeichen enden müssen.)

Beschreibung

Line Input # liest Zeichenkettendaten einer Zeile aus dem Gerät, das durch den *handle*-Parameter Spezifiziert wurde, und ordnet die Daten der Zeichenkettenvariablen *stringVar$* zu.

Verwandte Befehle

Input, Input #, Line Input

Beispiel einer Line Input #-Anweisung

In diesem Beispiel werden die Zeichenkettendaten des Kommunikationsports Nr. 1 empfangen und der Zeichenkettenvariablen A$ zugeordnet.

```
Function lintest
    String a$
    Print #1, "Please input string to be sent to robot"
    Line Input #1, a$
    Print "Value entered = ", a$
Fend
```

LoadPoints-Anweisung

Lädt eine Punktedatei in den Punktspeicherbereich für den aktuellen Roboter.

> S

Syntax

LoadPoints *fileName* **[, Merge]**

Parameter

fileName	Zeichenkettenausdruck, der die spezifische Datei enthält, die in den Punktspeicherbereich des aktuellen Roboters geladen werden soll. Die Endung .PTS wird angehängt, so dass der Benutzer keine Endung angeben muss. Die Datei muss im aktuellen Projekt vorhanden sein. Es kann kein Pfad angegeben werden.
Merge	Optional. Wenn Merge angegeben wird, werden die aktuellen Punkte nicht gelöscht, bevor die neuen Punkte geladen werden. Die Punkte in der Datei werden zu den aktuellen Punkten hinzugefügt. Wenn ein Punkt in der Datei bereits existiert, wird der Punkt im Speicher überschrieben.

Beschreibung

LoadPoints lädt Punktedateien in den Hauptspeicherbereich der Steuerung.

Verwenden Sie **Merge,** um Punktedateien zu kombinieren. Es kann beispielsweise eine Hauptpunktedatei vorhanden sein, die gemeinsame Punkte für Locals Parken, etc. im Bereich von 0-100 enthält. In diesem Fall wird **Merge** verwendet, um andere Punktedateien für jedes verwendete Objekt zu laden, ohne die gemeinsam verwendeten Punkte zu löschen. Der Bereich kann z. B. zwischen 101 und 999 liegen.

Mögliche Fehler

Die Datei existiert nicht:

Wenn *fileName* (Dateiname) nicht existiert, wird ein Fehler ausgegeben.

Verwandte Befehle

ClearPoints, SavePoints

Beispiel einer LoadPoints-Anweisung

```
Function main

  ' Lädt gemeinsame Punkte für den aktuellen Roboter.
  LoadPoints "R1Common.pts"

  ' Verbindet Punkte für Teilmodell 1
  LoadPoints "R1Model1.pts", Merge

Fend
```

Local-Anweisung

Definiert lokale Koordinatensysteme und zeigt sie an.

> S

Syntax

(1) Local *localNumber*, (*pLocal1 : pBase1*), *(pLocal2 : pBase2),* [{ **L** | **R** }], [**BaseU**]
(2) Local *localNumber, pOrigin*
(3) Local *localNumber, pOrigin,* [*pXaxis*], [*pYaxis*], [{ **X** | **Y** }]
(4) Local ***localNumber***

Parameter

localNumber	Die Nummer des lokalen Koordinatensystems. Es können bis zu 15 lokale Koordinatensysteme definiert werden (mit Integer-Werten von 1 bis 15).
pLocal1, pLocal2	Punktvariablen mit Punktedaten im lokalen Koordinatensystem.
pBase1, pBase2	Punktvariablen mit Punktedaten im Basiskoordinatensystem.
L | **R**	Optional. Richtet den lokalen Ursprung am linken (ersten) oder rechten (zweiten) Basispunkt aus.
BaseU	Optional. Wird BaseU angegeben, befinden sich die U-Achsen-Koordinaten im Basiskoordinatensystem. Wird BaseU weggelassen, befinden sich die U-Achsen-Koordinaten im lokalen Koordinatensystem.
pOrigin	Ursprung des 3D-Locals.
pXaxis	Optional. Ein Punkt auf der X-Achse des dreidimensionalen lokalen Koordinatensystems, wenn die X-Ausrichtung spezifiziert ist.
pYaxis	Optional. Ein Punkt auf der Y-Achse des dreidimensionalen lokalen Koordinatensystems, wenn die Y-Ausrichtung spezifiziert ist.
X | **Y**	Optional. Wenn die X-Ausrichtung spezifiziert ist, liegt *pXaxis* auf der X-Achse des lokalen Koordinatensystems und nur die Z-Koordinate von *pYaxis* wird verwendet. Wenn die Y-Ausrichtung spezifiziert ist, liegt *pYaxis* auf der Y-Achse des lokalen Koordinatensystems und nur die Z-Koordinate von *pXaxis* wird verwendet. Wird die Ausrichtung weggelassen, wird von einer X-Ausrichtung ausgegangen.

Beschreibung

(1) Local definiert ein lokales Koordinatensystem durch die Spezifizierung zweier Punkte, *pLocal1* und *pLocal2*, die darin enthalten sind, und die mit zwei Punkten *pBase1* und *pBase2* übereinstimmen, die im Basiskoordinatensystem vorhanden sind.

Beispiel:

```
Local 1, (P1:P11), (P2:P12)
```

P1 und P2 sind Punkte im lokalen Koordinatensystem. P1 und P2 sind Punkte im Basiskoordinatensystem.

Wenn die Entfernung zwischen den zwei definierten Punkten im lokalen Koordinatensystem nicht mit der Entfernung zwischen den zwei definierten Punkten im Basiskoordinatensystem übereinstimmt, wird die XY-Ebene des lokalen Koordinatensystem in der Position definiert, in der der Mittelpunkt zwischen den beiden definierten Punkten im lokalen Koordinatensystem mit dem zwischen den beiden definierten Punkten im Basiskoordinatensystem übereinstimmt.

Gleichermaßen wird die Z-Achse des lokalen Koordinatensystems in der Position definiert, in der die Mittelpunkte miteinander übereinstimmen.

(2) Definiert ein lokales Koordinatensystem durch Definition des Ursprungs und der Achsrotationswinkel im Verhältnis zum Basiskoordinatensystem.

Beispiel:
```
Local 1, XY(x, y, z, u)
Local 1, XY(x, y, z, u, v, w)
Local 1, P1
```

(3) Definiert ein dreidimensionales lokales Koordinatensystem durch Definition des Ursprungs, des X-Achsenpunktes und des Y-Achsenpunktes. Es werden nur die X-, Y- und Z-Koordinaten eines jeden Punktes verwendet. Die U-, V- und W-Koordinaten werden nicht beachtet. Wenn der X-Ausrichtungsparameter verwendet wird, liegt *pXaxis* auf der X-Achse des lokalen Koordinatensystems und nur die Z-Koordinate von *pYaxis* wird verwendet. Wenn der Y-Ausrichtungsparameter verwendet wird, liegt *pYaxis* auf der Y-Achse des lokalen Koordinatensystems und nur die Z-Koordinate von *pXaxis* wird verwendet.

Beispiel:
```
Local 1, P1, P2, P3
Local 1, P1, P2, P3, X
Local 1, P1, P2, P3, Y
```

(4) Zeigt die spezifizierten Einstellungen des lokalen Koordinatensystems an.

Verwendung der Parameter L und R

Während Local, wie oben beschrieben, im Wesentlichen Mittelpunkte für die Positionierung der Achsen des lokalen Koordinatensystems verwendet, können Sie optional das linke oder rechte Local auch über die Parameter **L** und **R** definieren.

Left Local (Linkes Local)
Left local definiert ein lokales Koordinatensystem durch Definition des Punktes *pLocal1* entsprechend Punkt *pBase1* im Basiskoordinatensystem (Z-Achsen-Richtung eingeschlossen).

Right Local (Rechtes Local)
Right local definiert ein lokales Koordinatensystem durch Definition des Punktes *pLocal2* entsprechend Punkt *pBase2* im Basiskoordinatensystem (Z-Achsen-Richtung eingeschlossen).

Verwendung des BaseU-Parameters

Wenn der **BaseU**-Parameter weggelassen wird, wird die U-Achse des lokalen Koordinatensystems automatisch gemäß der X- und Y-Koordinatenwerte der definierten vier Punkte korrigiert. Die zwei Punkte im Basiskoordinatensystem können daher zunächst U-Koordinaten-Werte haben.

Es kann wünschenswert sein, die U-Achse des lokalen Koordinatensystems auf der Grundlage der U-Koordinaten-Werte der zwei Punkte im Basiskoordinatensystem selbst zu korrigieren, anstatt sie automatisch korrigieren zu lassen (z. B. Korrektur der Rotationsachse durch Teach-In). Um dies zu erreichen, geben Sie bitte den **BaseU**-Parameter an.

Verwandte Befehle

ArmSet, Base, ECPSet, LocalClr, TLSet, Where

Beispiel einer Local-Anweisung

Hier sind einige Beispiele vom Befehlseingabefenster:

Nach links ausgerichtetes Local:

```
> p1 = 0, 0, 0, 0/1
> p2 = 100, 0, 0, 0/1
> p11 = 150, 150, 0, 0
> p12 = 300, 150, 0, 0
> local 1, (P1:P11), (P2:P12), L
> p21 = 50, 0, 0, 0/1
> go p21
```

Local, das nur durch den Ursprungspunkt definiert wurde:

```
> local 1, 100, 200, -20
```

Local, das nur durch den Ursprungspunkt definiert wurde, der um 45 Grad um die X-Achse rotiert wurde.

```
> local 2, 50, 200, 0, 0, 45
```

Dreidimensionales Local, in dem sich p2 nach der X-Achse des Locals ausrichtet.

```
> local 3, p1, p2, p3, x
```

Dreidimensionales Local, in dem sich p3 nach der Y-Achse des Locals ausrichtet.

```
> local 4, p1, p2, p3, y
```

Local-Funktion

F

Gibt die Local-Nummer eines Punktes aus.

Syntax

Local(*localNumber*)

Parameter

localNumber	Nummer des lokalen Koordinatensystems (ganze Zahl von 1 bis 15) unter Verwendung eines Ausdrucks oder eines numerischen Werts.

Rückgabewerte

Spezifizierte Daten des lokalen Koordinatensystems als Punktedaten.

Verwandte Befehle

Local-Anweisung

Beispiel einer Local-Funktion

```
P1 = Local(1)
```

LocalClr-Anweisung

Löscht ein lokales Koordinatensystem (macht Definitionen rückgängig). > S

Syntax

LocalClr *localNumber*

Parameter

localNumber	Integer-Ausdruck, der angibt, welches von 15 Locals (ganze Zahl von 1 bis 15) gelöscht werden soll.

Verwandte Befehle

Arm, ArmSet, ECPSet, Tool, TLClr, TLSet

Beispiel einer LocalClr-Anweisung

```
LocalClr 1
```

LocalDef-Funktion

Gibt den Status einer Local-Definition aus.

> F

Syntax

LocalDef (*localCoordinateNumber*)

Parameter

localCoordinateNumber	Integer-Ausdruck, der angibt, für welche lokale Koordinate der Status ausgegeben werden soll.

Rückgabewerte

Wahr, wenn das spezifizierte Local definiert wurde, sonst Falsch.

Verwandte Befehle

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLClr, TLSet

Beispiel einer LocalDef-Funktion

```
Function DisplayLocalDef(localNum As Integer)

   If LocalDef(localNum) = False Then
      Print "Local ", localNum, "is not defined"
   Else
      Print "Local 1: ",
      Print Local(localNum)
   EndIf
Fend
```

Lof-Funktion

F

Überprüft, ob im angegebenen RS-232- oder TCP/IP-Port Eingabezeilen im Puffer vorhanden sind.

Syntax

Lof(*portNumber*)

Parameter

portNumber Nummer des Kommunikationsports.

Rückgabewerte

Die Anzahl der Datensätze im Puffer. Wenn sich keine Daten im Puffer befinden, beträgt der Rückgabewert von **Lof** Null (0).

Beschreibung

Lof überprüft, ob der angegebene Port Daten empfangen hat. Die empfangenen Daten werden unabhängig vom Input#-Befehl im Puffer gespeichert.

Verwandte Befehle

ChkCom, ChkNet, Input#

Beispiel einer Lof-Funktion

Dieses Beispiel für das Befehlseingabefenster druckt die Anzahl der von Kommunikationsport Nr.1 empfangenen Datensätze aus.

```
>print lof(1)
 5
>
```

Long-Anweisung

Deklariert Long Integer-Variablen (4 Byte-Integer). **S**

Syntax

Long *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]**...**]

Parameter

varName	Variablenname, den der Benutzer als **Long** deklarieren möchte.
subscripts	Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus: (ubound1, [ubound2], [ubound3]) Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension. Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert. Die Gesamtzahl der Elemente der Matrixelemente für lokale und Global Preserve-Variablen ist 1000. Die Gesamtzahl der Matrixelemente für globale und Modulvariablen ist 10000. Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)

Beschreibung

Long wird verwendet, um Variablen als **Long** zu deklarieren. **Long**-Variabelen können ganze Zahlen im Wertebereich zwischn -2.147.483.648 und 2.147.483.647 enthalten. Lokale Variablen sollten in einer Funktion ganz oben deklariert sein. Globale und Modulvariabeln müssen außerhalb von Funktionen deklariert werden.

Verwandte Befehle

Boolean, Byte, Double, Global, Integer, Real, String

Beispiel einer Long-Anweisung

Das folgende Beispiel zeigt ein einfaches Programm, das unter Verwendung von **Long** einige Variablen als Long deklariert.

```
Function longtest
    Long A(10)         'Eindimensionale Matrix aus Longs
    Long B(10, 10)     'Zweidimensionale Matrix aus Longs
    Long C(10, 10, 10) 'Dreidimensionale Matrix aus Longs
    Long var1, arrayVar(10)
    Long i
    Print "Please enter a Long Number"
    Input var1
    Print "The Integer variable var1 = ", var1
    For I = 1 To 5
        Print "Please enter a Long Number"
        Input arrayVar(i)
        Print "Value Entered was ", arrayVar(i)
    Next I
Fend
```

LSet$-Funktion

F

Gibt die angegebene Zeichenkette mit abschließenden Leerzeichen aus, die bis zur angegebenen Länge angehängt werden.

Syntax

LSet$ (*string, length*)

Parameter

string	Zeichenkettenausdruck.
length	Integer-Ausdruck für die Gesamtlänge der ausgegebenen Zeichenkette.

Rückgabewerte

Spezifizierte Zeichenkette mit angehängten abschließenden Leerzeichen.

Verwandte Befehle

RSet$, Space$

Beispiel einer LSet$-Funktion

```
str$ = "123"
str$ = LSet$(str$, 10)  ' str$ = "123       "
```

LShift-Funktion

F

Verschiebt numerische Daten um eine vom Benutzer definierte Bitanzahl nach links.

Syntax

LShift*(number, shiftBits)*

Parameter

number	Der Integer-Ausdruck, der verschoben werden soll.
shiftBits	Die Bitanzahl (Integer zwischen 0 und 31), um die *number* nach links verschoben werden soll.

Rückgabewerte

Gibt ein numerisches Ergebnis aus, das dem *number*-Wert entspricht, um den die Bits um die in *shiftBits* definierte Anzahl von Bits nach links verschoben wurden.

Beschreibung

LShift verschiebt die spezifizierten numerischen Daten (*number*) um die festgelegte Bitanzahl (*shiftBits*) nach links (an eine höhereStelle). Die nachrückenden Bits niedrigeren Wertes werden auf 0 gesetzt.

Die einfachste Erklärung für LShift ist, dass es das Ergebnis von $number * 2^{shiftBits}$ ausgibt.

Hinweis

Typ numerischer Daten:

Die numerischen Daten (*number*) können aus jedem gültigen numerischen Datentyp bestehen. **LShift** arbeitet mit den Datentypen Byte, Integer, Long und Real.

Verwandte Befehle

And, Not, Or, RShift, Xor

Beispiel einer LShift-Funktion

```
Function lshiftst
    Integer i
    Integer num, snum
    num = 1
    For i = 1 to 10
         Print "i =", i
         snum = LShift(num, i)
         Print "The shifted num is ", snum
    Next i
Fend
```

Weitere beispielhafte Ergebnisse des LShift-Befehls vom Befehlseingabefenster.

```
> Print LShift(2,2)
8
> Print LShift(5,1)
10
> Print LShift(3,2)
12>
```

LTrim$-Funktion

F

Gibt eine Zeichenkette aus, die mit der definierten Zeichenkette identisch ist, wobei die vorangehenden Stellen nicht aufgeführt werden.

Syntax

LTrim$ (*string*)

Parameter

string Zeichenkettenausdruck.

Rückgabewerte

Spezifizierte Zeichenkette, bei der die vorangehenden Leerzeichen entfernt wurden.

Verwandte Befehle

RTrim$, Trim$

Beispiel einer LTrim$-Funktion

```
str$ = "  data  "
str$ = LTrim$(str$)  ' str$ = "data  "
```

Mask-Operator

S

Bitweise Maske für die Bedingung der Wait-Anweisung.

Syntax

Wait *expr1* **Mask** *exrp2*

Parameter

expr1	Jeder gültige Ausdruck für die Eingangsbedingung für Wait.
expr2	Jeder gültige Ausdruck, der ein numerisches Ergebnis ausgibt.

Beschreibung

Der **Mask**-Operator ist eine bitweise And-Verknüpfung für Eingangsbedingungen für Wait-Anweisungen.

Verwandte Befehle

Wait

Beispiel eines Mask-Operators

```
' Warten, bis die unteren 3 Bits des Eingangsports 0 1 entsprechen.
Wait In(0) Mask 7 = 1
```

MemIn-Funktion

F

Gibt den Status des angegebenen Merkerports aus. Jeder Port enthält 8 Merkerbits.

Syntax

MemIn(*portNumber***)**

Parameter

portNumber Integer-Ausdruck, der für die Merker-Bytes steht.

Rückgabewerte

Gibt einen Integer-Wert zwischen 0 und 255 aus. Der Rückgabewert beträgt 8 Bits. Jedes dieser Bits entspricht einem Merkerbit.

Beschreibung

MemIn gibt Ihnen die Möglichkeit, die Werte von 8 Merkerbits gleichzeitig zu betrachten. Der **MemIn**-Befehl kann verwendet werden, um den Status der 8 Merkerbits in einer Variablen zu speichern. Alternativ kann er auch mit dem Wait-Befehl verwendet werden, bis eine spezifische Bedingung erfüllt ist, die mehr als ein Merkerbit einschließt.

Da 8 Bits zeitgleich ausgegeben werden, liegen die Rückgabewerte im Bereich von 0 bis 255. Siehe Tabelle unten für die Entsprechung der Integer-Rückgabewerte zu den einzelnen Merkerbits entsprechen.

Merker-Ergebnis (unter Verwendung von Port 0)

Rückgabewerte	7	6	5	4	3	2	1	0
1	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
5	Aus	Aus	Aus	Aus	Aus	Ein	Aus	Ein
15	Aus	Aus	Aus	Aus	Ein	Ein	Ein	Ein
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Merker-Ergebnis (unter Verwendung von Port 31)

Rückgabewerte	255	254	253	252	251	250	249	248
3	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
7	Aus	Aus	Aus	Aus	Aus	Ein	Ein	Ein
32	Aus	Aus	Ein	Aus	Aus	Aus	Aus	Aus
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Hinweise

Unterschied zwischen MemIn und MemSw

Der MemSw-Befehl ermöglicht es dem Benutzer, den Wert eines Merkerbits zu lesen. Der Rückgabewert von MemSw ist entweder eine 1 oder eine 0 für ein ein- bzw. ausgeschaltetes Merkerbit. MemSw kann jedes Merkerbit einzeln überprüfen. Der **MemIn**-Befehl ist dem MemSw-Befehl sehr ähnlich, da auch er verwendet wird, um den Status der Merkerbits zu überprüfen. Es gibt jedoch einen grundsätzlichen Unterschied. Der MemIn-Befehl überprüft gleichzeitig 8 Merkerbits, wohingegen der MemSw-Befehl immer nur ein einziges Merkerbit überprüfen kann. **MemIn** gibt einen Wert zwischen 0 und 255 aus, der dem Benutzer anzeigt, welches der 8 Merkerbits ein- und welches ausgeschaltet ist.

Verwandte Befehle

In, InBCD, Off, MemOff, On, MemOn, OpBCD, Oport, Out, MemOut, Sw, MemSw, Wait

Beispiel einer MemIn-Funktion

Das Programmbeispiel unten liest den Wert der ersten 8 Merkerbits und stellt vor dem Fortfahren sicher, dass momentan alle 8 E/As auf 0 gesetzt sind. Wenn sie nicht auf 0 stehen, wird eine Fehlermeldung an den Bediener ausgegeben und der Task wird angehalten.

```
Function main
    Integer var1

    var1 = MemIn(0)  'Holt die ersten 8 Merkerbit-Werte
    If var1 = 0 Then
        Go P1
        Go P2
    Else
        Print "Error in initialization!"
        Print "First 8 memory I/O bits were not all set to 0"
    EndIf
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> memout 0, 1
> print MemIn(0)
1
> memon 1
> print MemIn(0)
3
> memout 31,3
> print MemIn(31)
3
> memoff 249
> print MemIn(31)
1
>
```

MemOff-Anweisung

> S

Schaltet das spezifizierte Bit des Merkers aus.

Syntax

MemOff { *bitNumber* | *memIOLabel* }

Parameter

bitNumber	Integer-Ausdruck, der für Merkerbits steht.
memIOLabel	Merker-Label.

Beschreibung

MemOff schaltet das spezifizierte Bit des Merkers aus. Die 512 Merkerbits sind normalerweise sehr gut geeignet für die Verwendung als Status-Bits für Zwecke wie Ein / Aus, Wahr / Falsch, Erledigt / Nicht erledigt etc. Der MemOn-Befehl schaltet das Merkerbit ein, während **MemOff** ihn ausschaltet. Der MemSw-Befehl wird verwendet, um den aktuellen Status des spezifizierten Merkerbits zu überprüfen. Der Wait-Befehl kann ebenfalls mit dem Merkerbit verwendet werden, um das System dazu zu veranlassen zu warten, bis ein definierter Merker-Status eingestellt ist.

Hinweis

Merkerausgänge aus

Alle Merkerbits werden ausgeschaltet, wenn die Steuerung neu gestartet wird. Sie werden nicht durch Not-Aus, eine offene Sicherheitsabschrankung, Programmende, den Reset-Befehl oder einen EPSON RC+-Neustart ausgeschaltet.

Verwandte Befehle

In, MemIn, InBCD, Off, On, MemOn, OpBCD, Oport, Out, MemOut, Sw, MemSw, Wait

Beispiel einer MemOff-Anweisung

Das folgende Beispiel zeigt zwei Tasks. Jeder der beiden Tasks kann Bewegungsbefehle initiieren. Jedoch wird ein Sicherungsmechanismus zwischen den beiden Tasks verwendet, um sicherzustellen, dass ein Task erst dann die Kontrolle über die Bewegungsbefehle des Roboters erhält, wenn der andere Task deren Verwendung abgeschlossen hat. Dadurch können zwei Tasks Bewegungsbefehle korrekt, geordnet und vorhersehbar ausführen. MemSw wird in Kombination mit dem Wait-Befehl verwendet, um zu warten, bis der 1. Merker den richtigen Wert erreicht hat, von dem an es sicher ist, eine neue Bewegung auszuführen. MemOn und MemOff werden verwendet, um den Merker für die richtige Synchronisierung ein- oder auszuschalten.

```
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For I = 1 to 100
        Wait Sw($1) = 0
        Go P(i)
        MemOn 1
    Next I
Fend

Function task2
    Integer I
    For I = 101 to 200
        Wait MemSw(1) = 1
        Go P(i)
        MemOff 1
    Next I
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> MemOn 1      'Schaltet Merkerbit Nr. 1 ein
> Print MemSw(1)
1
> MemOff 1     'Schaltet Merkerbit Nr.1 aus
> Print MemSw(1)
0
```

MemOn-Anweisung

Schaltet das spezifizierte Bit des Merkers ein.

> S

Syntax

MemOn { *bitNumber* | *memIOLabel* }

Parameter

bitNumber	Integer-Ausdruck, der für Merkerbits steht.
memIOLabel	Merker-Label.

Beschreibung

MemOn schaltet das spezifizierte Bit des Robotermerkers ein. Die 512 Merkerbits werden normalerweise als Statusbits für die Taskkommunikation verwendet. Der MemOn-Befehl schaltet den Merker ein, während MemOff ihn ausschaltet. Der Befehl MemSw wird verwendet, um den aktuellen Status des angegebenen Merkers zu überprüfen. Der Befehl Wait kann ebenfalls mit dem Merker verwendet werden, um das System dazu zu veranlassen zu warten, bis ein definierter Status eingestellt ist.

Hinweis

Merkerausgänge aus

Alle Merkerbits werden ausgeschaltet, wenn die Steuerung neu gestartet wird. Sie werden nicht durch Not-Aus, eine offene Sicherheitsabschrankung, Programmende, den Reset-Befehl oder einen EPSON RC+-Neustart ausgeschaltet.

Verwandte Befehle

In, MemIn, InBCD, Off, MemOff, On, OpBCD, Oport, Out, MemOut, Sw, MemSw, Wait

Beispiel einer MemOn-Anweisung

Das folgende Beispiel zeigt zwei Tasks. Jeder der beiden Tasks kann Bewegungsbefehle initiieren. Jedoch wird ein Sicherungsmechanismus zwischen den beiden Tasks verwendet, um sicherzustellen, dass ein Task erst dann die Kontrolle über die Bewegungsbefehle des Roboters erhält, wenn der andere Task deren Verwendung abgeschlossen hat. Dadurch können zwei Tasks Bewegungsbefehle korrekt, geordnet und vorhersehbar ausführen. MemSw wird in Kombination mit dem Wait-Befehl verwendet, um zu warten, bis der 1. Merker den richtigen Wert erreicht hat, von dem an es sicher ist, eine neue Bewegung auszuführen. MemOn und MemOff werden verwendet, um den Merker für die richtige Synchronisierung ein- oder auszuschalten.

```
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For I = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
        MemOn 1
    Next I
Fend

Function task2
    Integer I
    For I = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
        MemOff 1
    Next I
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> memon 1
> print memsw(1)
1
> memoff 1
> print memsw(1)
0
```

MemOut-Anweisung

Definiert 8 Merkerbits gleichzeitig.

> S

Syntax

MemOut *portNumber, outData*

Parameter

portNumber Integer-Ausdruck, der für die Portnummer von Merkerbits steht. Die *portNumber*-Auswahl entspricht den folgenden Ausgängen:

Portnummer	Ausgänge
0	0-7
1	8-15
.	.

outData Integer zwischen 0 und 255, der für das Ausgangsmuster für die mit der Anweisung *portNumber* ausgewählte Ausgangsgruppe steht. Wenn es in hexadezimaler Form dargestellt wird, wird der Bereich von &H0 bis &HFF abgedeckt. Die niedrigere Ziffer steht für die niedrigeren Stellen (oder die ersten 4 Ausgänge) und die obere Ziffer für die höheren Stellen (oder die zweiten 4 Ausgänge).

Beschreibung

MemOut setzt unter Nutzung der vom Benutzer angegebenen Kombination aus *portNumber*- und *outData*-Werten gleichzeitig 8 Merkerbits, um festzusetzen, welche Ausgänge eingestellt werden. Der *portNumber*-Parameter definiert, welche 8er-Ausgangsgruppe verwendet werden soll, wobei *portNumber* = 0 für die Ausgänge 0-7 steht, *portNumber* = 1 für die Ausgänge 8-15 steht, etc.

Sobald eine Portnummer (*portNumber*) ausgewählt wurde, muss ein spezifisches Ausgangsmuster definiert werden. Dafür wird der *outData*-Parameter verwendet. Der *outData*-Parameter kann zwischen 0 und 255 liegen und in hexadezimalem oder im Integer-Format dargestellt werden (d. h. &H0-&HFF oder 0-255.)

Die Tabelle unten zeigt mögliche E/A-Kombinationen und ihre entsprechenden *outData*-Werte, davon ausgehend, dass die *portNumber* 0 oder entsprechend 1 ist.

Ausgangseinstellungen wenn *portNumber*=0 (Ausgangsnummer)

OutData-Wert	7	6	5	4	3	2	1	0
01	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
02	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Aus
03	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
08	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Aus
09	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Ein
10	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Aus
11	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Ein
99	Aus	Ein	Ein	Aus	Aus	Aus	Ein	Ein
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Ausgangseinstellungen wenn *portNumber*=1 (Ausgangsnummer)

OutData-Wert	15	14	13	12	11	10	9	8
01	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
02	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Aus
03	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
08	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Aus
09	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Ein
10	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Aus
11	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Ein
99	Aus	Ein	Ein	Aus	Aus	Aus	Ein	Ein
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Verwandte Befehle

In, MemIn, InBCD, MemOff, MemOn, MemSw, Off, On, OpBCD, Oport, Out, Sw, Wait

Beispiel einer MemOut-Anweisung

Das Beispiel unten zeigt, wie der Haupttask einen Hintergrundtask namens *iotask* startet. Der *iotask* ist ein einfacher Task, der die Merkerbits 0 - 3 ein- und ausschaltet. Der MemOut-Befehl macht dies möglich, indem nur ein Befehl verwendet wird, anstatt jedes Merkerbit einzeln ein- und auszuschalten.

```
Function main
  Xqt 2, iotask
  Go P1
  .
  .
Fend

Function iotask

  Do
    MemOut 0, &HF
    Wait 1
    MemOut 0, &H0    Wait 1
  Loop
Fend
```

Andere Beispiele vom Befehlsfenster sehen wie folgt aus:

```
> MemOut 1,6      'Schaltet Merkerbits 9 und 10 ein
> MemOut 2,1      'Schaltet Merkerbit 8 ein
> MemOut 3,91     'Schaltet Merkerbits 24, 25, 27, 28 und 30 ein
```

MemSw-Funktion

F

Gibt den Status des angegebenen Merkerbits aus.

Syntax

MemSw(*$bitNumber***)**

Parameter

bitNumber Integer-Ausdruck, der für die Nummern von Merkerbits steht.

Rückgabewerte

Gibt eine 1 aus, wenn das spezifizierte Bit eingeschaltet und eine 0, wenn das spezifizierte Bit ausgeschaltet ist.

Beschreibung

MemSw gibt den Status eines Merkerbits aus. Gültige Einträge für MemSw liegen im Bereich von Bit 0 bis Bit 511. MemOn schaltet das spezifizierte Bit ein, MemOff schaltet es aus.

Verwandte Befehle

In, MemIn, InBCD, MemOff, MemOn, MemOut, Off, On, OpBCD, Oport, Out, Sw, Wait

Beispiel einer MemSw-Funktion

Das folgende Beispiel zeigt zwei Tasks. Jeder der beiden Tasks kann Bewegungsbefehle initiieren. Jedoch wird ein Sicherungsmechanismus zwischen den beiden Tasks verwendet, um sicherzustellen, dass ein Task erst dann die Kontrolle über die Bewegungsbefehle des Roboters erhält, wenn der andere Task deren Verwendung abgeschlossen hat. Dadurch können zwei Tasks Bewegungsbefehle korrekt, geordnet und vorhersehbar ausführen. MemSw wird in Kombination mit dem Wait-Befehl verwendet, um zu warten, bis Merkerbit Nr. 1 den richtigen Wert erreicht hat, von dem an es sicher ist, eine neue Bewegung auszuführen.

```
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For I = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
        MemOn 1
    Next I
Fend

Function task2
    Integer I
    For I = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
        MemOff 1
    Next I
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> memon 1
> print memsw(1)
1
> memoff 1
> print memsw(1)
0
```

Mid$-Funktion

F

Gibt eine Teilkette einer Zeichenkette aus, die an einer bestimmten Position startet.

Syntax

Mid$*(string, position,* [*count*])

Parameter

string	Quell-Zeichenkettenausdruck.
position	Die Startposition in der Zeichenkette für das Kopieren von *count*-Zeichen.
count	Optional. Anzahl der zu kopierenden Zeichen aus *string,* beginnend mit dem durch *position* definierten Zeichen. Wenn *count* weggelassen wird, werden alle Zeichen von *position* bis zum Ende der Zeichenkette ausgegeben.

Rückgabewerte

Gibt eine Teilkette von Zeichen aus *string* aus.

Beschreibung

Mid$ gibt eine Teilkette von so vielen *count*-Zeichen aus, wie mit dem *position*-Zeichen in *string* beginnen.

Verwandte Befehle

Asc, Chr$, InStr, Left$, Len, Right$, Space$, Str$, Val

Beispiel einer Mid$-Funktion

Das unten dargestellte Beispiel zeigt ein Programm, das die mittleren beiden Zeichen der Zeichenkette "ABCDEFHIJ“ extrahiert und den Rest der Zeichenkette, die in Position 5 startet.

```
Function midtest
    String basestr$, m1$, m2$
    basestr$ = "ABCDEFGHIJ"
    m1$ = Mid$(basestr$, (Len(basestr$) / 2), 2)
    Print "The middle 2 characters are: ", m1$
    m2$ = Mid$(basestr$, 5)
    Print "The string starting at 5 is: ", m2$
Fend
```

Mod-Operator

Gibt den Rest einer Division zweier ganzer Zahlen aus.

Syntax

number **Mod** *divisor*

Parameter

number	Die Zahl, die geteilt wird (der Dividend).
divisor	Die Zahl, durch die *number* geteilt wird.

Rückgabewerte

Gibt den Rest, nachdem *number* durch *divisor* geteilt wurde, aus.

Beschreibung

Mod wird verwendet, um den Rest zu erhalten, nachdem zwei Zahlen dividiert wurden. Bei dem Rest handelt es sich um eine ganze Zahl. Eine geschickte Verwendung des **Mod**-Befehls ist es, festzustellen, ob eine Zahl gerade oder ungerade ist. Die Methode, mit der der **Mod-**Befehl arbeitet, ist folgende: *number* wird durch *divisor* geteilt. Der nach dieser Division verbleibende Rest ist dann der Rückgabewert für den **Mod**-Befehl.

Verwandte Befehle

Abs, Atan, Atan2, Cos, Int, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Beispiel für den Mod-Operator

Das Beispiel unten zeigt an, ob eine Zahl (var1) gerade oder ungerade ist. Wenn die Zahl gerade ist, wird als Ergebnis des Mod-Befehls eine 0 ausgegeben. Ist die Zahl ungerade, wird als Ergebnis des Mod-Befehls eine 1 ausgegeben.

```
Function modtest
....Integer var1, result

....Print "Enter an integer number:"
....Input var1
....result = var1 Mod 2
....Print "Result = ", result
....If result = 0 Then
........Print "The number is EVEN"
....Else
........Print "The number is ODD"
....EndIf
Fend
```

Weitere Beispiele des Mod-Befehls vom Befehlseingabefenster.

```
> Print  36 Mod 6
> 0

> Print  25 Mod 10
> 5
>
```

Motor-Anweisung

Schaltet den Motorenstrom für alle Achsen des aktuellen Roboters ein oder aus. > S

Syntax

Motor ON | OFF

Parameter

ON | OFF Das Schlüsselwort **ON** wird verwendet, um den Motorenstrom einzuschalten. Das Schlüsselwort **OFF** wird verwendet, um den Motorenstrom auszuschalten.

Beschreibung

Der Befehl **Motor** On wird verwendet, um den Motorenstrom einzuschalten und die Bremsen für alle Achsen freizuschalten. **Motor** Off wird verwendet, um den Motorenstrom auszuschalten und die Bremsen anzuziehen.

Um den Roboter bewegen zu können, muss der Motorenstrom eingeschaltet sein.

Führen Sie Reset nach einem Not-Aus aus, oder nachdem ein Fehler aufgetreten ist, der ein Reset mit dem Reset-Befehl notwendig macht, und führen Sie dann **Motor** On aus.

Motor On stellt die folgenden Werte automatisch ein:

Power	Low
Fine	Vorgabewerte
Speed	Vorgabewerte
SpeedR	Vorgabewerte
SpeedS	Vorgabewerte
Accel	Vorgabewerte
AccelS	Vorgabewerte
AccelR	Vorgabewerte
PTPBoost	Vorgabewerte
LimZ	0

Verwandte Befehle

Brake, Power, Reset, SFree, SLock

Beispiel einer Motor-Anweisung

Die folgenden Beispiele werden vom Befehlseingabefenster aus ausgeführt:

```
> Motor On

> Motor Off
```

Motor-Funktion

F

Gibt den Status des Motorenstroms für den aktuellen Roboter aus.

Syntax

Motor

Rückgabewerte

0 = Motoren ausgeschaltet, 1 = Motoren eingeschaltet.

Verwandte Befehle

Motor-Anweisung

Beispiel einer Motor-Funktion

```
If Motor = Off Then
    Motor On
EndIf
```

Move-Anweisung

> S

Bewegt den Arm aus der aktuellen Position mittels Linearinterpolation an den definierten Punkt (d. h. Bewegung in einer geraden Linie bei gleich bleibender Werkzeugmittelpunkt-Geschwindigkeit).

Syntax

Move *destination* [**ROT**] [**ECP**] [**CP**] [*searchExpr*] [*!...!*]

Parameter

destination	Der Zielort einer Bewegung, die einen Punktausdruck verwendet.
ROT	Optional. :Bestimmt Geschwindigkeit / Beschleunigung / Verzögerung zugunsten der Werkzeugrotation.
ECP	Optional. Externe Kontrollpunkt-Bewegung. Dieser Parameter ist gültig, wenn die ECP-Option aktiviert ist.
CP	Optional. Spezifiziert die CP-Bewegung.
searchExpr	Optional. Ein Till- oder Find-Ausdruck. **Till** | **Find** **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können hinzugefügt werden, um E/A-Befehle und andere Befehle während der Bewegung auszuführen.

Beschreibung

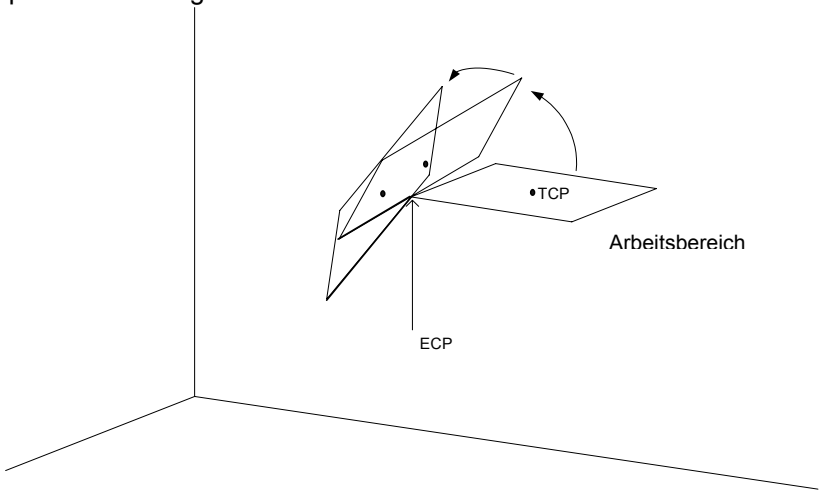
Move bewegt den Arm in einer geraden Linie aus der aktuellen Position zur *destination*. **Move** koordiniert alle Achsen so, dass sie zur selben Zeit starten und stoppen. Die *destination*-Koordinaten müssen vor der Ausführung des Move-Befehls geteacht worden sein. Beschleunigung und Verzögerung für die **Move**-Anweisung werden durch den AccelS-Befehl gesteuert. Die Geschwindigkeit für die Bewegung wird durch den SpeedS-Befehl gesteuert. Wenn der SpeedS-Geschwindigkeitswert die erlaubte Geschwindigkeit für eine beliebige Achse überschreitet, wird der Strom für alle vier Achsmotoren abgeschaltet und der Roboter hält an.

Move verwendet den Geschwindigkeitswert aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS. Lesen Sie den Abschnitt *Move mit CP verwenden* für Informationen über die Beziehung Geschwindigkeit-Beschleunigung und Beschleunigung-Verzögerung. Wenn jedoch der ROT-Parameter verwendet wird, verwendet **Move** den Geschwindigkeitswert aus SpeedR und die Beschleunigungs- und Verzögerungswerte aus AccelR. In diesem Fall haben die Geschwindigkeitswerte aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS keine Wirkung.

Wenn der Bewegungsabstand bei 0 liegt und nur die Werkzeugausrichtung geändert wird, tritt für gewöhnlich ein Fehler auf. Wenn jedoch der ROT-Parameter verwendet wird und die Beschleunigung und Verzögerung der Werkzeugrotation bevorrechtigt werden, ist eine fehlerfreie Bewegung möglich. Wenn keine Ausrichtungsänderung durch den ROT-Parameter vorliegt und die Bewegungsdistanz nicht bei 0 liegt, tritt ein Fehler auf.

Auch wenn die Werkzeugrotation im Vergleich zur Bewegungsdistanz groß ist und wenn die Rotationsgeschwindigkeit die spezifizierte Geschwindigkeit des Manipulators überschreitet, tritt ein Fehler auf. In diesem Fall reduzieren Sie die Geschwindigkeit oder hängen Sie den ROT-Parameter an, um die Rotationsgeschwindigkeit / -beschleunigung /-verzögerung zu bevorrechtigen.

Wenn ECP verwendet wird, bewegt sich der Trajektoriebereich des externen Kontrollpunktes entsprechend der ECP-Nummer, die durch die ECP-Anweisung festgelegt wurde, gerade in Bezug auf das Werkzeugkoordinatesystem. In diesem Fall folgt der Trajektoriebereich des Werkzeugmittelpunktes keiner geraden Linie.



Die optionale Till-Bedingung ermöglicht es dem Benutzer, eine Bedingung zu definieren, die den Roboter dazu veranlasst, bis zum vollständigen Stopp zu verzögern, bevor der **Move**-Befehl vollständig ausgeführt wurde. Die spezifizierte Bedingung ist einfach eine Gegenprüfung zu einem der Eingänge. Dies wird durch Verwenden des Sw-Befehls erreicht. Der Benutzer kann überprüfen, ob die Eingänge aus- oder eingeschaltet sind und den Arm dazu veranlassen, anzuhalten, je nachdem, welche Bedingung spezifiziert ist. Dieses Feature funktioniert ähnlich wie eine Unterbrechung (Interrupt), bei der **Move** unterbrochen (gestoppt) wird, sobald die Eingangs-Bedingung erfüllt ist. Wenn die Bedingung während des **Move**-Befehls nie erfüllt wird, erreicht der Arm erfolgreich den durch *destination* definierten Punkt. Für weitere Informationen bezüglich der Till-Bedingung lesen Sie bitte unter ‚Till-Befehl' weiter.

Hinweise

Was die Move-Anweisung nicht kann:

1) Move kann keine Bewegung ausführen, bei der nur die Werkzeugausrichtung geändert wird.

2) Move kann keine Bereichsüberprüfung der Bewegungsbahn durchführen, bevor die Bewegung selbst gestartet wird. Es ist dem System daher möglich, sogar für Zielpositionen, die sich innerhalb eines zulässigen Bereichs befinden, auf dem Weg zum Zielpunkt eine verbotene Position zu finden. In diesem Fall kann der Arm abrupt zum Stehen kommen, was einen Stoß und den Servo-Aus-Zustand des Arms hervorrufen kann. Um dem vorzubeugen, sollten Sie sicherstellen, dass bei niedriger Geschwindigkeit Bereichsüberprüfungen durchgeführt werden, bevor Sie **Move** bei hohen Geschwindigkeiten verwenden. Zusammenfassend heißt das, dass obschon sich die Zielposition innerhalb des Armbereichs befindet, es einige Bewegungen gibt, die nicht ausgeführt werden können. Das liegt daran, dass der Arm einige der Zwischenpositionen nicht erreichen kann, die während der **Move**-Bewegung benötigt werden.

Move mit CP verwenden

Der CP-Parameter veranlasst den Arm, zur *destination* zu fahren, ohne langsamer zu werden oder an dem durch *destination* definierten Punkt anzuhalten. Dadurch wird es dem Benutzer ermöglicht, eine Serie von Bewegungsbefehlen miteinander zu verketten, was den Arm veranlasst, sich entlang eines kontinuierlichen Weges zu bewegen und während dieser Bewegung eine bestimmte Bewegung einzuhalten. Der **Move**-Befehl ohne CP veranlasst den Arm immer dazu, bis zum vollständigen Halt herunterzubremsen, bevor er das Punktziel *destination* erreicht hat.

Korrekte Geschwindigkeits- und Beschleunigungs-Befehle mit Move:

Die Befehle SpeedS und AccelS werden verwendet, um Geschwindigkeit und Beschleunigung des Roboters während der **Move-**Bewegung zu definieren. Beachten Sie, dass SpeedS und AccelS sich auf linearinterpolierte und kreisinterpolierte Bewegungen beziehen, während sich die Befehle Speed und Accel aus PTP-Bewegungen beziehen..

Mögliche Fehler

Versuch, nur die Werkzeugausrichtung zu ändern

Während einer Bewegung ist es nicht möglich, nur die Werkzeugausrichtung zu ändern. Wird dies versucht, tritt ein Fehler auf.

Überdrehzahl-Fehler der Achse

Wenn die Geschwindigkeit einer Achse durch eine angeforderte Bewegung überschritten wird, tritt ein Überdrehzahl-Fehler auf. Im Falle eines Überdrehzahl-Fehlers der Motoren wird der Roboterarm angehalten und die Servoachse wird abgeschaltet.

Es folgt kein Bewegungsbefehl auf den Move CP-Befehl

Wenn auf den **Move** CP-Befehl keine weiteren Bewegungsbefehle folgen, die den Arm reibungslos bewegen (da der Arm noch nicht verzögert hat), kann der Arm durch plötzliche Stoßbewegungen beschädigt werden. In diesem Fall tritt ein Fehler auf, der Arm hält an und die Servoachse wird freigeschaltet. (Das Eingeben einer Pause während einer CP-Bewegung kann ebenfalls einen Fehler hervorrufen, da der Befehl ebenfalls versucht, den Roboter zum sofortigen Stillstand zu bringen.)

Verwandte Befehle

AccelS, Arc, Go, Jump, Jump3, Jump3CP, SpeedS, Sw, Till

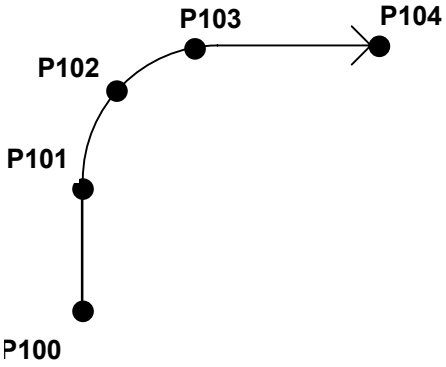
Beispiel einer Move-Anweisung

Das folgende Beispiel zeigt eine einfache PTP-Bewegung vom Punkt P0 zum Punkt P1 mit einer geradlinigen Rückbewegung zum Punkt P0. Später bewegt sich der Arm geradlinig zum Punkt P2, bis der 2. Eingang eingeschaltet wird. Wenn der 2. Eingang während der Bewegung einschaltet, verzögert sich der Arm bis zum vollständigen Stopp, bevor er den Punkt P2 erreicht und der nächste Programmbefehl ausgeführt wird.

```
Function movetest
  Home
  Go  P0
  Go P1
  Move P2 Till Sw(2) = On
  If Sw(2) = On Then
    Print "Input #2 came on during the move and"
    Print "the robot stopped prior to arriving on"
    Print "point P2."
  Else
    Print "The move to P2 completed successfully."
    Print "Input #2 never came on during the move."
  EndIf
Fend
```

In diesem Beispiel wird **Move** mit CP verwendet. Das Diagramm unten zeigt eine Bogenbewegung, die am Punkt P100 ihren Anfang nimmt, in einer geraden Linie durch Punkt P101 läuft. Hier beginnt der Arm, einen Bogen zu schlagen. Der Bogen wird dann fortgesetzt durch den Punkt P102 und weiter bis P103. Danach bewegt sich der Arm in einer geraden Linie zum Punkt P104, wo er schließlich bis zum vollständigen Stopp verzögert. Bitte beachten Sie, dass der Arm zwischen den einzelnen Punkten nicht verzögert, bis er den Punkt P104 erreicht.

Die folgende Funktion würde eine solche Bewegung generieren.




```
Function CornerArc
   Go P100
   Move CP P101        'Stoppt nicht bei P101
   Arc CP P102, P103   'Stoppt nicht bei P103
   Move P104           'Verzögert, um bei P104 zu stoppen
Fend
```

MyTask-Funktion

F

Gibt die Tasknummer des aktuellen Programms aus.

Syntax

MyTask

Rückgabewerte

Die Tasknummer des aktuellen Tasks. Gültige Einträge sind die Integer 1-16.

Beschreibung

MyTask gibt die Tasknummer des aktuellen Programms mit einem Zahlzeichen aus. Der **MyTask-**Befehl wird in einem bestimmten Programm eingefügt. Wenn dieses Programm die **MyTask-**Funktion ausführt, wird die Tasknummer ausgegeben, in der das Programm läuft.

Verwandte Befehle

Xqt

Beispiel einer MyTask-Funktion

Das folgende Programm schaltet die E/A-Ports von 1 bis 8 ein oder aus.

```
Function main
  Xqt 2, task    'Führt Task 2 aus.
  Xqt 3, task    'Führt Task 3 aus.
  Xqt 4, task    'Führt Task 4 aus.
  Xqt 5, task    'Führt Task 5 aus.
  Xqt 6, task    'Führt Task 6 aus.
  Xqt 7, task    'Führt Task 7 aus.
  Xqt 8, task    'Führt Task 8 aus.
  Call task
Fend

Function task
  Do
    On MyTask           'Schaltet den E/A-Port ein, dessen
                        'Nummer der aktuellen Tasknummer entspricht
    Off MyTask          'Schaltet den E/A-Port aus, dessen
                        'Nummer der aktuellen Tasknummer entspricht
  Loop
Fend
```

Next

> S

Die Befehle For und Next werden zusammen verwendet, um eine Schleife zu erzeugen, in der Befehle, die sich zwischen den Befehlen For und Next befinden, mehrfach ausgeführt werden, wie vom Benutzer angegeben.

Syntax

For *var1* **=** *initval* **To** *finalval* **[Step** *Increment* **]**
statements
Next *var1*

Parameter

var1	Die Zählvariable, die mit der For / Next-Schleife verwendet wird. Diese Variable wird normalerweise als Integer definiert, kann jedoch auch als Realvariable definiert werden.
initval	Der Anfangswert für den Zähler *var1*.
finalval	Der Endwert des Zählers var1. Sobald dieser Wert erreicht ist, ist die For / Next-Schleife vollendet und die Ausführung wird mit der Anweisung, die auf den Next-Befehl folgt, fortgesetzt.
Increment	Ein optionaler Parameter, der das Zähl-Inkrement für jede Ausführung der Next-Anweisung innerhalb der For / Next-Schleife definiert wird. Diese Variable kann positiv oder negativ sein. Wenn der Wert negativ ist, muss der Erstwert der Variable größer sein als ihr Endwert. Wird der Inkrement-Wert weggelassen, inkrementiert (erhöht) das System automatisch um 1.
statements	Jede gültige SPEL+-Anweisung kann in die For / Next-Schleife eingefügt werden.

Rückgabewerte

Keine

Beschreibung

For / Next führt einen Satz von Anweisungen innerhalb einer Schleife eine definierte Anzahl von Malen aus. Die *For*-Anweisung stellt den Anfang der Schleife dar. Die **Next**-Anweisung ist das Ende der Schleife. Wie oft die Anweisungen innerhalb der Schleife ausgeführt werden, wird mithilfe einer Variable gezählt.

Der erste numerische Ausdruck (*initval)* ist der Erstwert des Zählers. Dieser Wert kann positiv oder negativ sein, solange die Variable *finalval* und die Step-Inkrementierung einander korrekt entsprechen.

Der zweite numerische Ausdruck (*finalval)* ist der Endwert des Zählers. Dies ist der Wert der, sobald er erreicht ist, die Beendigung der For / Next-Schleife auslöst. Die Steuerung des Programms wird an den nächsten auf den **Next**-Befehl folgenden Befehl weitergegeben.

Programmanweisungen, die der *For*-Anweisung folgen, werden ausgeführt, bis ein **Next**-Befehl erreicht wird. Die Zählervariable (*var1*) wird dann durch den Step-Wert inkrementiert, der durch den Parameter *increment* definiert ist. Wird die Step-Option nicht genutzt, wird der Zähler um 1 inkrementiert.

Die Zählervariable (*var1*) wird dann mit dem Endwert (*finalval*) verglichen. Wenn der Zählerstand kleiner oder gleich dem Endwert (*finalval*) ist, werden die Anweisungen, die dem **For-**Befehl folgen, erneut ausgeführt. Wenn die Zählervariable größer als der Endwert (*finalval*) ist, wird die Ausführung außerhalb der For/**Next-**Schleife verzweigt und fährt mit dem Befehl fort, der direkt auf den **Next**-Befehl folgt.

Eine Verschachtelung von For/**Next**-Anweisungen wird für bis zu 10 Ebenen unterstützt. Dies bedeutet, dass eine For / **Next**-Schleife in einer weiteren For / **Next**-Schleife verschachtelt werden kann usw., bis

10 „Verschachtelungen" von For / **Next**-Schleifen vorhanden sind.

Hinweise

Negative Step-Werte:

Wenn der Wert der Step-Inkrementierung (*increment*) negativ ist, wird die Zählervariable (*var1*) bei jedem Durchlauf durch die Schleife dekrementiert (verringert) und der Erstwert (*initval*) muss größer sein als der Endwert, damit die Schleife funktioniert.

Verwandte Befehle

For

Beispiel für For / Next

```
Function fornext
    Integer ctr
    For ctr = 1 to 10
        Go Pctr
    Next ctr
    '
    For ctr = 10 to 1 Step -1
        Go Pctr
    Next ctr
Fend
```

Not-Operator

F

Führt die bitweise Umkehrung des Operandenwertes aus.

Syntax

Not *operand*

Parameter

operand	Integer-Ausdruck.

Rückgabewerte

Umkehrung des Operandenwertes.

Beschreibung

Die **Not**-Funktion führt die bitweise Umkehrung des Operandenwertes aus. Jedes resultierende Bit ist die Umkehrung des entsprechenden Bits im Operanden; 0-Bits werden in 1 und 1er-Bits in 0 umgewandelt.

Verwandte Befehle

Abs, And, Atan, Atan2, Cos, Int, LShift, Mod, Or, RShift, Sgn, Sin, Sqr, Str$, Tan, Val, Xor

Beispiel eines Not-Operators

Dies ist ein einfaches Beispiel aus dem Befehlseingabefenster zur Verwendung des **Not**-Befehls.

```
>print not(1)
 -2
>
```

Off-Anweisung

> S

Schaltet den angegebenen Ausgang aus, der nach einer definierten Zeit wieder eingeschaltet werden kann.

Syntax

Off { *bitNumber* | *outputLabel* }, [*time],* [*parallel*]

Parameter

bitNumber	Integer-Ausdruck, der angibt, welcher Ausgang ausgeschaltet werden soll.
outputLabel	Ausgangslabel.
time	Optional. Gibt ein Zeitintervall in Sekunden an, für das der Ausgang ausgeschaltet bleibt. Nachdem das Zeitintervall abgelaufen ist, wird der Ausgang wieder eingeschaltet. Das Zeitintervall umfasst mindestens 0,01 Sekunden und höchstens 10 Sekunden.
parallel	Optional. Wenn ein Timer eingestellt ist, kann der *parallel*-Parameter verwendet werden, um festzulegen, wann der nächste Befehl ausgeführt wird: 0 - sofort, nachdem der Ausgang ausgeschaltet wurde 1 - nach Ablauf des definierten Zeitintervalls (Vorgabewert).

Beschreibung

Off schaltet den angegebenen Ausgang aus (setzt ihn auf 0).

Wenn der *time*-Intervallparameter definiert ist, wird der durch *bitNumber* spezifizierte Ausgang ausgeschaltet und nach Ablauf des Zeitintervalls wieder eingeschaltet. Wenn der Ausgang vor Ausführung von Off bereits ausgeschaltet war, wird er nach Ablauf des Zeitintervalls wieder eingeschaltet.

Die *parallel*-Parametereinstellungen sind anwendbar, wenn das Zeitintervall wie folgt definiert ist:
1: Schaltet den Ausgang aus, nach Ablauf eines definierten Zeitintervalles wieder ein und führt dann den nächsten Befehl aus. (Dies ist auch der Vorgabewert für den *parallel*-Parameter. Wird dieser Parameter weggelassen, ist dies das gleiche, als würde der Parameter auf 1 gesetzt.)
0: Schaltet den Ausgang aus und führt gleichzeitig den nächsten Befehl aus.

Hinweise

Ausgänge, die als Remote-Steuerausgänge konfiguriert sind

Wenn ein Ausgang angegeben wird, der als Systemausgang eingestellt war, so tritt ein Fehler auf. Remote-Steuerausgänge werden je nach Systemstatus automatisch ein- oder ausgeschaltet.

Verhalten der Ausgänge im Falle von Not-Aus:

EPSON RC+ verfügt über ein Feature, das bei Eintreten eines Not-Aus alle Ausgänge ausschaltet. Dieses Feature wird über Einstellungen | Steuerung | Voreinstellungen aktiviert oder deaktiviert.

Verwandte Befehle

In, InBCD, MemOn, MemOff, MemOut, MemSw, OpBCD, Oport, Out, Wait

Beispiel einer Off-Anweisung

Das Beispiel unten zeigt, wie der Haupttask einen Hintergrundtask namens *iotask* startet. *iotask* ist ein einfacher Task, um die getrennten Ausgänge 1 und 2 ein- und wieder auszuschalten, 10 Sekunden zu warten und den Vorgang dann zu wiederholen.

```
Function main
  Xqt 2, iotask
  Go P1
  .
  .
  .
Fend


Function iotask
  Do
    On 1
    On 2
    Off 1
    Off 2
    Wait 10
  Loop
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> on  1
> off 1, 10
'Schaltet Ausgang 1 aus, wartet 10 Sekunden und schaltet ihn wieder ein.
> on  2
> off 2
```

OLRate-Anweisung

Anzeige des Überlastgrades für eine oder alle Achsen des aktuellen Roboters. > S

Syntax

OLRate [*jointNumber*]

Parameter

jointNumber Integer-Ausdruck, dessen Wert zwischen 1 und 6 liegt.

Beschreibung

OLRate kann verwendet werden, um festzustellen, ob ein bestimmter Zyklus das Servosystem überbeansprucht. Faktoren wie Temperatur und Strom können Servofehler während der Anwendung in Hochleistungszyklen verursachen. **OLRate** kann dabei helfen zu überprüfen, ob das Robotersystem kurz vor einem Servofehler steht.

Während eines Zyklus sollten Sie einen weiteren Task ausführen, um **OLRate** zu befehlen. Wenn die **OLRate** bei einer Achse 1,0 überschreitet, tritt ein Servofehler auf.

Servofehler treten am ehesten bei hohen Nutzlasten auf. Sie können dazu beitragen, dass die Geschwindigkeits- und Beschleunigungseinstellungen keinen Servofehler in einem Produktionszyklus hervorrufen, indem Sie **OLRate** in einem Testzyklus verwenden.

Um gültige Messwerte zu erhalten, müssen Sie **OLRate** ausführen, während sich der Roboter bewegt.

Verwandte Befehle

OLRate-Funktion

Beispiel einer OLRate-Anweisung

```
>olrate
0.10000  0.20000
0.30000  0.40000
0.50000  0.60000

Function main
  Power High
  Speed 50
  Accel 50,  50
  Xqt 2, MonitorOLRate
  Do
    Jump P0
    Jump P1
  Loop
Fend

Function MonitorOLRate
  Do
    ' Zeigt OLRate an
    OLRate
    Wait 1
  Loop
Fend
```

OLRate-Funktion

F

Gibt den Überlastgrad für eine Achse des aktuellen Roboters aus.

Syntax

OLRate(*jointNumber***)**

Parameter

jointNumber Integer-Ausdruck, dessen Wert zwischen 1 und 6 liegt.

Rückgabewerte

Gibt die OLRate für die angegebene Achse aus. Die Werte variieren zwischen 0,0 und 2,0.

Beschreibung

OLRate kann verwendet werden, um festzustellen, ob ein bestimmter Zyklus das Servosystem überbeansprucht. Faktoren wie Temperatur und Strom können Servofehler während der Anwendung in Hochleistungszyklen verursachen. **OLRate** kann dabei helfen, zu überprüfen, ob das Robotersystem kurz vor einem Servofehler steht.

Während eines Zyklus sollten Sie einen weiteren Task ausführen, um **OLRate** zu befehlen. Wenn die **OLRate** bei einer Achse 1,0 überschreitet, tritt ein Servofehler auf.

Servofehler treten am ehesten bei hohen Nutzlasten auf. Sie können dazu beitragen, dass die Geschwindigkeits- und Beschleunigungseinstellungen keinen Servofehler in einem Produktionszyklus hervorrufen, indem Sie **OLRate** in einem Testzyklus verwenden.

Um gültige Messwerte zu erhalten, müssen Sie **OLRate** ausführen, während sich der Roboter bewegt.

Verwandte Befehle

OLRate-Anweisung

Beispiel einer OLRate-Funktion

```
Function main
  Power High
  Speed 50
  Accel 50, 50
  Xqt 2, MonitorOLRate
  Do
    Jump P0
    Jump P1
  Loop
Fend

Function MonitorOLRate
  Integer i
  Real olRates(4)
  Do
    For i = 1 to 4
      olRates(i) = OLRate(i)
      If olRate(i) > .5 Then
        Print "Warning: OLRate(", i, ") is over .5"
      EndIf
    Next i
  Loop
Fend
```

On-Anweisung

> S

Schaltet den angegebenen Ausgang ein, der nach einer definierten Zeit wieder ausgeschaltet werden kann.

Syntax

On { *bitNumber* | *outputLabel* }, [*time],* [*parallel*]

Parameter

bitNumber	Integer-Ausdruck, der angibt, welcher Ausgang eingeschaltet werden soll.
outputLabel	Ausgangslabel.
time	Optional. Gibt ein Zeitintervall in Sekunden an, für das der Ausgang eingeschaltet bleiben soll. Nachdem das Zeitintervall abgelaufen ist, wird der Ausgang wieder ausgeschaltet. (Das minimale Zeitintervall beträgt 0,01 Sekunden.)
parallel	Optional. Wenn ein Timer eingestellt ist, kann der *parallel*-Parameter dazu verwendet werden, festzulegen, wann der nächste Befehl ausgeführt wird: 0 - sofort nachdem der Ausgang eingeschaltet wurde 1 - nach Ablauf des definierten Zeitintervalls (Vorgabewert)

Beschreibung

On schaltet den angegebenen Ausgang ein (setzt ihn auf 1).
Wenn der Zeitintervall-Parameter definiert ist, wird der durch *outnum* spezifizierte Ausgang eingeschaltet und nach Ablauf des Zeitintervalls wieder ausgeschaltet.

Die *parallel*-Parametereinstellungen sind anwendbar, wenn das Zeitintervall wie folgt definiert ist:
1: Schaltet den Ausgang ein, nach Ablauf eines definierten Zeitintervalles wieder aus und führt dann den nächsten Befehl aus. (Dies ist auch der Vorgabewert für den *parallel*-Parameter. Wird dieser Parameter weggelassen, ist dies das gleiche, als würde der Parameter auf 1 gesetzt.)
0: Schaltet den Ausgang ein und führt gleichzeitig den nächsten Befehl aus.

Hinweise

Ausgänge, die als Remote konfiguriert sind

Wird ein Ausgang angegeben, der als Remote eingestellt war, tritt ein Fehler auf. Remote-Ausgänge werden je nach Systemstatus automatisch ein- oder ausgeschaltet. Für weitere Remote-Informationen lesen Sie bitte das EPSON RC+-Benutzerhandbuch. Die einzelnen Bits für den Remote-Anschluss können vom EPSON RC+ Remote-Konfigurationsdialog als Remote oder als E/A eingestellt werden. In diesen Dialog gelangen Sie über das Einstellungen-Menü.

Verhalten der Ausgänge im Falle von Not-Aus

Die RC170 Steuerung verfügt über ein Feature, das bei Eintreten eines Not-Aus alle Ausgänge ausschaltet. Dieses Feature wird über einen der Option-Schalter aktiviert oder deaktiviert. Zur Konfiguration gelangen Sie über Einstellungen | Steuerung | Voreinstellungen.

Verwandte Befehle

In, InBCD, MemOff, MemOn, Off, OpBCD, Oport, Out, Wait

Beispiel einer On-Anweisung

Das Beispiel unten zeigt, wie der Haupttask einen Hintergrundtask namens *iotask* startet. *iotask* ist ein einfacher Task, um die getrennten Ausgänge 1 und 2 ein- und danach wieder auszuschalten, 10 Sekunden zu warten und den Vorgang dann zu wiederholen.

```
Function main
    Xqt iotask
    Go P1
    .
    .
    .
Fend


Function iotask
    '
    Do
        On 1
        On 2
        Off 1
        Off 2
        Wait 10
    Loop
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> On 1
> Off 1
> On 2
> Off 2
```

OnErr

Legt Interrupt-Verzweigungen an, die die Steuerung veranlassen, zu einem Fehlerbehandlungs-Unterprogramm zu wechseln, wenn ein Fehler auftritt. Ermöglicht dem Benutzer die Fehlerbehandlung.

S

Syntax

OnErr GoTo {*label* | **0**}

Parameter

label	Anweisungslabel zu dem gewechselt werden soll, wenn ein Fehler auftritt.
0	Der Parameter, der verwendet wird, um die OnErr-Einstellung zu löschen.

Beschreibung

OnErr ermöglicht dem Benutzer die Fehlerbehandlung. Wenn ein Fehler auftritt, ohne dass **OnErr** verwendet wird, wird der Task abgebrochen und der Fehler wird angezeigt. Wird **OnErr** jedoch verwendet, kann der Benutzer den Fehler „abfangen“ und zu einer Fehlerbehandlungsroutine gehen, um den Fehler automatisch zu beheben. Bei Empfang eines Fehlers verzweigt **OnErr** die Steuerung zur im **EResume**-Befehl festgelegten Zeilennummer oder dem entsprechenden Label. Der Task wird also nicht abgebrochen und der Benutzer hat die Möglichkeitden Fehler automatisch zu beheben. Dadurch istder Arbeitsablauf in Roboter-Arbeitszellen wesentlich reibungsloser, da mögliche Probleme immer auf die gleiche Weise behandelt und behoben werden.

Wenn der **OnErr**-Befehl mit dem 0-Parameter spezifiziert wird, wird die aktuelle **OnErr**-Einstellung gelöscht. (D. h. nach der Ausführung von **OnErr** 0 wird die Programmausführung unterbrochen, wenn ein Fehler auftritt).

Verwandte Befehle

Err, EResume

Beispiel für OnErr

Das folgende Beispiel zeigt ein einfaches Dienstprogramm, das prüft, ob die Punkte P0-P399 existieren. Wenn ein Punkt nicht existiert, erscheint eine Meldung für den Benutzer auf dem Bildschirm. Das Programm verwendet den CX-Befehl, um bei jedem einzelnen Punkt zu testen, ob er definiert wurde oder nicht. Wenn ein Punkt nicht definiert ist, wird die Steuerung an die Fehlerbehandlungsroutine übergeben und auf dem Bildschirm erscheint eine Meldung, die dem Benutzer mitteilt, welcher Punkt nicht definiert war.

```
Function errDemo
    Integer i, errNum

    OnErr GoTo errHandler

    For i = 0 To 399
        temp = CX(P(i))
    Next i
    Exit Function
    '
    '
    '*****************************************
    '* Error Handler                         *
    '*****************************************
errHandler:
    errNum = Err
    'Überprüft, ob ein nicht definierter Punkt verwendet wird
    If errNum = 7007 Then
       Print "Point number P", i, " is undefined!"
    Else
       Print "ERROR: Error number ", errNum, " occurred while"
       Print "       trying to process point P", i, " !"
    EndIf
    EResume Next
Fend
```

OpBCD-Anweisung

> S

Setzt 8 Ausgänge gleichzeitig unter Verwendung des BCD-Formats.
(Binär codierte Dezimazahl)

Syntax

OpBCD portNumber, outData

Parameter

portNumber Integer-Ausdruck, der für die E/A-Ausgangsbytes steht. Die *portNumber*-Auswahl entspricht den folgenden Ausgängen:

Portnummer	Ausgänge
0	0-7
1	8-15
2	16-23
3	24-31
...	...

outData Integer-Ausdruck zwischen 0 und 99, der für das Ausgangsmuster für die mit der Anweisung *portNumber* ausgewählte Ausgangsgruppe steht. Die 2. Ziffer (Einer-Ziffer genannt) wird durch die unteren 4 Ausgänge in der ausgewählten Gruppe und die 1. Ziffer (Zehner-Ziffer genannt) durch die oberen 4 Ausgänge in der ausgewählten Gruppe angegeben.

Beschreibung

OpBCD setzt 8 Ausgänge gleichzeitig unter Verwendung des BCD-Formats. Die Standard- und Erweiterungsausgänge werden in Gruppen von 8 unterteilt. Der *portNumber*-Parameter für den OpBCD-Befehl definiert, welche 8er-Ausgangsgruppe verwendet werden soll, wobei *portNumber* = 0 für die Ausgänge 0-7 steht, *portNumber* = 1 für die Ausgänge 8-15, etc.

Sobald eine Portnummer ausgewählt wurde (d. h. es wurde eine Gruppe von acht Ausgängen ausgewählt), muss ein spezifisches Ausgangsmuster definiert werden. Dies wird im BCD-Format unter Zuhilfenahme des *outdata*-Parameters getan. Der *outdata*-Parameter kann aus ein oder zwei Ziffern bestehen. (Gültige Einträge liegen im Bereich von 0 bis 99.) Die erste Stelle (Zehnerstelle) entspricht den oberen 4 Ausgängen der Gruppe von 8 Ausgängen, ausgewählt von *portNumber*. Die zweite Stelle (Einerstelle) entspricht den unteren 4 Ausgängen in der Gruppe von 8 Ausgängen, ausgewählt von *portNumber*.

Da gültige Ausgänge im BCD-Format im Bereich von 0 bis 9 für jede Stelle liegen, kann nicht jede E/A-Kombination ausgeben werden. Die Tabelle unten zeigt mögliche E/A-Kombinationen und ihre entsprechenden *outnum*-Werte, davon ausgehend, dass die *portNumber* 0 ist.

Einstellungen des Ausgangs (Ausgangsnummer)

Outnum-Wert	7	6	5	4	3	2	1	0
01	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
02	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Aus
03	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
08	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Aus
09	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Ein
10	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Aus
11	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Ein
99	Ein	Aus	Aus	Ein	Ein	Aus	Aus	Ein

Beachten Sie, dass das BCD-Format nur die Spezifizierung von Dezimalwerten zulässt. Dies bedeutet, dass es durch die Verwendung des BCD-Formats nicht möglich ist, alle Ausgänge mit dem

OpBCD-Befehl einzuschalten. Beachten Sie, dass der Maximalwert für jede *outnum*-Ziffer 9 ist. Stattdessen werden die Ausgänge 0, 3, 4 und 7 ein- und alle anderen ausgeschaltet.

Hinweise

Der Unterschied zwischen OpBCD und Out

Die Befehle **OpBCD** und Out in der SPEL+- Sprache sind einander sehr ähnlich. Es gibt jedoch einen wesentlichen Unterschied. Dieser Unterschied wird im Folgenden aufgezeigt:

- Der **OpBCD**-Befehl verwendet das BCD-Format zur Spezifizierung eines 8-Bit-Wertes, der verwendet wird, um die Ausgänge ein- oder auszuschalten. Da das BCD-Format den Gebrauch der Werte &HA, &HB, &HC, &HD, &HE oder &HF ausschließt, können nicht alle Kombinationen für die Einstellung der 8 Ausgänge erfüllt werden.
- Der Out-Befehl funktioniert ähnlich wie der OpBCD-Befehl, nur, dass der Out-Befehl es gestattet, den Bereich der 8-Bit-Werte, der für das Ein- oder Ausschalten der Ausgänge verwendet werden darf, zwischen 0 und 255 anzusiedeln (gegenüber 0 bis 99 für OpBCD). Somit können alle möglichen Kombinationen für die 8 Ausgangsgruppen den Benutzerspezifikationen folgend initialisiert werden.

Ausgänge, die als Remote konfiguriert sind:

Wird ein als Remote eingestellter Ausgang so spezifiziert, dass er durch **OpBCD** eingeschaltet wird, tritt ein Fehler auf. Remote-Ausgänge werden je nach Systemstatus automatisch ein- oder ausgeschaltet. Für weitere Remote-Informationen lesen Sie bitte das EPSON RC+-Benutzerhandbuch. Die einzelnen Bits für den Remote-Anschluss können vom EPSON RC+ Remote-Konfigurationsdialog als Remote oder als E/A eingestellt werden. In diesen Dialog gelangen Sie über das Einstellungen-Menü.

Verhalten der Ausgänge im Falle von Not-Aus:

Die RC170 Steuerung verfügt über ein Feature, das bei Eintreten eines Not-Aus alle Ausgänge ausschaltet. Dieses Feature wird über einen der Option-Schalter aktiviert oder deaktiviert. Zur Konfiguration gelangen Sie über Einstellungen | Steuerung | Voreinstellungen.

Verwandte Befehle

In, InBCD, MemOff, MemOn, MemSw, Off, On, Oport, Out, Sw, Wait

Beispiel einer OpBCD-Funktion

Das Beispiel unten zeigt, wie der Haupttask einen Hintergrundtask namens *iotask* startet. *iotask* ist ein einfacher Task, um abwechselnd die Ausgänge 1 und 2 und die Ausgänge 0 und 3 einzuschalten. Wenn 1 und 2 eingeschaltet sind, sind 0 und 3 ausgeschaltet und umgekehrt.

```
Function main
    Xqt 2, iotask
    Go P1
    .
    .
Fend

Function iotask
    Do
        OpBCD 0, 6
        OpBCD 0, 9
        Wait 10
    Loop
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> OpBCD 1,6  'Schaltet die Ausgänge 1 und 2 ein
> OpBCD 2,1  'Schaltet den Ausgang 8 ein
> OpBCD 3, 91 'Schaltet die Ausgänge 24, 28 & 31 ein
```

OpenCom-Anweisung

S

Öffnet einen RS-232-Kommunikationsport.

Syntax

OpenCom #*portNumber*

Parameter

portNumber Integer-Ausdruck für den zu öffnenden Port.

Beschreibung

Der spezifizierte RS-232-Port muss an der Steueurng installiert sein.

Verwandte Befehle

ChkCom, CloseCom, SetCom

Beispiel einer OpenCom-Anweisung

```
OpenCom #1
```

OpenNet-Anweisung

Öffnen eines TCP / IP-Netzwerkports. **S**

Syntax

OpenNet #*portNumber* **As** { **Client** | **Server** }

Parameter

portNumber Integer-Ausdruck für den zu öffnenden Port. Der Bereich liegt zwischen 201 und 208.

Beschreibung

OpenNet öffnet einen TCP / IP-Port für die Kommunikation mit einem anderen Rechner im Netzwerk.

Ein System sollte als Server öffnen und das andere als Client. Die Reihenfolge ist dabei nicht von Bedeutung.

Verwandte Befehle

ChkNet, CloseNet, SetNet

Beispiel einer OpenNet-Anweisung

In diesem Beispiel lautet die Konfiguration der TCP / IP-Einstellungen in zwei Steuerungen wie folgt:

Steuerung Nr.1:
Port: #201
Host-Name: 192.168.0.2
TCP/IP-Port: 1000

```
Function tcpip
  OpenNet #201 As Server
  WaitNet #201
  Print #201, "Data from host 1"
Fend
```

Steuerung Nr.2:
Port: #201
Host-Name: 192.168.0.1
TCP/IP-Port: 1000

```
Function tcpip
  String data$
  OpenNet #201 As Client
  WaitNet #201
  Input #201, data$
  Print "received '", data$, "' from host 1"
Fend
```

Oport-Funktion

F

Gibt den Status des spezifizierten Ausgangs aus.

Syntax

Oport(*outnum***)**

Parameter

outnum Integer-Ausdruck, der für die E/A-Ausgänge steht.

Rückgabewerte

Gibt den angegebenen Ausgangsstatus entweder als 0 oder als 1 aus.

0: Off-Status (Aus)
1: On-Status (Ein)

Beschreibung

Oport ermöglicht eine Status-Überprüfung der Ausgänge. Er funktioniert weitgehend so, wie der Sw-Befehl für die Ausgänge funktioniert. **Oport** wird im Allgemeinen verwendet, um den Status eines Ausgangs zu überprüfen, der an eine Aufgabevorrichtung, ein Förderband, einen Greifermagneten oder an den Host eines anderen Gerätes angeschlossen werden könnte, das über getrennte E/As arbeitet. Natürlich hat der Ausgang, der mit dem **Oport**-Befehl überprüft wird, 2 Status (1 oder 0). Diese zeigen an, ob der spezifizierte Ausgang ein- oder ausgeschaltet ist.

Hinweise

Der Unterschied zwischen Oport und Sw

Es ist wichtig, den Unterschied zwischen den Befehlen **Oprt** und Sw zu verstehen. Mit beiden Befehlen wird der E/A-Status ermittelt. Der E/A-Typ ist jedoch unterschiedlich. Der Sw-Befehl arbeitet mit Eingängen. Der **Oport**-Befehl arbeitet mit den Standard- und mit Erweiterungs-Hardware-Ausgängen. Diese Hardwareports sind einzelne Ausgänge, die mit Geräten außerhalb der Steuerung kommunizieren.

Verwandte Befehle

In, InBCD, MemIn, MemOn, MemOff, MemOut, MemSw, Off, On, OpBCD, Out, Sw, Wait

Beispiel einer OPort-Funktion
Im Beispiel unten wird der Ausgang 5 eingeschaltet und dann sichergestellt, dass dieser eingeschaltet ist, bevor fortgefahren wird.

```
Function main
    TMOut 10
    OnErr errchk
    Integer errnum
    On 5    'Schaltet Ausgang 5 ein
    Wait Oport(5)
    Call mkpart1
    Exit Function

errchk:
    errnum = Err(0)
    If errnum = 94 Then
       Print "TIME Out Error Occurred during period"
       Print "waiting for Oport to come on. Check"
       Print "Output #5 for proper operation. Then"
       Print "restart this program."
    Else
       Print "ERROR number ", errnum, "Occurred"
       Print "Program stopped due to errors!"
    EndIf
    Exit Function
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus:

```
> On 1
> Print Oport(1)
1
> Off 1
> Print Oport(1)
0
>
```

Or-Operator

Führt einen bitweisen oder logischen ODER-Vorgang an zwei Operanden aus.

Syntax

expr1 **oder** *expr2*

Parameter

expr1, expr2 Integer- oder Boolean-Ausdruck.

Rückgabewerte

Bitweiser Or-Wert der Operanden, wenn es sich um Integer-Ausdrücke handelt. Logische Or-Verknüpfung, wenn es sich um Boolean-Ausdrücke handelt.

Beschreibung

Bei Integer-Ausdrücken führt der **Or**-Operator den bitweisen Or-Vorgang an den Werten der Operanden aus. Jedes Bit des Ergebnisses ist 1, wenn eins oder beide entsprechenden Bits der beiden Operanden 1 sind. Bei Boolean-Ausdrücken ist das Ergebnis Wahr, wenn einer der Ausdrücke als Wahr bewertet wird.

Verwandte Befehle

And, LShift, Mod, Not, RShift, Xor

Beispiel eines Or-Operators

Das folgende Beispiel zeigt eine bitweise Or-Verknüpfung.

```
>print 1 or 2
 3
```

Das folgende Beispiel zeigt eine logische Or-Verknüpfung.

```
If a = 1 Or b = 2 Then
  c = 3
EndIf
```

Out-Anweisung

Setzt 8 Ausgänge gleichzeitig.

> S

Syntax

Out *portNumber, outData*

Parameter

portNumber Integer-Ausdruck, der für die E/A-Ausgangsbytes steht. Die *portnum*-Auswahl entspricht den folgenden Ausgängen:

Portnummer	Ausgänge
0	0-7
1	8-15
...	...

outData Integer zwischen 0 und 255, der für das Ausgangsmuster für die mit der Anweisung *portNumber* ausgewählte Ausgangsgruppe steht. Wenn es in hexadezimaler Form dargestellt wird, wird der Bereich von &H0 bis &HFF abgedeckt. Die niedrigere Ziffer steht für die niedrigeren Stellen (oder die ersten 4 Ausgänge) und die obere Ziffer für die höheren Stellen (oder die zweiten 4 Ausgänge).

Beschreibung

Out setzt unter Nutzung der vom Anwender angegebenen Kombination aus *portNumber*- und *outData*-Werten 8 Ausgänge gleichzeitig. Der *portNumber*-Parameter definiert, welche 8er-Ausgangsgruppe verwendet werden soll, wobei *portNumber* = 0 für die Ausgänge 0-7 steht, *portNumber* = 1 für die Ausgänge 8-15, etc.

Sobald eine Portnummer ausgewählt wurde (d. h. es wurde eine Gruppe von 8 Ausgängen ausgewählt), muss ein spezifisches Ausgangsmuster definiert werden. Dafür wird der *outData*-Parameter verwendet. Der *outData*-Parameter kann einen Wert zwischen 0 und -255 haben und in hexadezimalem oder Integer-Format dargestellt werden (d. h. &H0-&HFF oder 0-255).

Die Tabelle unten zeigt mögliche E/A-Kombinationen und ihre entsprechenden *outData*-Werte, davon ausgehend, dass die *portNumber* 0 oder entsprechend 1 ist.

Ausgangseinstellungen wenn *portNumber*=0 (Ausgangsnummer)

OutData-Wert	7	6	5	4	3	2	1	0
01	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
02	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Aus
03	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
08	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Aus
09	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Ein
10	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Aus
11	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Ein
99	Aus	Ein	Ein	Aus	Aus	Aus	Ein	Ein
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Ausgangseinstellungen wenn *portNumber*=1 (Ausgangsnummer)

OutData-Wert	15	14	13	12	11	10	9	8
01	Aus	Aus	Aus	Aus	Aus	Aus	Aus	Ein
02	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Aus
03	Aus	Aus	Aus	Aus	Aus	Aus	Ein	Ein
08	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Aus
09	Aus	Aus	Aus	Aus	Ein	Aus	Aus	Ein
10	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Aus
11	Aus	Aus	Aus	Ein	Aus	Aus	Aus	Ein
99	Aus	Ein	Ein	Aus	Aus	Aus	Ein	Ein
255	Ein	Ein	Ein	Ein	Ein	Ein	Ein	Ein

Hinweise

Der Unterschied zwischen OpBCD und Out

Die Befehle **Out** und OpBCD in der Sprache SPEL+ sind einander sehr ähnlich. Es gibt jedoch einen wesentlichen Unterschied. Dieser Unterschied wird im Folgenden aufgezeigt:

- Der OpBCD-Befehl verwendet das BCD-Format zur Spezifizierung eines 8-Bit-Wertes, der verwendet wird, um die Ausgänge ein- oder auszuschalten. Da das BCD-Format den Gebrauch der Werte &HA, &HB, &HC, &HD, &HE oder &HF ausschließt, können nicht alle Kombinationen für die Einstellung der 8 Ausgänge erfüllt werden.
- Der **Out**-Befehl funktioniert ähnlich wie der OpBCD-Befehl, nur, dass der Out-Befehl es ermöglicht, den Bereich der 8-Bit-Werte, der für das Ein- oder Ausschalten der Ausgänge verwendet werden darf, zwischen 0 und 255 anzusiedeln (gegenüber 0 bis 99 bei OpBCD). Somit können alle möglichen Kombinationen für die 8 Ausgangsgruppen den Benutzerspezifikationen folgend initialisiert werden.

Verwandte Befehle

In, InBCD, MemOff, MemOn, MemOut, MemSw, Off, On, Oport, Sw, Wait

Beispiel einer Out-Anweisung

Das Beispiel unten zeigt, wie der Haupttask einen Hintergrundtask namens *iotask* startet. *iotask* ist ein einfacher Task, um abwechselnd die Ausgänge 0-3 ein- und dann wieder auszuschalten. Der Out-Befehl macht dies möglich, indem nur ein Befehl verwendet wird, anstatt jeden Ausgang einzeln ein- und auszuschalten.

```
Function main

  Xqt iotask
  Do
    Go P1
    Go P2
  Loop
Fend

Function iotask

  Do
    Out 0, &H0F
    Out 0, &H00
    Wait 10
  Loop
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus :

```
> Out 1,6  'Schaltet die Ausgänge 9 und 10 ein
> Out 2,1  'Schaltet den Ausgang 8 ein
> Out 3,91 'Schaltet die Ausgänge 24, 25, 27, 28 und 30 ein
```

Out-Funktion

F

Gibt den Status eines Ausgangsbytes aus.

Syntax

Out(*portNumber***)**

Parameter

portNumber Integer-Ausdruck, der für die E/A-Ausgangsbytes steht. Die *portNumber*-Auswahl entspricht den folgenden Ausgängen:

Portnummer	Ausgänge
0	0-7
1	8-15
...	...

Rückgabewerte

Der Ausgangsstatus 8-Bit-Wert für den spezifizierten Port.

Verwandte Befehle

Out-Anweisung

Beispiel einer Out-Funktion

```
Print Out(0)
```

OutW-Anweisung

Setzt 16 Ausgänge gleichzeitig.

Syntax

OutW *wordPortNum, outputData*

Parameter

wordPortNum	Integer-Ausdruck, der für die E/A-Ausgangsworte steht.
outputData	Spezifiziert Ausgangsdaten (Integer von 0 bis 65535) unter Verwendung eines Ausdrucks oder eines numerischen Wertes.

Beschreibung

Ändert den aktuellen Status der E/A-Ausgangsportgruppe, spezifiziert durch die Wort-Portnummer, in die angegebenen Ausgangsdaten.

Verwandte Befehle

In, InW, Out

Beispiel einer OutW-Anweisung

```
OutW 0, 25
```

OutW-Funktion

F

Gibt den Status eines Ausgangswortes (2 Bytes) aus.

Syntax

OutW(*wordPortNum***)**

Parameter

wordPortNum Integer-Ausdruck, der für die E/A-Ausgangsworte steht.

Rückgabewerte

Der Ausgangsstatus 16-Bit-Wert für den spezifizierten Port.

Verwandte Befehle

OutW-Anweisung

Beispiel einer OutW-Funktion

```
OutW 0, &H1010
```

PAgl-Funktion

F

Gibt den Achswert eines bestimmten Punktes aus.

Syntax

PAgl (*point*, *jointNumber)*

Parameter

point	Punktausdruck.
jointNumber	Definiert die Achsnummer (ganze Zahl von 1 bis 6) unter Verwendung eines Ausdrucks oder eines numerischen Werts.

Rückgabewerte

Gibt die berechnete Achsposition aus (reeller Wert, in Grad bei der Rotationsachse, in mm bei der linearen Achse).

Verwandte Befehle

Agl, CX, CY, CZ, CU, CV, CW, PPls

Beispiel einer PAgl-Funktion

```
Real joint1
joint1 = PAgl(P10, 1)
```

Pallet-Anweisung

Definiert Paletten und zeigt sie an.

> S

Syntax

Pallet [**Outside**,] [*palletNumber, Pi, Pj, Pk* [,*Pm*], *columns, rows*]

Parameter

Outside	Optional. Lässt Zeilen- und Spaltenindizes außerhalb des Bereichs der spezifizierten Reihen und Spalten zu. Der Bereich umfasst -32768 bis 32767.
palletNumber	Palettennummer, angegeben durch einen Integer zwischen 0 und 15.
Pi, Pj, Pk	Punktvariablen, die die Standard-3-Punkte-Palette definieren.
Pm	Optional. Punktvariable, die mit Pi, Pj und Pk verwendet wird, um eine 4-Punkte-Palette zu definieren.
columns	Integer-Ausdruck, der für die Anzahl von Punkten auf der Pi-bis-Pj-Seite der Palette steht. Der Bereich umfasst 1 bis 32767.
rows	Integer-Ausdruck, der für die Anzahl von Punkten auf der Pi-bis-Pk-Seite der Palette steht. Der Bereich umfasst 1 bis 32767.

Rückgabewerte

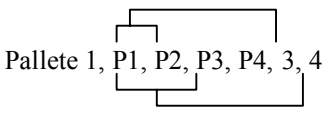
Wenn die Parameter weggelassen werden, werden alle definierten Paletten angezeigt.

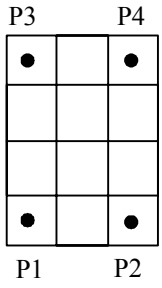
Beschreibung

Definiert eine Palette, indem dem Roboter mindestens die Punkte Pi, Pj und Pk geteacht werden und indem die Anzahl von Punkten von Pi bis Pj und von Pi bis Pk angegeben wird.

Wenn die Palette eine gleichmäßige, rechteckige Form hat, müssen nur 3 der 4 Eckpunkte definiert werden. In den meisten Fällen ist es jedoch besser, 4 Eckpunkte zu verwenden, um eine Palette zu definieren.

Um eine Palette zu definieren, teachen Sie dem Roboter zunächst entweder 3 oder 4 Eckpunkte und definieren Sie dann die Palette wie folgt:
Eine mit 4 Punkten definierte Palette: P1, P2, P3 und P4 wird unten gezeigt. Es gibt 3 Positionen von P1-P2 und 4 Positionen von P1-P3. Dies ergibt eine Palette mit einer Gesamtzahl von 12 Positionen. Die Syntax für die Definition dieser Palette lautet wie folgt:





10	11	12
7	8	9
4	5	6
1	2	3

Punkten, die für Teile einer Palette stehen, werden automatisch Teilnummern zugeordnet, die in diesem Beispiel bei P1 beginnen. Diese Teilnummern werden auch von der Pallet-Funktion benötigt. Wenn *Outside* angegeben wird, können die Zeilen- und Spaltenindizes außerhalb des Bereichs der Zeilen und Spalten spezifiziert werden. *Outside* sollte durch eine zweidimensionale Teilung spezifiziert werden.

Beispiel:

```
Palette außen 1, P1, P2, P3, 4, 5
Jump Pallet(1, -2, 10)
```

-2,10						
			1,5	2,5	3,5	4,5
			1,4	2,4	3,4	4,4
			1,3	2,3	3,3	4,3
			1,2	2,2	3,2	4,2
			1,1	2,1	3,1	4,1

Beispiel

		1,6			4,6		
-1,4		1,4	2,4	3,4	4,4		6,4
		1,3	2,3	3,3	4,3		
		1,2	2,2	3,2	4,2		
-1,1		1,1	2,1	3,1	4,1		6,1
		1,-1			4,-1		

Hinweise

Die maximale Palettengröße:

Die Gesamtanzahl von Punkten, die durch eine spezifische Palette definiert ist, muss weniger als 32767 betragen.

Nicht korrekte Palettenform-Definitionen:

Beachten Sie, dass eine falsche Punktreihenfolge oder eine falsche Anzahl von Teilen zwischen den Punkten eine nicht korrekte Palettenform-Definition zur Folge hat.

Definition der Palettenebene:

Die Palettenebene wird durch Z-Achsen-Koordinatenwerte der drei Palletteneckpunkte definiert. Daher können auch vertikale Paletten definiert werden.

Palettendefinition für eine Palette mit einer einzigen Reihe:

Eine Palette mit nur einer einzigen Reihe kann mit einer 3 Punkte-Pallet-Anweisung oder einem 3 Punkte-Pallet-Befehl definiert werden. Teachen Sie einen Punkt an jedem Ende und definieren Sie dann wie folgt: Spezifizieren Sie 1 als Anzahl der Teile zwischen demselben Punkt.

```
> Palette 2, P20, P21, P20, 5, 1        'Definiert eine 5x1 Palette
```

Verwandte Befehle

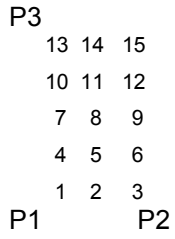
Pallet-Funktion

Beispiel einer Pallet-Anweisung

Der folgende Befehl aus dem Befehlseingabefenster erstellt die Palette, die durch die Punkte P1, P2 und P3 definiert ist und teilt die Palettenebene in 15 gleich verteilte Palettenpunkt-Positionen, bei denen Palettenpunkt Nr. 1, Palettenpunkt Nr. 2 und Palettenpunkt Nr. 3 entlang der Seite P1-bis-P2 angeordnet sind.

```
> pallet 1, P1, P2, P3, 3, 5
> jump pallet(1, 2)          'Springt zur Position auf der Palette
```

Die resultierende Palette sieht aus wie folgt:



Pallet-Funktion

F

Spezifiziert eine Position in einer vorher definierten Palette.

Syntax

(1) Pallet (*palletNumber*, *palletPosition* **)**
(2) Pallet (*palletNumber*, *column, row* **)**

Parameter

palletNumber	Palettennummer, angegeben durch einen Integer-Ausdruck zwischen 0 und 15.
PalletPosition	Die Palettenposition wird durch einen Integer zwischen 1 und 32767 angegeben.
column	Die Palettenspalte wird durch einen Integer-Ausdruck zwischen 1 und 32767 angegeben.
row	Die Palettenzeile wird durch einen Integer-Ausdruck zwischen 1 und 32767 angegeben.

Beschreibung

Pallet gibt eine Position in einer Palette aus, die zuvor durch die Pallet-Anweisung definiert wurde. Verwenden Sie die Funktion zusammen mit den Befehlen Go oder Jump, um den Arm zur spezifizierten Palettenposition zu bewegen.

Die Palettenpositionsnummer kann arithmetisch oder durch einen Integer definiert werden.

Verwandte Befehle

Pallet-Anweisung

Beispiel einer Pallet-Funktion

Das folgende Programm transferiert Objekte von Palette 1 nach Palette 2.

```
Function main
  Integer index
  Pallet 1, P1, P2, P3, 3, 5      'Definiert Palette 1
  Pallet 2, P12, P13, P11, 5, 3   'Definiert Palette 2
  For index = 1 To 15
    Jump Pallet(1, index)     'Bewegt den Arm zum Punkt-Index auf
Palette 1
    On 1     'Hält das Werkstück
    Wait 0.5
    Jump Pallet(2, index)     'Bewegt den Arm zum Punkt-Index auf
Palette 2
    Off 1     'Gibt das Werkstück frei
    Wait 0.5
  Next I
Fend
```

ParseStr-Anweisung / -Funktion

Analysiert eine Zeichenkette syntaktisch und gibt eine Textelement-Matrix aus.

F S

Syntax

ParseStr *inputString$, tokens$(), delimiters$*
numTokens = **ParseStr(***inputString$, tokens$(), delimiters$***)**

Parameter

inputString$	Zeichenkettenausdruck, der syntaktisch analysiert (geparst) werden soll.
tokens$()	Ausgangsmatrix von Zeichenketten, die die Textelemente enthalten. Die durch ByRef deklarierte Matrix kann nicht spezifiziert werden.
delimiters$	Zeichenkettenausdruck, der eine oder mehrere Textelement-Begrenzungen enthält.

Rückgabewerte

Wird ParseStr als Funktion verwendet, wird die Anzahl der geparsten Textelemente ausgegeben.

Verwandte Befehle

Redim, String

Beispiel einer ParseStr-Anweisung

```
String toks$(0)
Integer i

ParseStr "1 2 3 4", toks$(), " "

For i = 0 To UBound(toks)
   Print "token ", i, " = ", toks$(i)
Next i
```

Pass-Anweisung

Führt gleichzeitig vier PTP-Bewegungen aus, führt an festgelegten Punkten vorbei, aber nicht durch diese hindurch.

> S

Syntax

Pass *point* [, {**On | Off | MemOn | MemOff**} *bitNumber* [*, point ...*]]

Parameter

point	**P***number* oder **P**(*expr*) oder Punktlabel. Wenn die Punktedaten fortgesetzt werden und aufsteigend oder absteigend angeordnet sind, geben Sie zwei Punktnummern an und verbinden Sie diese mit einem Doppelpunkt als P(1:5).
bitNumber	Schaltet den E/A-Ausgang oder den Merker ein oder aus. Integer zwischen 0 und 511 oder Ausgangslabel.

Beschreibung

Der **Pass**–Befehl bewegt den Roboterarm in die Nähe, aber nicht direkt durch die spezifizierte Punktserie.

Um eine Punktserie zu spezifizieren, verwenden Sie Punkte (P0,P1, ...) und trennen Sie diese durch Kommata voneinander ab.

Um Ausgänge während der Bewegung ein- oder auszuschalten, fügen Sie einen durch Kommata abgetrennten On- oder Off-Befehl zwischen die Punkte ein. On oder Off wird ausgeführt, bevor der Roboter den Punkt unmittelbar vor On oder Off erreicht.

Wenn auf **Pass** direkt ein weiterer **Pass-**Befehl folgt, wechselt die Steuerung zum folgenden **Pass-**Befehl, ohne dass der Roboter am letzen definierten Punkt des vorangegangenen **Pass-**Befehls anhält.

Wenn direkt auf **Pass** ein Bewegungsbefehl folgt, der nicht **Pass** lautet, hält der Roboter am letzen definierten Punkt des vorangegangenen **Pass-**Befehls an. Eine **Fine**-Positionierung wird jedoch nicht durchgeführt.

Wenn direkt auf **Pass** ein Befehl, eine Anweisung oder eine Funktion folgt, der bzw. die kein Bewegungsbefehl ist, kann der direkt darauf folgende Befehl, die Anweisung oder die Funktion ausgeführt werden, bevor der Roboter den letzen Punkt des vorangegangenen Pass-Befehls erreicht hat.

Wenn eine Fine-Positionierung an der Zielposition gewünscht wird, lassen Sie auf den Pass- einen Go-Befehl folgen und spezifizieren Sie die Zielposition wie im folgenden Beispiel gezeigt.

```
Pass P5; Go P5; On 1; Move P10
```

Je größer die Werte für Beschleunigung und Verzögerung sind, desto näher fährt der Arm an den angegebenen Punkt. Der **Pass-**Befehl kann auch so verwendet werden, dass der Roboterarm Hindernisse umgeht.

Verwandte Befehle

Accel, Go, Jump, Speed

Beispiel einer Pass-Anweisung

Das Beispiel zeigt die Handhabung des Roboterarms mit Hilfe des Pass-Befehls:

```
Function main
  Jump P1
  Pass P2   'Bewegt den Arm zu P2 und führt
            'den nächsten Befehl durch, bevor P2 erreicht wird
  On 2
  Pass P3
  Pass P4
  Off 0
  Pass P5
Fend
```

Pause-Anweisung

S

Vorübergehender Stopp der Ausführung aller Programme, für die die Pause-Anweisung aktiviert ist.

Syntax

Pause

Beschreibung

Wenn der **Pause**-Befehl ausgeführt wird, wird die Programmausführung für alle Tasks unterbrochen, für die *Pause* aktiviert ist. Ebenso werden alle Tasks unterbrochen, die gerade einen Bewegungsbefehl ausführen, selbst wenn *Pause* für diese Tasks nicht aktiviert ist.

Hinweise

QP und die Auswirkungen auf Pause:

Der QP-Befehl wird verwendet, um den Arm direkt bei *Pause* zu stoppen, oder um die aktuelle Bewegung abzuschließen und das Programm dann zu pausieren. Lesen Sie die Hilfe zum QP-Befehl für weitere Informationen.

Beispiel einer Pause-Anweisung

Das Beispiel unten zeigt die Verwendung des **Pause**-Befehls zur vorübergehenden Aussetzung der Programmausführung. Der Task führt Programmanweisungen aus, bis er die Zeile erreicht, die einen Pause-Befehl enthält. An diesem Punkt wird der Task ausgesetzt. Der Anwender kann dann auf den Button *Continue* im Run-Fenster klicken, um mit der Ausführung fortzufahren.

```
Function main

  Xqt monitor
  Go P1
  On 1
  Jump P2
  Off 1
  Pause    'Unterbricht die Programmausführung
  Go P40
  Jump P50
Fend
```

PauseOn-Funktion

F

Gibt den aktuellen *Pause*-Status aus.
Diese Funktion wird aus Kompatibilitätsgründen freigehalten.
Alle Tasks werden im Pausestatus der Steuerung unterbrochen.

PDef-Funktion

F

Gibt den Definitionsstatus eines spezifizierten Punktes aus.

Syntax

PDef (*point*)

Parameter

point Integer-Ausdruck oder **P***number* oder **P**(*expr*) oder Punktlabel.

Rückgabewerte

Wahr, wenn der Punkt definiert ist, sonst Falsch.

Verwandte Befehle

Here-Anweisung, Pdel

Beispiel einer PDef-Funktion

```
If Not PDef(1) Then
    Here P1
Endif
```

PDel

Löscht spezifizierte Positionsdaten.

Syntax

PDel *firstPointNum , [lastPointNum]*

Parameter

firstPointNum	Die erste Punktnummer in einer zu löschenden Punktsequenz. *firstPointNum* muss ein Integer sein.
lastPointNum	Die letzte Punktnummer in einer zu löschenden Punktsequenz. *lastPointNum* muss ein Integer sein.

Beschreibung

Löscht spezifizierte Postitionsdaten aus dem Punktspeicher der Steuerung für den aktuellen Roboter. Löscht alle Positionsdaten angefangen bei *firstPointNum* bis einschließlich *lastPointNum*. Um zu verhindern, dass der Fehler Nr. 2 auftritt, muss *firstPointNum* kleiner sein als *lastPointNum*.

Beispiel für PDel

```
> p1=10,300,-10,0/L
> p2=0,300,-40,0
> p10=-50,350,0,0
> pdel 1,2        'Löscht die Punkte 1 und 2
> plist
P10  =   -50.000,  350.000,    0.000,    0.000 /R /0
> pdel 50         'Löscht Punkt 50
> pdel 100,200    'Löscht die Punkte 100 bis 200
>
```

PLabel$-Funktion

F

Gibt das mit einer Punktnummer verbundene Punktlabel aus.

Syntax

PLabel$(*point***)**

Parameter

point	Integer-Ausdruck oder **P***number* oder **P**(*expr*) oder Punktlabel.

Verwandte Befehle

PDef-Funktion, PLabel-Anweisung, PNumber-Funktion

Beispiel einer PLabel$-Funktion

```
Print PLabel$(1)
Print PLabel$(P(i))
```

PLabel-Anweisung

Definiert ein Label für einen spezifizierten Punkt.

> S

Syntax

PLabel *pointNumber, newLabel*

Parameter

pointNumber	Ein Integer-Ausdruck, der für eine Punktnummer steht.
newLabel	Ein Zeichenkettenausdruck, der für das Label steht, das für den spezifizierten Punkt verwendet wird.

Verwandte Befehle

PDef-Funktion, PLabel-Funktion, PNumber-Funktion

Beispiel einer PLabel-Anweisung

```
PLabel 1, "pick"
```

PList

Zeigt Punktedaten im Speicher für den aktuellen Roboter an.

Syntax

(1) PList
(2) PList *pointNumber*
(3) PList *startPoint*,
(4) PList *startPoint*, **endpoint**

Parameter

pointNumber	Der Nummernbereich umfasst 0 bis 999.
startPoint	Die Startpunktnummer. Der Nummernbereich umfasst 0 bis 999.
endPoint	Der Endpunkt-Index. Der Nummernbereich umfasst 0 bis 999.

Rückgabewerte

Punktedaten.

Beschreibung

Plist zeigt Punktedaten im Speicher für den aktuellen Roboter an.

Wenn innerhalb des spezifizierten Punktebereichs keine Punktedaten vorhanden sind, werden keine Daten angezeigt.
Wenn eine Startpunktnummer größer festgelegt wird als seine Endpunktnummer, tritt ein Fehler auf.

(1) PList

Zeigt die Koordinatendaten für alle Punkte an.

(2) PList pointIndex

Zeigt die Koordinatendaten für den spezifizierten Punkt an.

(3) PList *startPoint*,

Zeigt die Koordinatendaten für alle Punkte an, die mit *startPoint* anfangen.

(4) PList *startPoint*, endpoint

Zeigt die Koordinatendaten für alle Punkte an, die mit *startPoint* anfangen und mit *endPoint* enden.

Beispiel für PList

Dieses Beispiel zeigt die Punktedaten der Armposition des aktuellen Roboters an.

```
> plist *
WORLD:  X:  360.000 mm  Y:   10.000 mm  Z:    0 mm  U:    0.000 deg
JOINT:  1:    0.000 deg 2:    0.000 deg 3:    0.000 mm  4:    0.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:        0 pls
```

Dieses Beispiel enthält die Punktedaten für einen Punkt.

```
> plist 1
P1   =   290.000,    0.000,  -20.000,    0.000 /R /0
>
```

Dieses Beispiel enthält die Punktedaten innerhalb des Bereichs von 10 und 20. In diesem Beispiel wurden in diesem Bereich nur drei Punkte gefunden.

```
> plist 10, 20
P10  =   290.000,    0.000,  -20.000,    0.000 /R /0
P12  =   300.000,    0.000,    0.000,    0.000 /R /0
P20  =   285.000,   10.000,  -30.000,   45.000 /R /0
>
```

Dieses Beispiel enthält die Punktedaten, die mit der Punktnummer 10 beginnen.

```
> plist 10,
P10  =   290.000,    0.000,  -20.000,    0.000 /R /0
P12  =   300.000,    0.000,    0.000,    0.000 /R /0
P20  =   285.000,   10.000,  -30.000,   45.000 /R /0
P30  =   310.000,   20.000,  -50.000,   90.000 /R /0
>
```

PLocal-Anweisung

> S

Stellt das lokale Attribut für einen Punkt ein.

Syntax

PLocal(*point***) =** *localNumber*

Parameter

point	Integer-Ausdruck oder **P***number* oder **P**(*expr*) oder Punktlabel.
localNumber	Ein Integer Ausdruck, der für die neue Local-Nummer steht. Der Bereich umfasst 0 bis 15.

Verwandte Befehle

PLocal-Funktion

Beispiel einer PLocal-Anweisung

```
PLocal(pick) = 1
```

PLocal-Funktion

F

Gibt die Local-Nummer für einen spezifizierten Punkt aus.

Syntax

PLocal(*point***)**

Parameter

point Integer-Ausdruck oder **P***number* oder **P**(*expr*) oder Punktlabel.

Rückgabewerte

Local-Nummer für einen spezifizierten Punkt.

Verwandte Befehle

PLocal-Anweisung

Beispiel einer PLocal-Funktion

```
Integer localNum
localNum = PLocal(pick)
```

Pls-Funktion

F

Gibt den aktuellen Encoder-Pulse-Zählstand für jede Achse an der aktuellen Position aus.

Syntax

Pls*(jointNumber)*

Parameter

jointNumber Die Achse, für die der aktuelle Encoder-Pulse-Zählstand ermittelt werden soll.

Rückgabewerte

Gibt einen Nummernwert aus, der den aktuellen Encoder-Pulse-Zählstand für die durch *jointNumber* spezifizierte Achse ausgibt.

Beschreibung

Pls wird verwendet, um die aktuelle Encoderposition (Pulse-Zählstand) jeder Achse zu lesen. Diese Werte können gespeichert und später mit dem Pulse-Befehl verwendet werden.

Verwandte Befehle

CX, CY, CZ, CU, CV, CW, Pulse

Beispiel einer Pls-Funktion

Im Folgenden wird ein einfaches Beispiel für die Ermittlung und das Drucken der Pulse-Werte jeder Achse aufgezeigt.

```
Function plstest
  Real t1, t2, z, u
  t1 = pls(1)
  t2 = pls(2)
  z = pls(3)
  u = pls(4)
  Print "T1 joint current Pulse Value: ", t1
  Print "T2 joint current Pulse Value: ", t2
  Print "Z joint current Pulse Value: ", z
  Print "U joint current Pulse Value: ", u
Fend
```

PNumber-Funktion

Gibt die mit einem Punktlabel verbundene Punktnummer aus.

Syntax

PNumber(*pointLabel***)**

Parameter

pointLabel Ein Punktlabel, das in der aktuellen Punktedatei verwendet wird oder ein Zeichenkettenausdruck, der ein Punktlabel enthält.

Verwandte Befehle

PDef-Funktion, PLabel$-Funktion

Beispiel einer PNumber-Funktion

```
Integer pNum
String pointName$

pNum = PNumber(pick)

pNum = PNumber("pick")

pointName$ = "place"
pNum = PNumber(pointName$)
```

Punkt-Zuweisung

Definiert einen Roboterpunkt durch Zuordnung zu einem Punktausdruck.

> S

Syntax

point = pointExpr

Parameter

point Ein folgendermaßen spezifizierter Roboterpunkt:
P*number*
P(*expr*)
pointLabel

pointExpr Punktausdruck.

Beschreibung

Definiert einen Roboterpunkt durch Gleichsetzung mit einem anderen Punkt oder Punktausdruck.

Verwandte Befehle

Local, Pallet, PDef, PDel, Plist

Beispiel einer Punkt-Zuweisung

Die folgenden Beispiele werden vom Befehlseingabefenster aus ausgeführt:

Zuordnung von Koordinaten zu P1:

```
> P1 = 300,200,-50,100
```

Spezifizierung der Linksarmstellung:

```
> P2 = -400,200,-80,100/L
```

Addieren Sie 20 zur X-Koordinate von P2 und definieren Sie den resultierenden Punkt als P3:

```
> P3 = P2 +X(20)
> plist 3
P3=-380,200,-80,100/L
```

Subtrahieren Sie 50 von der Y-Koordinate von P2, ersetzen Sie die Z-Koordinate durch -30, und definieren Sie den resultierenden Punkt P4 als Stellung des rechten Arms:

```
>P4=P2 -Y(50) :Z(-30) /R
```

Addieren Sie 90 zur U-Koordinate der Palette(n) (3, 5), und definieren Sie den resultierenden Punkt als P6:

```
> P5 = Here
> P6 = pallet(3,5) +U(90)
```

Punktausdruck

Spezifiziert einen Roboterpunkt für die Zuordnung und für Bewegungsbefehle.

> S

Syntax

point [{ **+** | **-** } *point*] [*local*] [*hand*] [*elbow*] [*wrist*] [*j4flag*] [*j6flag*] [*relativeOffsets*] [*absoluteCoords*]

Parameter

point	Spezifizierung des Basis-Punktes. Dabei kann es sich um eine der folgenden Optionen handeln: **P***number* **P**(*expr*) **Here** **Pallet**(*palletNumber*, *palletIndex*) *pointLabel* **XY**(*X*, *Y*, *Z*, *U,* [*V*], [*W*]) **JA**(*J1, J2, J3, J4,* [*J5*], [*J6*]) **Pulse**(*J1, J2, J3, J4,* [*J5*], [*J6*])
local	Optional. Local-Nummer von 1 bis 15 mit einem vorangestellten Schrägstrich (/**0** bis /**15**) oder @-Zeichen (**@0** bis **@15**). Der Schrägstrich bedeutet, dass sich die Koordinaten im lokalen Koordinatensystem befinden. Die @-Zeichen bedeuten, dass die Koordinaten in lokale Koordinaten umgewandelt werden.
hand	Optional bei SCARA- und 6-Achsrobotern. Geben Sie /**L** oder /**R** für Linksarm- oder Rechtsarmausrichtung an.
elbow	Optional bei 6-Achsrobotern. Geben Sie /**A** oder /**B** für die Ausrichtung oberhalb oder Ausrichtung unterhalb an.
wrist	Optional bei 6-Achsrobotern. Geben Sie /**F** oder /**NF** für die Ausrichtung im negativen oder im positiven Winkelbereich an.
j4flag	Optional bei 6-Achsrobotern. Geben Sie /**J4F0** oder /**J4F1** an.
j6flag	Optional bei 6-Achsrobotern. Geben Sie /**J6F0** - /**J6F127** an.
relativeOffsets	Optional. Eine oder mehrere relative Koordinaten. {**+** | **-**} {**X** | **Y** | **Z** | **U** | **V** | **W**} **(***expr***)** Die TL-Offsets sind relative Versätze im aktuellen Werkzeug-Koordinatensystem. {**+** | **-**} **{TLX** | **TLY** | **TLZ** | **TLU** | **TLV** | **TLW**} **(***expr***)**
absoluteCoords	Optional. Eine oder mehrere absolute Koordinaten. **:** {**X** | **Y** | **Z** | **U** | **V** | **W**} **(***expr***)**

Beschreibung

Punktausdrücke werden in Punktzuordnungs-Anweisungen und Bewegungsbefehlen verwendet. Punkte können addiert und subtrahiert werden, wenn keine direkten Koordinaten verwendet werden. Beispiel:

```
Go P1 + P2
P1 = P2 + XY(100, 100, 0, 0)
```

Relative Versätze verwenden

Sie können eine oder mehrere Koordinaten relativ zum Basis-Punkt versetzen. Beispielsweise bewegt die folgende Anweisung den Roboter aus der aktuellen Position 20 mm die positive X-Achse entlang:

```
Go Here +X(20)
```

Wenn Sie die gleiche Anweisung erneut ausführen, bewegt sich der Roboter erneut um 20mm an der X-Achse entlang, da dies eine relative Bewegung ist.

Sie können auch relative Werkzeug-Versätze verwenden:

```
Go Here +TLX(20) -TLY(5.5)
```

Verwendung absoluter Koordinaten

Sie können eine oder mehrere Koordinaten des Basis-Punktes durch Verwendung absoluter Koordinaten ändern. Die folgende Anweisung bewegt den Roboter zur 20 mm-Position auf der X-Achse:

```
Go Here :X(20)
```

Wenn Sie dieselbe Anweisung erneut ausführen, bewegt sich der Roboter nicht, da er sich aufgrund der vorangegangenen Bewegung bereits in der absoluten Position für X befindet.

Relativer Versatz und absolute Koordinaten machen es einfach, einen Punkt vorübergehend zu modifizieren. Beispielsweise bewegt dieser Code den Roboter schnell an die Position 10 mm oberhalb des Pick-Punktes. Dabei wird ein relativer Versatz für Z von 10 mm verwendet. Anschließend bewegt sich der Roboter dann langsam zum Pick-Punkt selbst.

```
Speed fast
Jump pick +Z(10)
Speed slow
Go pick
```

Dieser Code bewegt den Roboter von der aktuellen Position direkt aufwärts, indem ein absoluter Wert von 0 (Null) für die Z-Achse spezifiziert wird:

```
LimZ 0
Jump Here :Z(0)
```

Verwendung von Locals

Sie können mit einem Schrägstrich oder einem @-Zeichen eine Local-Nummer spezifizieren. Jedes dieser Zeichen hat eine separate Funktion.

Verwenden Sie den Schrägstrich, um die Koordinaten in einer Local-Nummer zu markieren. Beispielsweise bedeutet das Hinzufügen von /1 in der folgenden Anweisung, dass P1 sich am Ort 0,0,0,0 in Local 1 befindet.

```
P1 = XY(0, 0, 0, 0) /1
```

Das @-Zeichen wird verwendet, um Koordinaten in Local-Koordinaten zu übertragen. Dieses Beispiel zeigt, wie ein Punkt in einer Local-Nummer geteacht wird:

```
P1 = Here @1
```

P1 wird auf die aktuelle Position gesetzt, die in die Position in Local 1 übertragen wurde.

Verwandte Befehle

Go, Local, Pallet, Pdel, Plist, Hand, Elbow, Wrist, J4Flag, J6Flag

Beispiel eines Punktausdrucks

Hier sind einige Beispiele der Verwendung von Punktausdrücken in Zuordnungsanweisungen und Bewegungsbefehlen:

```
P1 = XY(300,200,-50,100)
P2 = P1 /R
P3 = pick /1
P4 = P5 + P6
P(i) = XY(100, 200, CZ(P100), 0)
Go P1 -X(20) :Z(-20) /R
Go Pallet(1, 1) -Y(25.5)
Move pick /R
Jump Here :Z(0)
Go Here :Z(-25.5)
Go JA(25, 0, -20, 180)
pick = XY(100, 100, -50, 0)

P1 = XY(300.200, -50.100, -90, 0)
P2 = P1 /F /B
P2 = P1 +TLV(25)
```

PosFound-Funktion

F

Gibt den Status des Find-Vorgangs aus.

Syntax

PosFound

Rückgabewerte

Wahr, wenn die Position während der Bewegung gefunden wurde. Falsch, wenn die Position nicht gefunden wurde.

Verwandte Befehle

Find

Beispiel einer PosFound-Funktion

```
Find Sw(5) = ON
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Error: Cannot find the sensor signal."
EndIf
```

Power-Anweisung

In älteren Fassungen Lp genannt.

> S

Schaltet den Power-Modus auf High oder Low und zeigt den aktuellen Status an.

Power-Syntax

(1) Power { **High | Low** }
(2) Power

Parameter

High | Low Die Einstellung kann High oder Low sein. Die Vorgabeeinstellung ist Low.

Rückgabewerte

Zeigt den aktuellen Power-Status an, wenn der Parameter weggelassen wird.

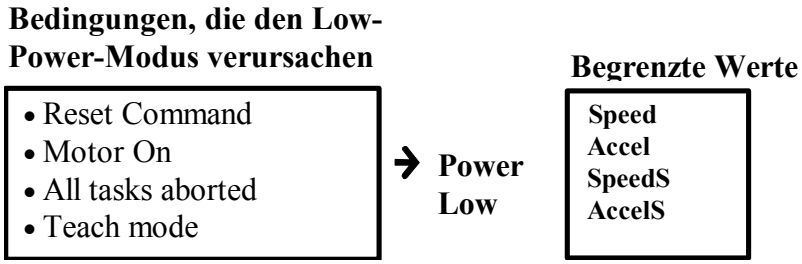
Beschreibung

Die Anweisung schaltet den Power-Modus auf High oder Low. Außerdem wird der aktuelle Modusstatus angezeigt.

Low – Wenn die Leistung auf Low eingestellt ist, ist der Low-Power-Modus eingeschaltet. Das bedeutet, dass der Roboter langsam verfährt (unter 250 mm/s) und die Servo-Steifigkeit auf niedrig eingestellt wird, damit die Servoachse freigeschaltet wird, wenn der Roboter gegen ein Objekt stößt. Dies ist die normale Betriebsart zum Teachen von Punkten.

High – Wenn die Leistung auf High eingestellt ist, ist der Low-Power-Modus ausgeschaltet. Das bedeutet, dass der Roboter mit voller Geschwindigkeit und mit voller Servo-Steifigkeit verfahren kann. Dies ist die normale Betriebsart für den Ablauf tatsächlicher Anwendungen.

Die folgenden Vorgänge bewirken ein Umschalten in den Low-Power-Modus. In diesem Fall werden die Einstellungen für Geschwindigkeit und Beschleunigung auf niedrige Werte begrenzt.



Hinweise

Der Low-Power-Modus (Power Low) und seine Auswirkungen auf die maximale Geschwindigkeit (Max Speed):

Im Low-Power-Modus ist die Motorkraft begrenzt und die Einstellung der effektiven Bewegungsgeschwindigkeit ist niedriger als der Vorgabewert. Wenn im Low-Power-Modus eine höhere Geschwindigkeit (direkt) vom Befehlseingabefenster aus oder in einem Programm spezifiziert wurde, wird die Geschwindigkeit auf den Vorgabewert gesetzt. Wenn eine höhere Bewegungsgeschwindigkeit benötigt wird, setzen Sie die Leistung auf High.

Der High-Power-Modus (Power High) und seine Auswirkungen auf die maximale Geschwindigkeit (Max Speed):

Im High-Power-Modus können höhere Geschwindigkeiten als der Vorgabewert eingestellt werden.

Verwandte Befehle

Accel, AccelS, Speed, SpeedS

Beispiel einer Power-Anweisung

Die folgenden Beispiele werden vom Befehlseingabefenster aus ausgeführt:

```
> Speed 50          'Bestimmt hohe Geschwindigkeit im Low-Power-Modus

> Accel 100, 100    'Spezifiziert hohe Beschleunigung

> Jump P1           'Bewegt den Roboter mit niedriger Geschwindigkeit und
                     niedriger Beschleunigung

> Speed             'Zeigt die aktuellen Geschwindigkeitswerte an
Low-Power-Modus:
   50
   50     50

> Accel             'Zeigt die aktuellen Beschleunigungswerte an
Low-Power-Modus:
   100    100
   100    100
   100    100

> Power High        'Setzt den High-Power-Modus

> Jump P2           'Bewegt den Roboter mit hoher Geschwindigkeit
```

Power-Funktion

F

Gibt den Power-Status aus.

Syntax

Power

Rückgabewerte

0 = Power Low, 1 = Power High.

Verwandte Befehle

Power-Anweisung

Beispiel einer Power-Funktion

```
If Power = 0 Then
    Print "Low Power Mode"
EndIf
```

PPls-Funktion

F

Gibt die Pulse-Position einer bestimmten Achse für einen angebenen Punkt aus.

Syntax

PPls (*point*, *jointNumber)*

Parameter

point	Punktausdruck.
jointNumber	Definiert die Achsnummer (ganze Zahl von 1 bis 6) unter Verwendung eines Ausdrucks oder eines numerischen Werts.

Rückgabewerte

Gibt die errechnete Achsposition aus (Long-Wert, in Pulsen).

Verwandte Befehle

Agl, CX, CY, CZ, CU, CV, CW, Pagl

Beispiel einer PPls-Funktion

```
Long pulses1
pulses1 = PPls(P10, 1)
```

Print-Anweisung

> S

Gibt Daten zum aktuellen Anzeigefenster aus, einschließlich des Run-Fensters, des Benutzerfensters, des Befehlseingabefensters und des Makrofensters.

Syntax

Print *expression* [*, expression...*] [,]
Print

Parameter

expression	Optional. Eine Zahl oder ein Zeichenkettenausdruck.
, (comma)	Optional. Wenn am Ende der Anweisung ein Komma steht, wird kein CRLF hinzugefügt.

Rückgabewerte

Variablendaten oder die spezifizierte Zeichenkette.

Beschreibung

Print zeigt die Variablendaten oder die Buchstabenzeichenkette auf dem Anzeigegerät an.

Ans Ende der Zeile wird automatisch CRLF (carriage return and line feed / Wagenrücklauf mit Zeilenvorschub) angehängt, es sei denn, am Ende der Anweisung steht ein Komma.

Verwandte Befehle

Print #

Beispiel einer Print-Anweisung

Das folgende Beispiel extrahiert den Wert der U-Achsen-Koordinate von Punkt P100 und verlegt den Koordinatenwert in die Variable *uvar*. Der Wert wird dann im aktuellen Anzeigefenster ausgegeben.

```
Function test
  Real uvar
  uvar = CU(P100)
  Print "The U Axis Coordinate of P100 is ", uvar
Fend
```

Print #-Anweisung

Gibt Daten an den angegebenen Kommunikationsport oder das angegebene Gerät aus. > S

Syntax

Print #*portNumber, expression* [*, expression...*]

Parameter

portNumber	Kommunikationsport oder Geräte-ID. Kommunikationsports können in OpenCom- (RS232) und OpenNet- (TCP/IP) Anweisungen definiert werden. *Die Integerwerte der Geräte-IDs lauten wie folgt:* *21 RC+* *23 OP* *24 TP*
expression	Ein numerischer- oder ein Zeichenkettenausdruck.

Beschreibung

Print # gibt Variablendaten, numerische Werte oder Zeichenketten an den durch *portNumber* spezifizierten Kommunikationsport oder das Gerät aus.

Verwandte Befehle

Print

Beispiel einer Print #-Anweisung

Im Folgenden sehen Sie einige Beispiele für Print #-Anweisungen.

```
Function printex
  String temp$
  Print #1, "5"     'Sendet das Zeichen "5" an den seriellen Port 1
  temp$ = "hello"
  Print #1, temp$
  Print #2, temp$
  Print #1 " Next message for port 1"
  Print #2 " Next message for port 2"
Fend
```

PTCLR-Anweisung

Löscht und initialisiert das maximale Drehmoment für eine oder mehrere Achsen.

> S

Syntax

PTCLR [*j1*], [*j2*], [*j3*], [*j4*], [*j5*], [*j6*]

Parameter

j1 - j6	Optional. Integer-Ausdruck, der für die Achsnummer steht. Wenn keine Parameter vorhanden sind, werden die Werte des maximalen Drehmoments für alle Achsen gelöscht.

Beschreibung

PTCLR löscht die Werte des maximalen Drehmoments für die spezifizierten Achsen.

Bevor Sie PTRQ ausführen, müssen Sie PTCLR ausführen.

Verwandte Befehle

ATRQ, PTRQ

Beispiel einer PTCLR-Anweisung

```
> ptclr
> go p1
> ptrq 1
      0.227
> ptrq
      0.227    0.118
      0.249    0.083
      0.000    0.000
>
```

PTPBoost-Anweisung

> S

Definiert die algorithmischen Verstärkungsparameter von Beschleunigung, Verzögerung und Gechwindigkeit für kurze PTP-Bewegungen.

Syntax

(1) PTPBoost *boost*, [*departBoost*], [*approBoost*]
(2) PTPBoost

Parameter

boost	Integer-Ausdruck zwischen 0 und 100.
departBoost	Optional. Jump-Depart-Verstärkungswert. Integer-Ausdruck zwischen 0 und 100.
approBoost	Optional. Jump-Approach-Verstärkungswert. Integer-Ausdruck zwischen 0 und 100.

Rückgabewerte

Wenn Parameter ausgelassen werden, werden die aktuellen PTPBoost-Einstellungen angezeigt.

Beschreibung

PTPBoost definiert die Beschleunigung, Verzögerung und Geschwindigkeit bei kurzen PTP-Bewegungen. Die Anweisung ist nur dann wirksam, wenn die Bewegungsdistanz kurz ist. Die PTPBoostOK-Funktion kann verwendet werden, um zu bestätigen, ob eine bestimmte Bewegungsdistanz zum Zielpunkt kurz genug ist, um von **PTPBoost** betroffen zu sein.

PTPBoost muss unter normalen Umständen nicht modifiziert werden. Verwenden Sie **PTPBoost** nur dann, wenn die Zykluszeit verringert werden muss, auch wenn dadurch die Vibration zunimmt. Sie haben außerdem die Möglichkeit, die Vibration zu reduzieren, auch wenn dadurch die Zykluszeit verlängert wird.

Wenn der **PTPBoost**-Wert groß ist, wird die Zykluszeit kürzer. Die Positionierungsvibration nimmt jedoch zu. Wenn der **PTPBoost**-Wert klein ist, nimmt die Positionierungsvibration zu, die Zykluszeit wird jedoch länger. Das Angeben von nicht angemessenen **PTPBoost**-Werten verursacht Fehler oder Schäden am Manipulator. Dies kann die Qualität des Roboters beeinträchtigen und seine Lebensdauer verkürzen.

Der **PTPBoost**-Wert wird auf seine Standardwerte zurückgesetzt, wenn einer der folgenden Befehle ausgeführt wird:

Controller Power On
Motor On
SFree, SLock
Reset
Stop-Taste oder Tasten Strg + C

Verwandte Befehle

PTPBoost-Funktion, PTPBoostOK

Beispiel einer PTPBoost-Anweisung

```
PTPBoost 50, 30, 30
```

PTPBoost-Funktion

F

Gibt den spezifizierten PTPBoost-Wert aus.

Syntax

PTPBoost(*paramNumber***)**

Parameter

paramNumber Integer-Ausdruck, der die folgenden Werte haben kann:
1: Verstärkungswert
2: Jump-Depart-Verstärkungswert
3: Jump-Approach-Verstärkungswert

Rückgabewerte

Integer-Wert zwischen 0 und 100.

Verwandte Befehle

PTPBoost-Anweisung, PTPBoostOK

Beispiel einer PTPBoost-Funktion

```
Print PTPBoost(1)
```

PTPBoostOK-Funktion

F

Gibt an, ob die PTP-Bewegung von der aktuellen Position zur Zielposition eine kurze Laufweite ist.

Syntax

PTPBoostOK(*targetPos***)**

Parameter

targetPos Punktausdruck für die Zielposition.

Rückgabewerte

Wahr, wenn es möglich ist, von der aktuellen Position zur Zielposition zu gelangen, ansonsten Falsch.

Beschreibung

Verwenden Sie **PTPBoostOK**, um festzustellen, ob die Distanz zwischen der aktuellen Position und der Zielposition kurz genug ist, damit PTPBoost wirksam ist.

Verwandte Befehle

PTPBoost

Beispiel einer PTPBoostOK-Funktion

```
If PTPBoostOK(P1) Then
  Go P1
EndIf
```

PTPTime-Funktion

F

Gibt die geschätzte Zeit für einen PTP-Bewegungsbefehl aus, ohne ihn auszuführen.

Syntax

(1) PTPTime(*destination, destArm, destTool*)
(2) PTPTime(*start, startArm, startTool, destination, destArm, destTool*)

Parameter

start	Punktausdruck für die Startposition.
destination	Punktausdruck für die Zielposition.
destArm	Integer-Ausdruck für die Zielpunkt-Armnummer.
destTool	Integer-Ausdruck für die Zielpunkt-Werkzeugnummer.
startArm	Integer-Ausdruck für die Startpunkt-Armnummer.
startTool	Integer-Ausdruck für die Startpunkt-Werkzeugnummer.

Rückgabewerte

Reeller Wert in Sekunden.

Beschreibung

Verwenden Sie PTPTime, um die Zeit zu berechnen, die für einen PTP-Bewegungsbefehl (Go) benötigt wird. Verwenden Sie Syntax 1, um die Zeit zu berechnen, die von der aktuellen Position zum Zielpunkt benötigt wird. Verwenden Sie Syntax 2, um die Zeit zu berechnen, die vom Startpunkt zum Zielpunkt zu benötigt wird.

Der tatsächliche Bewegungsvorgang wird nicht durchgeführt, wenn diese Funktion ausgeführt wird. Die aktuellen Positions-, Arm- und Werkzeugeinstellungen werden nicht verändert.

Wenn die Position nicht erreichbar ist oder die Arm- oder Werkzeugeinstellungen nicht korrekt sind, wird eine 0 ausgegeben.

Verwandte Befehle

ATRQ, Go, PTRQ

Beispiel einer PTPTime-Funktion

```
Real secs

secs = PTPTime(P1, 0, 0, P2, 0, 1)
Print "Time to go from P1 to P2 is:", secs

Go P1
secs = PTPTime(P2, 0, 1)
Print "Time to go from P1 to P2 is:", secs
```

PTran-Anweisung

S

Führt eine relative Bewegung einer Achse in Pulsen aus.

Syntax

PTran *joint, pulses*

Parameter

joint	Integer-Ausdruck, der für die zu bewegende Achse steht.
pulses	Integer Ausdruck, der für die Anzahl der zu verfahrenden Pulse steht.

Beschreibung

PTran wird verwendet, um eine Achse über eine definierte Anzahl von Pulsen von der aktuellen Position zu wegzubewegen.

Verwandte Befehle

Go, JTran, Jump, Move

Beispiel einer PTran-Anweisung

```
PTran 1, 2000
```

PTRQ-Anweisung

> S

Zeigt das maximale Drehmoment für die spezifizierte Achse an.

Syntax

PTRQ [*jointNumber*]

Parameter

jointNumber Optional. Integer-Ausdruck, der für die Achsnummer steht.

Rückgabewerte

Zeigt die Werte des maximalen Drehmoments für alle Achsen an.

Beschreibung

Verwenden Sie **PTRQ**, um den maximalen Drehmomentwert für eine oder alle Achsen seit der Ausführung der PTCLR-Anweisung anzuzeigen.

Das maximale Drehmoment ist eine reelle Zahl zwischen 0 und 1.

Verwandte Befehle

ATRQ-, PTCLR-, PTRQ-Funktion

Beispiel einer PTRQ-Anweisung

```
> ptclr
> go p1
> ptrq 1
      0.227
> ptrq
      0.227    0.118
      0.249    0.083
      0.000    0.000
>
```

PTRQ-Funktion

F

Gibt das maximale Drehmoment für die spezifizierte Achse aus.

Syntax

PTRQ(*jointNumber*)

Parameter

jointNumber Integer-Ausdruck, der für die Achsnummer steht.

Rückgabewerte

Reeller Wert zwischen 0 und 1.

Verwandte Befehle

ATRQ-, PTCLR-, PTRQ-Anweisung

Beispiel einer PTRQ-Funktion

In diesem Beispiel wird die **PTRQ**-Funktion in einem Programm verwendet:

```
Function DisplayPeakTorque
   Integer i

   Print "Peak torques:"
   For i = 1 To 4
      Print "Joint ", i, " = ", PTRQ(i)
   Next i
Fend
```

Pulse-Anweisung

Bewegt den Roboterarm mittels einer PTP-Bewegung zu dem Punkt, der durch die Pulse-Werte für jede Achse festgelegt wurde.

> S

Syntax

(1) Pulse *J1, J2, J3, J4 ,* [*J5*] *,* [*J6*]
(2) Pulse

Parameter

J1, J2, J3, J4	Der Pulse-Wert für jede der ersten vier Achsen. Der Pulse-Wert muss in dem durch den Range-Befehl definierten Bereich liegen und sollte ein Integer- oder ein Long-Ausdruck sein.
J5, J6	Optional. Für die Verwendung mit 6-Achsrobotern.

Rückgabewerte

Wenn Parameter weggelassen werden, werden die Pulse-Werte für die aktuelle Roboterposition angezeigt.

Beschreibung

Pulse verwendet den Achsen-Pulsewert der Pulse-Nullposition, um die Roboterarmposition darzustellen und nicht das orthogonale Koordinatensystem. Der Pulse-Befehl bewegt den Roboterarm mittels einer PTP-Bewegung.

Der Range-Befehl stellt die oberen und unteren Grenzwerte ein, die im Pulse-Befehl verwendet werden können.

Hinweis

Sorgen Sie dafür, dass der Pfad frei von Hindernissen ist, bevor Sie die Pulse-Anweisung verwenden.

Im Gegensatz zu Jump bewegt **Pulse** alle Achsen gleichzeitig, einschließlich der Hub-und Fallbewegung der Z-Achse beim Verfahren zur Zielposition. Daher ist es sehr wichtig, dass Sie bei der Verwendung von **Pulse** darauf achten, dass sich der Arm entlang eines hindernisfreien Pfades bewegen kann.

Mögliche Fehler

Der Pulse-Wert ist höher als der Grenzwert:

Wenn der mit dem Pulse-Befehl spezifizierte Pulse-Wert über dem durch den Range-Befehl eingestellten Grenzwert liegt, tritt ein Fehler auf.

Verwandte Befehle

Go, Accel, Range, Speed, Pls, Pulse-Funktion

Beispiel einer Pulse-Anweisung

Im Folgenden sehen Sie Beispiele vom Befehlseingabefenster:

Dieses Beispiel bewegt den Roboteram zu der Position, die durch jeden Achsen-Pulse-Wert definiert wird.

```
> pulse 16000, 10000, -100, 10
```

Dieses Beispiel zeigt die Pulsnummern von der ersten bis zur vierten Achse der aktuellen Roboterarmposition.

```
> pulse
PULSE:  1:  27306 pls 2:  11378 pls 3:  -3072 pls 4:   1297 pls
>
```

Pulse-Funktion

F

Gibt einen Roboterpunkt aus, dessen Koordinaten in Pulsen für jede Achse spezifiziert sind.

Syntax

Pulse (*J1, J2, J3, J4 ,* [*J5*] *,* [*J6*])

Parameter

J1, J2, J3, J4	Der Pulse-Wert für die Achsen 1 bis 4. Der Pulse-Wert muss in dem durch den Range-Befehl definierten Bereich liegen und sollte ein Integer- oder ein Long-Ausdruck sein.
J5, J6	Optional. Für die Verwendung bei 6-Achsrobotern.

Rückgabewerte

Ein Roboterpunkt, der die angegebenen Pulse-Werte verwendet.

Verwandte Befehle

Go, JA, Jump, Move, Pulse-Anweisung, XY

Beispiel einer Pulse-Funktion

```
Jump Pulse(1000, 2000, 0, 0)
```

QP-Anweisung

Schaltet den Quick-Pause-Modus ein oder aus und zeigt den aktuellen Modusstatus an. **>** **S**

Syntax

(1) QP { **On** | **Off** }
(2) QP

Parameter

On | **Off**	Quick Pause kann entweder ein- oder ausgeschaltet sein.

Rückgabewerte

Zeigt die aktuelle **QP**-Moduseinstellung an, wenn der Parameter weggelassen wird.

Beschreibung

Wenn während der Ausführung eines Bewegungsbefehls entweder die Pausetaste gedrückt wird oder ein Pause-Signal an die Steuerung gesendet wird, bestimmt der Quick-Pause-Modus, ob der Roboter sofort angehalten wird oder ob er anhält, nachdem der Bewegungsbefehl durchgeführt wurde.

Sofortiges Verzögern und Anhalten wird "Quick Pause" genannt.

Wenn der Parameter On angegeben wurde, schaltet **QP** den Quick-Pause-Modus ein.
Wenn der Parameter Off angegeben wurde, schaltet **QP** den Quick-Pause-Modus aus.

QP zeigt die aktuelle Einstellung an, die entscheidet, ob der Roboterarm auf die Pause-Eingabe reagieren soll, indem er sofort anhält oder nachdem der aktuelle Arbeitsgang des Arms abgeschlossen ist. **QP** ist ein Status-Befehl, um anzuzeigen, ob der Quick-Pause-Modus ein- (On) oder ausgeschaltet (Off) ist.

Hinweise

Der Quick-Pause-Modus wird vorgabemäßig auf On gestellt, wenn der Strom eingeschaltet wird.

Der Quick-Pause-Modus, der durch den **QP**-Befehl eingestellt wurde, bleibt nach dem Reset-Befehl bestehen. Wenn jedoch die Stromversorgung des PCs oder der Drive Unit aus- und dann wieder eingeschaltet wird, wird der Quick-Pause-Modus auf die Vorgabeeinstellung On zurückgesetzt.

QP und der Eingang der Sicherheitsabschrankung:

Auch wenn der **QP**-Modus auf Off geschaltet ist, hält der Roboter sofort an, wenn sich die Sicherheitsabschrankung öffnet.

Verwandte Befehle

Pause

Beispiel einer QP-Anweisung

Dieses Beispiel aus dem Befehlseingabefenster zeigt die Einstellung, die dem Roboterarm sagt, ob er sofort anhalten soll, wenn Pause eingegeben wird (d. h. der QP-Modus ist ein- oder ausgeschaltet).

```
> qp
QP ON

> qp on  'Setzt QP in den Quick-Pause-Modus
>
```

Quit-Anweisung

S

Bricht die Ausführung eines bestimmten oder aller Tasks ab.

Syntax

Quit { *taskIdentifier* | **All** }

Parameter

taskIdentifier	Taskname oder Integer-Ausdruck, der für die Tasknummer steht. Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, liegt der Bereich für normale Tasks zwischen 1 und 16 und für Traptasks zwischen 257 und 261.
All	Bestimmt, dass alle Tasks abgebrochen werden sollen.

Beschreibung

Quit stoppt die Tasks, die gerade ausgeführt werden oder die vorübergehend mit Halt unterbrochen wurden.

Verwandte Befehle

Exit-Funktion, Halt, Resume, Xqt

Beispiel einer Quit-Anweisung

Dieses Beispiel zeigt zwei Tasks, die nach 10 Sekunden abgebrochen werden.

```
Function main
  Xqt winc1   'Startet die Funktion winc1
  Xqt winc2   'Startet die Funktion winc2
  Wait 10
  Quit winc1  'Bricht den Task winc1 ab
  Quit winc2  'Bricht den Task winc2 ab
Fend

Function winc1
  Do
     On 1; Wait 0.2
     Off 1; Wait 0.2
  Loop
Fend

Function winc2
  Do
     On 2; Wait 0.5
     Off 2; Wait 0.5
  Loop
Fend
```

RadToDeg-Funktion

> F

Konvertiert Radianten in Grad.

Syntax

RadToDeg(*radians***)**

Parameter

radians Reeller Ausdruck, der die Radianten darstellt, die in Grad konvertiert werden sollen.

Rückgabewerte

Ein Double-Wert in Grad.

Verwandte Befehle

ATan, ATan2, DegToRad-Funktion

Beispiel einer RadToDeg-Funktion

```
s = Cos(RadToDeg(x))
```

Randomize-Anweisung

Initialisiert den Zufallszahlengenerator.

S

Syntax

(1) Randomize *seedValue*
(2) Randomize

Parameter

seedValue **Spezifiziert einen reellen Wert (0 oder höher), der die Basis für die Ausgabe einer Zufallszahl darstellt.**

Verwandte Befehle

Rnd-Funktion

Beispiel einer Randomize-Anweisung

```
Function main
    Real r
    Randomize
    Integer randNum

    randNum = Int(Rnd(10)) + 1
    Print "Random number is:", randNum
Fend
```

Range-Anweisung

Spezifiziert die Bewegungsgrenzen für jede der Servoachsen und zeigt sie an. > S

Syntax

(1) Range *j1Min, j1Max, j2Min, j2Max, j3Min, j3Max, j4Min, j4Max,* [*j5Min*, *j5Max, j6Min, j6Max*]
(2) Range

Parameter

j1Min	Die Untergrenze für die 1. Achse, angegeben in Pulsen.
j1Max	Die Obergrenze für die 1. Achse, angegeben in Pulsen.
j2Min	Die Untergrenze für die 2. Achse, angegeben in Pulsen.
j2Max	Die Obergrenze für die 2. Achse, angegeben in Pulsen.
j3Min	Die Untergrenze für die 3. Achse, angegeben in Pulsen.
j3Max	Die Obergrenze für die 3. Achse, angegeben in Pulsen.
j4Min	Die Untergrenze für die 4. Achse, angegeben in Pulsen.
j4Max	Die Obergrenze für die 4. Achse, angegeben in Pulsen.
j5Min	Optional bei 6-Achsrobotern. Die Untergrenze für die 5. Achse, angegeben in Pulsen.
j5Max	Optional bei 6-Achsrobotern. Die Obergrenze für die 5. Achse, angegeben in Pulsen.
j6Min	Optional bei 6-Achsrobotern. Die Untergrenze für die 6. Achse, angegeben in Pulsen.
j6Max	Optional bei 6-Achsrobotern. Die Obergrenze für die 6. Achse, angegeben in Pulsen.

Rückgabewerte

Zeigt den aktuellen **Range**-Wert an, wenn **Range** ohne Parameter eingegeben wurde.

Beschreibung

Range bestimmt die Unter- und Obergrenzen für jede Motorenachse, angegeben in Pulsen. Diese Achsengrenzen werden in Pulseinheiten definiert. Dies ermöglicht es dem Benutzer, einen maximalen und einen minimalen Achsen-Arbeitsbereich für jede einzelne Achse zu definieren. XY-Koordinaten-Grenzwerte können auch mit Hilfe des Befehls XYLim eingestellt werden.

Die **Range**-Ausgangswerte sind für jeden Roboter unterschiedlich. Die Werte, die durch diesen Befehl definiert werden, bleiben auch erhalten, nachdem die Stromversorgung ausgeschaltet wurde.

Wenn die Parameter weggelassen werden, werden die aktuellen **Range**-Werte angezeigt.

Mögliche Fehler

Der Versuch, den zulässigen Bereich zu verlassen:

Wenn der Roboterarm versucht, sich über eine der Arbeitsbereichsgrenzen hinaus zu bewegen, tritt ein Fehler auf.

Die Achse verfährt nicht:

Wenn der Pulse-Wert der Untergrenze dem Pulse-Wert der Obergrenze entspricht oder größer als dieser ist, bewegt sich die Achse nicht.

Verwandte Befehle

JRange, SysConfig, XYLim

Beispiel einer Range-Anweisung

In diesem einfachen Beispiel aus dem Befehlseingabefenster werden die aktuellen Range-Einstellungen angezeigt und anschließend geändert.

```
> range
-18205, 182045, -82489, 82489, -36864, 0, -46695, 46695
>
> range 0, 32000, 0, 32224, -10000, 0, -40000, 40000
>
```

Read-Anweisung

Liest Zeichen aus einem Kommunikationsport.

S

Syntax

Read #*portNumber*, *stringVar*$, *count*

Parameter

portNumber	Kommunikationsport, aus dem gelesen werden soll.
stringVar$	Name einer Zeichenkettenvariablen, die die Buchstabenzeichenkette empfängt.
count	Maximale Anzahl der zu lesenden Bytes.

Verwandte Befehle

ChkCom, ChkNet, OpenCom, OpenNet, Write

Beispiel einer Read-Anweisung

```
Integer numOfChars
String data$

numOfChars = ChkCom(1)

If numOfChars > 0 Then
    Read #1, data$, numOfChars
EndIf
```

ReadBin-Anweisung

Liest binäre Daten aus einem Kommunikationsport.

S

Syntax

ReadBin #*portNumber*, *var*
ReadBin #*portNumber, array(), count*

Parameter

portNumber	Kommunikationsport, aus dem gelesen werden soll.
var	Name eines Bytes, Integers oder einer Long-Variablen, das bzw. die das Datenbyte empfängt.
array()	Name eines Bytes, Integers oder einer Long-Feldvariablen, das bzw. die das Datenbyte empfängt. Spezifiziert eine eindimensionale Feldvariable.
count	Spezifiziert die Anzahl der zu lesenden Bytes. Der angegebene Zählstand muss kleiner oder gleich der Anzahl der Matrixelemente sein.

Verwandte Befehle

Write, WriteBin

Beispiel einer ReadBin-Anweisung

```
Integer data
Integer dataArray(10)

numOfChars = ChkCom(1)

If numOfChars > 0 Then
    ReadBin #1, data
EndIf

NumOfChars = ChkCom(1)
 If numOfChars > 10 Then
    ReadBin #1, dataArray(), 10
EndIf
```

Real-Anweisung

Deklariert Real-Variablen (8 Bytes reelle Zahl). **S**

Syntax

Real *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]**...**]

Parameter

varName	Variablenname, den der Benutzer als **Real-**Typ deklarieren möchte.
subscripts	Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus: (ubound1, [ubound2], [ubound3]) Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension. Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert. Die Gesamtzahl der Elemente der Matrixelemente für lokale und Global Preserve-Variablen ist 1000. Die Gesamtzahl der Matrixelemente für globale und Modulvariablen ist 10000. Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)

Beschreibung

Real wird verwendet, um Variablen als **Real**-Variablen zu deklarieren. Lokale Variablen sollten am Anfang einer Funktion deklariert sein. Globale und Modulvariabeln müssen außerhalb von Funktionen deklariert werden.

Die Anzahl der gültigen Zahlzeichen bei Real ist sechs.

Verwandte Befehle

Boolean, Byte, Double, Global, Integer, Long, String

Beispiel einer Real-Anweisung

Das folgende Beispiel zeigt ein einfaches Programm, das unter Verwendung von **Real** einige Variablen deklariert.

```
Function realtest
  Real var1
  Real A(10)         'Eindimensionale Matrix aus Reals
  Real B(10, 10)     'Zweidimensionale Matrix aus Reals
  Real C(10, 10, 10) 'Dreidimensionale Matrix aus Reals
  Real arrayVar(10)
  Integer i
  Print "Please enter a Real Number:"
  Input var1
  Print "The Real variable var1 = ", var1
  For i = 1 To 5
    Print "Please enter a Real Number:"
    Input arrayVar(i)
    Print "Value Entered was ", arrayVar(i)
  Next i
Fend
```

RealPls-Funktion

F

Syntax

RealPls*(jointNumber)*

Parameter

jointNumber Die spezifische Achse, für die der aktuelle Pulse-Zählstand ermittelt werden soll.

Rückgabewerte

Gibt einen Integer-Wert aus, der den aktuellen Encoder-Pulse-Zählstand für die durch *jointNumber* spezifizierte Achse ausgibt.

Beschreibung

RealPls wird verwendet, um die aktuelle Encoderposition (oder Pulse-Zählstand) jeder Achse zu lesen. Diese Werte können gespeichert und später mit dem Pulse-Befehl verwendet werden.

Verwandte Befehle

CX, CY, CZ, CU, CV, CW, Pulse

Beispiel einer RealPls-Funktion

```
Function DisplayPulses

  Long joint1Pulses

  joint1Pulses = RealPls(1)
  Print "Joint 1 Current Pulse Value: ", joint1Pulses
Fend
```

RealPos-Funktion

F

Gibt die aktuelle Position des spezifizierten Roboters aus.

Syntax

RealPos

Rückgabewerte

Ein Roboterpunkt, der die aktuelle Position des angegebenen Roboters spezifiziert.

Beschreibung

RealPos wird verwendet, um die aktuelle Position des Roboters zu lesen.

Verwandte Befehle

CurPos, CX, CY, CZ, CU, CV, CW, RealPls

Beispiel einer RealPos-Funktion

```
Function ShowRealPos

  Print RealPos
Fend

P1 = RealPos
```

Redim-Anweisung

Redimensioniert eine Matrix während der Laufzeit.

S

Syntax

Redim [**Preserve**] *arrayName* **(***subscripts***)**

Parameter

Preserve	Optional. Spezifiziert, um den vorherigen Inhalt der Matrix zu erhalten. Wird die Angabe weggelassen, wird der Inhalt der Matrix gelöscht.
arrayName	Name der Feldvariablen; folgt den Standardkonventionen für Variablenbenennung. Die Feldvariable muss bereits deklariert worden sein.
subscripts	Neue Dimensionen der Feldvariablen. Sie müssen dieselbe Anzahl von Dimensionen wie bei der Deklaration der Variablen angeben. Die Syntax sieht folgendermaßen aus **(***dim1,* [*dim2*], [*dim3*]**)** *dim1*, *dim2*, *dim3* kann ein IntegerAusdruck zwischen 0 und 2147483646 sein.
subscripts	Optional. Es können neue Dimensionen einer Feldvariable deklariert werden. Sie müssen dieselbe Anzahl von Dimensionen wie bei der Deklaration der Variable angeben. Die Indexsyntax sieht folgendermaßen aus: (ubound1, [ubound2], [ubound3]) Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension. Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert. Die Gesamtzahl der Matrixelemente für lokale und Global Preserve-Variablen beträgt 100 bei Zeichenketten und 1000 bei allen anderen Typen. Die Gesamtzahl der Matrixelemente für globale und Modulvariablen beträgt 1000 bei Zeichenketten und 10000 bei allen anderen Typen. Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)

Beschreibung

Verwenden Sie Redim, um die Dimensionen einer Matrix während der Laufzeit zu ändern. Verwenden Sie Preserve, um die vorherigen Werte beizubehalten.
Die Feldvariable, die durch Byref deklariert wurde, kann Redim nicht verwenden.

Verwandte Befehle

UBound

Beispiel einer Redim-Anweisung

```
Integer i, numParts, a(0)

Print "Enter number of parts "
Input numParts

Redim a(numParts)

For i=0 to UBound(a)
    a(i) = i
Next

' Redimensioniert die Matrix mit 20 weiteren Elementen
Redim Preserve a(numParts + 20)

' Die Werte des ersten Elementes werden beibehalten
For i = 0 to UBound(a)
    Print a(i)
Next
```

Reset-Anweisung

Setzt die Steuerung auf einen initialisierten Status zurück. > S

Syntax

Reset

Beschreibung

Reset setzt die folgenden Parameter zurück:

Not-Aus-Status
Fehlerstatus
Ausgänge (alle Ausgänge außer E/As, die dem Remote-Ausgang zugeordnet werden, werden auf Off gesetzt; der Benutzer kann dieses Feature in EPSON RC+ deaktivieren)
Speed, SpeedR und SpeedS für den aktuellen Roboter (werden auf den Vorgabewert zurückgesetzt)
Accel, AccelR und AccelS für den aktuellen Roboter (werden auf den Vorgabewert zurückgesetzt)
LimZ-Paramter des aktuellen Roboters (wird auf 0 zurückgesetzt)
Fine des aktuellen Roboters (wird auf den Vorgabewert zurückgesetzt)
Power Low für den aktuellen Roboter (der Low-Power-Modus wird eingeschaltet)
PTPBoost des aktuellen Roboters (wird auf den Vorgabewert zurückgesetzt)

Für servobedingte Fehler, den Not-Aus-Status und alle anderen Bedingungen, die zurückgesetzt werden müssen, wird kein Befehl außer Reset akzeptiert. In diesem Fall führen Sie erst Reset und anschließend andere Prozesse, wie erforderlich, aus.

Nach einem Not-Aus sollten Sie z. B. zunächst überprüfen, dass Betriebsbedingungen den Sicherheitsvorschriften entsprechen, dann Reset ausführen und anschließend den Befehl Motor On.

Der Kritischer Fehler-Status wird durch **Reset** nicht abgebrochen.
Wenn ein kritischer Fehler auftritt, schalten Sie die Steuerung aus und beheben Sie die Fehlerursache.

Hinweise

Optionsschalter Reset

Wenn in der Steuerung die Voreinstellung “Reset schaltet die Ausgänge aus” (Reset turns off outputs) aktiviert ist, werden alle Ausgänge ausgeschaltet, wenn der **Reset**-Befehl ausgeführt wird. Es ist wichtig, dass Sie dies bei der Verdrahtung des Systems bedenken, damit das Ausschalten von Ausgängen nicht zum Herunterfallen von Werkzeugen oder ähnlichen Situation führt. Weitere Informationen finden Sie unter Einstellungen | Steuerung | Voreinstellungen im Benutzerhandbuch.

Verwandte Befehle

Accel, AccelS, Fine, LimZ, Motor, Off, On, PTPBoost, SFree, SLock, Speed, SpeedS

Beispiel einer Reset-Anweisung

Dieses Beispiel zeigt den Reset-Befehl, vom Befehlseingabefenster aus ausgegeben.

```
>reset
>
```

Resume-Anweisung

S

Setzt einen Task fort, der durch den Halt-Befehl unterbrochen wurde.

Syntax

Resume { *taskIdentifier* | ***All*** }

Parameter

taskIdentifier	Taskname oder Integer-Ausdruck, der für die Tasknummer steht. Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, umfasst der Bereich für normale Tasks 1 bis 16 und für Traptasks 257 bis 261.
All	Bestimmt, dass alle Tasks fortgesetzt werden sollen.

Beschreibung

Die **Resume**-Anweisung setzt die Ausführung der Tasks fort, die durch den Halt-Befehl unterbrochen wurden.

Verwandte Befehle

Halt, Quit, Xqt

Beispiel einer Resume-Anweisung

Dieses Beispiel zeigt die Verwendung des Resume-Befehls nach dem Halt-Befehl.

```
Function main
   Xqt 2, flicker   'Führt Flicker als Task 2 aus

   Do
      Wait 3         'Erlaubt, Flicker 3 Sekunden lang auszuführen
      Halt flicker   'Hält den Flicker-Task an
      Wait 3
      Resume flicker 'Setzt den Flicker-Task fort
   Loop
Fend

Function flicker
   Do
      On 1
      Wait 0.2
      Off 1
      Wait 0.2
   Loop
Fend
```

Return-Anweisung

S

Die **Return**-Anweisung wird zusammen mit der GoSub-Anweisung verwendet. **GoSub** übererträgt die Programmsteuerung an ein Unterprogramm. Sobald das Unterprogramm vollständig ausgeführt ist, veranlasst **Return,** dass die Programmausführung in der Zeile fortgesetzt wird, die auf den **GoSub**-Befehl folgt, der das Unterprogramm initiiert hatte.

Syntax

Return

Beschreibung

Die **Return**-Anweisung wird zusammen mit der GoSub-Anweisung verwendet. Der Hauptzweck der **Return**-Anweisung ist die Rückgabe der Programmsteuerung an den Befehl, der auf den GoSub-Befehl folgt, der das Unterprogramm ursprünglich ausgelöst hatte.

Der GoSub**-**Befehl bewirkt, dass die Programmsteuerung mit einer durch den Anwender definierten Anweisungs-Zeilennummer oder mit einem gleichermaßen definierten Label verzweigt wird. Das Programm führt dann die Anweisung in dieser Zeile aus und fährt mit der Ausführung in den darauf folgenden Zeilennummern fort, bis es auf einen **Return**-Befehl trifft. Der **Return**-Befehl veranlasst die Programmsteuerung dann, zu der Zeile zurück zu wechseln, die direkt auf diejenige folgt, die den GoSub-Befehl ursprünglich initiiert hat. (D. h., der **GoSub**-Befehl veranlasst die Ausführung eines Unterprogramms, wobei die Ausführung anschließend zu der Anweisung zurückkehrt, die auf den **GoSub**-Befehl folgt.)

Mögliche Fehler

Return-Befehl ohne GoSub-Befehl gefunden

Ein **Return**-Befehl wird verwendet, um aus einem Unterprogramm in das Ursprungsprogramm zurück zu wechseln, von dem aus der GoSub-Befehl ausgegeben wurde. Wenn ein **Return**-Befehl gefunden wird, dem kein GoSub-Befehl vorangegangen ist, wird ein Fehler ausgegeben. Ein allein stehender **Return**-Befehl ist bedeutungslos, da das System nicht weiß, wohin es zurückkehren soll.

Verwandte Befehle

OnErr, GoSub**,** GoTo

Beispiel einer Return-Anweisung

Das folgende Beispiel zeigt eine einfache Funktion, die einen GoSub-Befehl verwendet, um zu einem Label namens checkio zu verzweigen und die 16 ersten Anwender-Eingänge zu überprüfen. Danach kehrt das Unterprogramm zum Hauptprogramm zurück.

```
Function main
     Integer var1, var2
     GoSub checkio
     On 1
     On 2
     Exit Function

checkio:  'Hier startet das Unterprogramm
     var1 = In(0)
     var2 = In(1)
     If var1 <> 0 Or var2 <> 0 Then
          Print "Message to Operator here"
     EndIf
finished:
     Return  'Das Unterprogramm endet hier und kehrt zur Zeile 40 zurück
Fend
```

Right$-Funktion

F

Gibt eine Teilkette der ganz rechts stehenden Zeichen einer Zeichenkette aus.

Syntax

Right$*(string, count)*

Parameter

string	Zeichenkettenvariable oder Buchstabenzeichenkette mit bis zu 255 Zeichen, aus der die am weitesten rechts stehenden Zeichen kopiert werden.
count	Anzahl der zu kopierenden Zeichen aus *string,* beginnend mit dem Zeichen, das ganz rechts steht.

Rückgabewerte

Gibt eine Zeichenkette der ganz rechts stehenden *count*-Zeichen aus der vom Benutzer definierten Buchstabenzeichenkette aus.

Beschreibung

Right$** gibt die ganz rechts stehenden *count*-Zeichen einer vom Benutzer definierten Zeichenkette aus. **Right$ kann so viele Zeichen ausgeben, wie in der Buchstabenzeichenkette vorhanden sind.

Verwandte Befehle

Asc, Chr$, InStr, Left$, Len, Mid$, Space$, Str$, Val

Beispiel einer Right$-Funktion

Das Beispiel unten zeigt ein Programm, das eine teilweise Datenzeichenkette als Eingang verwendet und in Teilnummer, Teilnamen und Teilzählerstand aufteilt.

```
Function SplitPartData(DataIn$ As String, ByRef PartNum$ As String,
ByRef PartName$ As String, ByRef PartCount As Integer)

    PartNum$ = Left$(DataIn$, 10)

    DataIn$ = Right$(datain$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")

    PartName$ = Mid$(DataIn$, 11, 10)

    PartCount = Val(Right$(dataIn$, 5))

Fend


Weitere Beispielergebnisse des Right$-Befehls vom Befehlseingabefenster.
> Print Right$("ABCDEFG", 2)
 FG

> Print Right$("ABC", 3)
 ABC
```

Rnd-Funktion

F

Gibt eine Zufallszahl aus.

Syntax

Rnd(*maxValue*)

Parameter

maxValue Reeller Ausdruck, der für den maximalen Rückgabewert steht.

Rückgabewerte

Reelle Zufallszahl von 0 bis *range*.

Beschreibung

Verwenden Sie Rnd, um Zufallszahlenwerte zu generieren.

Verwandte Befehle

Int, Randomize

Beispiel einer Rnd-Funktion

Im folgenden Beispiel wird eine Zufallszahl zwischen 1 und 10 generiert.

```
Function main
    Real r
    Integer randNum

    Randomize
    randNum = Int(Rnd(9)) + 1
    Print "Random number is:", randNum
Fend
```

RobotInfo-Funktion

F

Gibt Statusinformationen für den Roboter aus.

Syntax

RobotInfo(*index*)

Parameter

index Integer-Ausdruck, der für den Index der auszugebenden Daten steht.

Rückgabewerte

Die spezifizierte Information wird als Integer ausgegeben.

Beschreibung

Die Information für jedes Bit des ausgegebenen Wertes wird in der folgenden Tabelle angezeigt:

Index	Bit	Wert	Beschreibung
0	0	&H1	Nicht definiert
	1	&H2	Ein zurücksetzbarer Fehler ist aufgetreten
	2	&H4	Ein nicht zurücksetzbarer Fehler ist aufgetreten
	3	&H8	Motoren sind eingeschaltet
	4	&H10	Die aktuelle Power-Einstellung ist High
	5	&H20	Nicht definiert
	6	&H40	Nicht definiert
	7	&H80	Nicht definiert
	8	&H100	Roboter ist angehalten
	9	&H200	Roboter ist nicht angehalten (führt eine Bewegung aus oder befindet sich im Quick-Pause-Modus)
	10	&H400	Roboter wurde durch Pause oder Sicherheitsabschrankung gestoppt
	11		Nicht definiert
	12		Nicht definiert
	13		Nicht definiert
	14	&H4000	TILL-Bedingung wurde durch vorangehenden Bewegungsbefehl erfüllt
	15	&H8000	SENSE-Bedingung wurde durch vorangehenden Bewegungsbefehl erfüllt
	16-31		Nicht definiert
1	0-31		Nicht definiert
2	0	&H1	Roboter befindet sich in seiner Home-Position
	1-31		Nicht definiert
3	0	&H1	Servo der 1. Achse ist aktiviert
	1	&H2	Servo der 2. Achse ist aktiviert
	2	&H4	Servo der 3. Achse ist aktiviert
	3	&H8	Servo der 4. Achse ist aktiviert
	4	&H10	Servo der 5. Achse ist aktiviert
	5	&H20	Servo der 6. Achse ist aktiviert
	6-31		Nicht definiert

4	N/A	0 - 16 -1	Nummer der Tasks, die Roboterbefehle ausführen 0 = Befehl, der vom Befehlseingabefenster oder Makro ausgeführt wird -1 = kein Task verwendet den Manipulator
5	0	&H1	Bremse der 1. Achse ist eingeschaltet
	1	&H2	Bremse der 2. Achse ist eingeschaltet
	2	&H4	Bremse der 3. Achse ist eingeschaltet
	3	&H8	Bremse der 4. Achse ist eingeschaltet
	4	&H10	Bremse der 5. Achse ist eingeschaltet
	5	&H20	Bremse der 6. Achse ist eingeschaltet
	6-31		Nicht definiert

Verwandte Befehle

CtrlInfo, RobotInfo$, TaskInfo

Beispiel einer RobotInfo-Funktion

```
If (RobotInfo(3) And &H1) = &H1 Then
  Print "Joint 1 is locked"
Else
  Print "Joint 1 is free"
EndIf
```

RobotInfo$-Funktion

F

Gibt Textinformationen für den Roboter aus.

Syntax

RobotInfo$(*index*)

Parameter

index Integer-Ausdruck, der für den Index der auszugebenden Daten steht.

Rückgabewerte

Eine Zeichenkette, die die spezifizierten Informationen enthält.

Beschreibung

Index	Beschreibung
0	Bezeichnung des Roboters
1	Bezeichnung des Modells
2	Nicht definiert
3	Nicht definiert
4	Seriennummer des Roboters

Verwandte Befehle

CtrlInfo, RobotInfo, TaskInfo

Beispiel einer RobotInfo$-Funktion

```
Print "Robot Name: ", RobotInfo$(0)
```

RobotModel$-Funktion

F

Gibt die Bezeichnung des Robotermodells aus.

Syntax

RobotModel$

Rückgabewerte

Eine Zeichenkette, die die Bezeichnung des Modells enthält. Hierbei handelt es sich um die Bezeichnung, die sich auf der Rückseite des Roboters befindet.

Verwandte Befehle

RobotType

Beispiel einer RobotModel$-Funktion

```
Print "The robot model is ", RobotModel$
```

RobotName$-Funktion

F

Gibt die Bezeichnung des Roboters aus.

Syntax

RobotName$

Rückgabewerte

Eine Zeichenkette, die die Bezeichnung des Roboters enthält.

Verwandte Befehle

RobotInfo, RobotModel$

Beispiel einer RobotName$-Funktion

```
Print "The robot name is ", RobotName$
```

RobotSerial$-Funktion

F

Gibt die Seriennummer des Roboters aus.

Syntax

RobotSerial$

Rückgabewerte

Eine Zeichenkette, die die Seriennummer des Roboters enthält.

Verwandte Befehle

RobotInfo, RobotName$, RobotModel$

Beispiel einer RobotSerial$-Funktion

```
Print "The robot serial number is ", RobotSerial$
```

RobotType-Funktion

F

Gibt den Robotertyp aus.

Syntax

RobotType

Rückgabewerte

1: JOINT

3: SCARA

5: 6-AXIS

Verwandte Befehle

RobotModel$

Beispiel einer RobotType-Funktion

```
If RobotType = 3 Then
  Print "Robot type is SCARA"
EndIf
```

RSet$-Funktion

F

Gibt die spezifizierte Zeichenkette mit vorangehenden Leerzeichen aus, die bis zur spezifizierten Länge eingefügt werden.

Syntax

RSet$ (*string, length*)

Parameter

string	Zeichenkettenausdruck.
length	Integer-Ausdruck für die Gesamtlänge der ausgegebenen Zeichenkette.

Rückgabewerte

Spezifizierte Zeichenkette mit vorangehenden Leerzeichen.

Verwandte Befehle

LSet$, Space$

Beispiel einer RSet$-Funktion

```
str$ = "123"
str$ = RSet$(str$, 10)  ' str$ = "       123"
```

RShift-Funktion

Verschiebt numerische Daten um eine vom Anwender definierte Bitanzahl nach rechts.

F

Syntax

RShift*(number, shiftBits)*

Parameter

number	Der numerische Ausdruck, der verschoben werden soll.
shiftBits	Die Bitanzahl (Integer zwischen 0 und 31), um die *number* nach rechts verschoben werden soll.

Rückgabewerte

Gibt ein numerisches Ergebnis aus, das dem *number*-Wert entspricht, um den die Bits um die in *shiftBits* definierte Anzahl von Bits nach rechts verschoben wurden.

Beschreibung

RShift verschiebt die spezifizierten numerischen Daten (*number*) um die festgelegte Bitanzahl (*shiftBits*) nach rechts (an eine niedrigere Stelle). Die verschobenen Bits höheren Wertes werden auf 0 gesetzt.

Die einfachste Erklärung für **RShift** ist, dass es das Ergebnis von *number* / $2^{shiftBits}$ ausgibt. (*Number* wird dividiert durch 2 *shiftBit* Male.)

Hinweise

Typ numerischer Daten:

Die numerischen Daten (*number*) können aus jeglichem gültigen numerischen Datentyp bestehen. **RShift** arbeitet mit den Datentypen Byte, Integer und Real.

Verwandte Befehle

And, LShift, Not, Or, Xor

Beispiel einer RShift-Funktion

Das unten gezeigte Beispiel zeigt ein Programm, das alle möglichen **RShift**-Werte für einen Integer-Datentyp aufzeigt, beginnend mit dem auf 0 gesetzten Integer.

```
Function rshiftst
  Integer num, snum, i
  num = 32767
  For i = 1 to 16
     Print "i =", i
     snum = RShift(num, 1)
     Print "RShift(32767, ", i, ") = ", snum
  Next i
Fend
```

Weitere beispielhafte Ergebnisse des **RShift**-Befehls vom Befehlseingabefenster.

```
> Print RShift(10,1)
5
> Print RShift(8,3)
1
> Print RShift(16,2)
4
```

RTrim$-Funktion

F

Gibt eine Zeichenkette aus, die mit der definierten Zeichenkette identisch ist, wobei die nachfolgenden Leerzeichen nicht angeführt werden.

Syntax

RTrim$(*string*)

Parameter

string Zeichenkettenausdruck.

Rückgabewerte

Spezifizierte Zeichenkette, bei der die nachfolgenden Leerzeichen entfernt wurden.

Verwandte Befehle

LTrim$, Trim$

Beispiel einer RTrim$-Funktion

```
str$ = "  data  "
str$ = RTrim$(str$)  ' str$ = "..data"

EndIf
```

SafetyOn-Funktion

F

Gibt den aktuellen Sicherheitsstatus aus.
Diese Funktion wird aus Kompatibilitätsgründen freigehalten.
Alle Tasks werden im Sicherheitsstatus unterbrochen.

SavePoints-Anweisung

Speichert Punktedaten im Hauptspeicher.

> S

Syntax

SavePoints *filename*

Parameter

fileName Zeichenkettenausdruck, der die Datei enthält, in der die Punkte gespeichert werden. Dem angegebenen *fileName* wird die Dateinamenerweiterung .PTS angehängt, so dass keine Dateinamenerweiterung vom Anwender spezifiziert werden muss.

Beschreibung

SavePoints speichert die Punkte in der spezifizierten Datei. Dem angegebenen *fileName* wird die Dateinamenerweiterung .PTS angehängt, so dass keine Dateinamenerweiterung vom Anwender spezifiziert werden muss.

Mögliche Fehler

Ungültiger Dateiname

Wird ein Dateiname eingegeben, dessen Bezeichnung Leerzeichen beinhaltet oder andere ungültige Dateinameneigenschaften aufweist, wird ein Fehler ausgegeben.

Verwandte Befehle

ClearPoints, LoadPoints

Beispiel einer SavePoints-Anweisung

```
ClearPoints
For i = 1 To 10
   P(i) = i, 100, 0, 0
Next i
SavePoints "TEST.PTS"
```

Select...Send

Führt eine von mehreren Anweisungsgruppen je nach Ausdruckswert aus.

S

Syntax

Select *selectExpr*
Case *caseExpr*
statements
[Case *caseExpr*
statements **]**
[Default
statements **]**

Send

Parameter

selectExpr	Ein numerischer Ausdruck oder ein Zeichenkettenausdruck.
caseExpr	Ein numerischer Ausdruck oder ein Zeichenkettenausdruck, der denselben Typ wie *selectExpr* bewertet.
statements	Eine oder mehrere gültige SPEL+-Anweisungen oder Mehrfachanweisungen.

Beschreibung

Wenn ein *caseExpr* einem *selectExpr* entspricht, werden die Anweisungen nach der Case-Anweisung ausgeführt. Nach der Ausführung übergibt die Programmsteuerung an die Anweisung, die auf Send folgt.

Wenn kein *caseExpr* einem *selectExpr* entspricht, werden die Standardanweisungen ausgeführt und die Programmsteuerung übergibt an die Anweisung, die auf Send folgt.

Wenn kein *caseExpr* einem *selectExpr* entspricht und Default weggelassen wird, wird nichts ausgeführt und die Programmsteuerung übergibt sofort an die Anweisung, die auf Send folgt.

selectExpr und *caseExpr* können Konstanten, Variablen und logische Operatoren beinhalten, die And, Or oder Xor verwenden.

Verwandte Befehle

If...Then...Else

Beispiel für Select…Send

Im Folgenden sehen Sie ein einfaches Beispiel für Select...Send:

```
Function Main
  Integer I
  For I = 0 To 10
    Select I
      Case 0
           Off 1;On 2;Jump P1
      Case 3
           On 1;Off 2
           Jump P2;Move P3;On 3
      Case 7
           On 4
      Default
           On 7
    Send
  Next
Fend
```

Sense-Anweisung

> S

Spezifiziert die Eingabebedingung, die, wenn sie erfüllt ist, die gerade in der Ausführung befindliche Bewegung durch Anhalten des Roboters über der Zielposition abschließt und diese anzeigt.

Syntax

Sense [*inputCondition*]

Parameter

inputcondition Diese Bedingung überprüft den Status des Eingangs und muss einen Boolean-Wert ausgeben. Die folgenden Funktionen und Operatoren können für die *inputCondition* verwendet werden:

Funktionen: In, InW, MemIn, MemSw, Sw, Force

Operatoren: And, Or, Xor, +, *

Andere: Klammern, um verschiedenen Operationen und Variablen Priorität einzuräumen.

Beschreibung

Sense wird verwendet, um die Approach-Bewegung während Jump-, Jump3- und Jump3CP-Befehlen anzuhalten. Die **Sense**-Bedingung muss mindestens eine Eingangs- oder Merkereingangsfunktion enthalten.

Wenn Variablen in eine Sense-Bedingung eingebunden sind, werden ihre Werte während der **Sense**-Ausführung berechnet. Mehrfache **Sense**-Anweisungen sind erlaubt. Die zuletzt aufgetretene **Sense**-Bedingung bleibt solange aktuell, bis sie durch eine andere **Sense**-Anweisung ersetzt wird.

Jump, Jump3, Jump3CP mit Sense-Bedingung

Überprüft, ob die aktuelle **Sense**-Bedingung erfüllt ist. Wenn die Bedingung erfüllt ist, wird der Jump-Befehl abgeschlossen, wenn der Roboter über dem Zielpunkt angehalten hat. (D. h., wenn die **Sense**-Bedingung Wahr ist, bleibt der Roboterarm genau über der Zielposition, ohne die Approach-Bewegung auszuführen. Wenn die **Sense**-Bedingung Faslch ist, beendet der Roboterarm die Jump-Befehl-Bewegung bis zur Zielposition.)

Wenn Parameter ausgelassen werden, werden die aktuellen Sense-Definitionen angezeigt.

Hinweise

Sense-Einstellung bei Einschalten des Hauptstroms

Beim Einschalten des Stroms lautet die **Sense**-Bedingung:

```
Sense Sw(0) = On      'Eingang 0 ist eingeschaltet
```

Verwendung von JS und Stat zur Verifizierung von Sense

Verwenden Sie JS oder Stat, um zu verifizieren, ob die Sense-Bedingung erfüllt wurde.

Verwandte Befehle

In, JS, Jump, Jump3, Jump3CP, MemIn, MemSw, Stat, Sw

Beispiel einer Sense-Anweisung

Dies ist ein einfaches Beispiel für die Verwendung der Sense-Anweisung.

```
Function test
   .
   .
TrySense:
    Sense Sw(1) = 0   'Bestimmt, dass der Arm über
                      'dem Ziel hält, wenn
                      'der Eingang 1 auf 0 gestellt ist
    Jump P1 C2 Sense
    'Wenn der Arm am durch Js(0)=1 definierten Punkt stehen bleibt,
    'wird ERRPRC ausgeführt und das Programm springt zu TrySense
    If Js(0) = 1 Then
         GoSub ERRPRC
         GoTo TrySense
    EndIf
    On 1; Wait 0.2; Off 1
   .
   .
Fend
```

Beispiele mit anderer Syntax

```
> Sense Sw(1)=1 And Sw($1)=1

> Sense Sw(0) Or Sw(1) And Sw($1)
```

SetCom-Anweisung

Definiert Anzeigeparameter für einen RS-232C-Port oder zeigt diese an.

S

Syntax

SetCom #*portNumber,* [*baud*], [*dataBits],* [*stopBits] ,* [*parity] ,* [*terminator] ,* [*HWFlow],* [*SWFlow] ,* [*timeOut*]

Parameter

portNumber	Gibt an, für welchen RS232-Port Parameter eingestellt werden sollen. Gültige Werte sind 1-16.
baud	Optional. Gibt die Baudrate an. Gültige Einträge sind: 110 2400 19200 300 4800 38400 600 9600 56000 1200 14400 115200 (Vorgabeeinstellung: 9600)
dataBits	Optional. Bestimmt die Anzahl der Datenbits pro Zeichen. Gültige Werte sind **7** und **8**.
stopBits	Optional. Bestimmt die Anzahl der Stopbits pro Zeichen. Gültige Werte sind **1** und **2**.
parity	Optional. Bestimmt die Parität. Gültige Werte sind **O** (Odd = ungerade), **E** (Even = gerade) und **N** (None = keine).
terminator	Optional. Gibt die Zeilenabschluss-Zeichen an. Gültige Werte sind **CR**, **LF**, **CRLF**.
HWFlow	Optional. Bestimmt die Hardwaresteuerung. Gültige Werte sind **RTS** und **NONE**.
SWFlow	Optional. Bestimmt die Softwaresteuerung. Gültige Werte sind **XON** und **NONE**.
timeOut	Optional. Gibt die die maximale Zeit für die Übermittlung oder den Empfang an, angegeben in Sekunden. Wenn der Wert 0 ist, gibt es keinen Zeitüberlauf.

Beschreibung

Wenn alle Parameter weggelassen werden, werden die Einstellungen des Kommunikationsports angezeigt.

Verwandte Befehle

OpenCom, CloseCom, SetNet

Beispiel einer SetCom-Anweisung

```
SetCom #1, 9600, 8, 1, N, CRLF, NONE, NONE, 0
SetCom #2, 4800
```

SetIn-Anweisung

> S

Bei virtuellen E/As wird ein spezifizierter Eingangsport (8 Bits) für den spezifizierten Wert eingestellt.

Syntax

SetIn *portNumber*, *value*

Parameter

portNumber Integer-Ausdruck, der für die Eingangsportnummer steht.

value Integer-Ausdruck zwischen 0 und 255 für den spezifizierten Port.

Beschreibung

SetIn ermöglicht die gleichzeitige Einrichtung von 8 virtuellen Eingängen.

Verwandte Befehle

SetSW, SetInW

Beispiel einer SetIn-Funktion

```
> setin 0, 1  ' Stellt das erste Bit von Port 0 auf On
```

SetInW-Anweisung

> S

Bei virtuellen E/As wird ein spezifiziertes Eingangswort (16 Bits) für den spezifizierten Wert eingestellt.

Syntax

SetInW *portNumber*, *value*

Parameter

portNumber Integer-Ausdruck, der für die Eingangsportnummer steht.

value Zahl zwischen 0 und 65535 für das spezifizierte Wort.

Beschreibung

SetInW ermöglicht die gleichzeitige Einrichtung von 16 virtuellen Eingängen.

Verwandte Befehle

SetSw, SetIn

Beispiel einer SetInW-Funktion

```
> setinw 0, 1  ' Stellt das erste Bit von Port 0 auf On
```

SetNet-Anweisung

S

Stellt die Parameter für einen TCP / IP-Port ein.

Syntax

SetNet #*portNumber, hostAddress, TCP_IP_PortNum, terminator, SWFlow, timeout*

Parameter

portNumber	Gibt an, für welchen Port Parameter eingestellt werden sollen. Gültige Werte sind 201-208.
hostAddress	Gibt die Host IP-Adresse an.
TCP_IP_PortNum	Gibt die TCP / IP-Portnummer für diesen Knoten an.
terminator	Gibt die Zeilenabschluss-Zeichen an. Gültige Werte sind **CR**, **LF**, **CRLF**.
SWFlow	Bestimmt die Softwaresteuerung. Der gültige Wert ist **NONE**.
timeOut	Gibt die die maximale Zeit für die Übermittlung oder den Empfang an, angegeben in Sekunden. Wenn der Wert 0 ist, gibt es keinen Zeitüberlauf.

Verwandte Befehle

OpenNet, CloseNet, SetCom

Beispiel einer SetNet-Anweisung

```
SetNet #201, "192.168.0.1", 2001, CRLF, NONE, 0
```

SetSw-Anweisung

> S

Bei virtuellen E/As wird ein spezifiziertes Eingangsbit für den spezifizierten Wert eingestellt.

Syntax

SetSw *bitNumber*, *value*

Parameter

bitNumber	Integer-Ausdruck, der für die Eingangsbitnummer steht.
value	Integer-Ausdruck mit einem Wert von 0 (Aus) oder 1 (Ein).

Beschreibung

SetSw ermöglicht das Ein- oder Ausschalten eines Eingangsbits.

Verwandte Befehle

SetIn, SetInW

Beispiel einer SetSw-Funktion

```
> setsw 2, on  ' Stellt das zweite Eingangsbit auf On
```

SFree-Anweisung

Schaltet die angegebene Servoachse frei.

> S

Syntax

SFree *jointNumber* [*, jointNumber,...*]

Parameter

jointNumber Ein Integer-Ausdruck, der für eine Servoachsennummer steht.

Beschreibung

SFree schaltet die angegebenen Servoachsen frei. Dieser Befehl wird für das direkte Teachen oder die Teil-Installation verwendet, indem eine bestimmte Achse teilweise abgeschaltet wird. Um eine Achse wieder in Betrieb zu nehmen, führen Sie den SLock-Befehl oder MotorOn aus.

Hinweise

SFree setzt einige Systemelemente in ihren Anfangsstatus zurück:

SFree setzt aus Sicherheitsgründen Parameter zurück, die die Roboterarmgeschwindigkeit (Speed und SpeedS), die Beschleunigung (Accel und AccelS) und den LimZ-Parameter betreffen.

Hinweise

SFree und seine Verwendung mit der Z-Achse bei SCARA-Robotern

Die Z-Achse hat elektronische Bremsen. Daher erlaubt es die Einstellung von SFree für die Z-Achse nicht, die Z-Achse sofort zu bewegen. Um die Z-Achse von Hand zu bewegen, muss die Bremse gelöst werden. Betätigen Sie dazu fortlaufend den Bremsfreigabetaster oben auf dem Roboterarm.

SFree und seine Verwendung mit 6-Achsrobotern

Alle Achsen eines 6-Achsroboters verfügen über eine elektromagnetische Bremse. Die Bremse kann mithilfe des Brake-Befehls freigegeben werden, wenn der Motor ausgeschaltet ist. Unter Verwendung von SFree kann keine Achse per Hand bewegt werden.

Ausführung von Bewegungsbefehlen während die Achsen im SFree-Status sind

Der Versuch, einen Bewegungsbefehl während des SFree-Status auszuführen, verursacht einen Fehler im Standardstatus der Steuerung. Um jedoch Bewegungen zu erlauben, solange eine oder mehrere der Achsen im SFree-Status sind, aktivieren Sie die Voreinstellung „Erlaube Bewegung mit einer odere mehreren freigegebenen Achsen“ ("Allow Motion with one or more axes free"). (Diese Voreinstellung kann unter Einstellungen | Steuerung | Voreinstellungen EPSON RC+ 5.0 vorgenommen werden.)

Verwandte Befehle

Brake, LimZ, Motor, SFree Function, SLock

Beispiel einer SFree-Anweisung

Dies ist ein einfaches Beispiel für die Verwendung einer SFree-Anweisung. Die Voreinstellung für Bewegungen mit SFree muss aktiviert sein, damit die Anweisung in diesem Beispiel funktioniert.

```
Function GoPick
   Speed pickSpeed
   'Gibt die Ansteuerung von J1 und J2 frei
   'und steuert die Z- und U-Achsen zur Teilinstallation
   SFree 1, 2
   Go pick
   'Stellt die Ansteuerung von J1 und J2 wieder her
   SLock 1, 2
Fend
```

SFree-Funktion

F

Gibt den SFree-Status für eine angegebene Achse aus.

Syntax

SFree(*jointNumber*)

Parameter

jointNumber Integer-Ausdruck, der für die zu prüfende Achsennummer steht.

Rückgabewerte

Wahr, wenn die Achse freigegeben ist, Falsch, wenn sie nicht freigegeben ist.

Verwandte Befehle

SFree-Anweisung

Beispiel einer SFree-Funktion

```
If SFree(1) Then
    Print "Joint 1 is free"
EndIf
```

Sgn-Funktion

F

Bestimmt das Vorzeichen des Operanden.

Syntax

Sgn(*Operand*)

Parameter

Operand	Ein numerischer Ausdruck.

Rückgabewerte

1: Wenn der Operand einen positiven Wert hat.
0: Wenn der Operand 0 ist.
-1: Wenn der Operand einen negativen Wert hat.

Beschreibung

Die **Sgn**-Funktion bestimmt das Vorzeichen des numerischen Wertes des Operanden.

Verwandte Befehle

Abs, And, Atan, Atan2, Cos, Int, Mod, Or, Not, Sin, Sqr, Str$, Tan, Val, Xor

Beispiel einer Sgn-Funktion

Dies ist ein einfaches Beispiel aus dem Befehlseingabefenster für die Verwendung der Sgn-Funktion.

```
>print sgn(123)
 1
>print sgn(-123)
 -1
>
```

Signal-Anweisung

S

Sendet ein Signal an die Tasks, die WaitSig ausführen.

Syntax

Signal *signalNumber*

Parameter

signalNumber Zu übertragende *signalNumber.* Der Bereich umfasst 0 bis 15.

Beschreibung

Das Signal kann verwendet werden, um Multi-Task-Ausführungen zu synchronisieren.

Signale, die vor der Ausführung von WaitSig ausgegeben wurden, werden ignoriert.

Verwandte Befehle

WaitSig

Beispiel einer Signal-Anweisung

```
Function Main
   Xqt 2, SubTask
   Call InitSys
   Signal 1

Fend

Function SubTask
   WaitSig 1

Fend
```

Sin-Funktion

F

Gibt den Sinus eines numerischen Ausdrucks aus.

Syntax

Sin(*radians***)**

Parameter

radians Reeller Ausdruck in Radianten.

Rückgabewerte

Numerischer Wert, der für den Sinus des numerischen Ausdrucks *radians* steht.

Beschreibung

Sin gibt den Sinus des numerischen Ausdrucks aus. Der numerische Ausdruck (*radians)* muss in Radianteneinheiten angegeben werden. Der von der **Sin-**Funktion ausgegebene Wert liegt zwischen -1 und 1.

Verwenden Sie die RadToDeg-Funktion, um Radiantenwerte in Gradwerte umzuwandeln.

Verwandte Befehle

Abs, Atan, Atan2, Cos, Int, Mod, Not, Sgn, Sqr, Str$, Tan, Val

Beispiel einer Sin-Funktion

Das folgende Beispiel zeigt ein einfaches Programm, das den **Sin**-Befehl nutzt.

```
Function sintest
    Real x
    Print "Please enter a value in radians:"
    Input x
    Print "Sin of ", x, " is ", Sin(x)
Fend
```

SLock-Anweisung

> S

Stellt die Servoleistung aus dem “Servo-free”-Zustand für die spezifizierte Servoachse wieder her.

Syntax

SLock *jointNumber* [*, jointNumber,...*]

Parameter

jointNumber Die Servoachsennummer.

Beschreibung

SLock stellt die Servoleistung für die angegebene Servoachse wieder her, die für das direkte Teachen oder die Teilinstallation durch den SFree-Befehl abgeschaltet war.

Wenn die Achsennummer ausgelassen wird, werden alle Achsen aktiviert.

Die Aktivierung der dritten Achse (Z) bewirkt, dass die Bremse gelöst wird.

Um alle Achsen zu aktivieren, sollte Motor On anstelle von **SLock** verwendet werden.

Die Verwendung von **SLock** im Motor Off-Status verursacht einen Fehler.

Hinweise

SLock setzt einige Systemelemente zurück in den Anfangsstatus:

SFree setzt aus Sicherheitsgründen Parameter zurück, die die Roboterarmvorschubgeschwindigkeit (Speed und SpeedS), die Beschleunigung (Accel und AccelS) und den LimZ Parameter betreffen.

Verwandte Befehle

Brake, LimZ, Reset, SFree

Beispiel einer SLock-Anweisung

Dies ist ein einfaches Beispiel für die Verwendung einer SLock-Anweisung. Die Voreinstellung für Bewegungen mit SFree muss aktiviert sein, damit die SLock-Anweisung in diesem Beispiel funktioniert.

```
Function test
   .
   .
   .
   'Gibt die Ansteuerung von J1 und J2 frei
   'und steuert die Z- und U-Achsen zur Teilinstallation
   SFree 1, 2
   Go P1
   'Stellt die Ansteuerung von J1 und J2 wieder her
   SLock 1, 2
   .
   .
   .
Fend
```

Space$-Funktion

F

Gibt eine Zeichenkette aus Leerzeichen aus.

Syntax

Space$(*count*)

Parameter

count Die Anzahl von Leerzeichen, die in die Ausgabe-Zeichenkette eingesetzt werden sollen.

Rückgabewerte

Gibt eine Zeichenkette von *count* Leerzeichen aus.

Beschreibung

Space$ gibt eine vom Anwender spezifizierte Zeichenkette von *count* Leerzeichen aus. **Space$** kann bis zu 255 Zeichen ausgeben (die maximale Anzahl von Zeichen, die in einer Zeichenkettenvariablen erlaubt ist).

Der **Space$-**Befehl wird für gewöhnlich verwendet, um Leerzeichen vor, hinter oder zwischen anderen Buchstabenzeichenketten einzufügen.

Verwandte Befehle

Asc, Chr$, InStr, Left$, Len, LSet$, Mid$, Right$, RSet$, Str$, Val

Beispiel einer Space$-Funktion

```
> Print "XYZ" + Space$(1) + "ABC"
XYZ ABC

> Print Space$(3) + "ABC"
   ABC
>
```

Speed-Anweisung

> S

Definiert die Armgeschwindigkeit für die Verwendung mit den PTP-Befehlen Go, Jump und Pulse oder zeigt diese an.

Syntax

(1) Speed *percent*, [*departSpeed]*, [*approSpeed*]
(2) Speed

Parameter

percent	Integer-Ausdruck zwischen 1-100, der die Armgeschwindigkeit als prozentualen Anteil der maximalen Geschwindigkeit darstellt.
departSpeed	Integer-Ausdruck zwischen 1 und 100, der die Depart-Bewegungsgeschwindigkeit für den Jump-Befehl darstellt.
approSpeed	Integer-Ausdruck zwischen 1 und 100, der die Approach-Bewegungsgeschwindigkeit für den Jump-Befehl darstellt.

Rückgabewerte

Zeigt die aktuellen **Speed**-Werte an, wenn sie ohne Parameter verwendet werden.

Beschreibung

Speed spezifiziert die Armgeschwindigkeit für alle PTP-Bewegungsbefehle. Dies schließt auch Bewegungen ein, die durch die Roboter-Bewegungsbefehle Go, Jump und Pulse ausgelöst wurden. Die Geschwindigkeit ist als prozentualer Anteil der maximalen Geschwindigkeit spezifiziert. Der Bereich der zulässigen Werte liegt zwischen 1 und 100 (1 steht für 1% der maximalen Geschwindigkeit und 100 für 100% der maximalen Geschwindigkeit). Speed 100 steht für die maximal mögliche Geschwindigkeit.

Die Werte der Depart- und Approach-Geschwindigkeit betreffen nur den Jump-Befehl. Wenn die Werte ausgelassen werden, wird jeder Wert auf den *percent-Wert* zurückgesetzt.

Der Geschwindigkeitswert wird auf den Standardwert zurückgesetzt, wenn einer der folgenden Befehle ausgeführt wird:

Controller Power On
Motor On
SFree, SLock
Reset
Stop-Button oder Tasten Strg+C

Im Low-Power-Modus ist die effektivste Geschwindigkeitseinstellung niedriger als der Vorgabewert. Wenn eine höhere Geschwindigkeit direkt (vom Befehlseingabefenster aus) oder in einem Programm spezifiziert wurde, wird die Geschwindigkeit auf den Vorgabewert gesetzt. Im High-Power-Modus ist die Bewegungsgeschwindigkeitseinstellung gleich dem mit **Speed** spezifizierten Wert.

Wenn eine Bewegung bei höherer Geschwindigkeit nötig ist, stellen Sie den High-Power-Modus unter Verwendung von Power High ein und schließen Sie die Schutzabschrankung. Wenn die Sicherheitstür offen ist, werden die **Speed**-Einstellungen auf die Vorgabewerte geändert.

Wenn **Speed** ausgeführt wird, während sich der Roboter im Low-Power-Modus befindet, wird eine Meldung angezeigt. Das folgende Beispiel zeigt, dass der Roboter sich mit der vorgegebenen Geschwindigkeit (5) bewegt, weil er im Low-Power-Modus ist, obwohl der eingestellte Geschwindigkeitswert von **Speed** bei 80 liegt.

```
> speed 80
> speed
Low-Power-Modus:
    80
    80     80
>
```

Verwandte Befehle

Accel, Go, Jump, Power, Pass, Pulse, SpeedS

Beispiel einer Speed-Anweisung

Speed kann vom Befehlseingabefenster aus oder in einem Programm verwendet werden. Unten angegeben sind einfache Beispiele beider Methoden.

```
Function speedtst
    Integer slow, fast, i
    slow = 10
    fast = 100
    For i = 1 To 10
        Speed slow
        Go  P0
        Go P1
        Speed fast
        Go  P0
        Go P1
    Next i
Fend
```

Vom Befehlseingabefenster aus kann der Anwender auch Geschwindigkeitswerte einstellen.

```
> Speed 100,100,50    'Die Abwärtsbewegung der Z-Achse wird auf 50
gesetzt
> Speed 50
> Speed
    Low-Power-Status: Speed is limited to 5
       50
       50        50
>
```

Speed-Funktion

F

Gibt eine der drei Geschwindigkeitseinstellungen aus.

Syntax

Speed[(*paramNumber*)]

Parameter

paramNumber	Integer-Ausdruck, der einen der unten gezeigten Werte bewertet. Wenn dieser Parameter ausgelassen wird, wird 1 verwendet. 1: Geschwindigkeit der PTP-Bewegung 2: Jump-Departgeschwindigkeit 3: Jump-Approachgeschwindigkeit

Rückgabewerte

Integer-Wert von 1 bis 100.

Verwandte Befehle

Speed-Anweisung

Beispiel einer Speed-Funktion

```
        Integer savSpeed

        savSpeed = Speed(1)
        Speed 50
        Go pick
        Speed savSpeed
    Fend
```

SpeedR-Anweisung

> S

Definiert die Werkzeugrotationsgeschwindigkeit für die CP-Bewegung, wenn ROT verwendet wird, oder zeigt diese an.

Syntax

(1) SpeedR *rotSpeed*
(2) SpeedR

Parameter

rotSpeed Reeller Ausdruck in Grad/Sekunde.

Rückgabewerte

Wenn Parameter ausgelassen werden, werden die aktuellen SpeedR-Einstellungen angezeigt.

Beschreibung

SpeedR wird wirksam, wenn die ROT-Bedingung in den Bewegungsbefehlen Move, Arc, Arc3, BMove, TMove und Jump3CP verwendet wird.

Der **SpeedR**-Wert wird auf den Standardwert zurückgesetzt (niedrige Geschwindigkeit), wenn eine der folgenden Aktionen ausgeführt wird:

Controller Power On
Motor On
SFree, SLock
Reset
Stop-Taste oder Tasten Strg + C

Verwandte Befehle

AccelR, Arc, Arc3, BMove, Jump3CP, Power, SpeedR Function, TMove

Beispiel einer SpeedR-Anweisung

```
SpeedR 200
```

SpeedR-Funktion

> F

Gibt einen Wert für die Werkzeugrotationsgeschwindigkeit aus.

Syntax

SpeedR

Rückgabewerte

Reeller Wert in Grad/Sekunde.

Verwandte Befehle

AccelR-Anweisung, SpeedR-Anweisung

Beispiel einer SpeedR-Funktion

```
Real currSpeedR
currSpeedR = SpeedR
```

SpeedS-Anweisung

Spezifiziert die Armgeschwindigkeit, die mit CP-Befehlen wie Move, Arc, Arc3, Jump3 und Jump3CP verwendet wird oder zeigt diese an.

> S

Syntax

(1) SpeedS *speed*, [*departSpeed*], [*approSpeed*]
(2) SpeedS

Parameter

speed	Reeller Ausdruck, der die CP-Bewegungsgeschwindigkeit in der Einheit mm/s darstellt. Gültige Einträge liegen bei 6-Achsrobotern zwischen 1 und 2000 und bei anderen Robotern zwischen 1 und 1120.
departSpeed	Optional. Reeller Ausdruck, der die Jump3-Departgeschwindigkeit in der Einheit mm/s darstellt. Gültige Einträge liegen bei 6-Achsrobotern zwischen 1 und 2000 und bei anderen Robotern zwischen 1 und 1120.
approSpeed	Optional. Reeller Ausdruck, der die Jump3-Approachgeschwindigkeit in der Einheit mm/sec darstellt. Gültige Einträge liegen bei 6-Achsrobotern zwischen 1 und 2000 und bei anderen Robotern zwischen 1 und 1120.

Rückgabewerte

Zeigt den aktuellen **SpeedS**-Wert an, wenn keine Parameter verwendet werden.

Beschreibung

SpeedS spezifiziert die Werkzeugmittelpunkt-Geschwindigkeit zur Verwendung mit allen CP-Bewegungsbefehlen. Dies schließt auch Bewegungen ein, die durch die Befehle Move und Arc initiiert wurden.

SpeedS wird in mm/s spezifiziert, was für die Werkzeugmittelpunkt-Geschwindigkeit für den Roboterarm steht. Gültige Einträge für **SpeedS** liegen bei 6-Achsrobotern zwischen 1 und 2000 und bei anderen Robotern zwischen 1 und 1120. Der Standardwert ist von Roboter zu Roboter unterschiedlich. Die SpeedS-Standardwerte für Ihr Robotermodell finden Sie in der Bedienungsanleitung des Roboters. Der SpeedS-Standardwert ist der **SpeedS**-Ausgangswert, der jedesmal automatisch vom Steuergerät eingerichtet wird, wenn der Hauptstrom eingeschaltet wird.

Der **SpeedS**-Wert wird auf den Standardwert zurückgesetzt, wenn einer der folgenden Befehle ausgeführt wird:

Controller Power On
Motor On
SFree, SLock
Reset
Stop-Taste oder Strg + C

Im Low-Power-Modus ist die effektivste **SpeedS**-Einstellung niedriger als der Vorgabewert. Wenn eine höhere Geschwindigkeit direkt (vom Befehlseingabefenster aus) oder in einem Programm spezifiziert wurde, wird die Geschwindigkeit auf den Vorgabewert gesetzt. Im High-Power-Modus ist die Bewegungseinstellung von **Speed** gleich dem Wert von Speed.

Wenn eine Bewegung bei höherer Geschwindigkeit nötig ist, stellen Sie den High-Power-Modus unter Verwendung von Power High ein und schließen Sie die Schutzabschrankung. Wenn die Sicherheitstür offen ist, werden die **SpeedS**-Einstellungen auf die Standardwerte geändert.

Wenn **SpeedS** ausgeführt wird, während sich der Roboter im Low-Power-Modus befindet, wird eine Meldung angezeigt. Das folgende Beispiel zeigt, dass der Roboter sich mit der vorgegebenen Geschwindigkeit (50) bewegt, weil er im Low-Power-Modus ist, auch wenn der eingestellte Geschwindigkeitswert von **SpeedS** bei 800 liegt.

```
> SpeedS 800
Low-Power-Status: SpeedS is limited to 50
>
> SpeedS
Low-Power-Status: SpeedS is limited to 50
  800
>
```

Verwandte Befehle

AccelS, Arc, Jump3, Move, Speed

Beispiel einer SpeedS-Anweisung

SpeedS kann vom Befehlseingabefenster aus oder in einem Programm verwendet werden. Unten angegeben sind einfache Beispiele beider Methoden.

```
Function speedtst
    Integer slow, fast, i
    slow = 50
    fast = 500
    For i = 1 To 10
        SpeedS slow
        Go  P0
        Move P1
        SpeedS fast
        Go  P0
        Move P1
    Next i
Fend
```

Vom Befehlseingabefenster aus kann der Anwender auch **SpeedS**-Werte einstellen.

```
> speeds 1000
> speeds 500
> speed 30     'Setzt die PTP-Geschwindigkeit
> go p0        'PTP-Bewegung
> speeds 100   'Setzt die geradlinige Geschwindigkeit in mm/s
> move P1      'Geradlinige Bewegung
```

SpeedS-Funktion

F

Gibt die aktuelle SpeedS-Einstellung aus.

Syntax

SpeedS [(*paramNumber*)]

Parameter

paramNumber	Optional. Integer-Ausdruck, der angibt, welcher SpeedS-Wert ausgegeben werden soll. 1: CP-Geschwindigkeit 2: Jump3-Departgeschwindigkeit 3: Jump3-Approachgeschwindigkeit

Rückgabewerte

Reelle Zahl, in mm/s

Verwandte Befehle

SpeedS-Anweisung

Beispiel einer SpeedS-Anweisung

```
Real savSpeeds
savSpeeds = SpeedS
Print "Jump3 depart speed = ", SpeedS(2)
```

SPELCom_Event-Anweisung

> S

Generiert ein Benutzerevent für eine VB Guide SPELCom-Steuerung, die in einem Host-Programm verwendet wird.

Syntax

SPELCom_Event *eventNumber* [*, msgArg1, msgArg2, msgArg3,...*]

Parameter

eventNumber	Ein Integer-Ausdruck, dessen Wert zwischen 1000 und 32767 liegt.
msgArg1, msgArg2, msgArg3...	Optional. Jedes Nachrichtenargument kann entweder eine Zahl, eine Buchstabensymbol-Zeichenkette oder ein Variablenname sein.

Beschreibung

Dieser Befehl erleichtert das Senden von Echtzeitinformationen an eine andere Anwendung, die die SPELCom ActiveX-Steuerung der VB Guide-Option verwendet. Sie können z. B. den Teilzählerstand, die Losnummer usw. aktualisieren, indem Sie ein Ereignis an Ihr Host-Programm schicken.

Hinweis

Dieser Befehl ist nur dann anwendbar, wenn die VB Guide-Option installiert ist.

Verwandte Befehle

VB Guide-Handbuch

Beispiel eines SPELCom_Event

In diesem Beispiel sendet ein SPEL+-Task Zyklusdaten an das Host-Programm.

```
Function RunParts
    Integer cycNum

    cycNum = 0
    ' Hauptschleife
    Do
        ...
        ...
        cycNum = cycNum + 1
        Spelcom_Event 3000, cycNum, lot$, cycTime
    Loop
Fend
```

Sqr-Funktion

F

Berechnet den nichtnegativen Quadratwurzelwert des Operanden.

Syntax

Sqr(*Operand*)

Parameter

Operand	Ein Ausdruck mit einer reellen Zahl.

Rückgabewerte

Quadratwurzelwert.

Beschreibung

Die Sqr-Funktion gibt den nichtnegativen Quadratwurzelwert des Operanden aus.

Mögliche Fehler

Negativer Operand:

Wenn der Operand ein negativer numerischer Wert ist oder einen negativen Wert hat, tritt ein Fehler auf.

Verwandte Befehle

Abs, And, Atan, Atan2, Cos, Int, Mod, Not, Or, Sgn, Sin, Str$, Tan, Val, Xor

Beispiel einer Sqr-Funktion

Dies ist ein einfaches Beispiel aus dem Befehlseingabefenster für die Verwendung der Sqr-Funktion.

```
>print sqr(2)
 1.414214
>
```

Das folgende Beispiel zeigt ein einfaches Programm, das den **Sqr**-Befehl nutzt.

```
Function sqrtest
    Real x
    Print "Please enter a numeric value:"
    Input x
    Print "The Square Root of ", x, " is ", Sqr(x)
Fend
```

Stat-Funktion

F

Gibt Informationen über den Ausführungsstatus der Steuerung aus.

Syntax

Stat*(address)*

Parameter

address Definiert, welche Status-Bits überprüft werden sollen.

Rückgabewerte

Gibt einen 4 Byte-Wert aus, der den Status der Steuerung darstellt. Siehe Tabelle unten.

Beschreibung

Der **Stat**-Befehl gibt Informationen aus, wie in der Tabelle unten aufgezeigt.

Adresse	Bit		Angegebener Status der Steuerung, wenn das Bit eingeschaltet ist.
0	0 bis 15	&H1 bis &H8000	Task 1 wird ausgeführt (Xqt) oder befindet sich im Halt-Status bis Task 16 wird ausgeführt (Xqt) oder befindet sich im Halt-Status
	16	&H10000	Task(s) wird / werden ausgeführt
	17	&H20000	Pausezustand
	18	&H40000	Fehlerzustand
	19	&H80000	Teach-Modus
	20	&H100000	Not-Aus-Zustand
	21	&H200000	Low-Power-Modus (Energie ist niedrig)
	22	&H400000	Eingang der Sicherheitsabschrankung ist geschlossen
	23	&H800000	Freigabeschalter ist offen.
	24	&H1000000	Nicht definiert
	25	&H2000000	Nicht definiert
	26-31		Nicht definiert
1	0	&H1	Protokolleintragung des Stopps über der Zielposition bei erfüllter Bedingung in der Jump...Sense Anweisung. (Dieser Protokolleintrag wird gelöscht, wenn eine andere Jump-Anweisung ausgeführt wird).
	1	&H2	Protokolleintragung des Stopps an einer Zwischenposition bei erfüllter Bedingung in der Go/Jump/Move...Till-Anweisung. (Dieser Protokolleintrag wird gelöscht, wenn eine andere Go/Jump/Move...Till-Anweisung ausgeführt wird.)
	2	&H4	Nicht definiert
	3	&H8	Protokolleintrag des Stopps an einer Zwischenposition bei erfüllter Bedingung in der Trap-Anweisung.
	4	&H10	Motor On-Modus
	5	&H20	Die aktuelle Position ist die Home-Position.
	6	&H40	Low Power-Status
	7	&H80	Nicht definiert
	8	&H100	Der Motor der 4. Achse ist eingeschaltet.
	9	&H200	Der Motor der 3. Achse ist eingeschaltet.
	10	&H400	Der Motor der 2. Achse ist eingeschaltet.
	11	&H800	Der Motor der 1. Achse ist eingeschaltet.
	12	&H1000	Der Motor der 6. Achse ist eingeschaltet.
	13	&H2000	Der Motor der 5. Achse ist eingeschaltet.
	14-31		Nicht definiert

Verwandte Befehle

EstopOn-Funktion, TillOn-Funktion, PauseOn-Funktion, SafetyOn-Funktion

Beispiel einer Stat-Funktion

```
Function StatDemo

    rbt1_sts = RShift((Stat(0) And &H070000), 16)
    Select TRUE
        Case (rbt1_sts And &H01) = 1
            Print "Tasks are running"
        Case (rbt1_sts And &H02) = 2
            Print "Pause Output is ON"
        Case (rbt1_sts And &H04) = 4
            Print "Error Output is ON"
    Send
Fend
```

Str$-Funktion

F

Konvertiert einen numerischen Wert in eine Zeichenkette und gibt diese aus.

Syntax

Str$*(number)*

Parameter

number Integer- oder reeller Ausdruck.

Rückgabewerte

Gibt eine Zeichenkettendarstellung des numerischen Wertes aus.

Beschreibung

Str$ konvertiert eine Zahl in eine Zeichenkette. Jede positive oder negative Zahl ist gültig.

Verwandte Befehle

Abs, Asc, Chr$, InStr, Int, Left$, Len, Mid$, Mod, Right$, Sgn, Space$, Val

Beispiel einer Str$-Funktion

Das folgende Beispiel zeigt ein Programm, das verschiedene Zahlen in Zeichenketten umwandelt und diese dann am Bildschirm ausgibt.

```
Function strtest
    Integer intvar
    Real realvar
    '
    intvar = -32767
    Print "intvar = ", Str$(intvar)
    '
    realvar = 567.9987
    Print "realvar = ", Str$(realvar)
    '
Fend
```

Andere Beispielergebnisse des Str$-Befehls vom Befehlseingabefenster.

```
> Print Str$(99999999999999)
 1.000000E+014

> Print Str$(25.999)
 25.999
```

String-Anweisung

Deklariert String-Variablen. (Zeichenkettenvariablen)

S

Syntax

String *varName$* [**(***subscripts***)**] [, *varName$* [**(***subscripts***)**]**...**]

Parameter

varName$	Variablenname, den der Benutzer als **String** deklarieren möchte.
subscripts	Optional. Dimensionen einer Feldvariable; es können bis zu drei Dimensionen deklariert werden. Die Indexsyntax sieht folgendermaßen aus: (ubound1, [ubound2], [ubound3]) Ubound1, ubound2 und ubound3 spezifizieren jeweils die Obergrenze der dazugehörigen Dimension. Die Elemente in jeder Dimension einer Matrix werden von 0 bis zum Wert der Obergrenze durchnummeriert. Die Gesamtzahl der Elemente der Matrixelemente für lokale und Global Preserve-Variablen ist 100. Die Gesamtzahl der Matrixelemente für globale und Modulvariablen ist 1000. Zur Berechnung aller in einer Matrix verwendeten Elemente verwenden Sie die folgende Formel: (Wenn eine Dimension nicht verwendet wird, ersetzen Sie den Ubound-Wert durch 0.) Alle Elemente = (ubound 1 +1) * (ubound2 + 1) * (ubound3 + 1)

Beschreibung

Die **String**-Anweisung wird verwendet, um Zeichenkettenvariablen zu deklarieren. Zeichenkettenvariablen können bis zu 255 Zeichen enthalten. Lokale Variablen sollten am Anfang einer Funktion deklariert sein. Globale und Modulvariabeln müssen außerhalb von Funktionen deklariert werden.

String-Operatoren

Die folgenden Operatoren können verwendet werden, um Zeichenkettenvariablen zu manipulieren:

+ Fügt Buchstaben-Zeichenketten zusammen**.** Kann in Zuweisungskommandos für Zeichenkettenvariablen oder im Print-Befehl verwendet werden.
Beispiel: name$ = fname$ + " " + lname$

= Vergleicht Buchstaben-Zeichenketten. Wahr wird nur ausgegeben, wenn die beiden Zeichenketten, auch in Bezug auf Groß- oder Kleinschreibung, völlig gleich sind.
Beispiel: If temp1$ = "A" Then GoSub test

< > Vergleicht Buchstaben-Zeichenketten. Wahr wird ausgegeben, wenn eines oder mehrere Zeichen unterschiedlich sind.
Beispiel: If temp1$ <> "A" Then GoSub test

Hinweise

Variablennamen müssen das Dollarzeichen „$" enthalten:

Bei Zeichenkettenvariablen muss an der letzten Stelle des Variablennamens das Dollarzeichen „$" stehen.

Verwandte Befehle

Boolean, Byte, Double, Global, Integer, Long, Real

Beispiel einer String-Anweisung

```
String password$
String A$(10)         'Eindimensionale Matrix aus Zeichenketten
String B$(10, 10)     'Zweidimensionale Matrix aus Zeichenketten
String C$(10, 10, 10) 'Dreidimensionale Matrix aus Zeichenketten

Print "Enter password:"
Input password$
If UCase$(password$) = "EPSON" Then
    Call RunMaintenance
Else
    Print "Password invalid!"
EndIf
```

Sw-Funktion

F

Gibt den Status des gewählten Eingangsports aus oder zeigt ihn an (d. h. abgetrennte Anwender-E/As).

Syntax

Sw(*bitNumber***)**

Parameter

bitNumber Integer-Ausdruck, der für die E/A-Eingänge steht.

Rückgabewerte

Gibt eine 1 aus, wenn der spezifizierte Eingang eingeschaltet ist und eine 0, wenn der spezifizierte Eingang ausgeschaltet ist.

Beschreibung

Sw bietet Ihnen die Möglichkeit, eine Statusüberprüfung der Hardwareeingänge durchzuführen. **Sw** wird im allgemeinen verwendet, um den Status eines Einganges zu überprüfen, der an eine Aufgabevorrichtung, ein Förderband, einen Greifermagneten oder an den Host eines anderen Gerätes angeschlossen ist, welches über getrennte E/As arbeitet. Natürlich hat der mit dem **Sw**-Befehl überprüfte Eingang zwei Status (1 oder 0). Diese zeigen an, ob das Gerät ein- oder ausgeschaltet ist.

Verwandte Befehle

In, InBCD, MemOn, MemOff, MemSw, Off, On, OpBCD, Oport, Out, Wait

Beispiel einer Sw-Funktion

Das Beispiel unten überprüft den getrennten Eingang Nr. 5 und verzweigt dementsprechend. Zur besseren Übersicht wird On anstelle von 1 verwendet.

```
Function main
    Integer i, feed5Ready
    feed5Ready = Sw(5)
    'Überprüft, ob der Feeder bereit ist
    If feed5Ready = On Then
        Call mkpart1
    Else
        Print "Feeder #5 is not ready. Please reset and"
        Print "then restart program"
    EndIf
Fend
```

Andere einfache Beispiele vom Befehlseingabefenster sehen wie folgt aus:

```
> print sw(5)
1
>
```

SyncLock-Anweisung

Synchronisiert Tasks mithilfe eines Mutex-Locks.

S

Syntax

SyncLock *syncID* [, *timeOut*]

Parameter

syncID	Integer-Ausdruck, der für die zu empfangende Signalnummer steht. Der Bereich umfasst 0 bis 15.
timeOut	Optional. Reeller Ausdruck, der für die maximale Wartezeit auf den Lock steht.

Beschreibung

Verwenden Sie **SyncLock**, um die Verwendung einer gemeinsamen Quelle zu sperren, so dass immer nur ein Task diese verwenden kann. Wenn der Task die Quelle nicht mehr benötigt, muss SyncUnlock aufgerufen werden, damit die Sperrung freigegeben wird und andere Tasks die Quelle verwenden können.

Ein Task kann nur eine SyncID freigeben, die vorher gesperrt war.

Um die Sperrung freizugeben, muss der Task SyncUnlock ausführen. Wenn der Task beendet ist, kann kein anderer Task die Sperrung verwenden, bis alle Tasks abgebrochen wurden.

Wenn SyncLock zweimal in Folge mit derselben Signalnummer verwendet wird, tritt ein Fehler auf.

Wenn der *timeout*-Parameter verwendet wird, muss die **Tw**-Funktion verwendet werden, um zu überprüfen, ob die Sperrung erfolgreich war.

Verwandte Befehle

Signal, SyncLock, Tw, Wait, WaitPos

Beispiel einer SyncLock-Anweisung

Im folgenden Beispiel werden SyncLock und SyncUnlock verwendet, damit immer nur ein Task eine Nachricht an den Kommunikationsport schreiben kann.

```
Function Main

    Xqt Func1
    Xqt Func2
Fend

Function Func1
  Long count
  Do
    Wait .5
    count = count + 1
    LogMsg "Msg from Func1, " + Str$(count)
  Loop
Fend

Function Func2
  Long count
  Do
    Wait .5
    count = count + 1
    LogMsg "Msg from Func2, " + Str$(count)
  Loop
Fend



Function LogMsg(msg$ As String)
  SyncLock 1
  OpenCom #1
  Print #1, msg$
  CloseCom #1
  SyncUnlock 1
Fend
```

Im folgenden Beispiel wird SyncLock mit einer optionalen Timeout-Funktion verwendet. Tw wird verwendet, um zu überprüfen, ob die Sperrung erfolgreich war. Durch die Verwendung der Timeout-Funktion können Sie periodisch andere Codes ausführen, während Sie darauf warten, dass eine Quelle gesperrt wird.

```
Function MySyncLock(syncID As Integer)
  Do
    SyncLock syncID, .5
    If Tw = 0 Then
      Exit Function
    EndIf
    If Sw(1) = On Then
      Off 1
    EndIf
  Loop
Fend
```

SyncUnlock-Anweisung

S

Gibt eine SyncID frei, die vorher unter SyncLock gesperrt wurde.

Syntax

SyncUnlock *syncID*

Parameter

syncID	Integer-Ausdruck, der für die zu empfangende Signalnummer steht. Der Bereich umfasst 0 bis 15.

Beschreibung

Verwenden Sie **SyncUnlock**, um eine SyncID freizugeben, die zuvor durch SyncLock gesperrt wurde.

Ein Task kann nur eine SyncID freigeben, die vorher gesperrt war.

Verwandte Befehle

Signal, SyncLock, Wait, WaitPos

Beispiel einer SyncUnlock-Anweisung

```
Function Main

    Xqt task
    Xqt task
    Xqt task
    Xqt task
Fend

Function task
  Do
    SyncLock 1
    Print "resource 1 is locked by task", MyTask
    Wait .5
    SyncUnlock 1
  Loop
Fend
```

SysConfig-Befehl

Zeigt die Systemkonfigurationsparameter an.

S

Syntax

SysConfig

Rückgabewerte

Zeigt die Systemkonfigurationsparameter an.

Beschreibung

Zeigt die momentan konfigurierten Werte für die Systemdaten an. Bei Lieferung des Roboters und der Steuerung oder bei Änderungen der Konfiguration sollten Sie diese Daten speichern. Verwenden Sie hierfür Backup Steuerung im Dialog Tools | Steuerung. Außerdem sollten Sie diese Daten ausdrucken und an einem sicheren Ort aufbewahren, da Informationen wie Kalibrierungsdaten und roboterspezifische Informationen mit dem SysConfig-Befehl angezeigt werden. Die Ausgabewerte des SysConfig-Befehls können aus dem Print-Dialog in EPSON RC+ gedruckt werden, indem der Ausgabe-Dialog des SysConfig-Befehls überprüft wird.

Die folgenden Daten werden angezeigt. (Die folgenden Daten dienen ausschließlich zu Referenzzwecken, da die Daten von Steuergerät zu Steuergerät variieren.)

```
' Version:
'   Firmware 1, 0, 0, 0

' Options:
'   External Control Point
'   VB Guide

' HOUR: 414.634

' Controller:
'   Serial #: 0001

' ROBOT 1:
' Name: Mnp01
' Model: PS3-AS10
'   Serial #: 0001
' Motor On Time: 32.738
'   Motor 1: Enabled, Power = 400
'   Motor 2: Enabled, Power = 400
'   Motor 3: Enabled, Power = 200
'   Motor 4: Enabled, Power = 50
'   Motor 5: Enabled, Power = 50
'   Motor 6: Enabled, Power = 50
```

```
  ARCH 0, 30, 30
  ARCH 1, 40, 40
  ARCH 2, 50, 50
  ARCH 3, 60, 60
  ARCH 4, 70, 70
  ARCH 5, 80, 80
  ARCH 6, 90, 90
  ARMSET 0, 0, 0, 0, 0, 0
  HOFS 0, 0, 0, 0, 0, 0
  HORDR 63, 0, 0, 0, 0, 0
  RANGE -7427414, 7427414, -8738134, 2621440, -3145728, 8301227, -5534152, 5534152,
-3640889, 3640889, -6553600, 6553600
  BASE 0, 0, 0, 0, 0, 0
  WEIGHT 2, 0
  INERTIA  0.1, 0
  XYLIM 0, 0, 0, 0, 0, 0


' Extended I/O Boards:
'   1: Installed
'   2: Installed
'   3: None installed
'   4: None installed

' Fieldbus I/O Board:
'   Installed
'   Type: PROFIBUS

' RS232C Boards:
'   1: Installed
'   2: None installed
```

Beispiel eines SysConfig-Befehls

```
> SysConfig
```

Tab$-Funktion

F

Gibt eine Zeichenkette aus, die die spezifizierte Anzahl von Tabulatoren beinhaltet.

Syntax

Tab$*(number)*

Parameter

number Integer-Ausdruck, der die Anzahl der Tabulatoren darstellt.

Rückgabewerte

Zeichenkette, die Tabulatoren enthält.

Beschreibung

Tab$ gibt eine Zeichenkette aus, die die spezifizierte Anzahl von Tabulatoren beinhaltet.

Verwandte Befehle

Left$, Mid$, Right$, Space$

Beispiel einer Tab$-Funktion

```
Print "X", Tab$(1), "Y"
Print
For i = 1 To 10
    Print x(i), Tab$(1), y(i)
Next i
```

Tan-Funktion

F

Gibt den Tangens eines numerischen Ausdrucks aus.

Syntax

Tan(*radians***)**

Parameter

radians Reeller Ausdruck, angegeben in Radianten.

Rückgabewerte

Reelle Zahl, die den Tangens des Parameters *radians* beinhaltet.

Beschreibung

Tan gibt den Tangens des numerischen Ausdrucks aus. Der numerische Ausdruck (*radians*) kann ein beliebiger numerischer Wert in Radianten sein.

Verwenden Sie die RadToDeg-Funktion, um Radiantenwerte in Gradwerte umzuwandeln.

Verwandte Befehle

Abs, Atan, Atan2, Cos, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Val

Beispiel einer Tan-Funktion

```
Function tantest
    Real num
    Print "Enter number in radians to calculate tangent for:"
    Input num
    Print "The tangent of ", num, "is ", Tan(num)
Fend
```

Das folgende Beispiel zeigt typische Ergebnisse der Verwendung des Tan-Befehls vom Befehlseingabefenster aus.

```
> print tan(0)
0.00
> print tan(45)
1.6197751905439
>
```

TargetOK-Funktion

Gibt an, ob die PTP-Bewegung von der aktuellen Position aus zur Zielposition möglich ist. **F**

Syntax

TargetOK(*targetPos***)**

Parameter

targetPos Punktausdruck für die Zielposition.

Rückgabewerte

Wahr, wenn es möglich ist, von der aktuellen Position zur Zielposition zu gelangen, ansonsten Falsch.

Beschreibung

Verwenden Sie **TargetOK**, um sicherzustellen, dass eine Zielposition und -ausrichtung erreicht werden können, bevor die eigentliche Bewegung dorthin ausgeführt wird. Der Trajektoriebereich der Bewegung zum Zielpunkt wird dabei nicht berücksichtigt.

Verwandte Befehle

CurPos, FindPos, InPos, WaitPos

Beispiel einer TargetOK-Funktion

```
If TargetOK(P1) Then
  Go P1
EndIf

If TargetOK(P10 /L /F) Then
  Go P10 /L /F
EndIf
```

TaskDone-Funktion

F

Gibt den Beedingungsstatus eines Tasks aus.

Syntax

TaskDone (*taskIdentifier*)

Parameter

taskIdentifier	Taskname oder Integer-Ausdruck, der für die Tasknummer steht. Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, umfasst der Bereich für normale Tasks 1 bis 16 und für Traptasks 257 bis 261.

Rückgabewerte

Wahr, wenn der Task abgeschlossen wurde, Falsch, wenn er nicht abgeschlossen wurde.

Beschreibung

Verwenden Sie TaskDone um zu ermitteln, ob ein Task abgeschlossen wurde.

Verwandte Befehle

TaskState, TaskWait

Beispiel einer TaskDone-Funktion

```
Xqt 2, conveyor
Do
    .
    .
Loop Until TaskDone(conveyor)
```

TaskInfo-Funktion

F

Gibt Statusinformationen für einen Task aus.

Syntax

TaskInfo(*taskIdentifier, index*)

Parameter

taskIdentifier Taskname oder Integer-Ausdruck, der für die Tasknummer steht.
Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, umfasst der Bereich für normale Tasks 1 bis 16 und für Traptasks 257 bis 261.

index Integer-Ausdruck, der für den Index der auszugebenden Daten steht.

Rückgabewerte

Ein Integer, der die spezifizierten Informationen enthält.

Beschreibung

Index	Beschreibung
0	Tasknummer
1	Nicht definiert
2	Nicht definiert
3	0- Der spezifizierte Task wird nicht ausgeführt. 1- Der spezifizierte Task wird ausgeführt. 2 – Der spezifizierte Task wartet auf ein Ereignis. 3 – Der spezifizierte Task wurde unterbrochen oder angehalten. 4 – Der spezifizierte Task befindet sich im Quick-Pause-Status. 5 – Der spezifizierte Task befindet sich im Fehlerstatus.
4	Während des Wartens auf ein Ereignis ist ein Zeitüberlauf aufgetreten (dasselbe wie TW).
5	Ereigniswartezeit (Millisekunden).
6	Aktuelle Roboternummer, die vom Task ausgewählt wurde.
7	Aktuelle Roboternummer, die vom Task verwendet wird.

Verwandte Befehle

CtrlInfo, RobotInfo, TaskInfo

Beispiel einer TaskInfo-Funktion

```
If (TaskInfo(1, 3) = 3 Then
  Print "Task 1 is runnning"
Else
  Print "Task 1 is not running"
EndIf
```

TaskInfo$-Funktion

F

Gibt Textinformationen für einen Task aus.

Syntax

TaskInfo$(*taskIdentifier, index*)

Parameter

taskIdentifier Taskname oder Integer-Ausdruck, der für die Tasknummer steht.
Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, umfasst der Bereich für normale Tasks 1 bis 16 und für Traptasks 257 bis 261.

index Integer-Ausdruck, der für den Index der auszugebenden Daten steht.

Rückgabewerte

Eine Zeichenkette, die die spezifizierten Informationen enthält.

Beschreibung

Die folgende Tabelle enthält Informationen, die mithilfe von **TaskInfo$** ausgegeben werden können.

Index	Beschreibung
0	Taskname
1	Anfangsdatum / -uhrzeit
2	Name der momentan ausgeführten Funktion
3	Zeilennummer in der Programmdatei, die die Funktion enthält

Verwandte Befehle

CtrlInfo, RobotInfo, TaskInfo

Beispiel einer TaskInfo$-Funktion

```
Print "Task 1 started: "TaskInfo$(1, 1)
```

TaskState-Funktion

F

Gibt den aktuellen Status eines Tasks aus.

Syntax

TaskState(*taskIdentifier*)

Parameter

taskIdentifier	Taskname oder Integer-Ausdruck, der für die Tasknummer steht. Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, umfasst der Bereich für normale Tasks 1 bis 16 und für Traptasks 257 bis 261.

Rückgabewerte

0: Der Task läuft nicht.
1: Der Task läuft.
2: Der Task wartet auf ein Ereignis.
3: Der Task wurde angehalten.
4: Der Task ist mit QuickPause angehalten worden.
5: Der Task befindet sich im Fehlerzustand.

Beschreibung

Verwenden Sie TaskState, um den Status für einen festgelegten Task zu erhalten. Sie können die Tasknummer oder den Tasknamen spezifizieren.

Verwandte Befehle

TaskDone, TaskWait

Beispiel einer TaskState-Funktion

```
If TaskState(conveyor) = 0 Then
    Xqt 2, conveyor
EndIf
```

TaskWait-Anweisung

Wartet darauf, dass ein Task beendet wird.

S

Syntax

TaskWait (*taskIdentifier*)

Parameter

taskIdentifier	Taskname oder Integer-Ausdruck, der für die Tasknummer steht. Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, umfasst der Bereich für normale Tasks 1 bis 16 und für Traptasks 257 bis 261.

Verwandte Befehle

TaskDone, TaskState

Beispiel einer TaskWait-Anweisung

```
Xqt 2, conveyor
TaskWait conveyor
```

TGo-Anweisung

Führt eine relative PTP-Bewegung im aktuellen Werkzeug-Koordinatensystem aus. **>** **S**

Syntax

TGo *destination* [**CP**] [*searchExpr*] [*!...!*]

Parameter

destination	Der Zielort einer Bewegung, die einen Punktausdruck verwendet.
CP	Optional. Spezifiziert CP-Bewegungen.
searchExpr	Optional. Ein Till- oder Find-Ausdruck. **Till** | **Find** **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können hinzugefügt werden, um E/A-Befehle und andere Befehle während der Bewegung auszuführen.

Beschreibung

Führt eine relative PTP-Bewegung im aktuellen Werkzeug-Koordinatensystem aus.

Armausrichtungsattribute, die unter dem Punktausdruck *destination* festgelegt wurden, werden ignoriert. Der Manipulator behält die aktuellen Armausrichtungspunkte bei. Bei einem 6-Achsmanipulator werden die Armausrichtungsattribute jedoch automatisch so geändert, dass die Achsenlaufweite so klein wie möglich ist.

Die Till-Bedingung wird verwendet, um TGo durch Verzögerung und Stoppen des Roboters an einer Zwischenposition des Verfahrweges abzuschließen, wenn die aktuelle Till-Bedingung erfüllt ist.

Der Find-Befehl wird verwendet, um einen Punkt in FindPos zu speichern, wenn die Find-Bedingung während der Bewegung erfüllt wird.

Wenn Till verwendet wird und die Till-Bedingung erfüllt wurde, hält der Manipulator sofort an und der Bewegungsbefehl ist beendet. Wenn die Till-Bedingung nicht erfüllt wird, bewegt sich der Manipulator in die Zielposition.

Wenn Find verwendet wird und die Find-Bedingung erfüllt wurde, wird die aktuelle Position gespeichert. Bitte lesen Sie den Abschnitt *Find* für weitere Informationen hierzu.

Wenn Parallelbearbeitung verwendet wird, können andere Prozesse parallel zu dem Bewegungsbefehl ausgeführt werden.

Verwandte Befehle

Accel, Find, !....! Parallelbearbeitung, Punktzuweisung, Speed, Till, TMove, Tool

Beispiel einer TGo-Anweisung

```
> TGo XY(100, 0, 0, 0)  'Verfährt 100mm in X-Richtung
                        '(im Werkzeug-Koordinatensystem)
Function TGoTest

  Speed 50
  Accel 50, 50
  Power High

  Tool 0
  P1 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L

  Go P1
  Print Here
  TGo XY(0, 0, -30, 0)
  Print Here

  Go P2
  Print Here
  TGo XY(0, 0, -30, 0)
  Print Here

Fend

[Output]
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

Till-Anweisung

> S

Spezifiziert und zeigt die Eingabebedingung an, die, wenn sie erfüllt ist, den durchgeführten Bewegungsbefehl (Jump, Go, Move usw.) abschließt, indem der Roboter an einer Zwischenposition verzögert und angehalten wird.

Syntax

Till [*inputCondition*]

Parameter

inputCondition Diese Bedingung überprüft den Status des Eingangs und muss einen Boolean-Wert ausgeben. Die folgenden Funktionen und Operatoren können für die *inputCondition* verwendet werden:

Funktionen: In, InW, MemIn, MemSw, Sw, Force

Operatoren: And, Or, Xor, +, *

Andere: Klammern, um verschiedenen Operationen und Variablen Prioriät einzuräumen.

Beschreibung

Die **Till**-Bedingung muss mindestens eine Eingangs- oder Merkereingangsfunktion enthalten. Fügen Sie in die **Till**-Bedingung nur die Operatoren ein, die im vorangegangenen Abschnitt "Parameter" beschrieben werden. (Die Verwendung eines anderen Operators erzeugt zwar keinen Fehler, führt aber zu unvorhersehbaren Bewegungen.)

Die **Till**-Anweisung kann in einer separaten Zeile oder als Suchausdruck in einer Bewegungsbefehl-Anweisung verwendet werden.

Wenn Variablen mit einbezogen sind, werden ihre Werte während der **Till**-Ausführung berechnet. Multiple **Till**-Anweisungen sind erlaubt. Die zuletzt aufgetretene **Till**-Bedingung bleibt solange aktuell, bis sie ersetzt wird.

Wenn Parameter ausgelassen werden, werden die aktuellen Till-Definitionen angezeigt.

Hinweise

Till-Einstellung bei Einschalten des Hauptstroms:

Wenn die Stromzufuhr eingeschaltet wird, wird die **Till**-Bedingung auf **Till** Sw(0) = On initialisiert.

Verwenden Sie Stat oder TillOn, um Till zu verifizieren:

Sie haben die Möglichkeit, nach der Ausführung eines Bewegungsbefehls, der die Till-Bedingung verwendet, zu überprüfen, ob die Till-Bedingung erfüllt wurde. Das können Sie tun, indem Sie die Stat-Funktion oder die TillOn-Funktion verwenden.

Verwandte Befehle

Find, Go, In, InW, Jump, MemIn, MemSw, Move, Stat, Sw, TillOn

Beispiel einer Till-Anweisung

Unten sind einige Zeilen eines Programmes dargestellt, das die Till-Anweisung verwendet.

```
Till Sw(1) = 0   'Definiert die Till-Bedingung (Eingang 1 aus)
Go P1 Till       'Stoppt, wenn die Bedingung der voherrigen Zeile
erfüllt ist
Till Sw(1) = 1 And Sw($1) = 1   'Definiert die neue Till-Bedingung
Move P2 Till     'Stoppt, wenn die Bedingung der vorherigen Zeile
erfüllt ist
Move P5 Till Sw(10) = 1         'Stoppt, wenn die Bedingung in dieser
                                'Zeile erfüllt ist
```

TillOn-Funktion

F

Gibt den aktuellen Till-Status aus.

Syntax

TillOn

Rückgabewerte

Wahr, wenn die Till-Bedingung im vorangegangenen Bewegungsbefehl, der Till verwendet hat, aufgetreten ist.

Beschreibung

TillOn gibt Wahr aus, wenn die Till-Bedingung aufgetreten ist.

TillOn entspricht ((Stat(1) And 2) <> 0).

Verwandte Befehle

EStopOn, SafetyOn, Sense, Stat, Till

Beispiel einer TillOn-Funktion

```
Go P0 Till Sw(1) = On
If TillOn Then
    Print "Till condition occurred during move to P0"
EndIf
```

Time-Befehl

Spezifiziert die aktuelle Uhrzeit und zeigt sie an.

Syntax

(1) Time *hours, minutes, seconds*
(2) Time

Parameter

hours	Die Stunde, auf die die Uhr der Steuerung eingestellt werden soll. Hierbei handelt es sich um einen Funktionsausdruck zwischen 1 und 24.
minutes	Die Minute, auf diedie Uhr der Steuerung eingestellt werden soll. Hierbei handelt es sich um einen Funktionsausdruck zwischen 0 und 59.
seconds	Die Sekunde, auf die die Uhr der Steuerung eingestellt werden soll. Hierbei handelt es sich um einen Funktionsausdruck zwischen 0 und 59.

Rückgabewerte

Wenn Parameter ausgelassen werden, wird die aktuelle Uhrzeit im 24-Stunden-Format angezeigt.

Beschreibung

Spezifiziert die aktuelle Uhrzeit.

Die Uhrzeit wird im 24-Stunden-Format angegeben.

Verwandte Befehle

Date, Time$

Beispiel eines Time-Befehls

```
> Time
The current time is 10:15:32

> Time 1,5,0
> Time
The current time is 01:05:15
```

Time-Funktion

F

Gibt die Betriebsstunden der Robotersteuerung aus.

Syntax

Time(*unitSelect*)

Parameter

unitSelect Ein Integer zwischen 0 und 2. Dieses Integer spezifiziert, welche Zeiteinheit die Steuerung ausgibt:

0: Stunden

1: Minuten

2: Sekunden

Beschreibung

Gibt die Betriebsstunden der Robotersteuerung als Integer aus.

Verwandte Befehle

Hour

Beispiel einer Time-Funktion

Im Folgenden werden einige Beispiele aus dem Befehlseingabefenster aufgezeigt:

```
Function main
    Integer h, m, s

    h = Time(0)   'Speichert die Zeit in Stunden
    m = Time(1)   'Speichert die Zeit in Minuten
    s = Time(2)   'Speichert die Zeit in Sekunden
    Print "This controller has been used:"
    Print h, "hours, ",
    Print m, "minutes, ",
    Print s, "seconds"
Fend
```

Time$-Funktion

F

Gibt die Systemzeit aus.

Syntax

Time$

Rückgabewerte

Eine Zeichenkette, die die aktuelle Zeit im 24-Stunden-Format *hh:mm:ss* enthält.

Beschreibung

Time$ wird verwendet, um die Systemzeit in einer Programmanweisung zu erhalten. Um die Systemzeit einzustellen, müssen Sie den Time-Befehl vom Befehlseingabefenster aus ausführen.

Verwandte Befehle

Date, Date$, Time

Beispiel einer Time$-Funktion

```
Print "The current time is: ", Time$
```

TLClr-Anweisung

Löscht ein Werkzeug-Koordinatensystem (macht Definitionen rückgängig). > S

Syntax

TLClr *toolNumber*

Parameter

toolNumber	Integer-Ausdruck, der angibt, welches von drei Werkzeugen zu löschen ist. (Werkzeug 0 ist das Vorgabewerkzeug und kann nicht gelöscht werden.)

Verwandte Befehle

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLSet

Beispiel einer TLClr-Anweisung

```
TLClr 1
```

TLDef-Funktion

> F

Gibt den Status einer Werkzeugdefinition aus.

Syntax

TLDef (*toolNumber*)

Parameter

toolNumber	Integer-Ausdruck, der angibt, für welches Werkzeug der Status ausgegeben werden soll.

Rückgabewerte

Wahr, wenn das spezifizierte Werkzeug definiert wurde, sonst Falsch.

Verwandte Befehle

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLClr, TLSet

Beispiel einer TLDef-Funktion

```
Function DisplayToolDef(toolNum As Integer)

   If TlDef(toolNum) = False Then
      Print "Tool ", toolNum, "is not defined"
   Else
      Print "Tool ", toolNum, ": ",
      Print TlSet(toolNum)
   EndIf
Fend
```

TLSet-Anweisung

Definiert ein Werkzeug-Koordinatensystem oder zeigt es an.

> S

Syntax

(1) TLSet *toolNum*, *toolDefPoint*
(2) TLSet *toolNum*
(3) TLSet

Parameter

toolNum	Integer von 1 bis 3, das darstellt, welches von 3 Werkzeugen definiert werden soll. (Werkzeug 0 ist das Vorgabewerkzeug und kann nicht modifiziert werden.)
toolDefPoint	**P***number* oder **P**(*expr*) oder Punktlabel oder Punktausdruck.

Rückgabewerte

Wenn Parameter ausgelassen werden, werden alle **TLSet**-Definitionen angezeigt.
Wenn nur die Werkzeugnummer spezifiziert wird, werden die spezifizierten **TLSet**-Definitionen angezeigt.

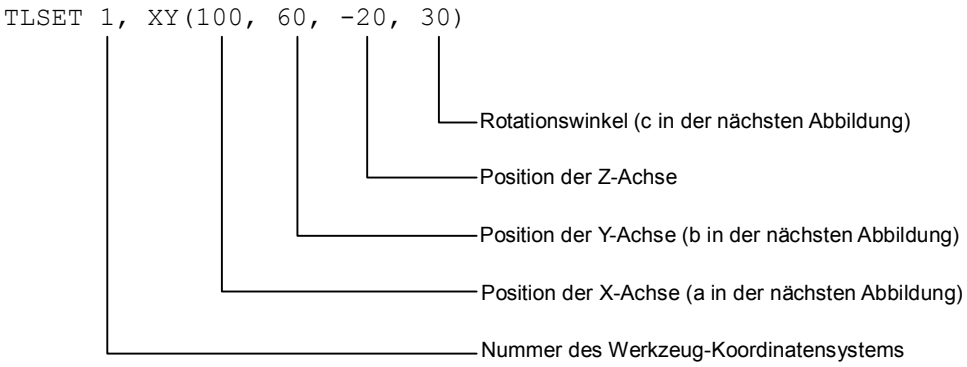
Beschreibung

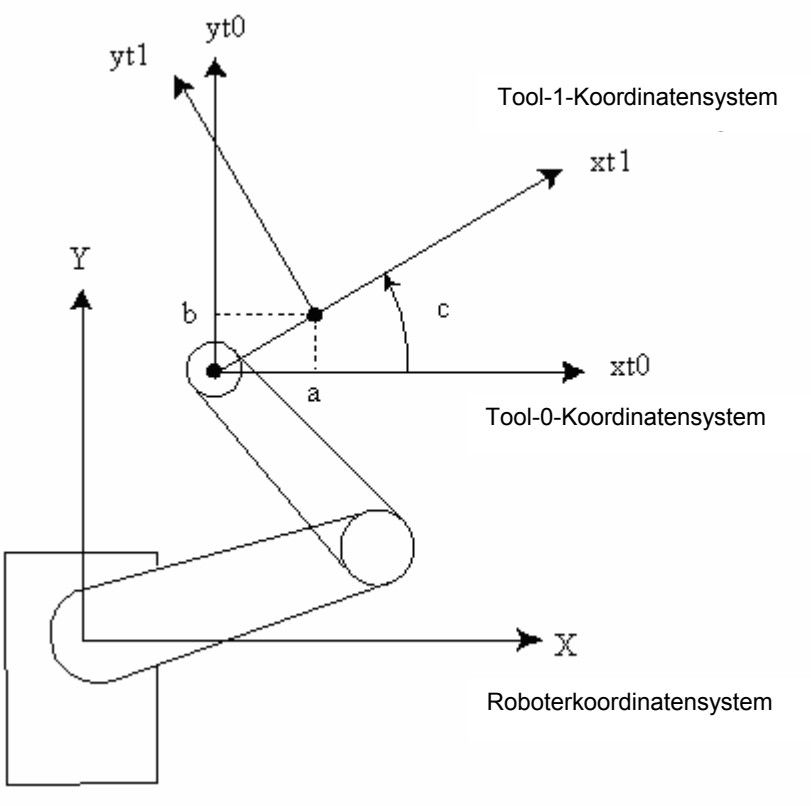
Definiert die Werkzeug-Koordinatensysteme Tool 1, Tool 2 und Tool 3 durch die Spezifizierung des Ursprungs und des Rotationswinkels des Werkzeug-Koordinatensystems in Relation zum Tool-0-Koordinatensystem (Armkoordinatensystem).

```
TLSet 1, XY(50,100,-20,30)

TLSet 2, P10 +X(20)
```

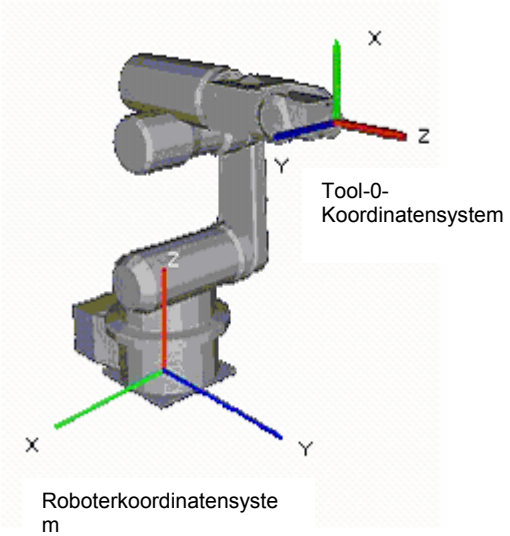
In diesem Fall wird Bezug auf die Koordinatenwerte von P10 genommen und zu dem X-Wert wird 20 addiert. Armattribut und Nummern des lokalen Koordinatensystems werden ignoriert.



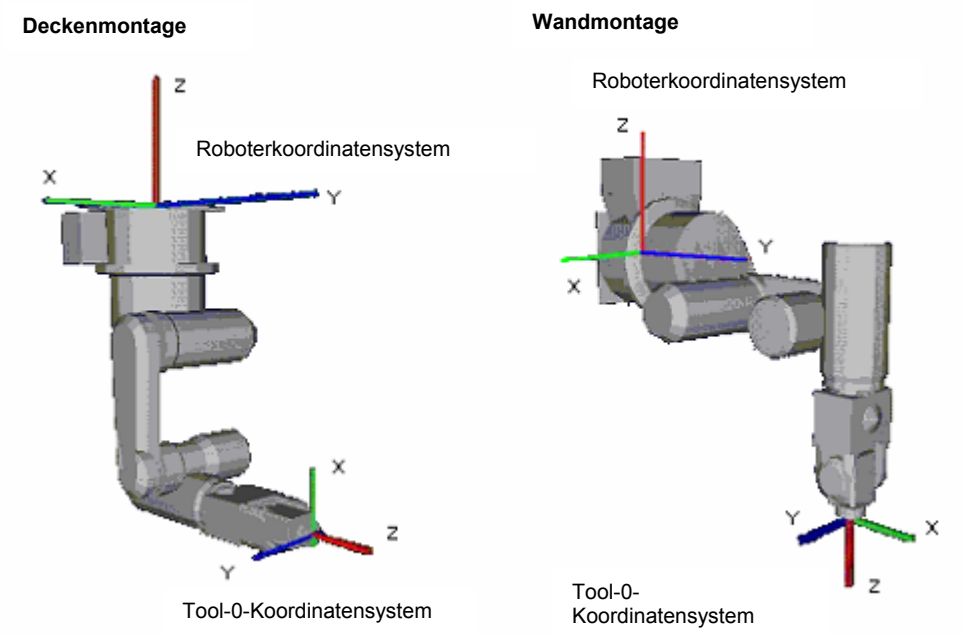


TlSet bei 6-Achsrobotern.

Der Ursprung von Tool 0 ist die Flanschseite der sechsten Achse. Wenn sich alle Achsen in der 0-Grad-Position befinden, wird die X-Achse des Tool-0-Koordinatensystems auf die Z-Achse des Roboterkoordinatensystems, die Y-Achse auf die X-Achse des Roboterkoordinatensystems und die Z-Achse senkrecht zur Flanschfläche und außerdem auf die Y-Achse des Roboterkoordinatensystems ausgerichtet, siehe Abbildung unten.



Tool-0-Koordinatensysteme werden für decken- und wandmontierte Roboter definiert, wie in der folgenden Abbildung dargestellt.



Hinweise

Die TLSet-Werte werden beibehalten

Die **TLSet**-Werte werden erhalten. Verwenden Sie TLClr, um eine Werkzeugdefinition zu löschen.

Verwandte Befehle

Tool, Arm, ArmSet, Point Expression, TLClr, TLSet Function

Beispiel einer TLSet-Anweisung

Das folgende Beispiel zeigt einen hilfreichen Test, der vom Befehlseingabefenster aus ausgeführt werden kann. Dieser Test dient dazu, den Unterschied zwischen Bewegungen mit einem definierten und einem nicht definierten Werkzeug zu verdeutlichen.

```
> TLSet 1, XY(100, 0, 0, 0)  'Definiert das Werkzeugkoordinatenystem für
                             'Werkzeug 1 (plus 100 mm in X-Richtung
                             'vom Armkoordinatensystem)
> Tool 1  'Wählt Werkzeug 1 wie durch TLSet definiert
> TGo P1  'Positioniert die Spitze des Werkzeugs 1 auf P1
> Tool 0  'Sagt dem Roboter, dass er für zukünftige Bewegungen kein
           Werkzeug verwenden soll
> Go P1   'Positioniert den Mittelpunkt der U-Achse auf P1
```

TLSet-Funktion

F

Gibt einen Punkt aus, der die Werkzeugdefinition für das spezifizierte Werkzeug enthält.

Syntax

TLSet(*toolNumber*)

Parameter

toolNumber Integer-Ausdruck, der die Nummer des auszugebenden Werkzeuges darstellt.

Rückgabewerte

Ein Punkt, der die Werkzeugdefinition beinhaltet.

Verwandte Befehle

TLSet-Anweisung

Beispiel einer TLSet-Funktion

```
P1 = TLSet(1)
```

TMOut-Anweisung

> S

Spezifiziert die Anzahl von Sekunden, die verstreichen können, bevor ein Zeitüberlauf gemeldet wird, weil die mit dem Wait-Befehl spezifizierte Bedingung noch nicht erfüllt ist.

Syntax

TMOut *seconds*

Parameter

seconds Reeller Ausdruck, der darstellt, wieviele Sekunden abzuwarten sind, bis ein Zeitüberlauf eintritt. Der gültige Bereich umfasst 0 bis 2147483 Sekunden, in 1-Sekunden-Intervallen.

Beschreibung

TMOut setzt die Zeit fest, die gewartet werden soll (wenn der Wait-Befehl verwendet wird), bis ein Zeitüberlauf auftritt. Wenn ein Zeitüberlauf von 0 Sekunden spezifiziert ist, wird der Zeitüberlauf ausgeschaltet. In diesem Fall wartet der Wait-Befehl für unbestimmte Zeit, bis die festgelegte Bedingung erfüllt ist.

Der vorgegebene Erstwert für **TMOut** ist 0.

Verwandte Befehle

In, MemSw, OnErr, Sw, TW, Wait

Beispiel einer TMOut-Anweisung

```
TMOut 5
Wait MemSw(0) = On
If TW Then
    Print "Time out occurred"
EndIf
```

TMove-Anweisung

> S

Führt im aktuellen Werkzeug-Koordinatensystem eine linearinterpolierte relative Bewegung aus.

Syntax

TMove *destination* [**ROT**] [**CP**] [*searchExpr*] [*!...!*]

Parameter

destination	Der Zielort einer Bewegung, die einen Punktausdruck verwendet.
ROT	Optional. :Bestimmt Geschwindigkeit / Beschleunigung / Verzögerung zugunsten der Werkzeugrotation.
CP	Optional. Spezifiziert CP-Bewegungen.
searchExpr	Optional. Ein Till- oder Find-Ausdruck. **Till** | **Find** **Till Sw**(*expr*) = {**On** | **Off**} **Find Sw**(*expr*) = {**On** | **Off**}
!...!	Optional. Parallelbearbeitungsanweisungen können hinzugefügt werden, um E/A-Befehle und andere Befehle während der Bewegung auszuführen.

Beschreibung

Führt im aktuellen Werkzeug-Koordinatensystem eine linearinterpolierte, relative Bewegung aus.

Armausrichtungsattribute, die unter dem Punktausdruck *destination* festgelegt wurden, werden ignoriert. Der Manipulator behält die aktuellen Armausrichtungspunkte bei. Bei einem 6-Achsmanipulator werden die Armausrichtungsattribute jedoch automatisch so geändert, dass die Achsenlaufweite so klein wie möglich ist.

TMove verwendet den Geschwindigkeitswert aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS. Lesen Sie den Abschnitt *TMove mit CP verwenden* für Informationen über die Beziehung Geschwindigkeit-Beschleunigung und Beschleunigung-Verzögerung. Wenn jedoch der ROT-Parameter verwendet wird, verwendet **TMove** den Geschwindigkeitswert aus SpeedR und die Beschleunigungs- und Verzögerungswerte aus AccelR. In diesem Fall haben die Geschwindigkeitswerte aus SpeedS und die Beschleunigungs- und Verzögerungswerte aus AccelS keine Wirkung.

Wenn der Bewegungsabstand bei 0 liegt und nur die Werkzeugausrichtung verändert wird, tritt für gewöhnlich ein Fehler auf. Wenn jedoch der ROT-Parameter verwendet wird und die Beschleunigung und Verzögerung der Werkzeugrotation bevorrechtigt werden, ist eine fehlerfreie Bewegung möglich. Wenn keine Ausrichtungsänderung durch den ROT-Parameter vorliegt und die Bewegungsdistanz nicht bei 0 liegt, tritt ein Fehler auf.

Auch wenn die Werkzeugrotation im Vergleich zur Bewegungsdistanz groß ist und wenn die Rotationsgeschwindigkeit die spezifizierte Geschwindigkeit des Manipulators überschreitet, tritt ein Fehler auf. In diesem Fall reduzieren Sie die Geschwindigkeit oder hängen Sie den ROT-Parameter an, um die Rotationsgeschwindigkeit / -beschleunigung / -verzögerung zu bevorrechtigen.

Die Till-Bedingung wird verwendet, um TMove durch Verzögerung und Stoppen des Roboters an einer Zwischenposition des Verfahrweges abzuschließen, wenn die aktuelle Till-Bedingung erfüllt ist.

Die Find-Bedingung wird verwendet, um einen Punkt in FindPos zu speichern, wenn die Find-Bedingung während der Bewegung erfüllt wird.

Wenn Till verwendet wird und die Till-Bedingung erfüllt wurde, hält der Manipulator sofort an und der Bewegungsbefehl ist beendet. Wenn die Till-Bedingung nicht erfüllt wird, bewegt sich der Manipulator zu seiner Zielposition.

Wenn Find verwendet wird und die Find-Bedingung erfüllt wurde, wird die aktuelle Position gespeichert. Lesen Sie den Abschnitt *Find* für weitere Informationen hierzu.

Wenn Parallelbearbeitung verwendet wird, können andere Prozesse parallel zu dem Bewegungsbefehl ausgeführt werden.

Hinweise

TMove mit CP verwenden

Der CP-Parameter veranlasst den Arm, zum Zielpunkt (*destination)* zu fahren, ohne langsamer zu werden oder an dem durch *destination* definierten Punkt anzuhalten. Dadurch wird es dem Benutzer ermöglicht, eine Serie von Bewegungsbefehlen miteinander zu verketten, was den Arm veranlasst, sich entlang eines kontinuierlichen Weges zu bewegen und während dieser Bewegung eine bestimmte Bewegung einzuhalten. Der T**Move**-Befehl ohne CP veranlasst den Arm immer dazu, bis zum vollständigen Halt herunterzubremsen, bevor er das Punktziel *destination* erreicht hat.

Verwandte Befehle

AccelS, Find, !....! Parallelbearbeitung, Punktzuweisung, SpeedS, TGo, Till, Tool

Beispiel einer TMove-Anweisung

```
> TMove XY(100, 0, 0, 0) 'Verfährt 100 mm in X-Richtung
                           '(im Werkzeug-Koordinatensystem)
Function Test

  Speed 50
  Accel 50, 50
  SpeedS 100
  AccelS 1000, 1000
  Power High

  Tool 0
  P1 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L

  Go P1
  Print Here
  TMove XY(0, 0, -30, 0)
  Print Here

  Go P2
  Print Here
  TMove XY(0, 0, -30, 0)
  Print Here

Fend

[Output]
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /R /0
 X:  300.000 Y:  300.000 Z:  -20.000 U:    0.000 V:    0.000 W:    0.000 /L /0
 X:  300.000 Y:  300.000 Z:  -50.000 U:    0.000 V:    0.000 W:    0.000 /L /0
```

Tmr-Funktion

F

Timer-Funktion, die die Zeit, seit der Timer gestartet wurde in Sekunden angibt.

Syntax

Tmr(*timerNumber*)

Parameter

timerNumber	Integer-Ausdruck, der darstellt, von welchem der 16 Timer die Zeit überprüft werden soll.

Rückgabewerte

Laufzeit des angegebenen Timers, dargestellt als reelle Zahl in Sekunden. Der Timer deckt den Bereich von 0 bis ca. 1.7E+31 ab. Die Timer-Auflösung beträgt 0,001 Sekunden.

Beschreibung

Gibt die Zeit, die vergangen ist, seit der spezifizierte Timer gestartet wurde, in Sekunden aus. 16 Timer sind verfügbar, mit den Nummern 0 – 15.

Die Timer werden mit TmReset auf Null gesetzt.

```
Real overhead

TmReset 0
overHead = Tmr(0)
```

Verwandte Befehle

TmReset

Beispiel einer Tmr-Funktion

```
TmReset 0          'Setzt den Timer 0 zurück
For I=1 To 10      'Führt Operationen zehnmal durch
    GoSub Cycle
Next
Print Tmr(0) / 10  'Berechnet die Zykluszeit und zeigt sie an
```

TMReset-Anweisung

S

Der Timer wird unter Verwendung der Tmr-Funktion zurückgesetzt.

Syntax

TmReset *timerNumber*

Parameter

timerNumber	Integer-Ausdruck von 0 – 15, der spezifiziert, welcher der 16 Timer zurückgesetzt werden soll.

Beschreibung

Setzt den Timer zurück und startet ihn unter Verwendung von *timerNum*ber. 16 Timer sind verfügbar, mit den Nummern 0 bis 15.

Verwenden Sie die Tmr-Funktion, um die Laufzeit für einen spezifizierten Timer auszugeben.

Verwandte Befehle

Tmr

Beispiel einer TmReset-Anweisung

```
TmReset 0           'Setzt den Timer 0 zurück
For I=1 To 10       'Führt Operationen zehnmal durch
    GoSub CYL
Next
Print Tmr(0)/10  'Berechnet die Zykluszeit und zeigt sie an
```

Toff-Anweisung

S

Schaltet die Anzeige der Ausführungszeile in der Sieben-Segment-LED aus.

Syntax

Toff

Beschreibung

Die Ausführungszeile in der Sieben-Segment-LED wird nicht mehr angezeigt.

Verwandte Befehle

Ton

Beispiel einer Toff-Anweisung

```
Function main
  Ton MyTask
  ...
  Toff
Fend
```

Tool-Anweisung

Wählt das aktuelle Werkzeug aus oder zeigt es an.

> S

Syntax

(1) Tool *toolNumber*
(2) Tool

Parameter

toolNumber Optional. Integer-Ausdruck von 0 bis 3, der darstellt, welche von 4 Werkzeugdefinitionen mit den anstehenden Bewegungsbefehlen verwendet werden soll.

Rückgabewerte

Zeigt das aktuelle **Werkzeug** (Tool) an, wenn keine Parameter verwendet werden.

Beschreibung

Tool wählt das Werkzeug aus, das durch die Werkzeugnummer (toolNum) spezifiziert ist. Wenn die Werkzeugnummer 0 ist, ist kein Werkzeug ausgewählt und alle Bewegungen werden in Bezug auf den Mittelpunkt der Greiferachse durchgeführt. Wenn jedoch der Werkzeugeintrag 1, 2 oder 3 ausgewählt ist, wird die Bewegung in Bezug auf das Ende des Werkzeuges, wie in der Werkzeugdefinition definiert, durchgeführt.

Hinweis

Ausschalten des Hauptstroms und die Auswirkungen auf die Werkzeugwahl:

Das Ausschalten des Hauptstroms ändert die Auswahl des Werkzeug-Koordinatensystems nicht.

Verwandte Befehle

TGo, TLSet, TMove

Beispiel einer Tool-Anweisung

Das folgende Beispiel zeigt einen hilfreichen Test, der vom Befehlseingabefenster aus ausgeführt werden kann. Dieser Test dient dazu, den Unterschied zwischen Bewegungen mit einem definierten und einem nicht definierten Werkzeug zu verdeutlichen.

```
>tlset 1, 100, 0, 0, 0  'Definiert das Werkzeug-Koordinatensystem für
                         'Werkzeug 1 (plus 100 mm in X-Richtung
                         'vom Armkoordinatensystem)
>tool 1    'Wählt Werkzeug 1 wie durch TLSet definiert
>tgo P1   'Positioniert die Spitze des Werkzeugs 1 auf P1
>tool 0  'Sagt dem Roboter, dass er für zukünftige Bewegungen kein
Werkzeug verwenden soll
>go P1   'Positioniert den Mittelpunkt der U-Achse auf P1
```

Tool-Funktion

F

Gibt die aktuelle Werkzeugnummer aus.

Syntax

Tool

Rückgabewerte

Integer, das die aktuelle Werkzeugnummer beinhaltet.

Verwandte Befehle

Tool-Anweisung

Beispiel einer Tool-Funktion

```
Integer savTool

savTool = Tool
Tool 2
Go P1
Tool savTool
```

Ton-Anweisung

Der angegebene Task schaltet die Debug-Ablaufverfolgung ein.

S

Syntax

Ton *taskIdentifier*
Ton

Parameter

taskIdentifier **Spezifiziert den Tasknamen oder die Tasknummer.**
Ein Taskname ist der Funktionsname, der in einer Xqt-Anweisung verwendet wird, oder eine Funktion, die vom Run- oder vom Benutzerfenster aus gestartet wird. Wenn ein Integer-Ausdruck verwendet wird, umfasst der Bereich für normale Tasks 1 bis 16 und für Traptasks 257 bis 261.

Beschreibung

Die Ausführungszeile von Task 1 wird im Ausgangsstatus angezeigt.
Die **Ton**-Anweisung zeigt die Ausführungszeile des spezifierten Tasks in der Sieben-Segment-LED an. Wenn *taskIdentifier* weggelassen wird, wird die Ausführungszeile mit der Ausführung der **Ton**-Anweisung in der Sieben-Segment-LED angezeigt.

Verwandte Befehle

Toff

Beispiel einer Ton-Anweisung

```
Function main
  Ton MyTask
  ...
  Toff
Fend
```

Trap-Anweisung

Definiert Traps und was beim Auftreten von Traps geschehen soll.

S

Syntax

Trap *trapNumber*, *condition* **GoTo** { *label*}
Trap *trapNumber*, *condition* **Call** *funcname*
Trap *trapNumber*, *condition* **Xqt** *funcname*
Trap *trapNumber*

Parameter

trapNumber	Integer von 1 bis 4, das darstellt, welche von 4 Trapnummern verwendet werden soll. (SPEL+ unterstützt bis zu 4 aktive Traps zur selben Zeit.)
condition	Bei dieser Bedingung handelt es sich um einen Boolean-Ausdruck. Die folgenden Funktionen und Operatoren können für den *condition*-Ausdruck verwendet werden: **Funktionen:** Ctr, MemIn, MemSw, In, InW, Oport, Out, OutW, Sw **Operatoren:** And, Or, Xor, +, * **Andere:** Klammern, um verschiedenen Operationen und Variablen Prioriät einzuräumen.
label	Das Label, an das die Programmausführung übertragen wird, wenn die Trap-Bedingung erfüllt wurde.
funcName	Die Funktion, die ausgeführt wird, wenn Call oder Xqt verwendet werden und die Trap-Bedingung erfüllt wurde. Die Funktion mit dem Argument kann nicht spezifiziert werden.

Hinweis

Die Funktionsweise **Trap Call** aus RC+ 4.x oder älteren Versionen wurde in RC+ 5.0 zu **Trap Xqt** geändert.
Die Funktionsweise **Trap GoSub** aus RC+ 4.x oder älteren Versionen wurde entfernt. Verwenden Sie **Trap Call** statt **Trap GoSub**.

Beschreibung

Ein Trap führt eine Interruptbearbeitung aus, die durch GoTo, Call oder Xqt spezifiziert ist, wenn die spezifizierte Bedingung erfüllt ist.
Sobald der Interrupt-Prozess ausgeführt wurde, wird die Trap-Einstellung gelöscht. Wenn derselbe Interrupt-Prozess noch einmal benötigt wird, muss der Trap-Befehl ihn erneut ausführen.

Um eine Trap-Einstellung zu löschen, führen Sie einfach den Trap-Befehl nur mit dem *trapNumber*-Parameter durch. Beispielsweise löscht „Trap 3“ den Trap Nr. 3.
Wenn die Funktion, die Trap GoTo ausgeführt hat, beendet wird, wird Trap GoTo automatisch abgebrochen.
Wenn der deklarierte Task endet, wird Trap Call abgebrochen.
Trap Xqt wird abgebrochen, wenn alle Tasks angehalten haben.

Wenn GoTo spezifiziert ist

Der Befehl, der ausgeführt wird, wird wie unten beschrieben verarbeitet. Dann verzweigt die Steuerung an die spezifizierte Zeilennummer oder das spezifizierte Label.
- Jede Armbewegung hält sofort an.
- Der Warte-Status durch die Warte- oder Eingabebefehle wird nicht fortgesetzt.
- Alle anderen Befehle beenden die Ausführung, bevor die Steuerung verzweigt.

Wenn Call spezifiziert ist
Nachdem derselbe Vorgang wie unter GoTo beschrieben ausgeführt wurde, verzweigt die Steuerung an die spezifizierte Zeilennummer oder das spezifizierte Label.
Sobald die Funktion endet, kehrt die Programmausführung zur nächsten Anweisung zurück, die auf die Anweisung folgt, bei der die Programmunterbrechung aufgetreten ist. Call-Anweisungen können in der Trap-Prozessfunktion nicht verwendet werden.
Wenn in der Trap-Prozessfunktion ein Fehler auftritt, ist die Fehlerbehandlung mit OnErr ungültig und ein Fehler tritt auf.

Wenn Xqt spezifiziert ist
Die Programmsteuerung führt die spezifizierte Funktion als einen Interruptbearbeitungstask aus. In diesem Fall wartet der Task, der den Trap-Befehl ausführt, nicht darauf, dass die Trap-Funktion vollendet ist, sondern fährt mit der Ausführung fort.

Verwandte Befehle

Call, Era, Erl, Err, Ert, ErrMsg$, GoSub, GoTo, OnErr, Xqt

Beispiel einer Trap-Anweisung

Beispiel 1: Fehlerprozess durch den Benutzer definiert.
Sw(0) Eingang wird als benutzerdefinierte Fehlereingabe betrachtet.

```
Function Main
    Trap 1 Sw(0)=On GoTo EHandle  'Definiert Trap
    .
    .
    .
EHandle:
    On 31   'Signalleuchten
    OpenCom #1
    Print #1, "Error is issued"
    CloseCom #1
Fend
```

Beispiel 2: Verwendungsweise wie beim Multi-Tasking

```
Function Main
    Trap 2 MemSw(0) = On Or MemSw(1) = On Call Feeder
    .
    .
    .
Fend
.
Function Feeder
  Select TRUE
    Case MemSw(0) = On
      MemOff 0
      On 2
    Case MemSw(1) = On
      MemOff 1
      On 3
  Send

  ' Definiert den Trap für den nächsten Zyklus
  Trap 2 MemSw(0) = On Or MemSw(1) = On Xqt Feeder
Fend
```

Trim$-Funktion

F

Gibt eine Zeichenkette aus, die mit der definierten Zeichenkette identisch ist, wobei die vorangehenden oder abschließenden Leerzeichen nicht angezeigt werden.

Syntax

Trim$(*string*)

Parameter

string Zeichenkettenausdruck.

Rückgabewerte

Spezifizierte Zeichenkette, bei der die vorangehenden und abschließenden Leerzeichen entfernt wurden.

Verwandte Befehle

LTrim$, RTrim$

Beispiel einer Trim$-Funktion

```
str$ = "  data  "
str$ = Trim$(str$)  ' str$ = "data"
```

TW-Funktion

F

Gibt den Status der Befehle Wait, WaitNet und WaitSig aus.

Syntax

TW

Rückgabewerte

Wenn die Wait-Bedingung innerhalb des Zeitintervalls erfüllt wurde, wird Falsch ausgegeben.
Wenn das Zeitintervall bereits abgelaufen ist, wird Wahr ausgegeben.

Beschreibung

Die Timer-Wait-Funktion **TW** gibt den Status der vorangehenden Wait-Bedingung mit Zeitintervall als Falsch (Wait-Bedingung erfüllt) oder Wahr (Zeitintervall abgelaufen) aus.

Verwandte Befehle

TMOut, Wait

Beispiel einer TW-Funktion

```
Wait Sw(0) = On, 5 'Wartet bis zu 5 Sekunden darauf, dass Eingang 0
eingeschaltet wird
If TW = True Then
    GoTo TIME_UP   'GoTo TIME_UP nach 5 Sekunden
EndIf
```

Ubound-Funktion

F

Gibt den größten verfügbaren Index für die angegebene Dimension einer Matrix aus.

Syntax

UBound *(arrayName [, dimension])*

Parameter

arrayName	Name der Feldvariablen; folgt den Standardkonventionen für Variablenbenennung.
dimension	Optional. Integer-Ausdruck, der signalisiert, welche Obergrenze der Abmessungen ausgegeben wird. Verwenden Sie 1 für die erste Dimension, 2 für die zweite und 3 für die dritte. Wenn *dimension* ausgelassen wird, wird von Dimension 1 ausgegangen.

Verwandte Befehle

Redim

Beispiel einer UBound-Funktion

```
Integer i, a(10)

For i=0 to UBound(a)
    a(i) = i
Next
```

Ucase$-Funktion

F

Gibt eine in Großbuchstaben umgewandelte Zeichenkette aus.

Syntax

UCase$ (*string*)

Parameter

string Zeichenkettenausdruck.

Rückgabewerte

Die konvertierte Großbuchstaben-Zeichenkette.

Verwandte Befehle

LCase$, LTrim$, Trim$, RTrim$

Beispiel einer UCase$-Funktion

```
str$ = "Data"
str$ = UCase$(str$)  ' str$ = "DATA"
```

Val-Funktion

F

Konvertiert eine aus Zahlen bestehende Zeichenkette in ihren numerischen Wert und gibt diesen Wert aus.

Syntax

Val*(string)*

Parameter

string Zeichenkettenausdruck, der nur aus numerischen Zeichen besteht. Die Zeichenkette kann außerdem ein Präfix enthalten: &H (hexadezimal), &O (oktal), oder &B (binär).

Rückgabewerte

Gibt ein Integer- oder Fließkommaergebnis aus, abhängig von der Zeichenketteneingabe. Wenn die Zeichenketteneingabe ein Dezimalpunktzeichen enthält, wird die Zahl in eine Fließkommazahl umgewandelt. Andernfalls ist der Ausgabewert ein Integer.

Beschreibung

Val konvertiert eine Zeichenkette aus Zahlen in einen numerischen Wert. Das Ergebnis kann ein Integer oder eine Fließkommazahl sein. Wenn die Zeichenkette, die dem Val-Befehl übergeben wird, ein Dezimalzeichen enthält, wird eine Fließkommazahlausgegeben. Andernfalls wird ein Integer ausgegeben.

Verwandte Befehle

Abs, Asc, Chr$, Int, Left$, Len, Mid$, Mod, Right$, Sgn, Space$, Str$

Beispiel einer Val-Funktion

Das folgende Beispiel zeigt ein Programm, das verschiedene Zeichenketten in Zahlen umwandelt und diese dann am Bildschirm ausgibt.

```
Function ValDemo
    String realstr$, intstr$
    Real realsqr, realvar
    Integer intsqr, intvar

    realstr$ = "2.5"
    realvar = Val(realstr$)
    realsqr = realvar * realvar
    Print "The value of ", realstr$, " squared is: ", realsqr

    intstr$ = "25"
    intvar = Val(intstr$)
    intsqr = intvar * intvar
    Print "The value of ", intstr$, " squared is: ", intsqr
Fend
```

Ein weiteres Beispiel für eine Ausgabe am Befehlseingabefenster:

```
> Print Val("25.999")
25.999
>
```

Wait-Anweisung

> S

Veranlasst das Programm, entweder eine bestimmte Zeit zu warten, oder solange zu warten, bis die spezifizierte Eingabebedingung (unter Verwendung der Befehle MemSw oder Sw) erfüllt ist. (An Stelle von Sw kann auch Oport verwendet werden, um die Hardware-Ausgänge zu überprüfen.)

Syntax

(1) Wait *time*
(2) Wait *inputCondition*
(3) Wait *inputCondition, time*

Parameter

time Reeller Ausdruck zwischen 0 und 2.147.483, der den Zeitraum darstellt, der abgewartet werden soll, wenn der Wait-Befehl verwendet wird, um die Wartezeit zeitlich zu definieren. Die Zeit wird in Sekunden angegeben. Das kleinste Inkrement beträgt 0,01 Sekunden.

inputCondition Diese Bedingung muss den Wert Wahr oder Falsch zurückgeben. Die folgenden Funktionen und Operatoren können für die inputCondition (Eingabebedingung) verwendet werden:

Funktionen: Ctr, In, InW, Lof, MemIn, MemSw, Lof, Motor, OPort, Out, OutW, Sw

Operatoren: And, Mask, Or, Xor, +, *

Andere: Klammern, um verschiedenen Operationen und Variablen Prioriät einzuräumen.

Beschreibung

(1) Wait mit Zeitintervall
Als Timer verwendet, veranlasst die **Wait**-Anweisung, das Programm, für eine bestimmte Zeitanzuhalten und führt dann das Programm weiter aus.

(2) Wait mit Eingangsbedingungen ohne Zeitintervall:
Wenn der **Wait**-Befehl als bedingte **Wait**-Sperre verwendet wird, bewirkt er, dass das Programm warten muss, bis angegebene Bedingungen erfüllt worden sind. Wenn nach TMOut das Zeitintervall abgelaufen ist und die **Wait**-Bedingungen noch nicht erfüllt wurden, tritt ein Fehler auf. Außerdem kann der Benutzer mehrere Bedingungen mit einem einzelnen Wait-Befehl überprüfen, indem er den And-, Mask-, Or- oder Xor-Befehl verwendet. (Mehr über **Wait** erfahren Sie im Beispielabschnitt.)

(3) Wait mit Eingangsbedingung und Zeitintervall:
Spezifiziert **Wait**-Bedingung und Zeitintervall. Wenn eine der Wait-Bedingungen erfüllt wurde oder das Zeitintervall abgelaufen ist, springt die Programmsteuerung zum nächsten Befehl. Verwenden Sie Tw, um zu prüfen, ob die Wait-Bedingung erfüllt wurde, oder ob das Zeitintervall abgelaufen ist.

Hinweise

Spezifikation eines Zeitüberlaufs für den Gebrauch mit Wait

Wenn der **Wait**-Befehl ohne Zeitintervall verwendet wird, kann ein Zeitüberlauf spezifiziert werden. So wird ein Zeitlimit für das Warten auf den definierten Befehl gesetzt. Der Zeitüberlauf wird durch den TMOut-Befehl gesetzt. Lesen Sie den Abschnitt über den TMOut-Befehl für weitere Informationen. (Die Vorgabeeinstellung für TMOut ist 0. Das bedeutet, dass keine Zeitsperre gesetzt ist.)

Verwandte Befehle

In, InBCD, MemIn, MemOff, MemOn, MemOut, MemSw, Off, On, Oport, Out, OutW, Sw, TMOut

Beispiel einer Wait-Anweisung

Das folgende Beispiel zeigt zwei Tasks. Jeder der beiden Tasks kann Bewegungsbefehle initiieren. Jedoch wird ein Sicherungsmechanismus zwischen den beiden Tasks verwendet, um sicherzustellen, dass ein Task erst dann die Kontrolle über die Bewegungsbefehle des Roboters erhält, wenn der andere Task deren Verwendung abgeschlossen hat. Dadurch können zwei Tasks Bewegungsbefehle korrekt, geordnet und vorhersehbar ausführen. MemSw wird in Kombination mit dem Wait-Befehl verwendet, um zu warten, bis der 1. Merker den richtigen Wert erreicht hat, von dem an es sicher ist, eine neue Bewegung auszuführen.

```
Function main
    Integer I
    MemOff 1
    Xqt !2, task2
    For I = 1 to 100
       Wait MemSw(1) = Off
       Go P(i)
       MemOn 1
    Next I
Fend

Function task2
Integer i
    For i = 101 to 200
       Wait MemSw(1) = On
       Go P(i)
       MemOff 1
    Next i
Fend


' Warten, bis Eingang 0 eingeschaltet ist
Wait Sw(0) = On

' 60,5 Sekunden warten und fährt dann mit der Ausführung fort
Wait 60.5

' Warten, bis Eingang 0 aus- und Eingang 1 eingeschaltet ist
Wait Sw(0) = Off And Sw(1) = On

' Warten, bis Merker 0 oder Merker 1 eingeschaltet sind
Wait MemSw(0) = On Or MemSw(1) = On

'Eine Sekunde warten und dann Ausgang 1 einschalten
Wait 1; On 1

' Warten, dass die unteren 3 Bits des Eingangsports 0 1 entsprechen.
Wait In(0) Mask 7 = 1
```

WaitNet-Anweisung

S

Wartet darauf, dass die TCP / IP-Port-Verbindung hergestellt wird.

Syntax

WaitNet #*portNumber* [, *timeOut*]

Parameter

portNumber	Integer-Ausdruck für die zu verbindende Portnummer. Der Bereich umfasst 201 bis 208.
timeOut	Optional. Maximale Zeit, die für die Verbindung abgewartet werden soll.

Verwandte Befehle

OpenNet, CloseNet

Beispiel einer WaitNet-Anweisung

In diesem Beispiel lautet die Konfiguration der TCP / IP-Einstellungen in zwei Steuerungen wie folgt:

Steuerung Nr.1:
Port: #201
Host-Name: 192.168.0.2
TCP/IP-Port: 1000

```
Function tcpip
  OpenNet #201 As Server
  WaitNet #201
  Print #201, "Data from host 1"
Fend
```

Steuerung Nr.2:
Port: #201
Host-Name: 192.168.0.1
TCP/IP-Port: 1000

```
Function tcpip
  String data$
  OpenNet #201 As Client
  WaitNet #201
  Input #201, data$
  Print "received '", data$, "' from host 1"
Fend
```

WaitPos-Anweisung

S

Wartet mit der nächsten Ausführung, bis der Roboter sich im CP-Modus bis zur Wait-Position verzögert hat.

Syntax

WaitPos

Verwandte Befehle

Wait, WaitSig, CP

Beispiel einer WaitPos-Anweisung

```
Off 1
CP On
Move P1
Move P2
WaitPos  ' Wartezeit, bis der Roboter verzögert
On 1
CP Off
```

WaitSig-Anweisung

S

Wartet auf ein Signal von einem anderen Task.

Syntax

WaitSig *signalNumber* [, *timeOut*]

Parameter

signalNumber	Integer-Ausdruck, der für die zu empfangende Signalnummer steht. Der Bereich umfasst 0 bis 15.
timeOut	Optional. Reeller Ausdruck, der für die maximale Wartezeit steht.

Beschreibung

Verwenden Sie **WaitSig**, um auf ein Signal eines anderen Tasks zu warten. Das Signal wird erst empfangen, nachdem **WaitSig** begonnen hat. Vorangegangene Signale werden ignoriert.

Verwandte Befehle

Wait, WaitPos, Signal

Beispiel einer WaitSig-Anweisung

```
Function Main
  Xqt SubTask
  Wait 1
  Signal 1
  .
  .
Fend

Function SubTask
  WaitSig 1
  Print "signal received"
  .
Fend
```

Weight-Anweisung

Definiert das Massenträgheitsmoment des Roboterarms bzw. zeigt es an.

> S

Syntax

Weight *payloadWeight* [*, distance*]
Weight

Parameter

payloadWeight	Das Gewicht, das der Greifer tragen muss, in kg.
distance	Der Abstand zwischen dem Rotationsmittelpunkt des zweiten Arms zum Schwerpunkt des Greifers, angegeben in mm. Gilt nur für SCARA-Roboter.

Rückgabewerte

Zeigt die aktuellen **Weight**-Einstellungen an, wenn Parameter ausgelassen werden.

Beschreibung

Spezifiziert Parameter zur Kalkulation der maximalen Beschleunigung von PTP-Bewegungen. Die **Weight**-Anweisung spezifiziert das Gewicht des Greifers und der zu tragenden Teile.

Die Spezifikation der Armlänge (*distance*) ist nur bei SCARA-Robotern notwendig. Die Armlänge ist der Abstand von der zweiten Arm-Rotationsgelenk-Mittellinie zum kombinierten Schwerpunkt aus Hand und Arbeitsstück.

Wenn der Äquivalenzwert des Gewichts des Werkstücks, der aus den spezifizierten Parametern berechnet wird, die maximal erlaubte Nutzlast überschreitet, tritt ein Fehler auf.

Mögliche Fehler

Das Gewicht überschreitet das Maximum.

Wenn das Äquivalenz-Lastgewicht, welches aus dem eingegebenen Wert kalkuliert wird, das maximale Lastgewicht überschreitet, tritt ein Fehler auf.

Möglicher Schaden des Manipulators

Beachten Sie, dass die Spezifikation eines **Weight-**Wertes für eine Roboterhand deutlich unter dem tatsächlichen Gewicht des Werkstücks zu überhöhter Beschleunigung und Verzögerung führen kann. Dies wiederum kann kann schwere Schäden am Manipulator verursachen.

Hinweis

Weight-Werte werden durch Abschalten des Hauptstroms nicht geändert.

Die **Weight**-Werte werden durch Abschalten der Stromzufuhr nicht geändert.

Verwandte Befehle

Accel, Inertia

Beispiel einer Weight-Anweisung

Dieser Weight-Befehl im Befehlseingabefenster zeigt die aktuelle Einstellung an.

```
> weight
2.000, 200.000
>
```

Weight-Funktion

F

Gibt einen Weight-Parameter aus.

Syntax

Weight(*paramNumber*)

Parameter

paramNumber Integer-Ausdruck, der einen der unten angegebenen Werte enthält:

1: Nutzlast-Gewicht

2: Armlänge

Rückgabewerte

Reelle Zahl, welche den Parameterwert beinhaltet.

Verwandte Befehle

Inertia, Weight Statement

Beispiel einer Weight-Funktion

```
Print "The current Weight parameters are: ", Weight(1)
```

Where-Anweisung

Zeigt die aktuellen Roboterpositionsdaten an.

Syntax

Where [*localNumber*]

Parameter

localNumber Optional. Spezifiziert die Nummer des lokalen Koordinatensystems. Local 0 ist die Vorgabeeinstellung.

Verwandte Befehle

Joint, PList, Pulse

Beispiel einer Where-Anweisung

```
> where
WORLD:  X:  350.000 mm  Y:    0 mm  Z:    0 mm  U:    0.000 deg
JOINT:  1:    0.000 deg 2:    0.000 deg 3:    0.000 mm  4:    0.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:        0 pls

> local 1, 100,100,0,0

> where 1
WORLD1: X:  250.000 mm  Y: -100.000 mm  Z:    0 mm  U:    0.000 deg
JOINT:  1:    0.000 deg 2:    0.000 deg 3:    0.000 mm  4:    0.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:        0 pls
```

Wrist-Anweisung

> S

Stellt die Handgelenkausrichtung eines Punktes ein.

Syntax

(1) Wrist *point*, [*Flip* | *NoFlip*]
(2) Wrist

Parameter

point **P***number* oder **P**(*expr*) oder Punktlabel.

Flip | *NoFlip* Steht für die Handgelenkausrichtung.

Rückgabewerte

Wenn beide Parameter ausgelassen werden, wird die Handgelenkausrichtung für die aktuelle Roboterposition angezeigt.
Wenn *Flip* | *NoFlip* weggelassen wird, wird die Handgelenkausrichtung für den spezifizierten Punkt angezeigt.

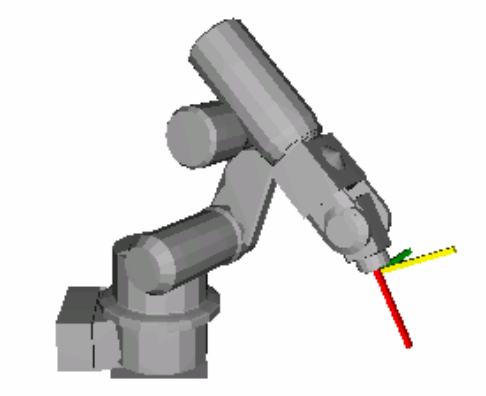
Verwandte Befehle

Elbow, Hand, J4Flag, J6Flag, Wrist-Funktion

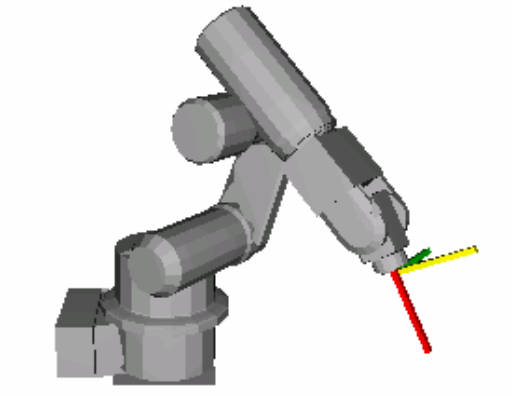
Beispiel einer Wrist-Anweisung

```
Wrist P0, Flip
Wrist P(mypoint), NoFlip

P1 = 320.000, 400.000, 350.000, 140.000, 0.000, 150.000
```



```
Wrist P1, NoFlip
Go P1
```



```
Wrist P1, Flip
Go P1
```

Wrist-Funktion

F

Gibt die Handegelenkausrichtung eines Punktes aus.

Syntax

Wrist [(*point*)]

Parameter

point Optional. **P***number* oder **P**(*expr*) oder Punktlabel oder Punktausdruck. Wird *point* weggelassen, wird die Handgelenkausrichtung der aktuellen Roboterposition ausgegeben.

Rückgabewerte

1 NoFlip (/NF)
2 Flip (/F)

Verwandte Befehle

Elbow-, Hand-, J4Flag-, J6Flag-, Wrist-Anweisung

Beispiel einer Wrist-Funktion

```
Print Wrist(pick)
Print Wrist(P1)
Print Wrist
Print Wrist(P1 + P2)
```

Write-Anweisung

S

Schreibt Zeichen ohne Endzeichen am Zeilenende an einen Kommunikationsport.

Syntax

Write #*portNumber*, *string*

Parameter

portNumber	Anzuschreibende Kommunikationsportnummer.
string	Zeichenkettenausdruck, der in die Datei geschrieben wird.

Beschreibung

Der Unterschied zwischen Write und Print besteht darin, dass bei Write am Zeilenende kein Endzeichen gesetzt wird.

Verwandte Befehle

Print, Read

Beispiel einer Write-Funktion

```
OpenCom #1
For i = 1 to 10
    Write #1, data$(i)
Next i
CloseCom #1
```

WriteBin-Anweisung

Schreibt binäre Daten an einen Kommunikationsport.

S

Syntax

WriteBin #*portNumber*, *data*
WriteBin #*portNumber*, *array(), count*

Parameter

portNumber	Kommunikationsport, aus dem gelesen werden soll.
data	Integer-Ausdruck, der die zu schreibenden Daten enthält.
array()	Name eines Bytes, Integers oder einer Long-Feldvariablen, das bzw. die das zu schreibende Datenbyte empfängt. Spezifiziert eine eindimensionale Feldvariable.
count	Spezifiziert die Anzahl der zu schreibenden Bytes. Die Anzahl muss geringer oder gleich der Anzahl der Matrixelemente sein.

Parameter

portNumber	Kommunikationsport, aus dem gelesen werden soll.
data	Integer-Ausdruck, der die zu schreibenden Daten enthält.

Verwandte Befehle

ReadBin, Write

Beispiel einer WriteBin-Anweisung

```
Integer i, data(100)

OpenCom #1
For i = 0 To 100
  WriteBin #1, i
Next I
WriteBin #1, data(), 100
CloseCom #1
```

Xor-Operator

Führt einen bitweisen Xor-Vorgang (exklusiver Or-Vorgang) an zwei Ausdrücken durch.

Syntax

result = *expr1* **Xor** *expr2*

Parameter

expr1, expr2 Ein numerischer Wert oder ein Variablenname.

result Ein Integer.

Beschreibung

Der Xor-Operator führt den bitweisen Xor-Vorgang zwischen den Werten der Operanden aus. Jedes Bit des Ergebnisses ist der Xor-verknüpfte Wert zweier einander entsprechender Bits der beiden Operanden.

If-Bit in expr1 ist	And-Bit in expr1 ist	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	0

Verwandte Befehle

And, LShift, Not, Or, Rshift

Beispiel eines Xor-Operators

```
>print 2 Xor 6
 4
>
```

Xqt-Anweisung

Beginnt die Ausführung eines Tasks aus einem anderen Task heraus.

S

Syntax

Xqt [*taskNumber,*] *funcName* [**(***argList***)**]

Parameter

taskNumber	Optional. Die Tasknummer des auszuführenden Tasks. Der Bereich für die Tasknummer umfasst 1 bis 16.
funcName	Der Name der auszuführenden Funktion.
argList	Optional. Liste von Argumenten, die an die Funktionsprozedur weitergegeben werden, wenn diese aufgerufen wird. Multiple Argumente werden durch Kommata voneinander getrennt.

Beschreibung

Xqt startet die spezifizierte Funktion und antwortet sofort.

Normalerweise wird der Parameter *taskNumber* nicht benötigt. Wenn *taskNumber* ausgelassen wird, ordnet SPEL+ der Funktion automatisch eine Tasknummer zu. So müssen Sie nicht die Übersicht darüber behalten, welche Tasknummern bereits verwendet werden.

Verwandte Befehle

Function...Fend, Halt, Resume, Quit

Beispiel einer Xqt-Anweisung

```
Function main
    Xqt flash         'Startet die Flash-Funktion als Task 2
    Xqt Cycle(5)       'Startet die Cycle-Funktion als Task 3

    Do
        Wait 3           'Führt den Task 2 drei Sekunden lang aus
        Halt flash     'Unterbricht den Task

        Wait 3
        Resume flash   'Setzt den Task fort
    Loop
Fend

Function Cycle(count As Integer)
    Integer i

    For i = 1 To count
        Jump pick
        On vac
        Wait .2
        Jump place
        Off vac
        Wait .2
    Next i
Fend

Function flash
    Do
        On 1
        Wait 0.2
        Off 1
        Wait 0.2
    Loop
Fend
```

XY-Funktion

F

Gibt einen Punkt von individuellen Koordinaten aus, die in einem Punktausdruck verwendet werden können.

Syntax

XY(*x, y, z, u,* [*v*, *w*]**)**

Parameter

x Reeller Ausdruck, der für die X-Koordinate steht.
y Reeller Ausdruck, der für die Y-Koordinate steht.
z Reeller Ausdruck, der für die Z-Koordinate steht.
u Reeller Ausdruck, der für die U-Koordinate steht.
v Optional bei 6-Achsrobotern. Reeller Ausdruck, der für die V-Koordinate steht.
w Optional bei 6-Achsrobotern. Reeller Ausdruck, der für die W-Koordinate steht.

Rückgabewerte

Ein Punkt, der auf der Basis der definierten Koordinaten festgelegt wird.

Verwandte Befehle

JA, Punktausdruck

Beispiel einer XY-Funktion

```
P10 = XY(60, 30, -50, 45) + P20
```

XYLim-Anweisung

> S

Stellt die zulässigen Grenzwerte des XY-Arbeitsbereichs für den Roboter ein oder zeigt diese an.

Syntax

XYLim *minX, maxX, minY, maxY,* [*minZ*], [*maxZ*]
XYLim

Parameter

minX	Die minimale X-Koordinatenposition, zu der der Manipulator verfahren kann. (Der Manipulator darf zu keiner Position verfahren, deren X-Koordinate kleiner als *minX* ist.)
maxX	Die maximale X-Koordinatenposition, zu der der Manipulator verfahren kann. (Der Manipulator darf zu keiner Position verfahren, deren X-Koordinate größer als *maxX* ist.)
minY	Die minimale Y-Koordinatenposition, zu der der Manipulator verfahren kann. (Der Manipulator darf zu keiner Position verfahren, deren Y-Koordinate kleiner als *minY* ist.)
maxY	Die maximale Y-Koordinatenposition, zu der der Manipulator verfahren kann. (Der Manipulator darf zu keiner Position verfahren, deren Y-Koordinate größer als *maxY* ist.)
minZ	Optional. Die minimale Z-Koordinatenposition, zu der der Manipulator verfahren kann. (Der Manipulator darf zu keiner Position verfahren, deren Z-Koordinate kleiner als *minZ* ist.)
maxZ	Optional. Die maximale Z-Koordinatenposition, zu der der Manipulator verfahren kann. (Der Manipulator darf zu keiner Position verfahren, deren Z-Koordinate größer als *maxZ* ist.)

Rückgabewerte

Zeigt die aktuellen **XYLim**-Werte an, wenn keine Parameter verwendet werden.

Beschreibung

XYLim wird verwendet, um die Grenzwerte des XY-Arbeitsbereichs zu definieren. Viele Robotersysteme ermöglichen es dem Benutzer, Achsenbegrenzungen zu definieren, die SPEL+-Sprache jedoch gestattet es, sowohl die Achsenbegrenzungen als auch die Bewegungsbegrenzungen zu definieren. Der Benutzer hat so die Möglichkeit, einen Arbeitsbereich für die Anwendung anzulegen. (Die Grenzpunkte des Achsenbereichs können ebenfalls mit SPEL definiert werden.)

Der Arbeitsbereich, der mit **XYLim**-Werten angelegt wurde, bezieht sich nur auf die Zielpositionen von Bewegungsbefehlen und nicht auf Bewegungsbahnen von der Start- zur Zielposition. Daher kann sich der Arm während der Bewegung außerhalb des XYLim-Bereichs bewegen. (Somit hat **der XYLim**-Bereich keine Auswirkungen auf den Pulse-Befehl.)

Hinweise

Überprüfung des Arbeitsbereichs ausschalten

Es gibt viele Anwendungen, für die keine Überprüfung der der Begrenzungen des Arbeitsbereichs erforderlich ist. Aus diesem Grund kann diese Überprüfung einfach ausgeschaltet werden. Um die Überprüfung der Arbeitsbereichbegrenzung auszuschalten, setzen Sie die Arbeitsbereichbegrenzungswerte *minX*, *maxX*, *minY* und *maxY* auf 0. Zum Beispiel XYLim 0, 0, 0, 0.

Vorgegebene Arbeitsbereichbegrenzungswerte

Die Vorgabewerte für den XYLim-Befehl lauten „0, 0, 0, 0“. (Überprüfung der Arbeitsbereichbegrenzung ist ausgeschaltet.)

Tipp

Point & Click-Installation für XYLim

EPSON RC+ 5.0 verfügt über ein “Point & Click”-Dialogfenster zur Definition der Grenzwerte der Arbeitsbereichbegrenzung. Die einfachste Methode, die XYLim-Werte einzustellen, ist die Verwendung der Seite XYZ Limits im Robotermanager.

Verwandte Befehle

Range

Beispiel einer XYLim-Anweisung

In diesem einfachen Beispiel vom Befehlseingabefenster werden die aktuellen XYLim-Einstellungen eingestellt und anschließend angezeigt.

```
> xylim -200, 300, 0, 500

> XYLim
-200.000, 300.000, 0.000, 500.000
```

XYLim-Funktion

F

Gibt Punktdaten für die obere oder die untere Grenze des XYLim-Bereichs aus.

Syntax

XYLim(*limit*)

Parameter

limit	Integer-Ausdruck, der spezifiziert, welche Grenze ausgegeben werden soll. 1: Untergrenze. 2: Obergrenze.

Rückgabewerte

Punkt, der die spezifizierten Grenzkoordinaten beinhaltet.

Verwandte Befehle

XYLim-Anweisung

Beispiel einer XYLim-Funktion

```
P1 = XYLim(1)
P2 = XYLim(2)
```

XYLimClr-Anweisung

> S

Löscht die XYLim-Definition.

Syntax

XYLimClr

Verwandte Befehle

XYLim, XYLimDef

Beispiel einer XYLimClr-Funktion

In diesem Beispiel wird die **XYLimClr**-Funktion in einem Programm verwendet:

```
Function ClearXYLim

   If XYLimDef = True Then
      XYLimClr
   EndIf
Fend
```

XYLimDef-Funktion

F

Gibt Informationen darüber aus, ob XYLim definiert wurde.

Syntax

XYLimDef

Rückgabewerte

Wahr, wenn XYLim definiert wurde, sonst Falsch.

Verwandte Befehle

XYLim, XYLimClr

Beispiel einer XYLimDef-Funktion

In diesem Beispiel wird die **XYLimDef**-Funktion in einem Programm verwendet:

```
Function ClearXYLim

   If XYLimDef = True Then
      XYLimClr
   EndIf
Fend
```

SPEL+-Fehlermeldungen

Um Hilfe zu einem SPEL+-Fehler zu erhalten, platzieren Sie den Cursor auf der Fehlermeldung, die auf dem Befehlseingabefenster angezeigt wird, und drücken Sie die Taste F1.

Es gibt die dreizehn folgenden Fehlerarten.

Ereignisse	Interpreter	Punkte
Warnungen	Parser	Feldbus
Steuerung	Motorsteuerung	Hardware
Bedienpult	Servo	EPSON RC+
Teach-Pendant		

Ereignisse

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1	Die Steuerungssoftware wird ausgeführt.			
2	Beendigung wegen Unterspannung des Netzteils.			
3	Die Steuerungssoftware wurde beendet.	Speichert dieses Protokoll, wenn die Steuerung über EPSON RC+ oder vom TP1 aus neu gestartet wird.		
4	Der Speicherbereich der Backup-Variablen wurde gelöscht.			
5	Die Hauptfunktion wurde gestartet.			
7	Die Seriennummer wurde gespeichert.			
8	Ein System-Backup wurde ausgeführt.			
9	Eine System-Wiederherstellung wurde ausgeführt.			
10	Die Roboterparameter wurden initialisiert.			
11	Der Wert des Pulse-Offsets zwischen dem Encoder-Ursprung und dem Home-Sensor (HOFS) wurde geändert. Zusätzlicher Wert ist der Wert der 1. Achse.		Wert der 1. Achse nach der Änderung	Wert der 1. Achse vor der Änderung
12	Der Wert des Pulse-Offsets zwischen dem Encoder-Ursprung und dem Home-Sensor (HOFS) wurde geändert. Zusätzlicher Wert ist der Wert der 2. Achse.		Wert der 2. Achse nach der Änderung	Wert der 2. Achse vor der Änderung
13	Der Wert des Pulse-Offsets zwischen dem Encoder-Ursprung und dem Home-Sensor (HOFS) wurde geändert. Zusätzlicher Wert ist der Wert der 3. Achse.		Wert der 3. Achse nach der Änderung	Wert der 3. Achse vor der Änderung
14	Der Wert des Pulse-Offsets zwischen dem Encoder-Ursprung und dem Home-Sensor (HOFS) wurde geändert. Zusätzlicher Wert ist der Wert der 4. Achse.		Wert der 4. Achse nach der Änderung	Wert der 4. Achse vor der Änderung
15	Der Wert des Pulse-Offsets zwischen dem Encoder-Ursprung und dem Home-Sensor (HOFS) wurde geändert. Zusätzlicher Wert ist der Wert der 5. Achse.		Wert der 5. Achse nach der Änderung	Wert der 5. Achse vor der Änderung
16	Der Wert des Pulse-Offsets zwischen dem Encoder-Ursprung und dem Home-Sensor (HOFS) wurde geändert. Zusätzlicher Wert ist der Wert der 6. Achse.		Wert der 6. Achse nach der Änderung	Wert der 6. Achse vor der Änderung
17	Gehe zum Meldungs-Speichermodus.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
	Keine allgemeine Protokollierung.			
101	Konsolengerät wurde gewechselt.		21:PC 22:Remote 23:OP1	
110	Steuerungsfirmware wurde installiert.		1:Einstellungen 2:Initialisierung 3:Upgrade 4:Recover	
111	IP-Adresse wurde wiederhergestellt.	Sie können dieses Protokoll speichern, wenn die Steuerungsfirmware installiert ist.		
120	PC an die Steuerung angeschlossen.		1:Ethernet 2:USB	
121	TP an die Steuerung angeschlossen.			
122	OP an die Steuerung angeschlossen.			
123	PC von der Steuerung getrennt.			
124	TP von der Steuerung getrennt.			
125	OP von der Steuerung getrennt.			
126	Betriebsart in den Automodus gewechselt.			
127	Betriebsart in den Programmiermodus gewechselt.			
128	Betriebsart in den Teachmodus gewechselt.			

Warnungen

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
501	Trace history ist aktiv.	Aktive Trace History beeinflusst die Systemleistung.		
502	Speicher wurde initialisiert.	Wenn dieser Fehler auftritt, wird der Wert der Global Preserve-Variablen initialisiert. Wechseln Sie die Batterie des CPU-Boards. Tauschen Sie das CPU-Board aus.		
511	Die Spanunng der Pufferbatterie des CPU-Boards ist zu niedrig. Die Batterie muss ausgetauscht werden.	Wechseln Sie die Batterie des CPU-Boards sofort. Lassen Sie die Stromversorgung der Steuerung so lange wie möglich EINgeschaltet, bevor Sie die Batterie wechseln.	Aktueller Wert	Randwert
512	Die 5V Eingangsspannung für das CPU-Board ist zu niedrig.	Wenn die Normalspannung nicht durch eine 5 V-Stromquelle allein erzeugt wird, tauschen Sie die Stromquelle aus.	Aktueller Wert	Randwert
513	Die 24V Eingangsspannung für Motorbremsen, Encoder und Lüfter ist zu niedrig.	Wenn die Normalspannung nicht durch eine 24 V-Stromquelle allein erzeugt wird, tauschen Sie die Stromquelle aus.	Aktueller Wert	Randwert
514	Innentemperatur der Steuerung zu hoch.	Halten Sie die Steuerung so bald wie möglich an und stellen Sie sicher, dass die Umgebungstemperatur der Steuerung nicht hoch ist. Stellen Sie sicher, dass der Filter nicht verstopft ist.	Aktueller Wert	Randwert
515	Die Rotationsgeschwindigkeit des Lüfters ist zu niedrig. (Lüfter 1)	Stellen Sie sicher, dass der Filter nicht verstopft ist. Wenn die Warnung immer noch ansteht, nachdem die Steuerung wieder hochgefahren wurde, wechseln Sie den Lüfter aus.	Aktueller Wert	Randwert
516	Die Rotationsgeschwindigkeit des Lüfters ist zu niedrig. (Lüfter 2)	Stellen Sie sicher, dass der Filter nicht verstopft ist. Wenn die Warnung immer noch ansteht, nachdem die Steuerung wieder hochgefahren wurde, wechseln Sie den Lüfter aus.	Aktueller Wert	Randwert
517	Innentemperatur der Steuerung zu hoch.			
700	Der Typ des Motortreibers passt nicht zum aktuellen Robotermodell. Robotermodell überprüfen. Motortreiber austauschen.	Robotermodell überprüfen.		
736	Der Encoder wurde zurückgesetzt. Steuerung neu starten.	Starten Sie die Steuerung neu.		
737	Unterspannung der Encoderbatterie. Batterie bei eingeschalteter Steuerung auswechseln.	Wechseln Sie die Batterie des Roboters bei eingeschalteter Steuerung.		

Steuerung

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1001	Ablauffehler. Außergewöhnlicher Fehler. Ungültiger Befehlsparameter.			
1002	Angeforderte Daten können nicht bezogen werden. Die Daten sind nicht angelegt oder der Bereich ist ungültig.	Kontrollieren Sie, ob Ziel-E/A, Variablen und Tasks existieren.		
1003	Ungültiges Passwort	Geben Sie das richtige Passwort ein.		
1004	Ausführung mit nicht unterstützter Version nicht möglich.	Verwenden Sie die richtige Version.		
1005	Ausführung mit ungültiger Seriennummer nicht möglich.	Verwenden Sie die Sicherungsdaten für dieselbe Steuerung, um die Konfiguration der Steuerung wiederherzustellen.		
1020	Ausführung im Recovery-Modus nicht möglich.	Fahren Sie die Steuerung normal hoch.		
1021	Ausführung nicht möglich wegen eines Initialisierungsfehlers der Steuerung.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1022	Ausführung ohne offenes Projekt nicht möglich.	Öffnen Sie ein Projekt.		
1023	Ausführung nicht möglich während das Projekt geöffnet ist.	Regenerieren Sie das Projekt.		
1024	Aktivierung von Remote nicht möglich.	Aktivieren Sie den Remote-Eingang.		
1025	Ausführung im Teach-Modus verboten.	Wechseln Sie in den Automatikmodus.		
1026	Ausführung im Teach-Modus nur vom TP möglich.	Wechseln Sie in den Automatikmodus.		
1027	Ausführung im Automodus nicht möglich.	Wechseln Sie in den Programmiermodus.		
1028	Ausführung im Automodus nur von Hauptkonsole möglich.	Wechseln Sie in den Programmiermodus.		
1029	Ausführung vom OP nicht möglich.	Aktivieren Sie den OP-Eingang.		
1030	Wechseln der Betriebsart nicht möglich.	Wechseln Sie mit einer Konsole im Programmiermodus in den Automatikmodus.		
1031	Ausführung nicht möglich während Tasks aktiv sind.	Halten Sie den Task an und führen Sie dann die Ausführung durch.		
1032	Ausführung nicht möglich, weil bereits die maximale Anzahl an normalen Tasks aktiv ist.	Halten Sie den Task an und führen Sie dann die Ausführung durch.		
1033	Ausführung während eines asynchronen Bewegungsbefehls nicht möglich.	Führen Sie die Ausführung durch, wenn die Bewegung abgeschlossen ist.		
1034	Asynchronen Befehl während der Ausführung gestoppt.	Asynchroner Befehl bereits angehalten, als die Steuerung den Befehl Stopp empfing.		
1035	Betriebsart kann nicht gewechselt werden.			
1041	Ausführung im Not-Aus-Zustand nicht möglich.	Beenden Sie den Not-Aus-Zustand.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1042	Ausführung nicht möglich, während die Sicherheitsabschrankung geöffnet ist.	Schließen Sie die Sicherheitsabschrankung.		
1043	Ausführung im Fehlerzustand nicht möglich.	Beenden Sie den Fehlerzustand.		
1044	Ausführung mit anstehendem Remote-Pause-Eingang nicht möglich.	Schalten Sie den Remote-Pause-Eingang aus.		
1045	Die Wartebedingung des Input-Befehls ist die einzige Eingabebedingung.	Die Steuerung hat eine Eingabe empfangen, als sie nicht in der Wartebedingung des Input-Befehls war.		
1046	Ausführung während der Dateiübertragung nicht möglich.	Führen Sie die Ausführung nach der Dateiübertragung durch.		
1047	Ein Befehl, der von anderen Geräten ausgeführt wird, kann nicht abgebrochen werden.	Brechen Sie den Bewegungsbefehl von dem Gerät aus ab, von dem aus er gegeben wurde.		
1048	Ausführung nicht möglich, nachdem Unterspannung erkannt wurde.			
1049	Andere Geräte befinden sich im Programmiermodus.			
1100	Dateifehler. Außergewöhnlicher Fehler. Zugriff auf die Datei nicht möglich.			
1102	Dateifehler. Außergewöhnlicher Fehler. Lese- und Schreibfehler der Registry.			
1103	Datei nicht gefunden.	Überprüfen Sie, ob die Datei existiert.		
1104	Projektdatei wurde nicht gefunden.	Regenerieren Sie das Projekt.		
1105	Objektdatei wurde nicht gefunden.	Regenerieren Sie das Projekt.		
1106	Punktedateien wurden nicht gefunden.	Regenerieren Sie das Projekt.		
1107	Ungültige Version der Objektdatei.	Regenerieren Sie das Projekt.		
1108	Das Datum der Objektdatei passt nicht zur Quelldatei.	Regenerieren Sie das Projekt.		
1109	Nicht genügend Speicherkapazität.	Erhöhen Sie die Speicherkapazität auf dem USB-Speicher.		
1120	Dateifehler. Außergewöhnlicher Fehler. Die Einstellungendatei ist beschädigt.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1121	Dateifehler. Außergewöhnlicher Fehler. Die Projektdatei ist beschädigt.	Regenerieren Sie das Projekt.		
1122	Dateifehler. Außergewöhnlicher Fehler. Die Punktedatei ist beschädigt.	Regenerieren Sie das Projekt.		
1123	Dateifehler. Außergewöhnlicher Fehler. Die E/A-Label-Datei ist beschädigt.	Regenerieren Sie das Projekt.		
1124	Dateifehler. Außergewöhnlicher Fehler. Die benutzerdefinierte Fehlerdatei ist beschädigt.	Regenerieren Sie das Projekt.		
1125	Dateifehler. Außergewöhnlicher Fehler. Die Fehlermeldungsdatei ist beschädigt.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1126	Dateifehler. Außergewöhnlicher Fehler. Die Information der Software-Optionen ist beschädigt.			
1130	Fehlermeldungsfehler. Außergewöhnlicher Fehler. Keine Enträge im Fehlerspeicher gefunden.			
1131	Zugriff auf den USB-Speicher nicht möglich.	Stecken Sie den USB-Speicher richtig ein. Wenn dieser Fehler immer noch auftritt, wenn der USB-Speicher richtig eingesteckt ist, kann die Steuerung den Speicher eventuell nicht erkennen. Stecken Sie einen anderen Speicher ein, um den Betrieb zu überprüfen.		
1132	Dateifehler. Außergewöhnlicher Fehler. Kopieren der Datei fehlgeschlagen.			
1133	Dateifehler. Außergewöhnlicher Fehler. Löschen der Datei fehlgeschlagen.			
1140	Dateifehler. Außergewöhnlicher Fehler. Öffnen der Objektdatei fehlgeschalgen.			
1141	Dateifehler. Außergewöhnlicher Fehler. Öffnen der Projektdatei fehlgeschalgen.			
1142	Dateifehler. Außergewöhnlicher Fehler. Lesen der Projektdatei fehlgeschlagen.			
1143	Dateifehler. Außergewöhnlicher Fehler. Öffnen der Zustandsspeicherdatei fehlgeschlagen.			
1144	Dateifehler. Außergewöhnlicher Fehler. Schreiben der Zustandsspeicherdatei fehlgeschlagen.			
1150	Dateifehler. Außergewöhnlicher Fehler. Der Fehlerspeicher ist ungültig.			
1151	Dateifehler. Außergewöhnlicher Fehler. Entschlüsseln des Fehlerspeichers fehlgeschlagen.			
1152	Dateifehler. Außergewöhnlicher Fehler. Öffnen der Fehlerspeicherdatei fehlgeschlagen.			
1153	Dateifehler. Außergewöhnlicher Fehler. Schreiben der Fehlerspeicherdatei fehlgeschlagen.			
1155	Dateifehler. Außergewöhnlicher Fehler. Öffnen der Einstellungendatei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1156	Dateifehler. Außergewöhnlicher Fehler. Speichern der Einstellungendatei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1157	Dateifehler. Außergewöhnlicher Fehler. Lesen der Einstellungendatei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1158	Dateifehler. Außergewöhnlicher Fehler. Schreiben der Einstellungendatei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1160	MCD-Fehler. Außergewöhnlicher Fehler. Öffnen der MCD-Datei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1161	MCD-Fehler. Außergewöhnlicher Fehler. Lesen der MCD-Datei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1162	MCD-Fehler. Außergewöhnlicher Fehler. Schreiben der MCD-Datei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1163	MCD-Fehler. Außergewöhnlicher Fehler. Speichern der MCD-Datei fehlgeschlagen.	Stellen Sie die Konfiguration der Steuerung wieder her.		
1165	MPD-Fehler. Außergewöhnlicher Fehler. Öffnen der MPD-Datei fehlgeschlagen.			
1166	MPD-Fehler. Außergewöhnlicher Fehler. Lesen der MPD-Datei fehlgeschlagen.			
1167	MPD-Fehler. Außergewöhnlicher Fehler. Schreiben der MPD-Datei fehlgeschlagen.			
1168	MPD-Fehler. Außergewöhnlicher Fehler. Speichern der MPD-Datei fehlgeschlagen.			
1170	MPL-Fehler. Außergewöhnlicher Fehler. Öffnen der MPL-Datei fehlgeschlagen.			
1171	MPL-Fehler. Außergewöhnlicher Fehler. Lesen der MPL-Datei fehlgeschlagen.			
1172	MPL-Fehler. Außergewöhnlicher Fehler. Schreiben der MPL-Datei fehlgeschlagen.			
1173	MPL-Fehler. Außergewöhnlicher Fehler. Speichern der MPL-Datei fehlgeschlagen.			
1175	MAL-Fehler. Außergewöhnlicher Fehler. Öffnen der MAL-Datei fehlgeschlagen.			
1176	MAL-Fehler. Außergewöhnlicher Fehler. Lesen der MAL-Datei fehlgeschlagen.			
1177	MAL-Fehler. Außergewöhnlicher Fehler. Schreiben der MAL-Datei fehlgeschlagen.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1178	MAL-Fehler. Außergewöhnlicher Fehler. Speichern der MAL-Datei fehlgeschlagen.			
1180	MTR-Fehler. Außergewöhnlicher Fehler. Erstellen der MTR-Datei fehlgechlagen.			
1181	PRM-Fehler. Außergewöhnlicher Fehler. Ersetzen der PRM-Datei fehlgeschlagen.			
1185	Dateifehler. Außergewöhnlicher Fehler. Öffnen der Backup-Informationsdatei fehlgeschlagen.			
1186	Dateifehler. Außergewöhnlicher Fehler. Lesen der Backup-Informationsdatei fehlgeschlagen.			
1187	Dateifehler. Außergewöhnlicher Fehler. Schreiben der Backup-Informationsdatei fehlgeschlagen.			
1188	Dateifehler. Außergewöhnlicher Fehler. Speichern der Backup-Informationsdatei fehlgeschlagen.			
1200	Kompilierfehler. Siehe Kompiliermeldung.	Dieser Fehler tritt bei der Kompilierung vom TP aus auf. Beheben Sie den Fehler dort, wo er aufgetreten ist.		
1201	Link-Fehler. Siehe Link-Meldung.	Dieser Fehler tritt bei der Kompilierung vom TP aus auf. Beheben Sie den Fehler dort, wo er aufgetreten ist.		
1500	Kommunikationsfehler.			
1501	Befehl nicht rechtzeitig abgeschlossen.	Führen Sie den Befehl nach einiger Zeit noch einmal aus. Überprüfen Sie die Verbindung zwischen PC und Steuerung.		
1502	Kommunikationsunterbrechung zwischen PC und Steuerung. Verbindung wird wiederhergestellt.	Überprüfen Sie die Verbindung zwischen PC und Steuerung.		
1503	Verbindungsunterbrechung während der Taskausführung.			
1510	IP-Adresse außerhalb des Bereichs.			
1550	Kommunikationsfehler. Außergewöhnlicher Fehler. Ethernet-Initialisierungfehler.			
1551	Kommunikationsfehler. Außergewöhnlicher Fehler. USB-Initialisierungsfehler.			
1552	Kommunikationsfehler. Außergewöhnlicher Fehler. Interner Kommunikationsfehler der Steuerung.			
1553	Kommunikationsfehler. Außergewöhnlicher Fehler. Ungültige Daten erkannt.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1555	Ethernet-Sendefehler.	Überprüfen Sie die Verbindung zwischen PC und Steuerung.		
1556	Ethernet-Empfangsfehler.	Überprüfen Sie die Verbindung zwischen PC und Steuerung.		
1557	USB-Sendefehler.	Überprüfen Sie die Verbindung zwischen PC und Steuerung.		
1558	USB-Empfangsfehler.	Überprüfen Sie die Verbindung zwischen PC und Steuerung.		

Bedienpult

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1600	Initialisierungsfehler. Außergewöhnlicher Fehler. Initialisierung des OP fehlgeschlagen.			
1603	Zeitüberlauf während der Kommunikation mit dem OP.	Überprüfen Sie, ob das Kabel fest eingesteckt ist. Tauschen Sie das Kabel aus.		
1604	Paritätsfehler während der Kommunikation mit dem OP.	Überprüfen Sie, ob das Kabel fest eingesteckt ist. Tauschen Sie das Kabel aus.		
1605	Framing-Fehler während der Kommunikation mit dem OP.	Überprüfen Sie, ob das Kabel fest eingesteckt ist. Tauschen Sie das Kabel aus.		
1606	Überlauf während der Kommunikation mit dem OP.	Überprüfen Sie, ob das Kabel fest eingesteckt ist. Tauschen Sie das Kabel aus.		
1607	Prüfsummenfehler während der Kommunikation mit dem OP.	Überprüfen Sie, ob das Kabel fest eingesteckt ist. Tauschen Sie das Kabel aus.		
1608	Wiederholungsfehler während der Kommunikation mit dem OP.	Überprüfen Sie, ob das Kabel fest eingesteckt ist. Tauschen Sie das Kabel aus.		
1609	OP kann nicht verbunden werden.	Aktualisieren Sie die Steuerungssoftware. Aktualisieren Sie die Firmware für das OP.		

Teach-Pendant

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1700	Initialisierungsfehler. Außergewöhnlicher Fehler. Initialisierung des TP fehlgeschlagen.			
1701	Initialisierungsfehler. Außergewöhnlicher Fehler. Initialisierung des TP fehlgeschlagen.			
1702	Initialisierungsfehler. Außergewöhnlicher Fehler. Initialisierung des TP fehlgeschlagen.			
1703	Dateifehler. Außergewöhnlicher Fehler. Lesen der Bildschirmdatendatei fehlgeschlagen.			
1704	Außergewöhnlicher Fehler. Lesen der Einstellungendatei fehlgeschlagen.			
1706	Außergewöhnlicher Fehler. Öffnen des TP-Ports fehlgeschlagen.			
1708	Außergewöhnlicher Fehler. Lesen der Schlüsseltabelle des TP fehlgeschlagen.			
1709	Außergewöhnlicher Fehler. Ändern der Sprache fehlgeschlagen.			
1800	Die Steuerung ist bereits an einen PC angeschlossen.	Nur ein PC kann an die Steuerung angeschlossen sein.		
1802	Es wurde versucht ohne Verbindung zur Steuerung einen Befehl auszuführen.			
1803	Lesen oder Schreiben der Datei auf dem PC fehlgeschlagen.			
1804	Initialisierungsfehler. Außergewöhnlicher Fehler. Speicherzuweisung auf dem PC fehlgeschlagen.			
1805	Verbindungsfehler. Die Startup-Einstellungen und die Verbindungskabel der Steuerung überprüfen.			
1806	Zeitüberlauf während des Verbindens über Ethernet.			
1807	Zeitüberlauf während des Verbindens über USB.			
1901	Nicht unterstützt. Außergewöhnlicher Fehler. Es wurde versucht einen nicht unterstützten Befehl auszuführen.			
1902	Nicht unterstützt. Außergewöhnlicher Fehler. Ein nicht unterstützter Parameter wurde angegeben.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
1903	Systemfehler. Außergewöhnlicher Fehler.			

Interpreter

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2000	Nicht unterstützt. Außergewöhnlicher Fehler. Es wurde versucht einen nicht unterstützten Befehl auszuführen.	Regenerieren Sie das Projekt.		
2001	Nicht unterstützt. Außergewöhnlicher Fehler. Es wurde versucht einen nicht unterstützten Verfahrbefehl auszuführen.	Regenerieren Sie das Projekt.		
2002	Nicht unterstützt. Außergewöhnlicher Fehler. Es wurde versucht einen nicht unterstützten Conveyor-Befehl auszuführen.	Regenerieren Sie das Projekt.		
2003	Nicht unterstützt. Außergewöhnlicher Fehler. Ein nicht unterstütztes Funktionsargument wurde angegeben.	Regenerieren Sie das Projekt.		
2004	Nicht unterstützt. Außergewöhnlicher Fehler. Ein nicht unterstützter Rückgabewert wurde angegeben.	Regenerieren Sie das Projekt.		
2005	Nicht unterstützt. Außergewöhnlicher Fehler. Ein nicht unterstützter Wait-Befehl wurde angegeben.	Regenerieren Sie das Projekt.		
2006	Nicht unterstützt. Außergewöhnlicher Fehler. Ein nicht unterstützter E/A-Befehl wurde angegeben.	Regenerieren Sie das Projekt.		
2010	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Außerhalb des internen Code-Bereichs.	Regenerieren Sie das Projekt.		
2011	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Funktionsargument fehlerhaft.	Regenerieren Sie das Projekt.		
2012	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Befehlsargument fehlerhaft.	Regenerieren Sie das Projekt.		
2013	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Code kann nicht verarbeitet werden.	Regenerieren Sie das Projekt.		
2014	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Variablentyp-Code kann nicht verarbeitet werden.	Regenerieren Sie das Projekt.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2015	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Stringtyp-Code kann nicht verarbeitet werden.	Regenerieren Sie das Projekt.		
2016	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Variablenkategory-Code kann nicht verarbeitet werden.	Regenerieren Sie das Projekt.		
2017	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Verarbeitung von unzulässigem Code nicht möglich.	Regenerieren Sie das Projekt.		
2018	Objektdateifehler. Außergewöhnlicher Fehler. Projektgenerierung. Variablengröße konnte nicht berechnet werden.	Regenerieren Sie das Projekt.		
2020	Anzahl der Stack-Tabelle überschritten. Funktionsaufruf oder lokale Variable außerhalb des Bereichs.	Stellen Sie sicher, dass keine Funktion unbegrenzt aufgerufen wird. Reduzieren Sie die Tiefe der Aufruffunktion.		
2021	Stack-Größe überschritten. Stack-Fehler. Funktionsaufruf oder lokale Variable überschreitet den Bereich.	Wenn Sie viele lokale Variablen verwenden, vor allem Stringtypvariablen, ersetzen Sie sie durch globale Variablen.		
2022	Stack-Fehler. Außergewöhnlicher Fehler. Erforderliche Daten nicht im Stack gefunden.	Regenerieren Sie das Projekt.		
2023	Stack-Fehler. Außergewöhnlicher Fehler. Unerwartetes Tag auf dem Stack gefunden.	Regenerieren Sie das Projekt.		
2030	Systemfehler. Außergewöhnlicher Fehler. Drive Unit Anzahl übersteigt den Maximalwert.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2031	Systemfehler. Außergewöhnlicher Fehler. Roboteranzahl übersteigt den Maximalwert.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2032	Systemfehler. Außergewöhnlicher Fehler. Unzulässige Tasknummer.	Regenerieren Sie das Projekt.		
2033	Systemfehler. Außergewöhnlicher Fehler. Zu viele Fehler.	Beheben Sie häufig auftretende Fehler.		
2040	Thread-Fehler. Außergewöhnlicher Fehler. Fehler beim Erstellen des Thread.			
2041	Thread-Fehler. Außergewöhnlicher Fehler. Zeitüberlauf beim Erstellen des Thread.			
2042	Thread-Fehler. Außergewöhnlicher Fehler. Zeitüberlauf beim Beenden des Thread.			
2043	Thread-Fehler. Außergewöhnlicher Fehler. Zeitüberlauf beim Beenden des Thread.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2044	Thread-Fehler. Außergewöhnlicher Fehler. Daemon-Prozess Zeitüberlauf.			
2045	Thread-Fehler. Außergewöhnlicher Fehler. Zeitüberlauf beim Fortsetzen des Tasks.			
2046	Thread-Fehler. Außergewöhnlicher Fehler. Zeitüberlauf beim Stoppen des Tasks.			
2047	Thread-Fehler. Außergewöhnlicher Fehler. Zeitüberlauf beim Starten des Tasks.			
2050	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Objektdatei zu groß.	Regenerieren Sie das Projekt.		
2051	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Die Objektdatei kann während der Ausführung nicht gelöscht werden.	Starten Sie die Steuerung neu.		
2052	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Der Objektdatei kann kein Speicher zugewiesen werden.	Starten Sie die Steuerung neu.		
2053	Objektdatei-Aktualisierung. Aktualisierung der Objektdatei.	Führen Sie dieselbe Bearbeitung nach einer Zeit noch einmal durch. Regenerieren Sie das Projekt.		
2054	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Synchronisierung des Projekts. Fehler der Funktions-ID.	Synchronisieren Sie die Dateien des Projekts. Regenerieren Sie das Projekt.		
2055	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Synchronisierung des Projekts. Fehler der ID der lokalen Variablen.	Synchronisieren Sie die Dateien des Projekts. Regenerieren Sie das Projekt.		
2056	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Synchronisierung des Projekts. Fehler der ID der globalen Variablen.	Synchronisieren Sie die Dateien des Projekts. Regenerieren Sie das Projekt.		
2057	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Synchronisierung des Projekts. Fehler der ID der Backup-Variablen.	Synchronisieren Sie die Dateien des Projekts. Regenerieren Sie das Projekt.		
2058	Fehler beim Ausführen der Objektdatei. Außergewöhnlicher Fehler. Variablengröße konnte nicht berechnet werden.	Synchronisieren Sie die Dateien des Projekts. Regenerieren Sie das Projekt.		
2059	Speicherbereich der globalen Variablen überschritten. Speicher für globale Variablen kann nicht zugewiesen werden.	Reduzieren Sie die Zahl der zu verwendenden globalen Variablen.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2070	SRAM-Fehler. Außergewöhnlicher Fehler. SRAM ist nicht gemappt.	Tauschen Sie das CPU-Board aus.		
2071	SRAM-Fehler. Außergewöhnlicher Fehler. Global Preserve Variable kann während der Benutzung nicht gelöscht werden.	Führen Sie dieselbe Bearbeitung nach einer Zeit noch einmal durch. Regenerieren Sie das Projekt.		
2072	Speicherbereichsüberschreitung für Global Preserve Variablen. Speicherzuweisung für Global Preserve Variablen nicht möglich.	Reduzieren Sie die Zahl der zu verwendenden Global Preserve-Variablen.	Maximalgröße	Größe, die Sie versucht haben zu verwenden
2073	SRAM-Fehler. Außergewöhnlicher Fehler. Fehler beim Löschen des Speicherbereichs der Global Preserve Variablen.	Regenerieren Sie das Projekt.		
2074	SRAM-Fehler. Außergewöhnlicher Fehler. Fehler beim Einrichten des Speicherbereichs der Global Preserve Variablen.	Starten Sie die Steuerung neu.		
2100	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Öffnen der Initialisierungdatei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2101	Initialisierungsfehler. Außergewöhnlicher Fehler. Doppelte Initialisierung.			
2102	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler bei der MNG-Initialisierung.			
2103	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler bei der Erstellung eines Events.			
2104	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Einrichten einer Priorität.			
2105	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Einrichten der Stack-Größe.			
2106	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Einrichten eines Interrupt-Prozesses.			
2107	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Starten eines Interrupt-Prozesses.			
2108	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Stoppen eines Interrupt-Prozesses.			
2109	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Beenden des MNG.	Starten Sie die Steuerung neu.		
2110	Initialisierungsfehler. Außergewöhnlicher Fehler. Speicherzuweisungsfehler.	Starten Sie die Steuerung neu.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2111	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Initialisieren einer Bewegung.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2112	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Beenden einer Bewegung.	Starten Sie die Steuerung neu.		
2113	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Mappen des SRAM.	Tauschen Sie das CPU-Board aus.		
2114	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Registrieren des SRAM.	Tauschen Sie das CPU-Board aus.		
2115	Initialisierungsfehler. Außergewöhnlicher Fehler. Die Anzahl der Feldbus-Boards übersteigt den Maximalwert.			
2116	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Initialisieren des Feldbusses.			
2117	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Beenden des Feldbusses.			
2118	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Einleiten einer Bewegung.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2120	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen des Systemspeicherbereichs.	Starten Sie die Steuerung neu.		
2121	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen des Speicherbereichs der Objektdatei.	Starten Sie die Steuerung neu.		
2122	Initialisierungsfehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen des Speicherbereichs des Roboters.	Starten Sie die Steuerung neu.		
2130	MCD-Fehler. Außergewöhnlicher Fehler. Fehler beim Öffnen der MCD-Datei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2131	MCD-Fehler. Außergewöhnlicher Fehler. Fehler beim Mappen der MCD-Datei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2132	PRM-Fehler. Außergewöhnlicher Fehler. PRM-Datei nicht gefunden.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2133	PRM-Fehler. Außergewöhnlicher Fehler. Fehler beim Mappen der PRM-Datei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2134	PRM-Fehler. Außergewöhnlicher Fehler. Fehlerhafter Inhalt der PRM-Datei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2150	Ablauffehler. Außergewöhnlicher Fehler. Tasknummer nicht gefunden.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2151	Ablauffehler. Außergewöhnlicher Fehler. Task wird ausgeführt.			
2152	Ablauffehler. Außergewöhnlicher Fehler. Fehlerhafte Größe des Objektcodes.			
2153	Ablauffehler. Außergewöhnlicher Fehler. Fehlerhafter Verfahrparameter.			
2154	Ablauffehler. Außergewöhnlicher Fehler. Bewegungsschritt wird ausgeführt.			
2155	Ablauffehler. Außergewöhnlicher Fehler. Ausführen der Bewegungsfunktion nicht möglich.			
2156	Ablauffehler. Außergewöhnlicher Fehler. Verfahrdaten sind nicht eingestellt.			
2157	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim Ändern der Verfahrparameter.			
2158	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen des Haltepunktspeicherbereichs.			
2159	Ablauffehler. Außergewöhnlicher Fehler. Haltepunkteanzahl übersteigt den Maximalwert.			
2160	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen der Funktions-ID.			
2161	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen der Adresse der lokalen Variablen.			
2162	Ablauffehler. Außergewöhnlicher Fehler. Puffer zum Speichern der lokalen Variablen zu klein.			
2163	Ablauffehler. Außergewöhnlicher Fehler. Das Ändern der Werte ist nur im Halt-Zustand des Tasks möglich.			
2164	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen der Adresse der globalen Variablen.			
2165	Ablauffehler. Außergewöhnlicher Fehler. Puffer zum Speichern der globalen Variablen zu klein.			
2166	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim Beziehen der Adresse der Global Preserve Variablen.			
2167	Ablauffehler. Außergewöhnlicher Fehler. Puffer zum Speichern der Global Preserve Variablen zu klein.			
2168	Ablauffehler. Außergewöhnlicher Fehler. SRAM ist nicht gemappt.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2169	Ablauffehler. Außergewöhnlicher Fehler. Löschen der Global Preserve Variable während des Ladens der Objektdatei nicht möglich.			
2170	Ablauffehler. Außergewöhnlicher Fehler. Puffer zum Speichern des Strings zu klein.			
2171	Ablauffehler. Außergewöhnlicher Fehler. Starten des Tasks nicht möglich, weil Unterspannung erkannt wurde.			
2172	Ablauffehler. Außergewöhnlicher Fehler. Doppelte Remote-E/A Konfiguration.			
2173	Remote-Einstellungsfehler. Nur einem existierenden Eingang kann eine Remote-Funktion zugewiesen werden.			
2174	Remote-Einstellungsfehler. Nur einem existierenden Ausgang kann eine Remote-Funktion zugewiesen werden.			
2175	Ablauffehler. Außergewöhnlicher Fehler. Remote-Funktion wurde nicht konfiguiert.			
2176	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim Warten auf Event.			
2177	Ablauffehler. Außergewöhnlicher Fehler. Fehler beim System-Backup.			
2178	Ablauffehler. Außergewöhnlicher Fehler. Fehler bei der Systemwiederherstellung.			
2200	Roboter wird bereits verwendet. Ein Bewegungsbefehl kann nicht ausgeführt werden, wenn andere Tasks den Roboter verwenden.	Bewegungsbefehl für den Roboter kann nicht von mehr als einem Task zur selben Zeit ausgeführt werden. Überprüfen Sie das Programm.		
2201	Roboter existiert nicht.	Überprüfen Sie, ob die Einrichtung des Roboters korrekt durchgeführt wird. Stellen Sie die Konfiguration der Steuerung wieder her.		
2202	Fehler des Bewegungssteuerungsmoduls. Außergewöhnlicher Fehler. Unbekannter Fehler wurde ausgegeben.			
2203	Local '0' kann nicht gelöscht werden.	Local '0' kann nicht gelöscht werden. Überprüfen Sie das Programm.		
2204	Ein Arm kann während der Verwendung nicht gelöscht werden.	Der Arm kann nicht gelöscht werden, während er verwendet wird. Stellen Sie sicher, dass der Arm nicht verwendet wird.	Arm, den Sie versucht haben zu löschen	

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2205	Arm '0' kann nicht gelöscht werden.	Arm '0' kann nicht gelöscht werden. Überprüfen Sie das Programm.		
2206	Ein Tool kann während der Verwendung nicht gelöscht werden.	Das Tool kann nicht gelöscht werden, während es verwendet wird. Stellen Sie sicher, dass das Tool nicht verwendet wird.	Tool, das Sie versucht haben zu löschen	
2207	Tool '0' kann nicht gelöscht werden.	Das Tool '0' kann nicht gelöscht werden. Überprüfen Sie das Programm.		
2208	ECP '0' kann nicht gelöscht werden.	ECP '0' kann nicht gelöscht werden. Überprüfen Sie das Programm.		
2209	ECP kann während der Verwendung nicht gelöscht werden.	Der ECP kann während der Verwendung nicht gelöscht werden. Stellen Sie sicher, dass der ECP nicht verwendet wird.	ECP, den Sie versucht haben zu löschen	
2210	'0' kann nicht als Local-Nummer verwendet werden.	Der Befehl, der das Local verarbeitet, kann die Local-Nummer 0 nicht verwenden. Überprüfen Sie das Programm.		
2220	PRM-Fehler. Außergewöhnlicher Fehler. Keine PRM-Dateidaten gefunden.	Starten Sie die Steuerung neu. Stellen Sie die Konfiguration der Steuerung wieder her.		
2221	PRM-Fehler. Außergewöhnlicher Fehler. Fehler beim Flashen der PRM-Datei.	Starten Sie die Steuerung neu. Stellen Sie die Konfiguration der Steuerung wieder her.		
2222	Local-Nummer ist nicht definiert.	Überprüfen Sie die Einstellungen des Locals. Überprüfen Sie das Programm.	Angegebene Local-Nummer	
2223	Local-Nummer nicht gefunden.	Local-Nummern von 1 bis 15 verfügbar. Überprüfen Sie das Programm.	Angegebene Local-Nummer	
2225	CalPls ist nicht definiert.	Überprüfen Sie die CalPls-Einstellungen.		
2226	Arm-Nummer nicht gefunden.	Armnummern 0 bis 3 sind verfügbar. Abhängig von den Befehlen ist die Armnummer 0 nicht verfügbar. Überprüfen Sie das Programm.	Angegebene Armnummer	
2227	Arm-Nummer ist nicht definiert.	Überprüfen Sie die Einstellungen des Arms. Überprüfen Sie das Programm.	Angegebene Armnummer	
2228	Pulse der Home-Position sind nicht definiert.	Überprüfen Sie die HomeSet-Einstellung.		
2229	Tool-Nummer nicht gefunden.	Tool-Nummern 0 bis 3 sind verfügbar. Abhängig von den Befehlen ist die Tool-Nummer 0 nicht verfügbar. Überprüfen Sie das Programm.	Angegebene Tool-Nummer	
2230	Tool-Nummer ist nicht definiert.	Überprüfen Sie die Tool-Einstellungen. Überprüfen Sie das Programm.	Angegebene Tool-Nummer	
2231	ECP-Nummer nicht gefunden.	Tool-Nummern 0 bis 15 sind verfügbar. Abhängig von den Befehlen ist die Tool-Nummer 0 nicht verfügbar. Überprüfen Sie das Programm.	Angegebene ECP-Nummer	

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2232	ECP-Nummer ist nicht definiert.	Überprüfen Sie die ECP-Einstellungen. Überprüfen Sie das Programm.	Angegebene ECP-Nummer	
2233	Keine Achse für Encoder-Reset angegeben.	Stellen Sie sicher, dass Sie eine Achse für Encoder-Reset angegeben.		
2234	Encoder-Reset mit eingeschalteten Motoren nicht möglich.	Schalten Sie den Motor vor dem Zurücksetzen aus.		
2235	XYLIM ist nicht definiert.	Überprüfen Sie die XYLim-Einstellungen. Überprüfen Sie das Programm.		
2236	PRM-Fehler. Außergewöhnlicher Fehler. Fehler beim Übertragen des Inhalts der PRM-Datei zum Statusmodul der Bewegungssteuerung.	Starten Sie die Steuerung neu. Stellen Sie die Konfiguration der Steuerung wieder her.		
2240	Feldindex außerhalb des benutzterdefinierten Bereichs. Zugriff außerhalb der Feldgrenzen nicht möglich.	Überprüfen Sie den Feldindex. Überprüfen Sie das Programm.	Dimensionen überschreiten die Definition.	Angegebener Index
2241	Felddimensionen entsprechen nicht der Definition.	Überprüfen Sie die Felddimensionen. Überprüfen Sie das Programm.		
2242	Null '0' wurde als Divisor verwendet.	Überprüfen Sie das Programm.		
2243	Variablenüberlauf. Die angegebene Variable war oberhalb des erlaubten Bereichs.	Überprüfen Sie den Variablentyp und das Ergebnis der Berechnung. Überprüfen Sie das Programm.		
2244	Variablenunterlauf. Die angegebene Variable war unterhalb des erlaubten Bereichs.	Überprüfen Sie den Variablentyp und das Ergebnis der Berechnung. Überprüfen Sie das Programm.		
2245	Befehl kann nicht mit Fließkommazahl ausgeführt werden.	Dieser Befehl kann nicht für den Real- oder Double-Typ ausgeführt werden. Überprüfen Sie das Programm.		
2246	Die Tan-Funktion kann nicht mit dem angegebenen Wert rechnen.	Überprüfen Sie den angegebenen Wert. Überprüfen Sie das Programm.	Angegebener Wert	
2247	Der angegebene Feldindex ist kleiner als '0'.	Überprüfen Sie den angegebenen Wert. Überprüfen Sie das Programm.	Angegebener Wert	
2248	Feldfehler. Außergewöhnlicher Fehler. Redim kann nur für eine Feldvariable ausgeführt werden.	Sie haben versucht, Redim für eine Variable auszuführen, die keine Feldvariable ist. Regenerieren Sie das Projekt.		
2249	Feldfehler. Außergewöhnlicher Fehler. Preserve kann nur für ein eindimensionales Feld verwendet werden.	Ein anderes als ein eindimensionales Feld wurde als Preserve für Redim angegeben. Regenerieren Sie das Projekt.		
2250	Feldfehler. Außergewöhnlicher Fehler. Fehler beim Berechnen des Variablenspeicherbereichs.	Regenerieren Sie das Projekt.		
2251	Nicht genug Speicher für Redim verfügbar.	Reduzieren Sie die Anzahl der anzugebenden Indizes für Redim. Führen Sie Redium reduziert durch.		
2252	Nicht genug Speicher für ByRef verfügbar.	Reduzieren Sie die Anzahl der Feldindizes, die von ByRef gesehen werden.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2253	Zeichen können nicht mit Werten verglichen werden.	Stellen Sie sicher, dass der String-Typ und der numerische Datentyp nicht verglichen werden. Überprüfen Sie das Programm.		
2254	Die angegebenen Daten liegen außerhalb der Grenzen des Feldes. Zugriff außerhalb der Feldgrenzen nicht möglich.	Überprüfen Sie die Anzahl der Feldindizes und die Daten. Überprüfen Sie das Programm.	Anzahl der Feldindizes	Anzahl der Daten, auf die zugegriffen werden soll.
2255	Variablen-Überlauf oder -Unterlauf. Die angegebene Variable ist außerhalb des Wertebereichs.	Wert, der den Bereich für den Double-Typ überschreitet, ist angegeben. Überprüfen Sie das Programm.		
2256	Der angegebene Feldindex überschreitet den erlaubten Bereich.	Reduzieren Sie die Anzahl der anzugebenden Indizes. Für verfügbare Indizes konsultieren Sie die Online-Hilfe.		
2260	Tasknummer außerhalb des Bereichs.	Für verfügbare Tasknummer konsultieren Sie die Online-Hilfe. Überprüfen Sie das Programm.	Angegebene Tasknummer	
2261	Die angegebene Tasknummer existiert nicht.	Überprüfen Sie das Programm.	Angegebene Tasknummer	
2262	Roboternummer außerhalb des Bereichs.	Die verfügbare Roboternummer ist 1. Überprüfen Sie das Programm.	Angegebene Roboternummer	
2263	Nummer des Ausgangs außerhalb des Bereichs. Die Portnummer oder die Gerätenummer ist außerhalb des Bereichs.	Für verfügbare Ausgangsnummer konsultieren Sie die Online-Hilfe. Überprüfen Sie das Programm.	Angegebene Ausgangsnummer	
2264	Befehlsargument außerhalb des Bereichs. Plausibiliät prüfen. 1: übergebenener Wert. 2: Argument-Reihenfolge	Für verfügbaren Argumentbereich konsultieren Sie die Online-Hilfe. Überprüfen Sie das Programm.	Angegebener Wert	Welche Argument-nummer?
2265	Achsennummer außerhalb des Bereichs.	Achsennummern von 1 bis 6 verfügbar. Überprüfen Sie das Programm.	Angegebene Achsennummer	
2266	Zeit für Wait außerhalb des Bereichs.	Zeit für Wait von 0 bis 2147483 verfügbar. Überprüfen Sie das Programm.	Angegebene Zeit für Wait	
2267	Timer-Nummer außerhalb des Bereichs.	Timer-Nummern von 0 bis 15 verfügbar. Überprüfen Sie das Programm.	Angegebene Timernummer	
2268	Trap-Nummer außerhalb des Bereichs.	Trap-Nummern von 1 bis 4 verfügbar. Überprüfen Sie das Programm.	Angegebene Trap-Nummer	
2269	Sprach-ID außerhalb des Bereichs.	Für verfügbare Sprach-ID konsultieren Sie die Online-Hilfe. Überprüfen Sie das Programm.	Angegebene Sprach-ID	
2270	Der im Parallelprozess angegebene D-Parameterwert ist außerhalb des Bereichs.	D-Parameterwerte von 0 bis 100 verfügbar. Überprüfen Sie das Programm.	Angegebener D-Parameterwert	
2271	Arch-Nummer außerhalb des Bereichs.	Arch-Nummern von 0 bis 7 verfügbar. Überprüfen Sie das Programm.	Angegebene Arch-Nummer	

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2272	Geräte-Nummer außerhalb des Bereichs.	Angegebene Nummer für ein Steuer- oder Anzeigegerät liegt außerhalb des Bereichs. Für verfügbare Geräte-Nummer konsultieren Sie die Online-Hilfe. Überprüfen Sie das Programm.	Angegebene Geräte-Nummer	
2273	Ausgangsdaten außerhalb des Bereichs.	Ausgangsdaten von 0 bis 255 verfügbar. Überprüfen Sie das Programm.	Ausgangsdaten	Welche Nummer für Bytedaten liegt außerhalb des Bereichs?
2274	Asin-Argument außerhalb des Bereichs -1 bis 1. .	Überprüfen Sie das Programm.		
2275	Acos-Argument außerhalb des Bereichs -1 bis 1. .	Überprüfen Sie das Programm.		
2276	Sqr-Argument außerhalb des Bereichs.	Überprüfen Sie das Programm.		
2277	Randomize-Argument außerhalb des Bereichs.	Überprüfen Sie das Programm.		
2278	Sin-, Cos-, Tan-Argument außerhalb des Bereichs.	Überprüfen Sie das Programm.		
2280	Die durch TMOut eingestellte Zeit lief ab, bevor die Wait-Bedingung erfüllt war.	Suchen Sie nach der Ursache für den Zeitüberlauf. Überprüfen Sie, ob der Zeitüberlauf richtig eingestellt ist.	Zeitüberlauf	
2281	Die durch TMOut eingestellte Zeit im WaitSig- oder SyncLock-Befehl ist abgelaufen.	Suchen Sie nach der Ursache für den Zeitüberlauf. Überprüfen Sie, ob der Zeitüberlauf richtig eingestellt ist.	Signalnummer	Zeitüberlauf
2282	Die durch TMOut eingestellte Zeit im WaitNet-Befehl ist abgelaufen.	Suchen Sie nach der Ursache für den Zeitüberlauf. Überprüfen Sie, ob der Zeitüberlauf richtig eingestellt ist.	Portnummer	Zeitüberlauf
2283	Zeitüberlauf. Außergewöhnlicher Fehler. Zeitüberlauf beim Einstellen des Anzeigegeräts.	Starten Sie die Steuerung neu.		
2300	Roboter wird bereits verwendet. Ein Bewegungsbefehl kann nicht ausgeführt werden, wenn andere Tasks den Roboter verwenden.	Bewegungsbefehl für den Roboter kann nicht von mehr als einem Task zur selben Zeit ausgeführt werden. Überprüfen Sie das Programm.	Tasknummer, die den Roboter verwendet	
2302	Es ist nicht möglich einen Call-Befehl in einem Trap-Call-Prozess auszuführen.	Eine andere Funktion kann nicht von der Funktion aufgerufen werden, die von Trap Call aufgerufen wurde. Überprüfen Sie das Programm.		
2303	Es ist nicht möglich einen Call-Befehl in einem Parallelprozess auszuführen.	Überprüfen Sie das Programm.		
2304	Es ist nicht möglich einen XQT-Befehl in einem Parallelprozess auszuführen.	Überprüfen Sie das Programm.		
2305	Es ist nicht möglich einen Call-Befehl vom Online-Fenster auszuführen.			
2306	Es ist nicht möglich einen XQT-Befehl von einem Task auszuführen, der durch Trap XQT gestartet wurde.	Überprüfen Sie das Programm.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2307	Dieser Befehl kann nicht ausgeführt werden, während Tasks aktiv sind.	Überprüfen Sie, ob alle Tasks abgeschlossen sind.		
2308	Motoren können wegen eines krtitischen Fehlers nicht eingeschaltet werden.	Suchen Sie den vorherigen Fehler im Fehlerspeicher und beheben Sie die Ursache. Starten Sie anschließend die Steuerung neu.		
2309	Bei offener Sicherheitsabschrankung kann kein Bewegungsbefehl ausgeführt werden.	Überprüfen Sie den Status der Sicherheitsabschrankung.		
2310	Beim Warten auf Continue kann kein Bewegungsbefehl ausgeführt werden.	Führen Sie Continue oder Stop durch und führen Sie anschließend den Bewegungsbefehl aus.		
2311	Beim Ausführen des Continue-Prozesses kann kein Bewegungsbefehl ausgeführt werden.	Warten Sie, bis Continue abgeschlossen ist und führen Sie anschließend den Bewegungsbefehl aus.		
2312	Bei Not-Aus kann kein Task ausgeführt werden.	Überprüfen Sie den Not-Aus-Status.		
2313	Die Ausführung kann nach dem Schließen der Sicherheitsabschrankung nicht sofort fortgesetzt werden.	Warten Sie 1,5 Sekunden, wenn die Sicherheitsabschrankung geöffnet ist, und führen Sie dann Continue aus.		
2314	Die Ausführung kann bei offener Sicherheitsabschrankung nicht fortgesetzt werden.	Überprüfen Sie den Status der Sicherheitsabschrankung.		
2315	Doppelte Continue-Ausführung.	Warten Sie, bis Continue abgeschlossen ist.		
2316	Die Ausführung kann nicht fortgestzt werden, nachdem ein Fehler erkannt wurde.	Überprüfen Sie den Fehlerstatus.		
2317	Es kann kein Task ausgeführt werden, nachdem ein Fehler erkannt wurde.	Setzen Sie den Fehler mit Reset zurück und führen Sie dann den Task aus.		
2318	Ein Bewegungsbefehl kann nicht ausgeführt werden, nachdem ein Fehler erkannt wurde.			
2319	Ein E/A-Befehl kann während Not-Aus nicht ausgeführt werden.			
2320	Fehler in Funktion. Außergewöhnlicher Fehler. Argumenttyp passt nicht.	Regenerieren Sie das Projekt.		
2321	Fehler in Funktion. Außergewöhnlicher Fehler. Rückgabewert passt nicht zur Funktion.	Regenerieren Sie das Projekt.		
2322	Fehler in Funktion. Außergewöhnlicher Fehler. ByRef-Typ passt nicht.	Regenerieren Sie das Projekt.		
2323	Fehler in Funktion. Außergewöhnlicher Fehler. Fehler bei der Bearbeitung der ByRef-Parameter.	Regenerieren Sie das Projekt.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2324	Fehler in Funktion. Außergewöhnlicher Fehler. ByRef-Parameter passt nicht.	Regenerieren Sie das Projekt.		
2325	Fehler in Funktion. Außergewöhnlicher Fehler. ByRef kann nicht in Xqt-Befehl verwendet werden.	Regenerieren Sie das Projekt.		
2330	Trap-Fehler. Außergewöhnlicher Fehler. Argumente nicht möglich in Trap Call- oder Xqt-Befehl.	Überprüfen Sie das Programm.		
2331	Trap-Fehler. Außergewöhnlicher Fehler. Fehler bei der Trap Goto-Bearbeitung.	Regenerieren Sie das Projekt.		
2332	Trap-Fehler. Außergewöhnlicher Fehler. Fehler bei der Trap Goto-Bearbeitung.	Regenerieren Sie das Projekt.		
2333	Trap-Fehler. Außergewöhnlicher Fehler. Trap ist bereits in Bearbeitung.	Regenerieren Sie das Projekt.		
2340	Der angegebene Wert für InBCD ist ein ungültiger BCD-Wert.	Überprüfen Sie das Programm.	Zehnerstelle	Einerstelle
2341	Der angegebene Wert für OpBCD ist ein ungültiger BCD-Wert.	Überprüfen Sie das Programm.	Angegebener Wert	
2342	Es ist nicht möglich, den Status eines Remote-Ausgangsbits zu ändern.	Überprüfen Sie die Remote-E/A-Einstellung.	E/A-Nummer	1: Bit, 2 Byte, 3: Wort
2343	Zeit für asynchrones Setzen eines Ausgangs durch On oder Off ist außerhalb des Bereichs.	Überprüfen Sie das Programm.	Angegebene Zeit	
2344	Nummer des Eingangs-/Ausgangsbits außerhalb des Bereichs oder Board nicht installiert.	Überprüfen Sie das Programm. Überprüfen Sie das Programm.Überprüfen Sie, ob das Erweiterungs-E/A-Board und das Feldbus-E/A-Board richtig erkannt werden.	Bitnummer	
2345	Nummer des Eingangs-/Ausgangsbytes außerhalb des Bereichs oder Board nicht installiert.	Überprüfen Sie das Programm. Überprüfen Sie das Programm.Überprüfen Sie, ob das Erweiterungs-E/A-Board und das Feldbus-E/A-Board richtig erkannt werden.	Bytenummer	
2346	Nummer des Eingangs-/Ausgangsworts außerhalb des Bereichs oder Board nicht installiert.	Überprüfen Sie das Programm. Überprüfen Sie das Programm.Überprüfen Sie, ob das Erweiterungs-E/A-Board und das Feldbus-E/A-Board richtig erkannt werden.	Wortnummer	
2347	Nummer des Merkerbits außerhalb des Bereichs.	Überprüfen Sie das Programm.	Bitnummer	
2348	Nummer des Merkerbytes außerhalb des Bereichs.	Überprüfen Sie das Programm.	Bytenummer	
2349	Nummer des Merkerworts außerhalb des Bereichs.	Überprüfen Sie das Programm.	Wortnummer	
2350	Befehl nur im Virtuelle-E/A-Modus möglich.	Befehl kann nur im Virtuelle E/A-Modus ausgeführt werden.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2351	Status des CC-Link-Systembereichs kann nicht geändert werden.			
2352	Remote-Einstellungsfehler. Zuweisen des CC-Link-Systembereichs zu einer Remote-Funktion nicht möglich.			
2360	Dateifehler. Außergewöhnlicher Fehler. Fehler beim Öffnen der Konfigurationsdatei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2361	Dateifehler. Außergewöhnlicher Fehler. Fehler beim Schließen der Konfigurationsdatei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2362	Dateifehler. Außergewöhnlicher Fehler. Fehler beim Öffnen des Schlüssels der Konfigurationsdatei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2363	Dateifehler. Außergewöhnlicher Fehler. Fehler beim Beziehen des Strings von der Konfigurationsdatei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2364	Dateifehler. Außergewöhnlicher Fehler. Fehler beim Schreiben der Konfigurationsdatei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2365	Dateifehler. Außergewöhnlicher Fehler. Fehler beim Aktualisieren der Konfigurationsdatei.	Stellen Sie die Konfiguration der Steuerung wieder her.		
2370	Die String-Kombination überschreitet die maximale String-Länge.	Die maximale String-Länge ist 255. Überprüfen Sie das Programm.	Länge einer String-Kombination	
2371	String-Länge außerhalb des Bereichs.	Die maximale String-Länge ist 255. Überprüfen Sie das Programm.	Angegebene Länge	
2372	Ungültiges Zeichen hinter dem &-Zeichen in der Val-Funktion angegeben.	Überprüfen Sie das Programm.		
2373	Nicht erlaubter String in der Val-Funktion angegeben.	Überprüfen Sie das Programm.		
2374	String-Fehler. Außergewöhnlicher Fehler. Ungültiges Zeichen im String enthalten.	Überprüfen Sie das Programm.		
2380	Step '0' für For...Next nicht möglich.	Überprüfen Sie den Step-Wert.		
2381	Beziehung zwischen For...Next und GoSub ungültig. For...Next mit Goto-Befehl verlassen.	Überprüfen Sie das Programm.		
2382	Return kann nicht ausgeführt werden, während OnErr ausgeführt wird.	Überprüfen Sie das Programm.		
2383	Return ohne GoSub verwendet. Überprüfen Sie das Programm.	Überprüfen Sie das Programm.		
2384	Case oder Send ohne Select verwendet. Überprüfen Sie das Programm.	Überprüfen Sie das Programm.		
2385	EResume kann nicht ausgeführt werden, während GoSub ausgeführt wird.	Überprüfen Sie das Programm.		
2386	EResume ohne OnErr verwendet. Überprüfen Sie das Programm.	Überprüfen Sie das Programm.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2400	Curve-Fehler. Außergewöhnlicher Fehler. Fehler beim Öffnen der Curve-Datei.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2401	Curve-Fehler. Außergewöhnlicher Fehler. Fehler beim Zuweisen der Header-Daten der Curve-Datei.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2402	Curve-Fehler. Außergewöhnlicher Fehler. Fehler beim Schreiben der Curve-Datei.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2403	Curve-Fehler. Außergewöhnlicher Fehler. Fehler beim Öffnen der Curve-Datei.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2404	Curve-Fehler. Außergewöhnlicher Fehler. Fehler beim Aktualisieren der Curve-Datei.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2405	Curve-Fehler. Außergewöhnlicher Fehler. Fehler beim Lesen der Curve-Datei.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2406	Curve-Fehler. Außergewöhnlicher Fehler. Curve-Datei ist beschädigt.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2407	Curve-Fehler. Außergewöhnlicher Fehler. Angegebene Datei ist keine Curve-Datei.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2408	Curve-Fehler. Außergewöhnlicher Fehler. Version der Curve-Datei ist ungültig.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2409	Curve-Fehler. Außergewöhnlicher Fehler. Roboternummer der Curve-Datei ist ungültig.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2410	Curve-Fehler. Außergewöhnlicher Fehler. Dem CVMove-Befehl kann nicht genug Speicher zugewiesen werden.	Starten Sie die Steuerung neu.		
2411	Anzahl der Punktedaten im Curve-Befehl übersteigt den Maximalwert.	Die Höchstzahl von im Curve-Befehl angegebenen Punkten ist 200. Überprüfen Sie das Programm.		
2412	Anzahl der Ausgangsbefehle im Curve-Befehl übersteigt den Maximalwert.	Die Höchstzahl von im Curve-Befehl angegebenen Ausgangsbefehlen ist 16. Überprüfen Sie das Programm.		
2413	Curve-Fehler. Außergewöhnlicher Fehler. Der angegebene interne Code im Curve-Befehl übersteigt den Maximalwert.	Starten Sie die Steuerung neu.		
2414	Der angegebene Punktverlauf P(:) übersteigt die maximale Anzahl.	Die Höchstzahl von kontinuierlich angegebenen Punkten ist 200. Überprüfen Sie das Programm.	Startpunkt	Endpunkt
2415	Curve-Fehler. Außergewöhnlicher Fehler. Curve-Datei kann nicht erstellt werden.	Starten Sie die Steuerung neu. Erstellen Sie eine weitere Curve-Datei.		
2416	Curve-Datei existiert nicht.	Überprüfen Sie, ob der angegebene Name der Curve-Datei korrekt ist.		
2417	Curve-Fehler. Außergewöhnlicher Fehler. Ausgangsbefehl vor den Punktedaten angegeben.	Stellen Sie sicher, dass kein Ausgangsbefehl vor den Punktdaten angegeben wird.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2418	Curve-Dateiname zu lang.	Überprüfen Sie, ob der angegebene Name der Curve-Datei korrekt ist. Die maximale String-Länge für einen Dateinamen ist 32.		
2419	Curve-Fehler. Außergewöhnlicher Fehler. Curve-Dateipfad zu lang.	Überprüfen Sie, ob der angegebene Name der Curve-Datei korrekt ist.		
2420	Curve-Dateiname ungültig.			
2430	Fehlermeldungsfehler. Außergewöhnlicher Fehler. Fehlermeldungsdatei existiert nicht.	Starten Sie die Steuerung neu.		
2431	Fehlermeldungsfehler. Außergewöhnlicher Fehler. Fehler beim Öffnen der Fehlermeldungsdatei.	Starten Sie die Steuerung neu.		
2432	Fehlermeldungsfehler. Außergewöhnlicher Fehler. Fehler beim Beziehen des Headers der Fehlermeldungsdatei.	Starten Sie die Steuerung neu.		
2433	Fehlermeldungsfehler. Außergewöhnlicher Fehler. Fehlermeldungsdatei ist beschädigt.	Starten Sie die Steuerung neu.		
2434	Fehlermeldungsfehler. Außergewöhnlicher Fehler. Die angegebene Datei ist keine Fehlermeldungsdatei.	Starten Sie die Steuerung neu.		
2435	Fehlermeldungsfehler. Außergewöhnlicher Fehler. Version der Fehlermeldungsdatei ungültig.	Starten Sie die Steuerung neu.		
2500	Die angegebenen Event-Bedingungen für Wait übersteigen die maximale Anzahl.	Die Höchstzahl von Event-Bedingungen ist 8. Überprüfen Sie das Programm.		
2501	Die angegebene Bitnummer in der Ctr-Funktion wurde nicht mit CTReset-Befehl eingerichtet.	Überprüfen Sie das Programm.	Angegebene Bitnummer	
2502	Tasknummer übersteigt die maximale Anzahl.	Anzahl der Tasks, die gleichzeitig ausgeführt werden können, ist 16. Überprüfen Sie das Programm.		
2503	Die in Xqt angegebene Tasknummer ist bereits aktiv.	Überprüfen Sie das Programm.	Angegebene Tasknummer	
2504	Task-Fehler. Außergewöhnlicher Fehler. Der angegebene Manipulator führt bereits einen Parallelprozess aus.	Regenerieren Sie das Projekt.		
2505	Nicht genug Daten für die Variablenzuweisung des Input-Befehls.	Überprüfen Sie die Kommunikationsdaten. Überprüfen Sie das Programm.		
2506	Anzahl der angegebenen Variablen für den Input-Befehl übersteigt den Maximalwert.	Für das OP kann nur eine Variable spezifiziert werden. Für andere Geräte können bis zu 32 Variablen spezifiziert werden.		
2507	Es werden bereits alle Zähler verwendet. Es kann kein neuer Zähler mit CTReset eingerichtet werden.	Anzahl der Zähler, die gleichzeitig eingerichtet werden können, ist 16. Überprüfen Sie das Programm.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2508	OnErr-Fehler. Außergewöhnlicher Fehler. Fehler bei der Bearbeitung des OnErr-Befehls.	Regenerieren Sie das Projekt.		
2509	OnErr-Fehler. Außergewöhnlicher Fehler. Fehler bei der Bearbeitung des OnErr-Befehls.	Regenerieren Sie das Projekt.		
2510	Das angegebene E/A-Label ist nicht definiert.	Angegebenes E/A-Label ist nicht registriert. Überprüfen Sie die E/A-Label-Datei.		
2511	SyncUnlock-Befehl ohne vorhergehenden SyncLock-Befehl verwendet. Überprüfen Sie das Programm.	Überprüfen Sie das Programm.	Signalnummer	
2512	SyncLock-Befehl wurde bereits ausgeführt.	Der SyncLock-Befehl kann nicht zweimal hintereinander ausgeführt werden. Überprüfen Sie das Programm.	Signalnummer	
2513	Das angegebene Punkt-Label ist nicht definiert.	Angegebenes Punkt-Label ist nicht registriert. Überprüfen Sie die Punktedatei.		
2514	Fehler beim Beziehen der Einschaltzeit des Roboters.	Starten Sie die Steuerung neu.		
2515	Fehler beim Einstellen von Datum oder Uhrzeit.	Überprüfen Sie, ob Datum und Zeit richtig eingestellt sind.		
2516	Fehler beim Beziehen der Debug-Daten oder beim Initialisieren.	Starten Sie die Steuerung neu.		
2517	Fehler beim Konvertieren in Datum oder Uhrzeit.	Überprüfen Sie die in der Steuerung eingestellte Zeit. Starten Sie die Steuerung neu.		
2518	Für den Startpunkt wurde eine größere Nummer angegeben als für den Endpunkt.	Geben Sie eine größere Anzahl Endpunktdaten als Startpunktdaten an.	Startpunkt	Endpunkt
2519	Das angegebene Format für FmtStr$ wurde nicht verstanden.	Überprüfen Sie das Format.		
2520	Punktedateiname zu lang.	Überprüfen Sie, ob der angegebene Name der Punktedatei korrekt ist. Die maximale String-Länge für einen Dateinamen ist 32.		
2521	Punktfehler. Außergewöhnlicher Fehler. Punktedateipfad zu lang.	Überprüfen Sie, ob der angegebene Name der Punktedatei korrekt ist.		
2522	Punktedateiname ungültig.			
2523	Der Continue-Prozess wurde bereits ausgeführt.			
2900	Fehler beim Öffnen des Ethernetports als Server.	Überprüfen Sie, ob der Ethernet-Port richtig eingerichtet ist. Überprüfen Sie, ob das Ethernet-Kabel richtig angeschlossen ist.		
2901	Fehler beim Öffnen des Ethernetports als Client.	Überprüfen Sie, ob der Ethernet-Port richtig eingerichtet ist. Überprüfen Sie, ob das Ethernet-Kabel richtig angeschlossen ist.		
2902	Fehler beim Lesen vom Ethernetport.	Stellen Sie sicher, dass der Port des Kommnikationsempfängers nicht geschlossen ist.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2904	Ungültige IP-Adresse wurde angegeben.			
2905	Ethernet-Fehler. Außergewöhnlicher Fehler. Keine Server/Client-Angabe.			
2906	Ethernetport wurde nicht konfiguiert.	Überprüfen Sie, ob der Ethernet-Port richtig eingerichtet ist.	Portnummer	
2907	Ethernetport wurde bereits von einem anderen Task verwendet.	Ein einzelner Port kann nicht von mehr als einem Task verwendet werden.	Portnummer	
2908	Port-Parameter können nicht verändert werden, während der Ethernetport offen ist.	Port-Parameter können nicht geändert werden, während der Port offen ist.	Portnummer	
2909	Ethernetport ist nicht offen.	Um den Ethernet-Port zu verwenden, führen Sie den OpenNet-Befehl aus.	Portnummer	
2910	Zeitüberlauf beim Lesen von einem Ethernetport.	Überprüfen Sie die Kommunikation.	Zeitüberlauf-Wert	
2911	Fehler beim Lesen von einem Ethernetport.	Überprüfen Sie die Kommunikation.		
2912	Ethernetport wurde bereits von einem anderen Task geöffnet.	Ein einzelner Port kann nicht von mehr als einem Task verwendet werden.	Portnummer	
2913	Fehler beim Schreiben auf den Ethernetport.	Überprüfen Sie, ob der Ethernet-Port richtig eingerichtet ist. Überprüfen Sie, ob das Ethernet-Kabel richtig angeschlossen ist.	Portnummer	
2914	Ethernetport-Verbindung nicht hergestellt.	Kontrollieren Sie, ob der Port des Kommnikationsempfängers offen ist.	Portnummer	
2915	Vom Ethernetport empfangene Daten übersteigen die Grenze einer Zeile.	Maximale Länge einer Zeile beträgt 255 Bytes.	Anzahl der Bytes in einer empfangenen Zeile	
2920	RS-232C-Fehler. Außergewöhnlicher Fehler. RS-232C-Port Bearbeitungsfehler.	Kontrollieren Sie, ob das RS-232C-Board richtig erkannt wird.		
2921	RS-232C-Fehler. Außergewöhnlicher Fehler. Fehler beim Lesen vom RS-232C-Port.			
2926	RS-232C-Hardware ist nicht installiert.	Kontrollieren Sie, ob das RS-232C-Board richtig erkannt wird.	Portnummer	
2927	RS-232C-Port wurde bereits von einem anderen Task geöffnet.	Ein einzelner Port kann nicht von mehr als einem Task verwendet werden.	Portnummer	
2928	Port-Parameter können nicht verändert werden, während der RS-232C-Port offen ist.	Port-Parameter können nicht geändert werden, während der Port offen ist.	Portnummer	
2929	RS-232C-Port ist nicht offen.	Um den RS-232C-Port zu verwenden, führen Sie den OpenCom-Befehl aus.	Portnummer	
2930	Zeitüberlauf beim Lesen von einem RS-232C-Port.	Überprüfen Sie die Kommunikation.	Zeitüberlauf-Wert	
2931	Fehler beim Lesen von einem RS-232C-Port.	Überprüfen Sie die Kommunikation.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
2932	RS-232C-Port wurde bereits von einem anderen Task geöffnet.	Ein einzelner Port kann nicht von mehr als einem Task verwendet werden.	Portnummer	
2933	Fehler beim Schreiben auf den RS-232C-Port.	Überprüfen Sie die Kommunikation.	Portnummer	
2934	RS-232C-Verbindung nicht hergestellt.			
2935	Vom RS-232C-Port empfangene Daten übersteigen die Grenze einer Zeile.	Maximale Länge einer Zeile beträgt 255 Bytes.	Anzahl der Bytes in einer empfangenen Zeile	
2950	Daemon-Fehler. Außergewöhnlicher Fehler. Fehler beim Erstellen des Daemon-Threads.			
2951	Daemon-Fehler. Außergewöhnlicher Fehler. Zeitüberlauf beim Erstellen des Daemon-Threads.			
2952	Fehler des Eingangssignals des TEACH/AUTO-Schlüsselschalters erkannt.	Stellen Sie den TP-Schlüsselschalter richtig auf TEACH oder AUTO. Überprüfen Sie, ob das TP richtig angeschlossen ist.		
2953	Fehler des Eingangssignals des Zustimmtasters erkannt.	Überprüfen Sie, ob das TP richtig angeschlossen ist.		
2954	Klebenden Relaiskontakt erkannt.	Überstrom, vielleicht aufgrund eines Kurzschlussfehlers. Suchen Sie nach der Ursache des Fehlers, treffen Sie die erforderlichen Maßnahmen und tauschen Sie anschließend das DPB aus.		
2955	Temperatur des Regenerationswiderstandes war höher als die festgelegte Temperatur.	Stellen Sie sicher, dass der Filter nicht verstopft ist und der Lüfter nicht anhält. Wenn der Filter und der Lüfter in Ordnung sind, tauschen Sie das Regenerations-Modul aus.		

Parser

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3000	Die OBJ-Datei ist zu groß. Das TP1 könnte nicht in der Lage sein, das Projekt zu generieren.			
3050	Main-Funktion ist nicht definiert.	Deklarieren Sie eine Main-Funktion.		
3051	Funktion existiert nicht.	Deklarieren Sie eine nicht definierte Funktion.		
3052	Variable existiert nicht.	Deklarieren Sie eine nicht definierte Variable.		
3100	Syntaxfehler.	Korrigieren Sie den Syntaxfehler.		
3101	Falsche Parameteranzahl.	Anzahl der Parameter ist zu hoch oder zu niedrig. Korrigieren Sie die Parameter.		
3102	Dateiname zu lang.	Kürzen Sie den Dateinamen.		
3103	Doppelter Funktionsname.	Ändern Sie den Funktionsnamen.		
3104	Doppelte Variablendeklaration ‘(%s).	Ändern Sie den Namen der Variablen.		
3105	Global und Global Preserve Variablen können nicht innerhalb einer Funktion deklariert werden.	Deklarieren Sie globale und Global Preserve-Variabeln außerhalb der Funktion.		
3106	Nicht definierte Funktion angegeben.	Geben Sie einen gültigen Funktionsnamen an.		
3107	While und Until für Do...Loop angegeben.	While- / Until-Befehl ist für den Do-Anweisung und den Loop-Befehl angegeben. Löschen Sie entweder den While- oder den Until-Befehl.		
3108	Die angegebene Zeilennummer oder Label ‘ (%s) existiert nicht.	Richten Sie das Zeilenlabel ein.		
3109	Überlauffehler.	Direkte numerische Angabe läuft über. Verringern Sie den numerischen Wert.		
3110	Eine nicht deklarierte Variable wurde angegeben ‘ (%s).	Eine Variable ist nicht definiert. Deklarieren Sie die Variable.		
3111	Die angegebene Variable ist keine Feldvariable.	Geben Sie die Feldvariable an.		
3112	Dimension der Feldvariablen kann nicht geändert werden.			
3113	Die angegebenen Elemente der Feldvariablen überschreiten die maximale Anzahl. (Wird nicht verwendet.)			
3114	Die angegebene Next-variable stimmt nicht mit der angegebenen For-Variable überein.	Korrigieren Sie den Namen der Variablen.		
3115	Ein Punktausdruck kann nicht als erstes Argument angegeben werden.	Spezifizieren Sie einen einzelnen Punkt für die Einstellung des Punkt-Flag. Spezifizieren Sie keinen Punktausdruck.		
3116	Anzahl der Felddimensionen passt nicht zur Deklaration.	Überprüfen Sie die Anzahl der Felddimensionen.		
3117	Datei nicht gefunden.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3118	Zugehöriges EndIf nicht gefunden.	Anzahl der EndIf-Befehle ist nicht ausreichend. Fügen Sie das EndIf hinzu.		
3119	Zugehöriges Loop nicht gefunden.	Anzahl der Loop-Befehle ist nicht ausreichend. Fügen Sie das Loop hinzu.		
3120	Zugehöriges Next nicht gefunden.	Anzahl der Next-Befehle ist nicht ausreichend. Fügen Sie das Next hinzu.		
3121	Zugehöriges Send nicht gefunden.	Anzahl der Send-Befehle ist nicht ausreichend. Fügen Sie das Send hinzu.		
3122	Der zweite Parameter kann nicht angegeben werden. (Wird nicht verwendet.)			
3123	Anzahl der On/Off-Befehle übersteigt das Maximum.	Eine Obergrenze für die Anzahl der On/Off-Befehle ist eingestellt. Überprüfen Sie die Obergrenze und korrigieren Sie das Programm.		
3124	Punktnummer übersteigt den Maximalwert.	Eine Obergrenze für die Anzahl der verfügbaren Punkte ist eingestellt. Überprüfen Sie die Obergrenze und korrigieren Sie das Programm.		
3125	Zugehöriges If nicht gefunden.	Anzahl der EndIf-Befehle ist zu hoch. Löschen Sie nicht das erforderliche EndIf.		
3126	Zugehöriges Do nicht gefunden.	Anzahl der Loop-Befehle ist zu hoch. Löschen Sie das nicht erforderliche Loop.		
3127	Zugehöriges Select nicht gefunden.	Anzahl der Send-Befehle ist zu hoch. Löschen Sie das nicht erforderliche Send.		
3128	Zugehöriges For nicht gefunden.	Anzahl der Next-Befehle ist zu hoch. Löschen Sie das nicht erforderliche Next.		
3129	'_' kann nicht als erstes Zeichen eines Bezeichners verwendet werden.	Ändern Sie das erste Zeichen eines Bezeichners in ein alphabetisches Zeichen um.		
3130	Rot-Parameter kann nicht angegeben werden.			
3131	Ecp-Parameter kann nicht angegeben werden.			
3132	Arch-Parameter kann nicht angegeben werden.			
3133	Limz-Parameter kann nicht angegeben werden.			
3134	Sense-Parameter kann nicht angegeben werden.			
3135	Ungültiger Parameter angegeben.			
3136	#include kann nicht verwendet werden.			
3137	Index der Feldvariablen kann nicht angegeben werden.	Index der Feldvariablen kann nicht angegeben werden.		
3138	ByRef wurde nicht in der Funktionsdeklaration angegeben.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3139	Xqt-Befehl kann nicht für eine Funktion ausgeführt werden, die einen ByRef-Parameter verlangt.	Xqt-Anweisung kann nicht für eine Funktion ausgeführt werden, für die ein ByRef-Parameter erforderlich ist. Löschen Sie den ByRef-Paramter.		
3140	Redim-Befehl kann nicht für eine ByRef-Variable verwendet werden.			
3141	Außergewöhnlicher Fehler. OBJ-Datei ist beschädigt.			
3142	Größe der OBJ-Datei übersteigt die verfügbare Größe.	Ergebnis der Kompilierung überschreitet den Grenzwert. Verkleinern Sie das Programm.		
3143	Ident-Länge übersteigt die verfügbare Größe.			
3144	'%s' wird bereits als Funktionsname verwendet.			
3145	'%s' wird bereits als Global Preserve-Variable verwendet.			
3146	'%s' wird bereits als Global Preserve-Variable verwendet.			
3147	'%s' wird bereits für eine Modulvariable verwendet.			
3148	'%s' wird bereits für eine Modulvariable verwendet.			
3149	'%s' wird bereits für ein E/A-Label verwendet.			
3150	'%s' wird bereits für einen eigendefinierten Fehler verwendet.			
3151	Ein Funtionsparameter kann nicht verwendet werden.	Argument kann nicht für die Funktion, die vom Trap-Befehl aufgeführt wird, spezifiziert werden.		
3152	Über Elementwert.			
3153	Parametertyp passt nicht.			
3154	'%s' ist kein Label eines Eingangsbits.			
3155	'%s' ist kein Label eines Eingangsbytes.			
3156	'%s' ist kein Label eines Eingangsworts.			
3157	'%s' ist kein Label eines Eingangsbits.			
3158	'%s' ist kein Label eines Eingangsbytes.			
3159	'%s' ist kein Label eines Ausgangsworts.			
3160	'%s' ist kein Label eines Merkerbits.			
3161	'%s' ist kein Label eines Merkerbytes.			
3162	'%s' ist kein Label eines Merkerworts.			
3163	Zu viele Funktionsargumente.			
3169	Ein nicht definiertes E/A-Label wurde angegeben.			
3170	Ungültige Bedingung wurde angegeben.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3171	Ein numerischer Wert kann nicht mit einem String verglichen werden.			
3172	Ein Befehl kann nicht als Variablenname verwendet werden.			
3172	'%s' wird bereits als Zeilenlabel verwendet.			
3173	Doppelte Zeilennummer oder -label (%s).			
3175	Nicht definiertes Punktlabel angegeben.			
3176	Nicht definierte Variable angegeben.			
3177	'%s' wird bereits als Punktlabel verwendet.			
3200	Wert fehlt.			
3201	',' erwartet.			
3202	'(' erwartet.			
3203	')' erwartet.			
3204	Bezeichner fehlt.			
3205	Kein Punkt angegeben.			
3206	Der Ausdruck der Event-Bedingung fehlt.			
3207	Formel fehlt.			
3208	String-Formel fehlt.			
3209	Punkformel fehlt.			
3210	Zeilenlabel wurde nicht angegeben.			
3211	Variable wurde nicht angegeben.			
3212	Zugehöriges Fend nicht gefunden.			
3213	':' erwartet. '.			
3214	True/False wurde nicht angegeben.			
3215	On/Off wurde nicht angegeben.			
3216	High/Low wurde nicht angegeben.			
3217	Label des Eingangsbits wurde nicht angegeben.			
3218	Label des Eingangsbytes wurde nicht angegeben.			
3219	Label des Eingangsworts wurde nicht angegeben.			
3220	Label des Ausgangsbits wurde nicht angegeben.			
3221	Label des Ausgangsbytes wurde nicht angegeben.			
3222	Label des Ausgangsworts wurde nicht angegeben.			
3223	Label des Merkerbits wurde nicht angegeben.			
3224	Label des Merkerbytes wurde nicht angegeben.			
3225	Label des Merkerworts wurde nicht angegeben.			
3226	Label des eigendefinierten Fehlers wurde nicht angegegeben.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3227	Funktionsname wurde nicht angegeben.			
3228	Variablentyp wurde nicht angegeben.			
3229	Ungültiger Parameter für Trap angegeben. Goto, Call oder Xqt verwenden.			
3230	For/Do/Function erwartet.			
3231	Above/Below wurde nicht angegeben.			
3232	Righty/lefty wurde nicht angegeben.			
3233	NoFlip/Flip wurde nicht angegeben.			
3234	Portnummer wurde nicht angegeben.			
3235	Stringtyp-Variable wurde nicht angegeben			
3236	RS232C-Portnummer wurde nicht angegeben.			
3237	Kommunikationsportnummer des Netzwerks wurde nicht angegeben.			
3238	Kommunikationgeschwindigkeit wurde nicht angegeben.			
3239	Datenbitnummer wurde nicht angegeben.			
3240	Stoppbitnummer wurde nicht angegeben.			
3241	Parität wurde nicht angegeben.			
3242	Terminator wurde nicht angegeben.			
3243	Hardware flow wurde nicht angegeben.			
3244	Software flow wurde nicht angegeben.			
3245	None wurde nicht angegeben.			
3246	Parameter 'O' oder 'C' wurde nicht angegeben.			
3247	NumAxes-Parameter wurde nicht angegeben.			
3248	J4Flag-Wert (0-1) wurde nicht angegeben.			
3249	J6Flag-Wert (0-128) wurde nicht angegeben.			
3250	Feldvariable wurde nicht angegeben.			
3251	Feldvariable wurde nicht angegeben.			
3252	Geräte-ID wurde nicht angegeben.			
3253	E/A-Typ wurde nicht angegeben.			
3254	E/A-Bitbreite wurde nicht angegeben.			
3256	Variablentyp wurde nicht angegeben.			
3257	Der Ausdruck gibt keinen Boolean-Wert zurück.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3255	ByRef wurde nicht angegeben.	Obwohl ByRef in der Funktionsdeklaration spezifiziert ist, ist kein ByRef für die Calling-Funktion spezifiziert.		
3300	Externes Definitonssysmbol wurde berücksichtigt. (Wird nicht verwendet.)			
3301	Version der verknüpften OBJ-Datei passt nicht.	Nicht alle Projektdateien werden in derselben Version kompiliert. Führen Sie eine Regenerierung durch.		
3302	Verknüpfte OBJ-Datei passt nicht zu den kompilierten E/A-Labels.	Projektkonfiguration wurde geändert. Führen Sie eine Regenerierung durch.		
3303	Verknüpfte OBJ-Datei passt nicht zu den kompilierten Labels der eigendefinierten Fehler.	Projektkonfiguration wurde geändert. Führen Sie eine Regenerierung durch.		
3304	Außergewöhnlicher Fehler. Verknüpfte OBJ-Datei passt nicht zu der kompilierten Kompilieroption.	Projektkonfiguration wurde geändert. Führen Sie eine Regenerierung durch.		
3305	Außergewöhnlicher Fehler. Verknüpfte OBJ-Datei passt nicht zu der kompilierten Linkoption.	Projektkonfiguration wurde geändert. Führen Sie eine Regenerierung durch.		
3306	Außergewöhnlicher Fehler. Verknüpfte OBJ-Datei passt nicht zu der kompilierten SPEL-Option.	Projektkonfiguration wurde geändert. Führen Sie eine Regenerierung durch.		
3307	Doppelte Funktion.	Derselbe Funktionsname wird für mehr als eine Datei verwendet.		
3308	Doppelte Global Preserve Variable.	Dieselbe Global Preserve-Variable wird für mehr als eine Datei verwendet.		
3309	Doppelte globale Variable.	Dieselbe globale Variable wird für mehr als eine Datei verwendet.		
3310	Doppelte Modulvariable.	Dieselbe Modulvariable wird für mehr als eine Datei verwendet.		
3311	Datei nicht gefunden.			
3312	Außergewöhnlicher Fehler. OBJ-Datei ist beschädigt.			
3313	Der angegebene Dateiname enthält unzulässige Zeichen.			
3314	Datei kann nicht geöffnet werden.	Datei wird in einer anderen Anwendung verwendet. Beenden Sie die andere Anwendung.		
3315	'%s' wird bereits als Funktionsname verwendet.			
3316	'%s' wird bereits für eine Global Preserve Variable verwendet.			
3317	'%s' wird bereits für eine globale Variable verwendet.			
3318	'%s' wird bereits für eine Modulvariable verwendet.			
3319	Dimension der Feldvariablen passt nicht zur Deklaration.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3320	Typ des Rückgabewerts der Funktion passt nicht zur Deklaration.			
3321	'%s' wird bereits als Funktionsname verwendet.			
3322	'%s' wird bereits als Global Preserve Variable verwendet.			
3323	'%s' wird bereits als globale Variable verwendet.			
3324	'%s' wird bereits als Modulname verwendet.			
3325	'%s' wird bereits als Local-Name verwendet.			
3326	Die Anzahl der Parameter entspricht nicht der Deklaration.			
3327	ByRef wurde nicht in der Funktionsdeklaration angegeben.			
3328	ByRef wurde nicht angegeben.			
3500	Doppeltes Macro im #define-Befehl.	Ein anderes Makro mit demselben Namen wurde definiert. Ändern Sie den Namen des Makros.		
3501	Makroname wurde nicht angegeben.			
3502	Name der Include-Datei nicht gefunden.			
3503	Die angegebene Include-Datei ist nicht im Projekt vorhanden.	Include-Datei, die nicht in der Projektkonfiguration registriert ist, ist angegeben. Fügen Sie die Include-Datei zur Projektkonfiguration hinzu.		
3504	Parameter der Makrofunktion passt nicht zur Deklaration.			
3505	Makro hat eine zirkulare Referenz.	Das Makro hat eine zirkulare Referenz. Beheben Sie die zirkulare Referenz.		
3506	#define, #ifdef, #ifndef, #else, #endif, #undef und Variablendeklartionsbefehle sind nur in Include-Dateien gültig.			
3507	Über #ifdef oder #ifndef Schachtelungstiefe.	Verringern Sie die Schachtelungstiefe, sodass sie unterhalb des Grenzwerts liegt.		
3508	Zugehöriges #ifdef oder #ifndef nicht gefunden.			
3509	Kein #endif für #ifdef oder #ifndef gefunden.			
3550	Parameter der Makrofunktion wurde nicht angegeben.	Als Makrofunktion deklariertes Makro wird ohne Argument aufgerufen.		
3800	Kompilierprozess abgebrochen.			
3801	Linkprozess abgebrochen.			
3802	Kompilierprozess abgebrochen. Anzahl der Kompilierfehler übersteigt den Maximalwert.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
3803	Linkprozess abgebrochen. Anzahl der Linkfehler übersteigt den Maximalwert.			
3804	Der angegebene Befehl kann nicht vom Befehlseingabefenster ausgeführt werden.			
3805	Der angegebene Befehl kann nur vom Befehlseingabefenster ausgeführt werden.			
3806	Die angegebene Funktion kann nicht vom Befehlseingabefenster ausgeführt werden.			
3900	Außergewöhnlicher Fehler. Interner Kommunikationpuffer kann nicht erreicht werden.			
3910	Außergewöhnlicher Fehler. Nicht definierter Befehl wurde angegeben.			
3911	Außergewöhnlicher Fehler. Dateiname kann nicht in den Dateinamenpuffer geschrieben werden.			
3912	Außergewöhnlicher Fehler. Interner Puffer kann nicht erreicht werden.			

Motorsteuerung

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4001	Arm erreicht Grenzen des Arbeitsbereichs.	Überprüfen Sie den Punkt, zu dem hin die Bewegung ausgeführt werden soll, den aktuellen Punkt und die Einstellung des Bereichs.		
4002	Der angegebene Wert ist außerhalb des Bereichs.	Überprüfen Sie die Einstellungsparameter.		Parameter, der den Fehler verursacht
4003	Fehler des Treibers der Bewegungseinheit. Außergewöhnlicher Fehler. Kommunikationfehler innerhalb des Bewegungssteuerungsmoduls.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4004	Fehler des Treibers der Bewegungseinheit. Außergewöhnlicher Fehler. Event-Wartefehler innerhalb des Bewegungssteuerungsmoduls.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4005	Die aktuelle Position liegt über dem angegebenen LimZ-Wert.	Senken Sie die Z-Achse. Erhöhen Sie den angegebenen LimZ-Wert.		
4006	Die Zielposition liegt über dem angegebenen LimZ-Wert.	Setzen Sie die Position der Z-Koordinate des Zielpunkts herab. Erhöhen Sie den angegebenen LimZ-Wert.		
4007	Koordinaten-Umwandlungsfehler. Der End-/Mittelpunkt liegt außerhalb des Arbeitsbereichs. Bewegungsschritt über die Grenze des Arbeitsbereichs.	Stellen Sie sicher, dass die Koordinate außerhalb des Arbeitsbereichs nicht angegeben wurde.		
4008	Die aktuelle Position oder der angegebene LimZ-Wert liegen außerhalb des Arbeitsbereichs.	Ändern Sie den spezifizierten LimZ-Wert.		
4009	Fehler des Treibers der Bewegungseinheit. Außergewöhnlicher Fehler. Zeitüberlauf innerhalb des Bewegungssteuerungsmoduls.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4010	Das angegebene Local ist nicht definiert.	Definieren Sie das lokale Koordinatensystem.		Local-Nummer
4011	Arm erreicht Grenzen des durch XYLim festgelegten Arbeitsbereichs.	Überprüfen Sie den Bereich, der durch den XYLim-Befehl begrenzt wird.		
4013	Interner Rechenfehler des Bewegungssteuermoduls. Außergewöhnlicher Fehler.			
4016	Es wurde versucht SFree für Achse(n) auszuführen, die für SFree nicht freigegeben sind.	Aufgrund der Einschränkung durch die Mechanik des Roboters dürfen eine bzw. mehrere Achsen nicht in den Servo-Free-Status gesetzt werden. Überprüfen Sie die technischen Daten des Roboters.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4018	Kommunikationfehler innerhalb des Bewegungssteuerungsmoduls. Außergewöhnlicher Fehler. Prüfsummenfehler.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4021	Der Abstand der Punkte ist zu gering um ein Local zu definieren.	Stellen Sie den Abstand zwischen den Punkten auf mehr als 1µm ein.		
4022	Punktekoordinate für Local-Definition ist ungültig.	Stellen Sie die Koordinate für die anzugebenden Punkte ein.		
4023	Kann nicht mit ausgeschalteten Motoren ausgeführt werden.	Schalten Sie den Motor EIN und führen Sie dann die Ausführung durch.		
4024	Armpositionierung mit der aktuellen Fine-Einstellung nicht möglich.	Stellen Sie sicher, dass der Roboter keine Erschütterungen erzeugt und dass alle Teile und Schrauben sicher befestigt sind. Erhöhen Sie die Fine-Einstellung.		
4025	Ein Bewegungsbefehl kann während Not-Aus nicht ausgeführt werden.	Beheben Sie den Not-Aus-Zustand und führen Sie dann den Bewegungsbefehl aus.		
4026	Kommunikationfehler innerhalb des Bewegungssteuerungsmoduls. Außergewöhnlicher Fehler. Servo-Schnittstellenfehler.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4028	Kommunikationfehler innerhalb des Bewegungssteuerungsmoduls. Außergewöhnlicher Fehler. Gerätetreiber-Statusfehler.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4030	Puffer für die Berechnung des mittleren Drehmoments übergelaufen. Zeit zwischen Atclr und Atrq verkürzen.	Verkürzen Sie den Abstand zwischen Atclr und Atrq auf unter zwei Minuten.		
4031	Bewegungsbefehl kann mit ausgeschalteten Motoren nicht ausgeführt werden.	Schalten Sie den Motor EIN und führen Sie dann den Bewegungsbefehl aus.		
4032	Bewegungsbefehl kann nicht ausgeführt werden, wenn eine oder mehrere Achsen freigeschaltet sind.	Setzen Sie alle Achsen in den SLock-Status und führen Sie dann den Bewegungsbefehl aus.		
4034	Der angegebene Befehl wird für dieses Manipulatormodell nicht unterstützt.	Verwenden Sie die Befehle Jump3 und Jump3CP.		
4035	Es wurde versucht, durch den CP-Befehl nur die Tool-Orientierung zu ändern.	Stellen Sie zwischen den Punkten einen Bewegungsabstand ein. Verwenden Sie die ROT-Bedingung, den SpeedR-Befehl und den AccelR-Befehl.		
4036	Die Geschwindigkeit der Tool-Rotation durch den CP-Befehl ist zu hoch.	Setzen Sie die Einstellwerte für die SpeedS- und AccelS-Befehle herab. Verwenden Sie die ROT-Bedingung, den SpeedR-Befehl und den AccelR-Befehl.		
4037	Die Punktattribute der aktuellen und der Zielposition für einen CP-Befehl sind unterschiedlich.	Gleichen Sie die Punktattribute an.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4038	Der Arc-Befehl kann nicht ausgeführt werden, weil zwei Punkte zu nah beieinander liegen.	Stellen Sie den Abstand zwischen den Punkten auf mehr als 1µm ein.		
4039	Die drei im Arc-Befehl angegebenen Punkte liegen auf einer Geraden.	Verwenden Sie den Move-Befehl.		
4041	Es wurde versucht, einen Bewegungsbefehl zur verbotenen Zone auf der Rückseite des Manipulators auszuführen.	Überprüfen Sie den Arbeitsbereich des Roboters.		
4042	Fehler des Treibers der Bewegungseinheit. Außergewöhnlicher Fehler. Kreisformatabschaltung konnte nicht erkannt werden.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4043	Der angegebene Befehl wird für dieses Manipulatormodell oder für diesen Achsentyp nicht unterstützt.			
4044	Curve-Fehler. Außergewöhnlicher Fehler. Angegebene Kurvenform wird nicht unterstützt.	Erstellen Sie eine neue Curve-Datei mit dem Curve-Befehl.		
4045	Curve-Fehler. Angebener Modus wird nicht unterstützt.	Geben Sie den richtigen Curve-Modus an. Erstellen Sie eine neue Curve-Datei mit der Curve-Anweisung.		
4046	Curve-Fehler. Angegebene Koordinatennummer außerhalb des Bereichs.	Verfügbare Anzahl von Koordinatenachsen ist 2, 3, 4 oder 6. Erstellen Sie mit der Curve-Anweisung eine neue Curve-Datei.		
4047	Curve-Fehler. Außergewöhnlicher Fehler. Keine Punktedaten angegeben.	Erstellen Sie eine neue Curve-Datei mit der Curve-Anweisung.		
4048	Curve-Fehler. Außergewöhnlicher Fehler. Parallelprozess wurde vor den Punkten angegeben.	Erstellen Sie eine neue Curve-Datei mit der Curve-Anweisung.		
4049	Curve-Fehler. Außergewöhnlicher Fehler. Anzahl der Parallelprozesse außerhalb des Bereichs.	Erstellen Sie eine neue Curve-Datei mit der Curve-Anweisung.		
4050	Curve-Fehler. Anzahl der Punkte außerhalb des Bereichs.	Anzahl der verfügbaren Punkte hängt von der Kurvenform ab. Überprüfen Sie nochmals die Anzahl der Punkte.		
4051	Curve-Fehler. Local-Attribute und Punktattribute aller angegebenen Punkte stimmen nicht überein.	Gleichen Sie das Local- und das Punkt-Flag für alle angegebenen Punkte an.		
4052	Curve-Fehler. Außergewöhnlicher Fehler. Nicht genug Speicher zum Formatieren der Kurvendatei.			
4053	Curve-Fehler. Außergewöhnlicher Fehler. Fehler beim Formatieren der Kurvendatei.	Überprüfen Sie die Punktdaten. Stellen Sie sicher, dass zwei nebeneinander liegende Punkt sich auf der spezifizierten Punktelinie nicht überlagern.		
4054	Curve-Fehler. Außergewöhnlicher Fehler. Kurvendatei fehlerhaft.	Kurvendatei ist defekt. Erstellen Sie eine neue Curve-Datei mit der Curve-Anweisung.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4055	Curve-Fehler. Keine Distanz für die Kurvenbewegung.	Überprüfen Sie die Punktdaten.		
4056	Curve-Fehler. Punktpositionen für den Curve-Befehl sind zu nah beieinander.	Stellen Sie den Abstand zwischen zwei neben dem angegebenen Punkt liegenden Punkten auf mehr als 0,001 mm ein.		
4059	Encoder-Reset mit eingeschalteten Motoren ausgeführt.	Schalten Sie den Motor AUS.		
4060	Mit eingeschalteten Motoren unzulässiger Befehl wurde ausgeführt.	Schalten Sie den Motor AUS.		
4061	Der angegebene Parameter wird bereits verwendet.	Sie haben versucht den aktuell angegebenen Arm und das aktuell angegebene Tool zu löschen. Wählen Sie einen anderen Arm und ein anderes Tool aus und führen Sie dann die Ausführung durch.		
4062	Die Variation der Orientierung ist größer als 360°.	Sie haben versucht, die 6. Achse mit einem CP-Bewegungsbefehl um mehr als 360 Grad zu rotieren.		
4063	Die Variation der Orientation des benachbarten Punktes ist größer als 90°.	Stellen Sie auf der angegebenen Punktelinie mit dem Curve-Befehl die Ausrichtungsänderung der U-, V-, und W-Koordinatenwerte zwischen zwei nebeneinander liegenden Punkten auf unter 90 Grad ein.		
4064	Die Orientierungskorrektur kann nicht automatisch durchgeführt werden.	Auf der angegebenen Punktelinie kann eine Kurve nicht durch automatische Orientierungskorrektur erstellt werden. Ändern Sie die angegebene Punktelinie, sodass die Ausrichtungsänderung der 6. Achse reduziert wird.		
4065	Es wurde versucht, mit derselben Orientierung im CP-Befehl die Achse 6 um eine Umdrehung zu rotieren.	Sie haben versucht, die 6. Achse mit einem CP-Bewegungsbefehl um mehr als 360 Grad zu rotieren. Sie haben versucht eine vollständige Rotation mit der 6. Achse mit derselben Ausrichtung wie der Bewegungsausgangsausrichtung durchzuführen. Ändern Sie den Zielpunkt, sodass die 6. Achse weniger als eine Umdrehung vollzieht.		
4066	Es wurde versucht, einen Bewegungsbefehl im verbotenen Bereich der Achsenkombinationen auszuführen.	Sie haben versucht, die Achsen in den Kollisionsbereich des Roboters zu bewegen.		
4068	Der ROT-Parameter wurde für einen CP-Befehl angegeben, ohne die Orientierung zu ändern.	Löschen Sie den ROT-Parameter aus dem CP-Befehl.		
4069	ECP wurde angegeben ohne ECP im CP-Befehl zu wählen.	Geben Sie einen gültigen ECP an.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4070	Die angegebene ECP-Nummer stimmt nicht mit der ECP-Nummer in Kurvendatei überein.	Geben Sie einen gültigen ECP an.		
4071	Bewegungsbefehl wurde versucht auszuführen, während die elektronische Bremse eingeschaltet war.			
4072	Initialisierungsfehler. Außergewöhnlicher Fehler. Hardware-Monitor wurde nicht initialisiert.			
4074	Der Motortyp passt nicht zur aktuellen Robotereinstellung.	Überprüfen Sie, ob das angegebene Robotermodell angeschlossen ist.		
4075	Die ECP-Option ist nicht aktiv.	Aktivieren Sie die ECP-Option.		
4099	Servo-Fehler während der Operation erkannt.			
4100	Kommunikationsfehler im Bewegungssteuerungsmodul. Außergewöhnlicher Fehler. Der aktuelle Punkt oder Pulswert kann nicht berechenet werden.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4101	Kommunikationsfehler im Bewegungssteuerungsmodul. Außergewöhnlicher Fehler. Der aktuelle Punkt oder Pulswert kann nicht berechenet werden.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4103	Initialisierungsfehler. Außergewöhnlicher Fehler. Initialisierungsfehler des Bewegungssteuerungsmoduls.	Starten Sie die Steuerung neu. Initialisieren Sie die Firmware der Steuerung. Tauschen Sie die Steuerung aus.		
4150	Fehler der redundanten Not-Aus-Kreise.	Der Eingangsstatus des redundanten Not-Aus-Eingangs weicht anhaltend mehr als zwei Sekunden ab. Stellen Sie sicher, dass keine Verbindungsunterbrechung, kein Erdschluss und kein Kurzschluss des Not-Aus-Eingangs vorliegt. Starten Sie anschließend die Steuerung neu.		
4151	Fehler der redundanten Kreise der Sicherheitsabschrankung.	Das Eingangssignal des redundanten Not-Aus-Eingangs weicht anhaltend mehr als zwei Sekunden ab. Stellen Sie sicher, dass keine Verbindungsunterbrechung, kein Erdschluss und kein Kurzschluss des Not-Aus-Eingangs vorliegt. Starten Sie anschließend die Steuerung neu.		
4152	Relaiskontakt des Hauptstromkreises klebt.	Es wurde erkannt, dass der Relaiskontakt des Hauptstromkreises aufgrund von Überstrom klebt. Tauschen Sie die Steuerung aus. Tauschen Sie den Roboter aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4153	Fehler der redundanten Kreise des Zustimmtasters.	Das Eingangssignal der redundanten Kreise weicht anhaltend mehr als zwei Sekunden ab. Überprüfen Sie den Anschluss des TP-Anschlusses. Tauschen Sie das TP aus. Tauschen Sie die Steuerung aus.		
4230	Servo-Echtzeit-Statusfehler. Außergewöhnlicher Fehler. Prüfsummenfehler.	Ein Prüfsummenfehler wurde in der Steuerung erkannt. Überprüfen Sie, ob ein Kurzschluss vorliegt oder ob die Peripheriegeräte nicht richtig verkabelt sind. (Not-Aus-, D-E/A- und Erweiterungs-E/A-Anschlüsse)Tauschen Sie die Steuerung aus.		
4232	Servo-Echtzeit-Statusfehler. Außergewöhnlicher Fehler. Freilaufzähler des Servos.	Ein Freilaufzähler wurde in der Steuerung erkannt. Überprüfen Sie, ob ein Kurzschluss vorliegt oder ob die Peripheriegeräte nicht richtig verkabelt sind. (Not-Aus-, D-E/A- und Erweiterungs-E/A-Anschlüsse)Tauschen Sie die Steuerung aus.		
4233	Servo-Echtzeit-Statusfehler. Außergewöhnlicher Fehler. Kommunikationfehler mit der Servo-CPU.	Ein Kommunikationsfehler wurde in der Steuerung erkannt. Überprüfen Sie, ob ein Kurzschluss vorliegt oder ob die Peripheriegeräte nicht richtig verkabelt sind. (Not-Aus-, D-E/A- und Erweiterungs-E/A-Anschlüsse)Tauschen Sie die Steuerung aus.		
4240	Irreguläre Unterbrechung der Bewegungssteuerung wurde erkannt. Außergewöhnlicher Fehler. Doppelte Unterbrechung.	Eine irreguläre Unterbrechung wurde in der Steuerung erkannt. Überprüfen Sie, ob ein Kurzschluss vorliegt oder ob die Peripheriegeräte nicht richtig verkabelt sind. (Not-Aus-, D-E/A- und Erweiterungs-E/A-Anschlüsse)Tauschen Sie die Steuerung aus.		
4241	Zu hohe Geschwindigkeit im Low-Power-Modus wurde erkannt.	Eine zu hohe Geschwindigkeit des Roboters im Low-Power-Modus wurde erkannt. Überprüfen Sie die Robotermechanik. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse). Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Tauschen Sie den Motortreiber aus. Tauschen Sie den Motor aus. (Motor- und Encoder-Fehler). Prüfen Sie die Verkabelung der Peripheriegeräte auf Kurzschluss und fehlerhaften Anschluss. (Not-Aus-, D-E/A- und Erweiterungs-E/A-Anschlüsse)		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4242	Ungültiger Beschleunigungswert wurde erkannt.	Sie haben versucht, den Roboter mit einem zu hohen Beschleunigungswert zu betreiben. Reduzieren Sie den AccelS-Wert für eine CP-Bewegung.		
4243	Ungültiger Geschwindigkeitswert im High-Power-Modus wurde erkannt.	Eine zu hohe Geschwindigkeit des Roboters im High-Power-Modus wurde erkannt. Überprüfen Sie die Robotermechanik. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse). Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Tauschen Sie den Motortreiber aus. Tauschen Sie den Motor aus. (Motor- und Encoder-Fehler). Prüfen Sie die Verkabelung der Peripheriegeräte auf Kurzschluss und fehlerhaften Anschluss. (Not-Aus-, D-E/A- und Erweiterungs-E/A-Anschlüsse)		
4250	Arm hat während der Bewegung die Grenzen des Arbeitsbereichs erreicht.	Überprüfen Sie, ob der Trajektoriebereich der CP-Bewegung im Arbeitsbereich liegt.		
4251	Arm hat während der Bewegung die Grenzen des durch XYLim festgelegten Arbeitsbereichs erreicht.	Überprüfen Sie die XYLim-Einstellungen.		
4252	Koordinaten-Umwandlungsfehler während der Bewegung.	Überprüfen Sie, ob der Trajektoriebereich der CP-Bewegung im Arbeitsbereich liegt.		
4267	Es wurde versucht, ohne Indikation das J4Flag-Attribut zu überschreiten.	Sie haben versucht, während der Bewegung das J4Flag-Attribut ohne die J4Flag-Indikation zu überschreiten. Ändern Sie das J4Flag für den Zielpunkt.		
4268	Es wurde versucht, ohne Indikation das J6Flag-Attribut zu überschreiten.	Sie haben versucht, während der Bewegung das J6Flag-Attribut ohne die J6Flag-Indikation zu überschreiten. Ändern Sie das J6Flag für den Zielpunkt.		
4269	Es wurde versucht, ohne Indikation die jeweilige Handgelenkorientierung zu überschreiten.	Sie haben versucht, während der Bewegung die jeweilige Handgelenkorientierung ohne die Wrist-Indikation zu überschreiten. Ändern Sie die Handegelenkorientierung für den Zielpunkt. Ändern Sie den Zielpunkt, um eine bestimmte Handgelenkorientierung zu verhindern.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
4270	Es wurde versucht, ohne Indikation die jeweilige Armorientierung zu überschreiten.	Sie haben versucht, während der Bewegung die jeweilige Armorientierung ohne die Hand-Indikation zu überschreiten. Ändern Sie die Armorientierung für den Zielpunkt. Ändern Sie den Zielpunkt, um eine bestimmte Armorientierung zu verhindern.		
4271	Es wurde versucht, ohne Indikation die jeweilige Ellenbogenorientierung zu überschreiten.	Sie haben versucht, während der Bewegung die jeweilige Ellenbogenorientierung ohne die Elbow-Indikation zu überschreiten. Ändern Sie die Ellenbogenorientierung für den Zielpunkt. Ändern Sie den Zielpunkt, um eine bestimmte Ellenbogenorientierung zu verhindern.		
4272	Das angegebene Punkt-Flag ist ungültig.	Für einen CP-Bewegungsbefehl entspricht die Armform am Zielpunkt nicht dem Punkt-Flag, das mit dem Zielpunkt angegeben wurde. Ändern Sie das Punkt-Flag oder den Zielpunkt.		

Servo

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5000	Gate-Array-Fehler der Servo-Steuerung. DMB überprüfen.	Überprüfen Sie, ob ein Kurzschluss vorliegt oder ob die Peripheriegeräte nicht richtig verkabelt sind. (Not-Aus- und E/A-Anschlüsse) Tauschen Sie das DMB aus. Tauschen Sie die zusätzliche Achseneinheit aus.		
5001	Unterbrechung des Parallelencodersignals. Die Signalkabelverbindung oder die interne Verdrahtung des Roboters überprüfen.	Überprüfen Sie das M/C-Kabelsignal. Überprüfen Sie die Signalleitungen des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Tauschen Sie den Motor aus. Tauschen Sie das DMB aus. Überprüfen Sie den Anschluss des Anschlusses in der Steuerung. (Lösen, an den Anschluss des seriellen Encoders am DMB anschließen.) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Verkabelung der Peripheriegeräte. (Not-Aus und E/A)		
5002	Motortreiber ist nicht installiert. Motortreiber installieren. DMB oder Motortreiber überprüfen.	Überprüfen Sie, ob der Motortreiber installiert ist. Überprüfen Sie die Modell- und Hardwareeinstellungen. Motortreiber austauschen. Tauschen Sie das DMB aus.		
5003	Initialisierungsfehler der Kommunikation des Inkrementalencoders. Signalkabelverbindungen und Robotereinstellungen überprüfen.	Überprüfen Sie die Modelleinstellungen. Tauschen Sie den Motor aus. Tauschen Sie das DMB aus.		
5004	Initialisierungsfehler des Absolutencoders. Signalkabelverbindungen und Robotereinstellungen überprüfen.	Überprüfen Sie die Modelleinstellungen. Tauschen Sie den Motor aus. Tauschen Sie das DMB aus.		
5005	Fehler der Einstellung der Encoderteilung. Robotereinstellungen überprüfen.	Überprüfen Sie die Modelleinstellungen.		
5006	Datenfehler während der Absolutencoderinitialisierung. Signalkabelverbindung, Steuerung und Motoren überprüfen.	Tauschen Sie den Motor aus. Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
5007	Absolutencoderumdrehungen über Maximalwert. Encoder-Reset durchführen.	Encoder-Reset durchführen. Tauschen Sie den Motor aus.		
5008	Position außerhalb des Pulsebereichs. Encoder-Reset durchführen.	Encoder-Reset durchführen. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5009	Keine Antwort vom seriellen Encoder. Signalkabelverbindung, Motor, DMB oder Encoderanschlussboard überprüfen.	Überprüfen Sie die Modelleinstellungen. (Ungültige Einstellungen des Parallelencodermodells) Überprüfen Sie die Signalkabelverbindung. Tauschen Sie das DMB und das Encoderanschlussboard aus.		
5010	Fehler beim Initialisieren des seriellen Encoders. Starten Sie die Steuerung neu. Motor, DMB oder Encoderanschlussboard überprüfen.	Überprüfen Sie die Konfiguration des Roboters. Überprüfen Sie die Signalkabelverbindung. Tauschen Sie das DMB und das Encoderanschlussboard aus.		
5011	Fehler bei der Initialisierung der Kommunikation des seriellen Encoders. Starten Sie die Steuerung neu. Motor, DMB oder Encoderanschlussboard überprüfen.	Überprüfen Sie die Konfiguration des Roboters. Überprüfen Sie die Signalkabelverbindung. Tauschen Sie das DMB und das Encoderanschlussboard aus.		
5012	Fehler des Watchdog-Timers der Servo-CPU. Starten Sie die Steuerung neu. Motor oder DMB überprüfen.	Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
5013	Fehler des WDT des aktuellen Steuerkreises. Starten Sie die Steuerung neu. Steuerung überprüfen.	Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die 15 V-Stromversorgung und die Kabelverbindung. Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
5015	Encoder-Reset durchgeführt. Starten Sie die Steuerung neu.	Starten Sie die Steuerung neu.		
5016	Spannungsversorgungsfehler des Absolutencoders. Batterie austauschen. Interne Verdrahtung des Roboters überprüfen.	Encoder-Reset durchführen. Überprüfen Sie die Signalkabelverbindung.		
5017	Fehler der Backup-Daten des Absolutencoders. Encoder-Reset durchführen.	Encoder-Reset durchführen. Überprüfen Sie die Signalkabelverbindung.		
5018	Absolutencoder Batteriealarm.	Batterie austauschen. Überprüfen Sie die Signalkabelverbindung.		
5019	Positionsfehler des Absolutencoders. Encoder-Reset durchführen. Tauschen Sie den Motor aus.	Encoder-Reset durchführen. Tauschen Sie den Motor aus.		
5020	Zu hohe Geschwindigkeit beim Einschalten der Steuerung. Roboter anhalten und Steuerung neu booten.	Starten Sie die Steuerung neu.		
5021	Absolutencoder Überhitzung.	Reduzieren Sie die Bewegungsleistung. Warten Sie, bis die Temperatur des Encoders sinkt.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5032	Servo-Alarm A.			
5040	Drehmomentfehler im High-Power-Modus. Motorkabelverbindung, Roboter, Treiber oder Motor überprüfen.	Geben Sie die Weight/Inertia-Einstellung an. Überprüfen Sie die Last. Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		
5041	Drehmomentfehler im Low-Power-Modus. Motorkabelverbindung, Roboter, Bremse, Treiber oder Motor überprüfen.	Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5042	Positionsfehlerüberlauf im High-Power-Modus. Motorkabelverbindung, Roboter, Treiber oder Motor überprüfen.	Geben Sie die Weight/Inertia-Einstellung an. Überprüfen Sie die Last. Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		
5043	Positionsfehlerüberlauf im Low-Power-Modus. Motorkabelverbindung, Roboter, Bremse, Treiber oder Motor überprüfen.	Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5044	Geschwindigkeitsfehlerüberlauf im High-Power-Modus. Motorkabelverbindung, Roboter, Bremse, Treiber oder Motor überprüfen.	Geben Sie die Weight/Inertia-Einstellung an. Überprüfen Sie die Last. Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		
5045	Geschwindigkeitsfehlerüberlauf im Low-Power-Modus. Motorkabelverbindung, Roboter, Bremse, Treiber oder Motor überprüfen.	Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5046	Zu hohe Geschwindigkeit im High-Power-Modus. Speed S reduzieren. Signalkabelverbindung, Roboter, Bremse, Treiber oder Motor überprüfen.	Reduzieren Sie SpeedS der CP-Bewegung. Ändern Sie die Ausrichtung der CP-Bewegung. Geben Sie die Weight/Inertia-Einstellung an. Überprüfen Sie die Last. Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		
5047	Zu hohe Geschwindigkeit im Low-Power-Modus. . Signalkabelverbindung, Roboter, Bremse, Treiber oder Motor überprüfen.	Überprüfen Sie die Bewegung im High-Power-Modus. Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5048	Überspannung des Hauptstromkreises. Versorgungsspannung oder regeneratives Bremsmodul überprüfen.	Geben Sie die Weight/Inertia-Einstellung an. Überprüfen Sie die Last. Überprüfen Sie den Roboter. (Gleichmäßigkeit, Flankenspiel, nicht gleichmäßige Bewegung, lockere Riemenspannung, Bremse) Stellen Sie sicher, dass der Roboter nicht mit Peripheriegeräten kollidiert. (Kollision, Berührung) Überprüfen Sie die Modelleinstellungen. Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die Verdrahtung des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Überprüfen Sie die Spannung der Stromversorgung. (Unterspannung des Netzteils) Tauschen Sie den Motortreiber aus. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus.		
5049	Überstrom des Motortreibers. Motorkabelverbindung oder interne Verdrahtung des Roboters überprüfen.	Überprüfen Sie, ob ein Kurzschluss oder Erdschluss der Stromleitung vorliegt. Motortreiber austauschen. Tauschen Sie das DMB aus.		
5050	Zu hohe Geschwindigkeit während Torque Control. Geschwindigkeitbereich der Arbeitsbewegung auf Plausibilität überprüfen.	Überprüfen Sie die Geschwindigkeit der Bewegung während Torque Control.		
5051	Netzteilfehler des 15V-PWM-Treibers. Starten Sie die Steuerung neu. 15V-Netzteil austauschen.	Überprüfen Sie die 15 V-Stromversorgung und die Kabelverbindung. Motortreiber austauschen. Tauschen Sie das DMB aus.		
5054	Motorüberlast. Accel oder die Bewegungsbelastung reduzieren.	Reduzieren Sie die Bewegungsleistung. Überprüfen Sie die Weight/Inertia-Einstellung. Überprüfen Sie den Roboter. (Flankenspiel, hohe Last, lockere Riemenspannung, Bremse)		
5055	Motorüberlast. Accel oder die Arbeitsbelastung reduzieren.	Reduzieren Sie die Bewegungsleistung. Überprüfen Sie die Weight/Inertia-Einstellung. Überprüfen Sie den Roboter. (Flankenspiel, hohe Last, lockere Riemenspannung, Bremse)		
5072	Servo-Alarm B.			

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
5080	Motor ist überlastet. Belastung und Accel reduzieren.	Reduzieren Sie die Bewegungsleistung. Überprüfen Sie die Weight/Inertia-Einstellung. Überprüfen Sie den Roboter. (Flankenspiel, hohe Last, lockere Riemenspannung, Bremse)		
5098	Zu hohe Temperatur des Encoders. Belastung reduzieren. Robotergetriebe überprüfen.	Warten Sie, bis die Temperatur des Encoders sinkt. Reduzieren Sie die Bewegungsleistung. Überprüfen Sie die Weight/Inertia-Einstellung. Überprüfen Sie den Roboter. (Flankenspiel, hohe Last, lockere Riemenspannung, Bremse)		
5099	Zu hohe Temperatur des Motortreibers. Luftfilter der Steuerung reinigen. Umgebungstemperatur überprüfen. Belastung reduzieren.	Reinigen Sie den Luftfilter. Reduzieren Sie die Bewegungsleistung. Überprüfen Sie die Weight/Inertia-Einstellung. Senken Sie die Umgebungstemperatur.		
5112	Servo-Alarm C.			

Punkte

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
7003	Außergewöhnlicher Fehler. Der angegebene Roboter wurde nicht gefunden.			
7004	Außergewöhnlicher Fehler. Doppelte Zuweisung des Punktespeicherbereichs.			
7006	Die angegebene Punktnummer wurde nicht gefunden. Eine gültige Punktnummer angeben.	Überprüfen Sie die angegebene Punktnummer.		
7007	Die angegebene Punktnummer ist nicht definiert. Einen Teachpunkt angeben.	Überprüfen Sie, Ob die Punktdaten im angegebenen Punkt registriert sind. Führen Sie das Teachen aus.		
7010	Außergewöhnlicher Fehler. Speicherbereich für die Palettendefinition kann nicht zugewiesen werden.			
7011	Außergewöhnlicher Fehler. Speicherbereich für die Palettendefinition kann nicht freigemacht werden.			
7012	Die angegebene Palettennummer wurde nicht gefunden. Eine gültige Palettennummer angeben.	Überprüfen Sie die Palettennummer.		
7013	Die angegebene Palette ist nicht definiert. Eine definierte Palette angeben oder eine Palette definieren.	Überprüfen Sie, ob die angegebene Palette im Palettenbefehl definiert ist. Deklarieren Sie die Palette.		
7014	Die angegebene Teilungszahl ist außerhalb der Teilungszahl der Palettendefinition. Eine gültige Teilung angeben.	Überprüfen Sie die angegebene Teilungszahl.		
7015	Außergewöhnlicher Fehler. Die angegebene Nummer der Koordinatenachse existiert nicht.			
7016	Außergewöhnlicher Fehler. Die angegebene Nummer der Armorientierung existiert nicht.			
7017	Außergewöhnlicher Fehler. Der erforderliche Speicher kann nicht zugewiesen werden.			
7018	Das angegebene Punktlabel wurde nicht gefunden. Ein gültiges Punktlabel angeben.	Überprüfen Sie das angegebene Punktlabel.		
7019	Außergewöhnlicher Fehler. Parametereinstellungen in der Initialisierungsdatei sind ungültig.			
7021	Doppeltes Punktlabel. Das angegebene Label wurde bereits eingetragen. Label ändern.	Ändern Sie das Punktlabel.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
7022	Das angegebene, lokale Koordinatensystem wurde nicht definiert. Eine gültige Nummer für das lokale Koordinatensystem angeben.	Überprüfen Sie die angegebene Local-Nummer. Definieren Sie das lokale Koordinatensystem.		
7023	Außergewöhnlicher Fehler. Der angegebene String hat das falsche Format.			
7024	Außergewöhnlicher Fehler. Speicherbereich der Punktedaten des angegebenen Roboters wurde nicht zugewiesen.			
7026	Punktedatei kann nicht geöffnet werden. Einen gültigen Punktedateinamen angeben.	Überprüfen Sie den Punktedateinamen. Überprüfen Sie, ob die angegebene Punktedatei für das Projekt existiert.		
7027	Aus der Punktedatei können keine Punktedaten gelesen werden.	Erstellen Sie eine neue Punktedatei.		
7028	Außergewöhnlicher Fehler. Punktespeicherbereich wurde über die maximale Punktezahl hinaus zugewiesen.			
7029	Der angegebene Punktedateiname ist falsch. Einen gültigen Punktedateinamen angeben.	Überprüfen Sie die Dateierweiterung.		
7030	Punktlabel ist zu lang. Ein gültiges Punktlabel angeben.	Ändern Sie das Punktlabel.		
7031	Beschreibung des angegebenen Punktes ist zu lang. Eine gültige Beschreibung angeben.	Ändern Sie den Kommentar.		
7032	Punktedatei ist beschädigt. Prüfsummenfehler.	Erstellen Sie eine neue Punktedatei.		
7033	Angegebene Punktedatei nicht gefunden. Einen gültigen Punktedateinamen angeben.			

Feldbus

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
7101	Kommunikationsfehler während der Übertragung.	Das Modul oder die Software der Steuerung sind beschädigt. Stellen Sie die Firmware der Steuerung wieder her.	1	
			2	
			3	
			4	
			10	
		Ein Kommunikationsfehler wurde während der Kommunikation erkannt. Das Kommunikationskabel ist fehlerhaft. Überprüfen Sie das Kommunikationskabel und die daran angeschlossenen Einheiten.	11	
			12	
		Das Modul oder die Software der Steuerung sind beschädigt. Stellen Sie die Firmware der Steuerung wieder her.	13	
			14	
			15	
7103	Zeitüberlauf während der Übertragung.	Das Modul oder die Software der Steuerung sind beschädigt. Stellen Sie die Firmware der Steuerung wieder her.	1	
			2	
			3	
		Ein Kommunikationsfehler wurde während der Kommunikation erkannt. Das Kommunikationskabel ist fehlerhaft. Überprüfen Sie das Kommunikationskabel und die daran angeschlossenen Einheiten.	4	

Hardware

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
9001	Fehler des Not-Aus-Kreises erkannt. Unterbrechung oder anderer Fehler in einem der redundanten Kreise.	Stellen Sie sicher, dass keine Verbindungsunterbrechung, kein Erdschluss und kein Kurzschluss des Not-Aus-Eingangs vorliegt. Starten Sie anschließend die Steuerung neu.		
9002	Fehler der Sicherheitsabschrankung erkannt. Unterbrechung oder anderer Fehler in einem der redundanten Kreise.	Stellen Sie sicher, dass keine Verbindungsunterbrechung, kein Erdschluss und kein Kurzschluss des Eingangs der Sicherheitsabschrankung vorliegt. Starten Sie anschließend die Steuerung neu.		
9014	Zu hohe Innentemperatur der Steuerung.	Halten Sie die Steuerung so bald wie möglich an und stellen Sie sicher, dass die Umgebungstemperatur der Steuerung nicht hoch ist. Stellen Sie sicher, dass der Filter nicht verstopft ist.	Aktueller Wert	Randwert
9015	Die Rotationsgeschwindigkeit des Lüfters ist zu niedrig. (Lüfter 1)	Stellen Sie sicher, dass der Filter nicht verstopft ist. Wenn die Warnung immer noch ansteht, nachdem die Steuerung wieder hochgefahren wurde, wechseln Sie den Lüfter aus.	Aktueller Wert	Randwert
9016	Die Rotationsgeschwindigkeit des Lüfters ist zu niedrig. (Lüfter 2)	Stellen Sie sicher, dass der Filter nicht verstopft ist. Wenn die Warnung immer noch ansteht, nachdem die Steuerung wieder hochgefahren wurde, wechseln Sie den Lüfter aus.	Aktueller Wert	Randwert
9017	Zu hohe Innentemperatur der Steuerung.			
9100	Initialisierungsfehler. Außergewöhnlicher Fehler. Speicherzuweisungsfehler.	Starten Sie die Steuerung neu.		
9233	Außergewöhnlicher Fehler. Feldbus-E/A-Treiber ist in einem unnormalen Zustand.	Das Modul oder die Software der Steuerung sind beschädigt. Stellen Sie die Firmware der Steuerung wieder her.		
9234	Initialisierungsfehler des Feldbus-E/A-Treibers.	Das Modul oder die Software der Steuerung sind beschädigt. Stellen Sie die Firmware der Steuerung wieder her.		
9610	Der RAS-Kreis erkannte eine Störung des Servosystems. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie die Steuerung aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9611	Interner RAM-Fehler der Servo CPU. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9612	Kommunikationsfehler des RAM der Main- und Servo-CPU. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9613	Interner RAM-Fehler der Servo CPU. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
9614	Initialisierungsfehler der Kommunikation der Main- und Servo-CPU. Steuerung neu booten. Störsignale messen. DBM austauschen.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9615	Initialisierungsfehler der Kommunikation der Main- und Servo-CPU. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9616	Kommunikationsfehler der Main- und Servo-CPU. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9617	Kommunikationsfehler der Main- und Servo-CPU. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9618	Servo-Langzeit-Befehlsüberlauf.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9619	Servo-Prüfsummenfehler im Langzeitbefehl.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9620	Fehler durch System-Watchdog-Timer erkannt. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9621	Drive Unit Prüffehler.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9622	RAM-Fehler der Servo-CPU. Starten Sie die Steuerung neu. Störsignale messen. Tauschen Sie das DMB aus.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9623	Fehler der redundanten Kreise des Not-Aus oder der Sicherheitsabschrankung. Verdrahtung überprüfen.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9624	Unterspannung des Hauptstromkreises des Netzteils erkannt. Spannungsversorgung überprüfen. Starten Sie die Steuerung neu.	Überprüfen Sie die Störungsreduzierung. Tauschen Sie das DMB aus.		
9625	Steuerrelaiskontakt des Netzteils des Hauptstromkreises klebt. DPB austauschen.	Tauschen Sie das DMB aus.		
9630	Servo-Echtzeit-Statusfehler. Außergewöhnlicher Fehler. Prüfsummenfehler.	Starten Sie die Steuerung neu. Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
9632	Servo-Echtzeit-Statusfehler. Außergewöhnlicher Fehler. Fehler im Freilaufzähler des Servos.	Starten Sie die Steuerung neu. Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
9633	Servo-Echtzeit-Statusfehler. Außergewöhnlicher Fehler. Kommunikationfehler mit der Servo-CPU.	Starten Sie die Steuerung neu. Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
9640	Irreguläre Unterbrechung der Bewegungssteuerung wurde erkannt. Außergewöhnlicher Fehler. Doppelte Unterbrechung.	Starten Sie die Steuerung neu. Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
9700	Gate-Array-Fehler der Servo-Steuerung. DMB überprüfen.	Überprüfen Sie, ob ein Kurzschluss vorliegt oder ob die Peripheriegeräte nicht richtig verkabelt sind. (Not-Aus- und E/A-Anschlüsse) Tauschen Sie das DMB aus. Tauschen Sie die zusätzliche Achseneinheit aus.		
9701	Unterbrechung des Parallelencodersignals. Die Signalkabelverbindung oder die interne Verdrahtung des Roboters überprüfen.	Überprüfen Sie das M/C-Kabelsignal. Überprüfen Sie die Signalleitungen des Roboters. (Fehlender Anschluss, Verbindungsunterbrechung, Kurzschluss) Tauschen Sie den Motor aus. (Encoder-Fehler) Tauschen Sie das DMB aus. (Kreisfehlererkennung) Überprüfen Sie den Anschluss des Anschlusses in der Steuerung. (Lösen, an den Anschluss des seriellen Encoders am DMB anschließen.) Überprüfen Sie die Modelleinstellungen. (Üngültige Einstellungen des Parallelencoders) Überprüfen Sie die Verkabelung der Peripheriegeräte. (Not-Aus und E/A)		
9702	Motortreiber ist nicht installiert. Motortreiber installieren. DMB oder Motortreiber überprüfen.	Überprüfen Sie, ob der Motortreiber installiert ist. Überprüfen Sie die Modell- und Hardwareeinstellungen. Motortreiber austauschen. Tauschen Sie das DMB aus.		
9703	Initialisierungsfehler der Kommunikation des Inkrementalencoders. Signalkabelverbindungen und Robotereinstellungen überprüfen.	Überprüfen Sie die Modelleinstellungen. Tauschen Sie den Motor aus. (Encoder-Fehler) Tauschen Sie das DMB aus.		
9704	Initialisierungsfehler des Absolutencoders. Signalkabelverbindungen und Robotereinstellungen überprüfen.	Überprüfen Sie die Modelleinstellungen. Tauschen Sie den Motor aus. (Encoder-Fehler) Tauschen Sie das DMB aus.		
9705	Fehler der Einstellung der Encoderteilung. Robotereinstellungen überprüfen.	Überprüfen Sie die Modelleinstellungen.		
9706	Datenfehler während der Absolutencoderinitialisierung. Signalkabelverbindung, Steuerung und Motoren überprüfen.	Tauschen Sie den Motor aus. (Encoder-Fehler) Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
9707	Absolutencoderumdrehungen über Maximalwert. Encoder-Reset durchführen.	Encoder-Reset durchführen. Tauschen Sie den Motor aus. (Encoder-Fehler)		
9708	Position außerhalb des Pulsebereichs. Encoder-Reset durchführen.	Encoder-Reset durchführen. Tauschen Sie das DMB aus. Tauschen Sie den Motor aus. (Encoder-Fehler)		
9709	Keine Antwort vom seriellen Encoder. Signalkabelverbindung, Motor, DMB oder Encoderanschlussboard überprüfen.	Überprüfen Sie die Modelleinstellungen. (Ungültige Einstellungen des Parallelencodermodells) Überprüfen Sie die Signalkabelverbindung. Tauschen Sie das DMB und das Encoderanschlussboard aus.		

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
9710	Fehler beim Initialisieren des seriellen Encoders. Starten Sie die Steuerung neu. Motor, DMB oder Encoderanschlussboard überprüfen.	Überprüfen Sie die Konfiguration des Roboters. Überprüfen Sie die Signalkabel. Tauschen Sie das DMB und das Encoderanschlussboard aus.		
9711	Fehler bei der Initialisierung der Kommunikation des seriellen Encoders. Starten Sie die Steuerung neu. Motor, DMB oder Encoderanschlussboard überprüfen.	Überprüfen Sie die Konfiguration des Roboters. Überprüfen Sie die Signalkabel. Tauschen Sie das DMB und das Encoderanschlussboard aus.		
9712	Fehler des Watchdog-Timers der Servo-CPU. Starten Sie die Steuerung neu. Motor oder DMB überprüfen.	Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
9713	Fehler des WDT des aktuellen Steuerkreises. Starten Sie die Steuerung neu. Steuerung überprüfen.	Überprüfen Sie die Stromkabelverbindung. Überprüfen Sie die 15 V-Stromversorgung und die Kabelverbindung. Tauschen Sie das DMB aus. Überprüfen Sie die Störungsreduzierung.		
9715	Encoder-Reset durchgeführt. Starten Sie die Steuerung neu.	Starten Sie die Steuerung neu.		
9716	Spannungsversorgungsfehler des Absolutencoders. Batterie austauschen. Interne Verdrahtung des Roboters überprüfen.	Encoder-Reset durchführen. Überprüfen Sie die Signalkabelverbindung.		
9717	Fehler der Backup-Daten des Absolutencoders. Encoder-Reset durchführen.	Encoder-Reset durchführen. Überprüfen Sie die Signalkabelverbindung.		
9718	Absolutencoder Batteriealarm.	Batterie austauschen. Überprüfen Sie die Signalkabelverbindung.		
9719	Positionsfehler des Absolutencoders. Encoder-Reset durchführen. Tauschen Sie den Motor aus.	Encoder-Reset durchführen. Tauschen Sie den Motor aus. (Encoder-Fehler)		
9720	Zu hohe Geschwindigkeit beim Einschalten der Steuerung. Roboter anhalten und Steuerung neu booten.	Starten Sie die Steuerung neu.		
9721	Absolutencoder Überhitzung.	Reduzieren Sie die Bewegungsleistung. Warten Sie, bis die Temperatur des Encoders sinkt.		
9732	Servo-Alarm A.			

EPSON RC+

Nr.	Meldung	Abhilfe	Anmerkung 1	Anmerkung 2
10000	Befehl durch Benutzer abgebrochen.			
10001	Befehlszeitüberlauf.			
10002	Falsche Punktedatei-Zeilensyntax.			
10003	Projekt konnte nicht generiert werden.			
10004	Spel-Klasseninstanz konnte nicht initialisiert werden.			
10005	Syntaxprüfung konnte nicht initialisiert werden.			
10006	wbproxy konnte nicht analysiert werden.			
10007	Projekt existiert nicht.			
10008	Kein Projekt angegeben.			
10009	Datei konnte nicht geöffnet werden.			
10010	Datei konnte nicht erzeugt werden.			
10011	Datei nicht gefunden.			
10012	Option nicht aktiviert.			

Vorkehrungen in Bezug auf die EPSON RC+ 4.0-Kompatibilität

Übersicht

Dieser Abschnitt enthält Informationen für Kunden, die EPSON RC+ 5.0 mit der RC170 Steuerung verwenden und die bereits EPSON RC+ 4.0 mit der RC520 Steuerung oder der RC420 Steuerung verwendet haben.
Die Unterschiede zwischen EPSON RC+ 5.0 und EPSON RC+ 4.0 liegen vor allem in der Hardware, den verwendbaren Manipulatoren, der zulässigen Achsenanzahl und der Softwareausführungsumgebung. Die Lektüre und das Verständnis dieses Abschnitts sind entscheidend für den sicheren Gebrauch des Robotersystems.
EPSON RC+ 5.0 ist eine verbesserte Software, die auch mit älteren Produkten kompatibel und dafür ausgelegt ist, fortgeschrittene Softwaretechnologien zu erneuern. Einzelne Teile sind jedoch nicht mit EPSON RC+ 4.0 kompatibel oder wurden gelöscht, um die Robotersteuerung zu spezialisieren und die Verwendung zu vereinfachen.
Die folgende Darstellung der Kompatibilität basiert auf dem Vergleich zwischen EPSON RC+ 4.0 und EPSON RC+ 5.0.

Allgemeine Unterschiede

Die folgenden allgemeinen Unterschiede bestehen zwischen EPSON RC+ 4.0 und EPSON RC+ 5.0.

Parameter	EPSON RC+ 5.0	EPSON RC+ 4.0
Anzahl der Tasks	Bis zu 16 Tasks	Bis zu 32 Tasks
Tasktyp	Kann NoPouse-Task nicht spezifizieren	Kann NoPouse-Task spezifizieren
Spezieller Trap, so wie TRAP ERROR	Nicht unterstützt	Unterstützt
Task startet mit TRAP-Nummer	Vorgesehene Tasknummer	Tasknummer nur zwischen 1 und 32
Anzahl der signifikanten Stellen für Realzahlentyp	6 Zahlzeichen	Keine Angabe
Anzahl der signifikanten Stellen für Double-Typ	14 Zahlzeichen	Keine Angabe
Anzahl der Matrixelemente	Keine Zeichenkettenvariablen Lokale Variable 1000 Globale Variable 1000 Modulvariable 0 Global Preserve-Variable 1000 0 1000 Zeichenkettenvariable Lokale Variable 100 Globale Variable 1000 Modulvariable 1000 Global Preserve-Variable 100	Abhängig vom freien Speicher
Zeilennummern	Nicht unterstützt	Unterstützt
Gerätenummer	21:PC 22:REMOTE 23:OP	1:Steuerung 2:REMOTE 3:OP
Timernummer-Bereich	0 bis 15	0 bis 63
Signalnummer-Bereich für SyncLock, SyncUnlock	0 bis 15	1 bis 32
Signalnummer-Bereich für Wait Sig, Signal	0 bis 5	0 bis 127
Portnummer in Ethernet	201 bis 208	128 bis 147
Portnummer in der RS-232C-Kommunikation	1 bis 8	1 bis 16
OpenCom-Ausführung des RS-232C-Kommunikationsports	Obligatorisch	Optional
Eingabe / Ausgabe an Dateien	Nicht unterstützt	Unterstützt
Feldbus-E/As	Verwendung normaler E/A-Befehle	Verwendung spezieller Befehle
Gruppe im Projekt	Nicht unterstützt	Unterstützt
Fehlernummer	Neue Fehlernummern	

Kompatibilitätsliste für Befehle

+	Funktionserweiterungen / Funktionsänderungen vorgenommen mit hoher Kompatibilität wurden vorgenommen.
–	Keine Änderungen.
!	Beachten. Es wurden Funktions- oder Syntaxänderungen vorgenommen.
!!	Beachten. Es wurden bedeutende Änderungen vorgenommen.
×	Gelöscht.

Befehl		Kompatibilität	Hinweis
A	Abs-Funktion	–	
	Accel-Anweisung	–	
	Accel-Funktion	–	
	AccelR-Anweisung	–	
	AccelR-Funktion	–	
	AccelS-Anweisung	–	
	AccelS-Funktion	–	
	Acos-Funktion	+	Eine Überprüfung des Argumentenbereichs wurde ergänzt.
	Agl-Funktion	–	
	AglToPls-Funktion	–	
	And-Operator	–	
	AOpen-Anweisung	×	
	Arc-Anweisung	–	
	Arc3-Anweisung	–	
	Arch-Anweisung	–	
	Arch-Funktion	–	
	Arm-Anweisung	–	
	Arm-Funktion	–	
	ArmClr-Anweisung	–	
	Armset-Anweisung	–	
	ArmSet-Funktion	–	
	Asc-Funktion	–	
	Asin-Funktion	+	Eine Überprüfung des Argumentenbereichs wurde ergänzt.
	Atan-Funktion	–	
	Atan2-Funktion	–	
	ATCLR-Anweisung	–	
	ATRQ-Anweisung	–	
	ATRQ-Funktion	–	
B	Base-Anweisung	–	
	BClr-Funktion	+	Eine Überprüfung des Argumentenbereichs wurde ergänzt.
	Beep-Anweisung	×	
	BGo-Anweisung	–	
	BMove-Anweisung	–	
	Boolean-Anweisung	–	
	BOpen-Anweisung	×	

Befehl		Kompatibilität	Hinweis
	Brake-Anweisung	+	Ermöglicht, die dritte Achse der E2-Serie freizugeben.
	BSet-Funktion	+	Eine Überprüfung des Argumentenbereichs wurde ergänzt.
	BTst-Funktion	+	Eine Überprüfung des Argumentenbereichs wurde ergänzt.
	Byte-Anweisung	–	
C	Calib-Anweisung	×	
	Call-Anweisung	–	
	CalPls-Anweisung	×	
	CalPls-Funktion	–	
	Chain-Anweisung	×	
	ChDir-Anweisung	×	
	ChDrive-Anweisung	×	
	ChkCom-Funktion	–	
	ChkNet-Funktion	–	
	Chr$-Funktion	–	
	Clear-Anweisung	!	In ClearPoints umbenannt
	Close-Anweisung	×	
	CloseCom-Anweisung	–	
	CloseNet-Anweisung	+	Ermöglicht, „All“ zu spezifizieren
	ClrScr-Anweisung	!	In Cls umbenannt. Geräte-ID kann für Argumente spezifiziert werden.
	Cnv_**	×	
	Cont-Anweisung	×	
	Copy-Anweisung	×	
	Cos-Funktion	–	
	CP-Anweisung	–	
	CP-Funktion	–	
	Ctr-Funktion	–	
	CTReset-Statement	–	
	CtrlDev-Anweisung	×	
	CtrlDev-Funktion	!	Geänderte Geräte-ID
	CtrlInfo-Funktion	!!	Die erhaltenen Inhalte wurden geändert.
	CurDir$-Funktion	×	
	CurDrive$-Funktion	×	
	CurPos-Funktion	–	
	Curve-Anweisung	–	
	CVMove-Anweisung	–	
	CX zu CW-Anweisung	–	
	CX zu CW-Funktion	–	
D	Date-Anweisung	–	
	Date$-Funktion	–	
	Declare-Anweisung	×	
	DegToRad-Funktion	–	
	Del-Anweisung	×	

Befehl		Kompatibilität	Hinweis
	Dir-Anweisung	×	
	Dist-Funktion	–	
	Do...Loop-Anweisung	–	
	Double-Anweisung	!	Anzahl der signifikaten Stellen umfasst 14 Zahlzeichen.
E	EClr-Anweisung	×	
	ECP-Anweisung	–	
	ECP-Funktion	–	
	ECPClr-Anweisung	–	
	ECPSet-Anweisung	–	
	ECPSet-Funktion	–	
	Elbow-Anweisung	–	
	Elbow-Funktion	–	
	ENetIO_****	×	
	Eof-Funktion	×	
	EPrint-Anweisung	×	
	Era-Funktion	+	Ermöglicht das Weglassen von Tasknummern.
	Erase-Anweisung	×	
	EResume-Anweisung	–	
	Erf$-Funktion	+	Ermöglicht das Weglassen von Tasknummern.
	Erl-Funktion	+	Ermöglicht das Weglassen von Tasknummern.
	Err-Funktion	–	
	ErrHist-Anweisung	×	
	ErrMsg$-Funktion	!	Das Argument verfügt über eine Sprach-ID.
	Error-Anweisung	+	Ermöglicht die Spezifizierung von Tasknummern für Argumente.
	Ert-Funktion	–	
	EStopOn-Funktion	–	
	Eval-Funktion	×	
	Exit-Anweisung	–	
F	FbusIO_****	×	Normaler E/A-Befehl möglich
	FileDateTime$-Funktion	×	
	FileExists-Funktion	×	
	FileLen-Funktion	×	
	Find-Anweisung	–	
	FindPos-Funktion	–	
	Fine-Anweisung	–	
	Fine-Funktion	–	
	Fix-Funktion	–	
	FmtStr$-Anweisung	!!	Die Funktion wurde deutlich begrenzt.
	FoldrExist-Funktion	×	
	For...Next	–	
	FreeFile-Funktion	×	

Befehl		Kompatibilität	Hinweis
	Function...Fend	–	
G	GetCurrentUser$-Funktion	×	
	Global-Anweisung	–	
	Go-Anweisung	–	
	Gosub...Return	–	
	Goto-Anweisung	–	
H	Halt-Anweisung	–	
	Hand-Anweisung	–	
	Hand-Funktion	–	
	Here-Anweisung	–	
	Here-Funktion	–	
	Hex$-Funktion	–	
	Hofs-Anweisung	×	
	Hofs-Funktion	–	
	Home-Anweisung	–	
	HomeSet-Anweisung	–	
	HomeSet-Funktion	–	
	HOrdr-Anweisung	–	
	HOrdr-Funktion	–	
	Hour-Anweisung	–	
	Hour-Funktion	–	
	HTest-Anweisung	×	
	HTest-Funktion	×	
I	If...EndIf	–	
	ImportPoints-Anweisung	×	
	In-Funktion	–	
	In($n)-Anweisung	×	Ersetzt durch MemIn
	InBCD-Funktion	–	
	Inertia-Anweisung	–	
	Inertia-Funktion	–	
	InPos-Funktion	–	
	Input-Anweisung	–	
	Input #-Anweisung	+	Eingabe ist von Geräten aus möglich.
	InputBox-Anweisung	×	
	InStr-Funktion	–	
	Int-Funktion	–	
	Integer-Anweisung	–	
	InW-Funktion	–	
	InW($n)-Anweisung	×	Ersetzt durch MemInW
	IONumber-Funktion	–	
J	J4Flag-Anweisung	–	
	J4Flag-Funktion	–	
	J6Flag-Anweisung	–	
	J6Flag-Funktion	–	

Befehl		Kompatibilität	Hinweis
	JA-Funktion	–	
	JRange-Anweisung	–	
	JRange-Funktion	–	
	JS-Funktion	!	Gibt Wahr / Falsch aus.
	JT-Funktion	–	
	JTran-Anweisung	–	
	Jump-Anweisung	–	
	Jump3-Anweisung	–	
	Jump3CP-Anweisung	–	
K	Kill-Anweisung	×	
L	LCase$-Funktion	–	
	Left$-Funktion	–	
	Len-Funktion	–	
	LimZ-Anweisung	–	
	LimZ-Funktion	–	
	Line Input-Anweisung	–	
	Line Input #-Anweisung	+	Eingabe ist von Geräten aus möglich.
	LoadPoints	!	Die Endung (.pnt) wurde in (.pts) geändert.
	Local-Anweisung	!	Die Localnummer „0” ist ein Fehler.
	Local-Funktion	!	Die Localnummer „0” ist ein Fehler.
	LocalClr-Anweisung	–	
	Lof-Funktion	–	
	LogIn Anweisung	×	
	Long-Anweisung	–	
	LPrint-Anweisung	×	
	LSet$-Funktion	–	
	LShift-Funktion	+	Eine Überprüfung des Argumentenbereichs wurde ergänzt.
	LTrim$-Funktion	–	
M	Mask-Operator	–	
	MCal-Anweisung	×	
	MCalComplete-Funktion	×	
	MCofs-Anweisung	×	
	MCofs-Funktion	×	
	MCordr-Anweisung	×	
	MCordr-Funktion	×	
	Mcorg-Anweisung	×	
	MemIn-Funktion	–	
	MemInW-Funktion	–	
	MemOff-Anweisung	–	
	MemOn-Anweisung	–	
	MemOut-Anweisung	–	
	MemOutW-Anweisung	–	
	MemSw-Funktion	–	

Befehl		Kompatibilität	Hinweis
	Mid$-Funktion	–	
	MKDir-Anweisung	×	
	Mod-Operator	–	
	Motor-Anweisung	–	
	Motor-Funktion	–	
	Move-Anweisung	–	
	MsgBox-Anweisung	×	
	MyTask-Funktion	–	
N	Not-Operator	–	
O	Off-Anweisung	–	
	Off$-Anweisung	×	Ersetzt durch MemOff
	OLRate-Anweisung	–	
	OLRate-Funktion	–	
	On-Anweisung	–	
	On-Anweisung	×	Ersetzt durch MemOn
	OnErr	–	
	OP_*	×	
	OpBCD-Anweisung	–	
	OpenCom-Anweisung	!	OpenCom ist obligatorisch.
	OpenNet-Anweisung	–	
	Oport-Funktion	–	
	Or-Operator	–	
	Out-Anweisung	–	
	Out-Funktion	–	
	Out$-Anweisung	×	Ersetzt durch MemOut
	OutW-Anweisung	–	
	OutW-Funktion	–	
	OutW$-Anweisung	×	Ersetzt durch MemOutW
P	PAgl-Funktion	–	
	Pallet-Anweisung	–	
	Pallet-Funktion	–	
	ParsStr-Anweisung	–	
	ParsStr-Funktion	–	
	Pass-Anweisung	+	Ermöglicht die Spezifikation eines fortlaufenden Punktes.
	Pause-Anweisung	–	
	PauseOn-Funktion	–	
	PDef-Funktion	–	
	PDel	+	Eine Argumentüberprüfung wurde ergänzt.
	PLabel$-Funktion	–	
	PLabel-Anweisung	–	
	PList	!	Eine Argumentüberprüfung wurde ergänzt. Die Funktion von Plist* wurde gelöscht.
	PLocal-Anweisung	–	

Befehl		Kompatibilität	Hinweis
	PLocal-Funktion	–	
	Pls-Funktion	–	
	PNumber-Funktion	–	
	Punkt-Zuweisung	–	
	Punktausdruck	–	
	POrient-Anweisung	×	
	POrient-Funktion	×	
	PosFound-Funktion	!	Gibt Wahr / Falsch aus.
	Power-Anweisung	–	
	Power-Funktion	–	
	PPls-Funktion	–	
	Print-Anweisung	!	Gibt alle Flags an einem Punktausgang aus. Stellt die Ausgangsnummer des Double-Typs oder des Realzahlentyps auf die signifikante Stelle ein.
	Print #-Anweisung	!	Siehe Print-Anweisung. Aktiviert Print für die einzelnen Geräte.
	PTCLR-Anweisung	–	
	PTPBoost-Anweisung	–	
	PTPBoost-Funktion	–	
	PTPBoostOK-Funktion	!	Gibt Wahr / Falsch aus.
	PTPTime-Funktion	–	
	PTran-Anweisung	–	
	PTRQ-Anweisung	–	
	PTRQ-Funktion	–	
	Pulse-Anweisung	–	
	Pulse-Funktion	–	
Q	QP-Anweisung	–	
	Quit-Anweisung	–	
R	RadToDeg-Funktion	–	
	Randmize-Anweisung	+	Der Geschwindigkeitswert kann spezifiziert werden.
	Range-Anweisung	–	
	Read-Anweisung	–	
	ReadBin-Anweisung	+	Ermöglicht das Einlesen mehrerer Bytes in die Feldvariable.
	Real-Anweisung	!	Sechsstellige signifikante Stelle
	Recover-Anweisung	×	
	Redim-Anweisung	!	Die Anzahl der Elemente ist begrenzt. Ein Feld, das by reference aufgerufen wurde, kann nicht ausgeführt werden.
	Rename-Anweisung	×	
	RenDir-Anweisung	×	
	Reset-Anweisung	–	
	Resume-Anweisung	–	
	Restart-Anweisung	×	
	Reset-Anweisung	–	

Befehl		Kompatibilität	Hinweis
	Return-Anweisung	–	
	Right$-Funktion	–	
	RmDir-Anweisung	×	
	Rnd-Funktion	–	
	Robot-Anweisung	×	
	Robot-Funktion	×	
	RobotModel$-Funktion	–	
	RobotType-Funktion	–	
	ROpen-Anweisung	×	
	RSet$-Funktion	–	
	RShift-Funktion	+	Eine Argumentüberprüfung wurde ergänzt.
	RTrim$-Funktion	–	
	RunDialog-Anweisung	×	
S	SafetyOn-Funktion	–	
	SavePoints-Anweisung	!	Die Endung (.pnt) wurde in (.pts) geändert.
	Seek-Anweisung	×	
	Select...Send	–	
	Sense	–	
	SetCom-Anweisung	!	Kann „56000” nicht für die Übertragungsrate spezifizieren. Ein Port mit OpenCom kann nicht ausgeführt werden.
	SetNet-Anweisung	–	
	SFree-Anweisung	–	
	SFree-Funktion	–	
	Sgn-Funktion	–	
	Shutdown-Anweisung	×	
	Signal-Anweisung	–	
	Sin-Funktion	–	
	SLock-Anweisung	–	
	Space$-Funktion	–	
	Speed-Anweisung	–	
	Speed-Funktion	+	Argument optional
	SpeedR-Anweisung	–	
	SpeedR-Funktion	–	
	SpeedS-Anweisung	–	
	SpeedS-Funktion	–	
	SPELCom_Event-Anweisung	–	
	SPELCom_Return-Anweisung	×	
	Sqr-Funktion	–	
	Stat-Funktion	!	Einige Informationen können nicht ausgegeben werden.
	Str$-Funktion	–	
	String-Anweisung	–	

Befehl		Kompatibilität	Hinweis
	Sw-Funktion	–	
	Sw($)-Funktion	×	Ersetzt durch MemSw
	SyncLock-Anweisung	!	Beim wiederholten Ausführen von SyncLock tritt ein Fehler auf.
	SyncUnlock-Anweisung	–	
T	Tab$-Funktion	–	
	Tan-Funktion	–	
	TargetOK-Funktion	!	Gibt Wahr / Falsch aus.
	TaskDone-Funktion	–	
	TaskState-Funktion	!	6 spezifizierte Tasks werden während der Ausführung der Wait-Anweisung nicht ausgegeben.
	TaskWait-Anweisung	–	
	TGo-Anweisung	–	
	TillOn-Funktion	–	
	Time-Befehl	–	
	Time-Funktion	–	
	Time$-Funktion	–	
	TLClr-Anweisung	–	
	TLSet-Anweisung	–	
	TLSet-Funktion	–	
	TMOut-Anweisung	–	
	TMove-Anweisung	–	
	Tmr-Funktion	–	
	TmReset-Anweisung	–	
	Tool-Anweisung	–	
	Tool-Funktion	–	
	Trap-Anweisung	!!	Kompatibel mit Trap Goto. Trap Gosub wurde durch Trap Call ersetzt. Trap Call wurde in Trap Xqt umbenannt. Trap Emergency, Trap Error, Trap Abort, Trap Pause, Trap SGOpen und Trap SGClose wurden gelöscht.
	Trim$-Funktion	–	
	Tw-Funktion	!	Gibt Wahr / Falsch aus.
	Type-Anweisung	×	
U	Ubound-Funktion	–	
	Ucase$-Funktion	–	
	UOpen-Anweisung	×	
V	Val-Funktion	–	
	Ver-Anweisung	×	Ersetzt durch SysConfig
	Verinit-Anweisung	×	
W	Wait-Anweisung	–	
	WaitNet-Anweisung	–	
	WaitPos-Anweisung	–	
	WaitSig-Anweisung	–	

Befehl		Kompatibilität	Hinweis
	Weight-Anweisung	–	
	Weight-Funktion	–	
	Where-Anweisung	!	Koordinatenwert zeigt immer 6-Achs an.
	While..Wend	×	Ersetzt durch Do…Loop
	WOpen-Anweisung	×	
	Wrist-Anweisung	–	
	Wrist-Funktion	–	
	Write-Anweisung	–	
	WriteBin-Anweisung	+	Multiple Bytes können von der Feldvariablen aufgelistet werden.
X	Xor-Operator	–	
	Xqt-Anweisung	!	NoPause kann nicht spezifiziert werden.
	XY-Funktion	–	
	XYLim-Anweisung	–	
	XYLim-Funktion	–	
Z	ZeroFlg-Funktion	–	

Liste neuer Befehle

ArmDef-Funktion
Base-Funktion
Brake-Funktion
Cls-Anweisung
DispDev-Anweisung
DispDev-Funktion
EcpDef-Funktion
HomeClr-Anweisung
HomeDef-Funktion
IOLabel$-Funktion
Joint-Anweisung
LocalDef-Funktion
RealPls-Funktion
RealPos-Funktion
RobotInfo$-Funktion
RobotInfo-Funktion
RobotName$-Funktion
RobotSerial$-Funktion
TaskInfo$-Funktion
TaskInfo-Funktion
TaskState-Anweisung
TlDef-Funktion
Toff-Anweisung
Ton-Anweisung
XYLimClr-Anweisung
XYLimDef-Funktion